

Thomas W. MacFarland
Jan M. Yates

Using R for Biostatistics

EXTRAS ONLINE

 Springer

Using R for Biostatistics

Thomas W. MacFarland • Jan M. Yates

Using R for Biostatistics

Thomas W. MacFarland
Senior Research Associate, Office of Institutional Effectiveness, Nova Southeastern University
Fort Lauderdale, FL, USA

Associate Professor, College of Computing and Engineering, Nova Southeastern University
Fort Lauderdale, FL, USA

Jan M. Yates
Professor Emerita, Abraham S. Fischler College of Education
Nova Southeastern University
Fort Lauderdale, FL, USA

ISBN 978-3-030-62403-3 ISBN 978-3-030-62404-0 (eBook)
<https://doi.org/10.1007/978-3-030-62404-0>

© Springer Nature Switzerland AG 2021

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Dedication

In appreciation for their patience, this text is dedicated to Andrew, Baylen, Courtney, Henry, and Lauren.

This text is also dedicated to the students and beginning researchers who have struggled with the transition from a graphical approach to statistics to the more empowering, but challenging, use of syntax. We hope you continue to explore the many learning resources available to the R community; with practice, the results gained from learning syntax will be more than worth the effort.

Preface

This text is about the use of R in biostatistics. It was prepared to help beginning students and researchers gradually increase their skills with the use of R syntax as they consider how R is used in the quickly expanding the world of biostatistics.

R has become one of the leading languages used for statistical analyses in the biological sciences, and it is increasingly used for data organization, statistical analyses, and the generation of high-quality publishable graphics. There are currently more than 15,000 R-focused packages freely available to the public, and many focus exclusively on applications in biostatistics. It is our view that those who work in biostatistics should have a good working knowledge of R and an understanding of how R fits into the professional toolkit.

Using R for Biostatistics begins with a brief discussion of biostatistics, with an emphasis on how biostatistics grew out of statistics. There is also a short history of the R language and how R developed from the prior S language. The beginning parts of this text also provide a glimpse of how the lessons have been structured, ranging from learning as much as possible about the data to the eventual development of an easy-to-understand summary of statistical analyses. Attention is also given to the many ways data can be imported into R, focusing on how R can accommodate datasets in various formats.

The major part of this text is presented in the form of lessons that address the leading statistical tests typically encountered early on among those who engage in research associated with biostatistics:

- Data Exploration, Descriptive Statistics, and Measures of Central Tendency
- Student's t-Test for Independent Samples
- Student's t-Test for Matched Pairs
- Oneway Analysis of Variance (Oneway ANOVA)

- Twoway Analysis of Variance (Twoway ANOVA)
- Correlation, Association, Regression, Likelihood, and Prediction

For each of these lessons, emphasis is placed on understanding the data, organizing and then working through the data, and subsequently understanding the outcomes of statistical analyses for each test by using many different and complementary approaches:

- The production of graphics is essential to understanding statistical outcomes, and each lesson provides many examples on how to produce beginning and eventually high-quality graphics associated with variables and the selected statistical test.
- Issues inherent to data distribution patterns are also emphasized in the lessons, where many datasets throughout the lessons are first analyzed by using a parametric approach to statistical analysis and analyses are then repeated by using a nonparametric approach. This approach toward analysis takes into account the complexities of real-world analyses in biostatistics, where data are not always as tidy as desired.
- Whenever possible, multiple R packages and multiple R functions are demonstrated, in an attempt to provide wide exposure to the many ways R can be used to gain the desired output.

A special feature in this text is the rich variety of bonus materials provided in the addenda after each lesson. Multiple approaches at statistical analysis, going beyond what was presented in the preceding lesson, are included in the addenda. Analyses that address parametric v nonparametric issues are further stressed in the addenda. Perhaps most importantly, the addenda often include additional datasets and guidance that provide the opportunity for incremental confidence-building practice activities with R. The complexity of R-based syntax is only gradually introduced as engagement with the text continues.

Using R for Biostatistics ends with a large and complex dataset and presentation of the many issues that need to be considered—an introduction to Big Data and breakout subsets of a large dataset. The ending parts of this text look at the future use of R for biostatistics, including a brief introduction to the increasing use of R Markdown.

All external datasets are available on this book's product Web page at Springer. Although multiple file formats are demonstrated in *Using R for Biostatistics*, most datasets are in comma-separated values (.csv) file format. By having access to the data in original format, it is possible to replicate outcomes by using the syntax presented in this text, but now as a self-guided practice activity where the outcomes are known.

Going beyond what is presented in this text, explore the many learning opportunities available to the R community. Join and review what is discussed in R-based discussion groups. View recorded R conference presentations made available through various media. Scan the long list (currently, more than 15,000) of R packages to see how R fits into the way biostatistics is approached by others. The opportunities to learn R syntax are many.

Fort Lauderdale, FL, USA
Fort Lauderdale, FL, USA
Summer 2020

Thomas W. MacFarland
Jan M. Yates

Acknowledgments

We want to thank the many individuals who believe in the open-source paradigm. This text is only possible because of the tireless and often unrecognized efforts of all who have contributed to R, core R, and the thousands of contributed R packages.

We also want to recognize our editor, Laura Aileen Briskman, and the entire Springer team. Thank you for your many ideas, feedback, help, and supporting our efforts.

Introduction

This text is focused on R, a freely available and open-source language that is among the leading languages used in biostatistics. R is typically dependent on user-generated syntax, not menu-driven point and click selections. However, the challenges of using R's syntax, whether working with an Integrated Development Environment (IDE), working interactively at the command line, or perhaps working offline in a separate text editor, are challenges that become a bridge too far for many.

The motivation for this text can be expressed in one simple word: frustration!

In many classes, and throughout the years, we have seen students and even beginning researchers experience frustration when using data science and information science curricular materials that, though excellent, are presented at a level beyond the capabilities of those who are learning the heuristics of a new programming language for the first time. Introductory texts need to focus on the incoming skills of learners through small confidence-building experiences and present incremental opportunities that build confidence and gradually develop proficiency among enthusiastic learners. Otherwise, frustration, not mastery, will be the main outcome for those learning the new language and its applications to biostatistics.

This text is structured to reduce frustration for learners who will use R to support the research process. To achieve this aim, this text starts with a brief introduction to biostatistics and how biostatistics developed into a distinct science, whether applied for agriculture, medicine, public health, or other subdisciplines. Then, in a set of consistent, structured lessons for many statistical tests the focus moves to a common framework for problem-solving using R. Each lesson is arranged as follows:

- Background
 - Description of the Data
 - Null Hypothesis (H_0)

- Import the Data into R
- Organize the Data and Display the Code Book
- Conduct a Visual Data Check Using Graphics
- Descriptive Statistics for Initial Analysis of the Data
- Quality Assurance, Data Distribution, and Tests for Normality
- Statistical Test(s)
- Summary of Outcomes

This consistent, step-by-step presentation provides a structured and organized introduction to R that supports the following:

- The researcher knows as much about the data as can be reasonably expected.
- The data are well-organized and a descriptive Code Book not only supports a thorough understanding of the data but also aids replication of all analyses, either at a future date or by others.
- Visual presentations improve immediate cognition of trends among the data and an understanding of these trends by others, especially audiences who may not be experienced in biostatistics.
- Descriptive statistics and measures of central tendency further improve understanding of the data.
- Quality assurance is promoted by carefully addressing data distribution patterns, which validates whether parametric, nonparametric, or both approaches should be used for later analyses.
- Statistical tests are conducted, often using multiple approaches in an effort to gain consistency of outcomes.
- The summary for each lesson is prepared, so that beginning biostatisticians can understand not only the initial outcomes but also the potential practical applications of outcomes.

These many precursor activities are often given only marginal attention by those in a rush to complete and then present research results. It is our view, however, that these many activities, although demanding, support appropriate statistical test selection, implementation, and presentation of outcomes in support of the biostatistics research process.

Most lessons in this text are enhanced by addenda, with new skills added to each advancing lesson, which are designed as value-added learning opportunities. The addenda often introduce and/or reinforce specialized packages and functions that go beyond what was previously presented, address parametric

v nonparametric approaches toward the data, and often end with additional practice datasets that support incremental practice with advanced skills.

Each external dataset has been placed at the publisher's Web site for this text, which makes it possible for the learner to practice with the syntax presented in the text and to see if self-generated outcomes match the known correct output.

Contents

1	Biostatistics and R	1
1.1	Purpose of This Text	2
1.2	Development of Biostatistics	3
1.3	Development of R	6
1.4	How R is Used in This Text	7
1.5	Import Data Into R	10
1.5.1	Import a .csv File of Comma-Separated Values into R	13
1.5.2	Import a .txt File of Tab-Separated Values into R	16
1.5.3	Import a .txt File of Fixed-Width Format Values into R	18
1.5.4	Import a .xlsx Spreadsheet File into R	24
1.5.5	Import a .csv File of Comma-Separated Values from an Online Source into R	27
1.5.6	Import a .csv File of Comma-Separated Values into R by Using Graphical User Interface (GUI) Selections	35
1.5.7	Import by Direct Keyboard <i>On the Fly</i> Data Entry into R	38
1.6	Addendum 1: Efficient Programming with R, Project Workflow, and Good Programming Practices (gpp)	41
1.7	Addendum 2: Preview of Descriptive Statistics and Graphics Using R	43
1.8	Addendum 3: R and <i>Beautiful Graphics</i>	46
1.9	Addendum 4: Research Designs Used in Biostatistics	51
1.9.1	Case Study and Clinical Trial	52
1.9.2	Pretest–Posttest for One Group	52
1.9.3	Pretest–Posttest for Control Group	52
1.9.4	Posttest Only for Control Group	53
1.9.5	Fixed Group Comparative Analysis of a Single Factor	53
1.9.6	Factorial Data Organization of Multiple Independent Variables	53

1.9.6.1	Goodness of Fit (e.g., Chi-Square)	53
1.9.6.2	Comparison of Group Means (e.g., Analysis of Variance)	54
1.9.7	Correlation, Association, Regression, Likelihood, and Prediction	54
1.10	Prepare to Exit, Save, and Later Retrieve This R Session	55
1.11	External Data and/or Data Resources Used in This Lesson	55
2	Data Exploration, Descriptive Statistics, and Measures of Central Tendency	57
2.1	Background	58
2.1.1	Description of the Data	58
2.1.2	Null Hypothesis	61
2.2	Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions	61
2.3	Organize the Data and Display the Code Book	63
2.4	Conduct a Visual Data Check Using Graphics (e.g., Figures)	65
2.5	Descriptive Statistics for Initial Analysis of the Data	70
2.6	Quality Assurance, Data Distribution, and Tests for Normality	80
2.7	Statistical Test(s)	86
2.8	Summary	87
2.9	Addendum 1: Specialized External Packages and Functions	88
2.10	Addendum 2: Parametric v Nonparametric	96
2.11	Addendum 3: Additional Practice Datasets for Data with Normal Distribution Patterns and Data That Do Not Exhibit Normal Distribution Patterns	97
2.11.1	Purpose of This Addendum	97
2.11.2	Background	97
2.11.3	Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions	98
2.11.4	Organize the Data and Display the Code Book	99
2.11.5	Conduct a Visual Data Check Using Graphics (e.g., Figures)	101
2.11.6	Descriptive Statistics for Initial Analysis of the Data	104
2.11.7	Quality Assurance, Data Distribution, and Tests for Normality	112
2.11.8	Statistical Test(s)	115
2.11.9	Summary of Outcomes for <code>SBPNormal</code> and <code>SBPNotNormal</code>	115
2.11.10	Additional Bonus Materials	116

2.12 Prepare to Exit, Save, and Later Retrieve This R Session	139
2.13 External Data and/or Data Resources Used in This Lesson	139
3 Student's t-Test for Independent Samples	141
3.1 Background	142
3.1.1 Description of the Data	142
3.1.2 Null Hypothesis	143
3.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions	144
3.3 Organize the Data and Display the Code Book	146
3.4 Conduct a Visual Data Check Using Graphics (e.g., Figures) . .	150
3.5 Descriptive Statistics for Initial Analysis of the Data	167
3.6 Quality Assurance, Data Distribution, and Tests for Normality	175
3.7 Statistical Test(s)	186
3.8 Summary of Outcomes	193
3.9 Addendum 1: t-Statistic v z-Statistic	196
3.9.1 Create an Enumerated Dataset	197
3.9.2 Calculate the t-Statistic	199
3.9.3 Calculate the z-Statistic	199
3.10 Addendum 2: Parametric v Nonparametric	200
3.11 Addendum 3: Additional Practice Datasets for Data with Normal Distribution Patterns and Data That Do Not Exhibit Normal Distribution Patterns	201
3.11.1 Data with Normal Distribution Patterns	202
3.11.2 Data That Do Not Exhibit Normal Distribution Patterns	227
3.12 Prepare to Exit, Save, and Later Retrieve This R Session	239
3.13 External Data and/or Data Resources Used in This Lesson	240
4 Student's t-Test for Matched Pairs	241
4.1 Background	242
4.1.1 Description of the Data	242
4.1.2 Null Hypothesis	244
4.1.3 Unstacked (e.g., Wide) Data and Stacked (e.g., Long) Data	244
4.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions	246
4.3 Organize the Data and Display the Code Book	248
4.4 Conduct a Visual Data Check Using Graphics (e.g., Figures) . .	252
4.5 Descriptive Statistics for Initial Analysis of the Data	256
4.6 Quality Assurance, Data Distribution, and Tests for Normality	262

4.7	Statistical Test(s)	267
4.8	Summary of Outcomes	271
4.9	Addendum 1: R-Based Tools for Unstacked (e.g., Wide) Data	272
4.10	Addendum 2: Stacked Data and Student's t-Test for Matched Pairs	275
4.11	Addendum 3: The Impact of N on Student's t-Test	279
4.12	Addendum 4: Parametric v Nonparametric	282
4.13	Addendum 5: Additional Practice Datasets for Data with Normal Distribution Patterns and Data That Do Not Exhibit Normal Distribution Patterns	285
4.14	Prepare to Exit, Save, and Later Retrieve This R Session	290
4.15	External Data and/or Data Resources Used in This Lesson	291
5	Oneway Analysis of Variance (ANOVA)	293
5.1	Background	294
5.1.1	Description of the Data	294
5.1.2	Null Hypothesis	295
5.2	Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions	295
5.3	Organize the Data and Display the Code Book	298
5.4	Conduct a Visual Data Check Using Graphics (e.g., Figures)	303
5.5	Descriptive Statistics for Initial Analysis of the Data	310
5.6	Quality Assurance, Data Distribution, and Tests for Normality	314
5.7	Statistical Test(s)	320
5.7.1	Exploratory Oneway ANOVA	334
5.7.2	Oneway ANOVA Method 1: lm() and anova() functions	335
5.7.3	Oneway ANOVA Method 2: aov() and TukeyHSD() Functions	336
5.8	Summary of Outcomes	340
5.9	Addendum 1: Other Packages for Display of Oneway ANOVA	346
5.10	Addendum 2: Parametric v Nonparametric	349
5.10.1	Parametric Approach to Oneway ANOVA	349
5.10.2	Nonparametric Alternative to Oneway ANOVA	349
5.11	Addendum 3: Additional Practice Datasets	352
5.11.1	Data with Normal Distribution Patterns	354
5.11.2	Data That Do Not Exhibit Normal Distribution Patterns	357
5.12	Prepare to Exit, Save, and Later Retrieve This R Session	358
5.13	External Data and/or Data Resources Used in This Lesson	359

6 Twoway Analysis of Variance (ANOVA)	361
6.1 Background	362
6.1.1 Description of the Data	363
6.1.2 Null Hypothesis	363
6.2 Import Data in Comma-Separated Values	364
6.3 Organize the Data and Display the Code Book	370
6.4 Conduct a Visual Data Check Using Graphics	371
6.5 Descriptive Statistics for Initial Analysis	383
6.6 Quality Assurance, Data Distribution, and Tests	387
6.7 Statistical Test(s)	394
6.8 Summary of Outcomes	405
6.9 Addendum 1: Other Packages for Display of Twoway ANOVA	408
6.10 Addendum 2: Parametric v Nonparametric	409
6.11 Addendum 3: Additional Practice Datasets	412
6.11.1 Data with Normal Distribution Patterns	413
6.11.2 Data That Do Not Exhibit Normal Distribution Patterns	418
6.12 Prepare to Exit, Save, and Later Retrieve	425
6.13 External Data and/or Data Resources Used in this Lesson	426
7 Correlation, Association, Regression, Likelihood, and Prediction	427
7.1 Background	428
7.1.1 Description of the Data	429
7.1.2 Null Hypothesis (H_0)	431
7.2 Import Data in Comma-Separated Values (.csv)	431
7.3 Organize the Data and Display the Code Book	435
7.3.1 Conduct a Visual Data Check Using Graphics (e.g., Figures)	443
7.3.2 Descriptive Statistics for Initial Analysis of the Data	449
7.4 Quality Assurance, Data Distribution	465
7.5 Statistical Test(s)	488
7.5.1 Correlation Using Pearson's r and Spearman's rho	489
7.5.2 Linear Regression Using a Single Predictor Variable	504
7.5.3 Linear Regression Using Multiple Predictor Variables	509
7.5.4 Ordinal Logistic Regression	511
7.5.5 Binary Logistic Regression	523
7.6 Summary of Outcomes	529
7.7 Addendum 1: Multiple Regression	532
7.7.1 Hand-Calculate Multiple Regression	534

7.7.2	Minimal Adequate Model (MAM) for Regression	537
7.7.3	Stepwise Regression	541
7.8	Addendum 2: Likelihood and Odds Ratio	546
7.9	Addendum 3: Parametric v Nonparametric	560
7.10	Addendum 4: Additional Practice Datasets	562
7.10.1	Data with Normal Distribution Patterns	569
7.10.2	Data That Do Not Exhibit Normal Distribution Patterns	574
7.11	Prepare to Exit, Save, and Later Retrieve	583
7.12	External Data and/or Data Resources Used in	584
8	Working with Large and Complex Datasets	585
8.1	Background	586
8.1.1	Description of the Data	586
8.1.2	First Null Hypothesis (H_0): Mean Comparisons by Breakout Groups	589
8.1.3	Second Null Hypothesis (H_0): Test of Association	589
8.2	Import Data in Comma-Separated Values (.csv)	589
8.3	Organize the Data and Display the Code Book	591
8.4	Conduct a Visual Data Check Using Graphics	606
8.5	Descriptive Statistics for Initial Analysis	625
8.5.1	Analyses of Object Variables in Original Format	630
8.5.2	Analyses of Boolean-Based Breakouts of Object Variables	632
8.5.2.1	Structure the Boolean-Based Data Selection Process	632
8.5.2.2	Create a New Dataset from an Existing Object Variable	632
8.5.2.3	Download the New Dataset	635
8.6	Quality Assurance, Data Distribution, and	650
8.6.1	Graphics for Normality	650
8.6.1.1	Histogram	651
8.6.1.2	Density Plot	652
8.6.1.3	Quantile-Quantile (Q-Q) Plot	652
8.6.2	Tests for Normality	654
8.6.2.1	Anderson–Darling Test for Normality	654
8.6.2.2	Jarque–Bera Test for Normality	655
8.6.2.3	Lilliefors (Kolmogorov–Smirnov) Test for Normality Null Hypothesis	656
8.6.2.4	Shapiro–Wilk Test for Normality	657

8.7	Statistical Test(s)	666
8.7.1	Null Hypothesis 1: Analysis of Variance	667
8.7.1.1	Parametric Oneway ANOVA and Tukey HSD Approach	667
8.7.1.2	Parametric Oneway ANOVA and Tukey HSD Approach	681
8.7.1.3	Nonparametric Kruskal–Wallis Approach	685
8.7.2	Null Hypothesis 2: Correlation—Association	690
8.8	Summary of Outcomes	724
8.8.1	Outcomes for First Null Hypothesis (H_0): Mean Comparisons by Breakout Groups	724
8.8.2	Outcomes for Second Null Hypothesis (H_0): Test of Association	725
8.9	Addendum 1: Additional Graphics, to Show Relationships Between and Among Data	725
8.9.1	Graphical Presentation of Grouped (e.g., Factor-Type) Data	726
8.9.1.1	Association Plot	726
8.9.1.2	Bar Plot	728
8.9.1.3	Mosaic Plot	729
8.9.1.4	Pie Chart	731
8.9.1.5	Waffle Chart (e.g., Squared Pie Chart)	735
8.9.2	Graphical Presentation of Interval and Other (e.g., Measured) Numeric Data	740
8.9.2.1	Bagplot (e.g., Bivariate Boxplot)	740
8.9.2.2	Beanplot	741
8.9.2.3	Beeswarm Plot	743
8.9.2.4	Boxplot (e.g., Box-Plot, Box-and-Whiskers Diagram, Box-and-Whiskers Plot)	747
8.9.2.5	Box-Percentile Plot	757
8.9.2.6	Density Plot	760
8.9.2.7	Dotplot (e.g., Dotchart)	760
8.9.2.8	Engelmann–Hecker (EH) Plot	762
8.9.2.9	Histogram	766
8.9.2.10	Line Chart (e.g., Line Graph)	772
8.9.2.11	Pirate Plot	778
8.9.2.12	Quantile-Quantile (Q-Q) Plot	779
8.9.2.13	Scatter Plot (e.g., Scatterplot, Scatter Diagram)	783
8.9.2.14	Color Gradient Plot	787
8.9.2.15	Correlogram	790
8.9.2.16	Hexbin Plot	792
8.9.2.17	Scatter Plot Matrix	794
8.9.2.18	Sunflower Scatterplot	796

8.9.2.19	Stem-and-Leaf Plot	797
8.9.2.20	Stripchart	799
8.9.2.21	Violin Plot	800
8.9.3	Specialized Graphics	801
8.9.3.1	Density Ridge	801
8.9.3.2	Gantt Chart	803
8.9.3.3	Interaction Plot	805
8.9.3.4	Staircase Plot	807
8.9.3.5	Triangular Plot for 3-D Representation	808
8.10	Addendum 2: Graphics Using the lattice Package	810
8.11	Addendum 3: Graphics Using the ggplot2 Package	817
8.12	Addendum 4: Beyond an Introduction to R—Use the tidyverse to Create Subsets of Original Datasets	870
8.12.1	Use the dplyr::filter() Function to Subset a Large Dataset	871
8.12.2	Use the magrittr Package Pipe-Like Operator	877
8.13	Prepare to Exit, Save, and Later Retrieve This R Session	881
8.14	External Data and/or Data Resources Used in This Lesson	881
9	Future Actions and Next Steps	883
9.1	Use of This Text	883
9.2	R and <i>Beautiful Reporting</i> with R Markdown	884
9.2.1	Static Reports (e.g., Documents)	885
9.2.2	Dynamic Output (e.g., Presentations)	886
9.3	Future Use of R for Biostatistics	887
9.4	Big Data and BioInformatics	888
9.5	External Resources	888
9.6	Contact the Authors	889
Index		891

List of Figures

1.1	Quality assurance of endurance	15
1.2	Quality assurance of MilkLb365	17
1.3	Quality assurance of corn yield	21
1.4	Quality assurance of waist	25
1.5	Quality assurance of sorghum yield	27
1.6	Quality assurance of fruit consumption by grade 9–12 students	32
1.7	Quality assurance of highway MPG	35
1.8	Quality assurance of rabbit weights	38
1.9	Quality assurance of systolic blood pressure for adult males . . .	40
1.10	Multiple histograms of birth weight	46
1.11	ggplot2 demonstration 1—simple to complex	49
1.12	ggplot2 demonstration 2—complex	50
2.1	Multiple visualization of weight	66
2.2	Bar plots of section and gender	67
2.3	Multiple visualizations of weight by section and gender	69
2.4	Weight Q-Q plot breakouts by section and gender	85
2.5	Section and gender: frequency distribution overall	90
2.6	Section and gender: frequency distribution breakouts	90
2.7	Weight by section and gender breakouts	92
2.8	Graphical representation of weight by gender	93
2.9	Graphical representation of weight by section	94
2.10	Multiple standard deviations with the same mean	119
2.11	Corn yield by year	123
2.12	Waffle chart of race-ethnicity	131
2.13	Bar plot of race-ethnicity	134
2.14	Dot chart of SBP by race-ethnicity and gender breakouts . . .	138
3.1	Distribution of breed by count—1	151
3.2	Distribution of percent butterfat—1	152

3.3	Distribution of percent protein—1	153
3.4	Distribution of breed by count—2	155
3.5	Distribution of percent butterfat—2	159
3.6	Distribution of percent butterfat—3	160
3.7	Distribution of percent butterfat—4	162
3.8	Distribution of percent protein—2	163
3.9	Distribution of percent protein—3	165
3.10	Distribution of percent protein—4	166
3.11	Percent butterfat by breed and percent protein by breed—1	170
3.12	Breakouts of breed by count	175
3.13	Distribution of percent butterfat—5	180
3.14	Distribution of percent protein—5	183
3.15	Percent butterfat and percent protein overall and by breed	186
3.16	Percent butterfat by breed and percent protein by breed—2	196
3.17	Distribution of systolic blood pressure by gender	212
3.18	Distribution of diastolic blood pressure by age breakouts	233
4.1	Comparison of pretest weights to posttest weights using unstacked data—1	254
4.2	Comparison of pretest weights to posttest weights using unstacked data—2	255
4.3	Distribution of pretest weights and posttest weights using unstacked data—1	256
4.4	Distribution of pretest weights and posttest weights using unstacked data—2	262
4.5	Distribution of pretest weights and posttest weights using unstacked data—3	267
4.6	Distribution of datapoints by groups using stacked data	274
4.7	Comparison of parametric pre and parametric post datapoints	288
5.1	Distribution of systolic blood pressure by lifestyle breakouts—1	302
5.2	Distribution of systolic blood pressure by lifestyle breakouts—2	304
5.3	Distribution of systolic blood pressure by lifestyle breakouts—3	305
5.4	Distribution of systolic blood pressure by lifestyle breakouts—4	307
5.5	Distribution of systolic blood pressure by lifestyle breakouts—5	308

5.6	Distribution of systolic blood pressure by lifestyle breakouts—6	309
5.7	Distribution of systolic blood pressure by lifestyle breakouts—7	310
5.8	Distribution of systolic blood pressure by lifestyle breakouts—8	320
5.9	Distribution of systolic blood pressure	328
5.10	Distribution of systolic blood pressure by lifestyle breakouts—9	328
5.11	Distribution of systolic blood pressure by lifestyle breakouts—10	329
5.12	Distribution of systolic blood pressure by lifestyle breakouts—11	330
5.13	Distribution of systolic blood pressure by lifestyle breakouts—12	331
5.14	Distribution of systolic blood pressure by lifestyle breakouts—13	332
5.15	Distribution of systolic blood pressure by lifestyle breakouts—14	333
5.16	Distribution of systolic blood pressure by lifestyle breakouts—15	333
5.17	Distribution of systolic blood pressure by lifestyle breakouts—16	334
5.18	Distribution of systolic blood pressure by lifestyle breakouts—17	340
5.19	Systolic blood pressure comparative family-wise confidence levels	342
5.20	Distribution of systolic blood pressure by lifestyle breakouts—18	343
5.21	Distribution of systolic blood pressure by lifestyle breakouts—19	343
5.22	Distribution of systolic blood pressure by lifestyle breakouts—20	344
5.23	Distribution of systolic blood pressure by lifestyle breakouts—21	345
5.24	Distribution of systolic blood pressure by lifestyle breakouts—22	345
5.25	Differences of systolic blood pressure means by lifestyle breakouts (Tukey—alpha = 0.05)	349
6.1	Distribution of systolic blood pressure overall	372
6.2	Distribution of systolic blood pressure overall and by breakouts—1	376
6.3	Distribution of systolic blood pressure overall and by breakouts—2	378

6.4	Distribution of systolic blood pressure overall and by breakouts—3	379
6.5	Distribution of systolic blood pressure overall and by breakouts—4	380
6.6	Factorial representation of systolic blood pressure overall and by breakouts	383
6.7	Distribution of systolic blood pressure overall and by breakouts—5	385
6.8	Distribution of systolic blood pressure overall and by breakouts—6	393
6.9	Distribution of systolic blood pressure by breakouts	403
6.10	Interaction of gender and drug for systolic blood pressure—1	403
6.11	Interaction of gender and drug for systolic blood pressure—2	405
6.12	Interaction of gender and drug for systolic blood pressure—3	405
6.13	Corn yield by county breakouts	422
6.14	Corn yield by management breakouts	423
7.1	Distribution of factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity	445
7.2	Distribution of numeric-type object variable age	446
7.3	Distribution of numeric-type object variable cholesterol	447
7.4	Distribution of numeric-type object variable systolic blood pressure	448
7.5	Distribution of numeric-type object variable diastolic blood pressure	448
7.6	Distribution of numeric-type object variable body mass index	449
7.7	Distribution of age by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity	460
7.8	Distribution of cholesterol by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity	461
7.9	Distribution of systolic blood pressure by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity	462
7.10	Distribution of diastolic blood pressure by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity	463
7.11	Distribution of body mass index by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity	464

7.12	Normality of numeric-type object variables AgeYears, TotalCholesterolmgdL, SBPmmHg, DBPmmHg, and BMIMetric	469
7.13	Normality of age by factor-type object variables gender, RaceEthnicity, BMISstatus, and obesity	480
7.14	Normality of cholesterol by factor-type object variables gender, RaceEthnicity, BMISstatus, and obesity	481
7.15	Normality of systolic blood pressure by factor-type object variables gender, RaceEthnicity, BMISstatus, and obesity	482
7.16	Normality of diastolic blood pressure by factor-type object variables gender, RaceEthnicity, BMISstatus, and obesity	483
7.17	Normality of body mass index by factor-type object variables gender, RaceEthnicity, BMISstatus, and obesity . . .	485
7.18	Normality of body mass index by obesity and accommodation for missing data	487
7.19	Brute force one-by-one display of correlation: SBP by Age, SBP by cholesterol, SBP by DBP, and SBP by BMI . .	498
7.20	Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Pearson’s r	499
7.21	Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Spearman’s rho	500
7.22	Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Pearson’s r and Spearman’s rho—1	502
7.23	Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Pearson’s r and Spearman’s rho—2	503
7.24	Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI	504
7.25	Mean systolic blood pressure by body mass index breakout groups and mean diastolic blood pressure by body mass index breakout groups	513
7.26	Probability of body mass index breakout group assignment by systolic blood pressure	521
7.27	Probability of body mass index breakout group assignment by diastolic blood pressure	522
7.28	Mean systolic blood pressure by obesity breakout groups and mean diastolic blood pressure by obesity breakout groups	525
7.29	Probability of obesity by systolic blood pressure and provability of obesity by diastolic blood pressure	529

7.30	Scatterplot of two variables (vigor at purchase by vigor at 100 pounds) with added regression line	537
7.31	Percentage distribution of coin tosses	549
7.32	Display of odds ratio—1	553
7.33	Display of odds ratio—2	556
7.34	Display of odds ratio—3	559
8.1	Quality assurance exploratory data analysis—1	607
8.2	Quality assurance exploratory data analysis—2	608
8.3	Quality assurance exploratory data analysis—3	608
8.4	Quality assurance exploratory data analysis—4	609
8.5	Quality assurance exploratory data analysis—5	609
8.6	Quality assurance exploratory data analysis—6	610
8.7	Quality assurance exploratory data analysis—7	611
8.8	Quality assurance exploratory data analysis—8	611
8.9	Quality assurance exploratory data analysis—9	612
8.10	Quality assurance exploratory data analysis—10	612
8.11	Quality assurance exploratory data analysis—11	612
8.12	Quality assurance exploratory data analysis—12	613
8.13	Quality assurance exploratory data analysis—13	613
8.14	Quality assurance exploratory data analysis—14	614
8.15	Quality assurance exploratory data analysis—15	615
8.16	Quality assurance exploratory data analysis—16	615
8.17	Quality assurance exploratory data analysis—17	616
8.18	Quality assurance exploratory data analysis—18	617
8.19	Quality assurance exploratory data analysis—19	617
8.20	Quality assurance exploratory data analysis—20	618
8.21	Quality assurance exploratory data analysis—21	618
8.22	Quality assurance exploratory data analysis—22	619
8.23	Quality assurance exploratory data analysis—23	619
8.24	Quality assurance exploratory data analysis—24	620
8.25	Quality assurance exploratory data analysis—25	621
8.26	Quality assurance exploratory data analysis—26	621
8.27	Quality assurance exploratory data analysis—27	622
8.28	Quality assurance exploratory data analysis—28	624
8.29	Quality assurance review of original dataset and new dataset using systolic blood pressure and age	635
8.30	Quality assurance review of new dataset using race and age	641
8.31	Quality assurance review of new dataset using race, sex, and age	643
8.32	Distribution (histogram) of two variables: XNormal and XNotNormal	651
8.33	Distribution (density plot) of two variables: XNormal and XNotNormal	652

8.34	Distribution (QQ plot) of two variables: XNormal and XNotNormal	653
8.35	Distribution pattern of systolic blood pressure with multiple subsets	662
8.36	Distribution pattern of diastolic blood pressure and age with multiple subsets	669
8.37	Distribution of diastolic blood pressure (age 040–049)—1	672
8.38	Distribution of diastolic blood pressure (age 040–049)—2	673
8.39	Distribution of diastolic blood pressure (age 040–049)—3	673
8.40	Distribution of diastolic blood pressure (age 040–049)—4	677
8.41	Distribution of diastolic blood pressure (age 040–049) and race—1	681
8.42	Distribution of diastolic blood pressure (age 040–049) and race—2	685
8.43	Distribution of systolic blood pressure by increasingly restrictive subsets—1	695
8.44	Distribution of weight by increasingly restrictive subsets—1	696
8.45	Distribution of systolic blood pressure by increasingly restrictive subsets—2	699
8.46	Distribution of systolic blood pressure by increasingly restrictive subsets—3	700
8.47	Distribution of systolic blood pressure by increasingly restrictive subsets—4	701
8.48	Distribution of weight by increasingly restrictive subsets—2	702
8.49	Distribution of weight by increasingly restrictive subsets—3	703
8.50	Distribution of weight by increasingly restrictive subsets—4	704
8.51	Distribution of systolic blood pressure from multiple perspectives—1	706
8.52	Normality of systolic blood pressure for selected subjects—1	707
8.53	Distribution of weight from multiple perspectives—1	709
8.54	Normality of weight for selected subjects—1	709
8.55	Side-by-side QQ plot of systolic blood pressure and weight	713
8.56	Correlation of systolic blood pressure and weight—1	719
8.57	Correlation of systolic blood pressure and weight—2	720
8.58	Correlation of systolic blood pressure and weight—3	721
8.59	Correlation of systolic blood pressure and weight—4	722
8.60	Association plot	727
8.61	Stacked bar plot and side-by-Sie bar plot	729

8.62	Mosaic plot—1	730
8.63	Mosaic plot—2	731
8.64	Pie chart—1	733
8.65	Pie chart—2	734
8.66	Pie chart—3	735
8.67	Pie chart—4	736
8.68	Waffle chart—1	738
8.69	Waffle chart—2	739
8.70	Bagplot	741
8.71	Beanplot—1	742
8.72	Beanplot—2	743
8.73	Beeswarm plot—1	745
8.74	Beeswarm plot—2	746
8.75	Boxplot—1	749
8.76	Boxplot—2	749
8.77	Boxplot—3	752
8.78	Boxplot—4	752
8.79	Beanplot and boxplot—1	754
8.80	Beanplot and boxplot—2	755
8.81	Boxplot—5	756
8.82	Boxplot—6	756
8.83	Box-percentile plot—1	758
8.84	Box-percentile plot—2	759
8.85	Box-percentile plot—3	760
8.86	Density plot	761
8.87	Dot plot—1	762
8.88	Dot plot—2	762
8.89	Engelmann–Hecker plot and boxplot—1	763
8.90	Engelmann–Hecker plot and boxplot—2	764
8.91	Engelmann–Hecker plot and boxplot—3	765
8.92	Engelmann–Hecker plot and boxplot—4	765
8.93	Histogram—1	766
8.94	Histogram—2	767
8.95	Histogram—3	768
8.96	Histogram—4	769
8.97	Histogram—5	770
8.98	Histogram—6	771
8.99	Histogram—7	772
8.100	Histogram—8	773
8.101	Line chart—1	775
8.102	Line chart—2	778
8.103	Pirate plot—1	779
8.104	Pirate plot—2	780
8.105	Quantile–quantile plot—1	782

8.106 Quantile–quantile plot—2	783
8.107 Scatter plot—1	785
8.108 Scatter plot—2	787
8.109 Scatter plot—3	788
8.110 Color gradient plot—1	789
8.111 Color gradient plot—2	790
8.112 Color gradient plot—3	791
8.113 Color gradient plot—4	791
8.114 Correlogram	793
8.115 Hexbin plot—1	794
8.116 Hexbin plot—2	794
8.117 Scatter plot matrix—1	795
8.118 Scatter plot matrix—2	796
8.119 Sunflower scatterplot	797
8.120 Stripchart	800
8.121 Violin plot—1	801
8.122 Violin plot—2	801
8.123 Density ridge—1	802
8.124 Density ridge—2	803
8.125 Density ridge—3	803
8.126 Density ridge—4	804
8.127 Gantt chart	805
8.128 Interaction plot—1	806
8.129 Interaction plot—2	807
8.130 Staircase plot	809
8.131 Triangular plot	810
8.132 lattice package—barchart	811
8.133 lattice package—box plot and density plot	813
8.134 lattice package—histogram and QQ plot	815
8.135 lattice package—scatter plot, scatter plot matrix (SPLOM), and 3D scatter plot	817
8.136 ggplot2 package—barchart 1	818
8.137 ggplot2 package—barchart 2	820
8.138 ggplot2 package—barchart 3	822
8.139 ggplot2 package—barchart 4	823
8.140 ggplot2 package—box plot 1	824
8.141 ggplot2 package—box plot 2	825
8.142 ggplot2 package—box plot 3	826
8.143 ggplot2 package—density plot 1	826
8.144 ggplot2 package—density plot 2	827
8.145 ggplot2 package—density plot 3	828
8.146 ggplot2 package—density plot 4	829
8.147 ggplot2 package—dot plot 1	830
8.148 ggplot2 package—dot plot 2	831

8.149	ggplot2 package—dot plot 3	832
8.150	ggplot2 package—dot plot 4	833
8.151	ggplot2 package—scatter plot 1	836
8.152	ggplot2 package—scatter plot 2	840
8.153	ggplot2 package and GGally package—scatter plot 1	842
8.154	ggplot2 package and GGally package—scatter plot 2	843
8.155	ggplot2 package and GGally package—scatter plot 3	844
8.156	ggplot2 package—violin plot 1	844
8.157	ggplot2 package—violin plot 2	845
8.158	ggplot2 package—violin plot 3	846
8.159	ggplot2 package—violin plot 4	847
8.160	ggplot2 package—violin plot 5	848
8.161	ggplot2 package—histogram 1	849
8.162	ggplot2 package—histogram 2	851
8.163	ggplot2 package—histogram 3	852
8.164	ggplot2 package—histogram 4	853
8.165	ggplot2 package—histogram 5	854
8.166	ggplot2 package—frequency polygon	857
8.167	ggplot2 package—stripchart	858
8.168	ggplot2 package—QQ plot 1	860
8.169	ggplot2 package—QQ plot 2	862
8.170	ggplot2 package—QQ plot 3	863
8.171	ggplot2 package—interaction plot	865
8.172	ggplot2 package—line chart	868
8.173	ggplot2 package—tile map	870
8.174	Multiple subsets of systolic blood pressure	877
8.175	Comparison of selected variables—original dataset and dataset after multiple subsets	880

List of Tables

3.1	Percent butterfat by breed	174
3.2	Percent protein by breed	175
5.1	Oneway ANOVA mean comparison tests	338



Chapter 1

Biostatistics and R

Abstract

The purpose of this lesson is to provide context for the science of biostatistics and to highlight a few of the major contributors. Emphasis is given to the role of data analysis for the various disciplines in the biological sciences (e.g., agriculture (animal science, crop science, and soil science), allied health, aquaculture, biology, clinical trials, dentistry, ecology, environmental studies, epidemiology, food production and technology, genetics, health sciences, medicine (allopathic and osteopathic), nursing, nutrition, oceanography, optometry, pharmacy, public health, etc.). The practice of biostatistics is then linked to the use of R, a free and open source software environment. As explained in this introductory lesson, each problem in this text is usually associated with: (1) background of the data, including a description of the data and a stated Null Hypothesis (H_0), (2) a .csv (comma-separated values) dataset, imported into R, (3) a Code Book detailing data organization, (4) visual data checks through the use of simple graphics (e.g., figures), (5) descriptive statistics of factor-type object variables (e.g., frequency distributions) and numeric-type object variables (e.g., mean, sd, median, mode, etc.),

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_1) contains supplementary material, which is available to authorized users.

(6) quality assurance processes including the use of tests of data distribution and tests for normality, (7) inferential statistical tests, including parametric tests and nonparametric tests, (8) a summary of outcomes (e.g., interpretation of output), (9) multiple addenda addressing additional examples for different types of data and additional functions, practice datasets, and/or bonus materials, and (10) procedures for a *graceful exit* and save process, to allow later retrieval of the R session.

Keywords: Agriculture (animal science, crop science, and soil science), Allied health, Aquaculture, Biology, Biostatistics, Census, Clinical trials, Code Book, Comma-separated values (.csv) text file, Command line interface (CLI), Comprehensive R archive network (CRAN), CRAN-contributed packages, Data analysis, Dentistry, Descriptive statistics, Ecology, Environmental studies, Epidemiology, Fixed-width format text file, Food production and technology, Genetics, Graphical user interface (GUI), Health sciences, Integrated development environment (IDE), Medicine (allopathic and osteopathic), Normal distribution, Nursing, Nutrition, Oceanography, Open source software, Optometry, Pharmacy, Public health, Quality assurance, R, S, Scheme, and Tab-separated values text file

1.1 Purpose of This Text

Scientists use empiricism to guide and validate decisions.¹ Precision, orderliness, analysis, and a sound background in statistics are directly associated with informed judgment, decision-making, and the subsequent allocation of human, physical, and fiscal resources—all to improve the human condition. The purpose of this text is to provide an introduction to the use of R software as a platform for problems related to biostatistics. Data identification and organization, descriptive portrayal of phenomena and statistical tests, and graphical presentation through the production of figures—all by using R—are found throughout this text.

R is typically used in one of three different means of operation: Command Line Interface (CLI), Graphical User Interface (GUI), and Integrated Development Environment (IDE). How R-based syntax is prepared is often a matter of personal choice from among these three options.

¹Review the writings of Francis Bacon, George Berkeley, Rene Descartes, Thomas Hobbes, David Hume, Gottfried Wilhelm Leibniz, John Locke, and similar luminaries associated with the 1600s and 1700s *Age of Enlightenment* and how attention to the senses and subsequently measurement of the same influenced thinking about ideas as simple as *cause and effect* and ultimately our more modern ideas about science.

- **CLI**—As stressed throughout this text, R supports a robust and useful syntax-based Command Line Interface (CLI) approach. This text is focused on the use of R-based syntax, working at the command line, to address data organization, statistical analyses, and graphical presentations as they relate to biostatistics.²
- **GUI**—It is possible to download external R packages that support the use of R at a Graphical User Interface (GUI). The use of point-and-click menu selection is at first easy and has value as an early learning activity. Eventually, however, the use of only menu selections becomes limited and puts restrictions on the full potential of R.
- **IDE**—It is also possible to download an Integrated Development Environment (IDE) specifically designed for the use of R. Typical to most IDEs, the IDE software *sits on top of R* and the screen is divided into multiple adjustable panes showing different actions, including an editor that links to syntax execution, the production and display of output and graphics, debugging aids, and file management. Most IDEs, as useful as they may be at first, are often seen as *middleware* which, similar to a GUI, limits and/or gets in the way of desired actions for those who know R syntax.

A series of small confidence-building activities is presented at the beginning of this text. More detail is gradually introduced as the text progresses, where the final lessons on R-based analyses and figures are quite detailed. For both the beginning small datasets and the ending larger datasets, all examples are for biostatistics and these datasets can be easily applied to all areas of biostatistics, regardless of major area of study.

1.2 Development of Biostatistics

The term *statistics* is derived from *status*, the Latin term for state. Thus, the science and practice of statistics, as we think of it today, was first associated with data relating to the state (e.g., government), such as census counts and health records. Given the importance of statistics as a part of state governance, there are more than a few accounts of census-taking and health records from the earliest days of recorded history.

Going beyond mere record-keeping, an interest in the mathematics of chance (e.g., probability) began to develop in the 1500s and 1600s, especially among

²It is certainly possible to interactively type all R-based syntax at the R prompt, but this approach is generally discouraged. Instead, it is far more efficient to use a separate text editor to prepare R syntax and to then determine the best way to put syntax into the active R session. A simple search of the Internet will provide a comprehensive list of recommended text editors (GNU Emacs and vim are commonly used text editors in the R community.) and many are available as free downloads. Experiment with the many possibilities on how to prepare R syntax using software and a programming environment that best meets personal needs, local computing infrastructure, and selected operating system.

those who engaged in European Renaissance court life. The early interest in probability may not have been altruistic but was instead focused on gaining advantage in card games and other forms of gambling. The use of probability to solve problems for societal gain may not have been the first interest but instead attention was focused on the question, *Given that there are X cards in the deck, if I discard the Y card from my hand, what is the chance that I will draw the Z card from the deck and improve my chance of winning this game of cards?*³

This early interest in probability, and eventually the evolving science of statistics as a vehicle for social improvement, eventually grew into what we think of as biostatistics. It is far beyond the purpose of this introductory text on the use of R in biostatistics to go into too much detail about the evolution of statistics as a science, but at a minimum it would be helpful to look into the biography and contributions of the following founders of what we now consider biostatistics and closely related sciences:

- **Gerolamo Cardano** (1501–1576) was an early writer on probability. Although well-regarded in his time as a polymath, he was also fond of gambling and published *Liber de Ludo Aleae*, translated as *Book on Games of Chance*, which some regard as a manual on gambling.⁴
- **Blaise Pascal** (1623–1662) prepared writings on probability and developed the Pascaline, an early business-oriented (e.g., tax records) mechanical calculator. The structured programming language *Pascal* was named in honor of Blaise Pascal.
- **John Graunt** (1620–1674) published *Natural and Political Observations Made Upon the Bills of Mortality*, perhaps the first widely-read text on demographics, public health, and epidemiology. Graunt, as one of the first demographers, established protocols in the 1600s that modeled standards now used in epidemiology, all based on available public health data.

³Basset (also called barbacole and hocca), faro, and lansquenet are examples of card games associated with Renaissance European court life that helped give rise to studies on probability. Separate from those caught up in the frenzy of gambling, some individuals instead determined the sequence of patterns in these card games and from this observation developed ideas about the probability of which cards would show in future hands—at least in a fair and honest game. These ideas on probability were then extended to other situations and eventually to observations about biological phenomena, such as crop yields, livestock production, medicine, etc.

⁴Look at the dates (mid-1500s to early-1800s) when considering the nature of the term *publication* and how scientific information was disseminated during this time period. Our modern view of scientific journals, monographs, textbooks, Internet-based blogs and discussion groups, etc., simply does not compare to how scientific communication occurred in the 1500s, 1600s, 1700s, and early-1800s. Journals were only in an early developmental stage during the later parts of this time period and instead, publications were often personal letters or lectures shared among a few select individuals, either close associates with common interests or small groups of individuals organized as some type of fraternal society.

- **John Snow** (1813–1858) advocated for the development of epidemiology in relation to public health. He provided on the ground empirically-based leadership to combat the *1854 Broad Street (London) Cholera Outbreak*. Because of his work, there were major improvements to the water and waste systems of Victorian London and by this successful example other large cities throughout the world saw improvements to public sanitation as the Industrial Revolution concentrated more and more people into emerging cities.
- **Florence Nightingale** (1820–1910) is likely best known as an advocate for our modern view of nursing. However, her presentation in *Diagram of the Causes of Mortality in the Army in the East* was a breakthrough publication that had strong implications for how biostatistics could be used to improve public health.
- **William Gosset** (1876–1937) wrote *The Probable Error of a Mean* (e.g., Student's t-Test), one of the most frequently used inferential tests for parametric data.
- **Alice Lee** (1858–1939) was one of the first women to receive a Doctor of Science degree from London University (1901). She worked for years at Karl Pearson's Biometric Laboratory, and the statistical tables she developed were used long after they were first developed.
- **Ronald Fisher** (1890–1962), through extensive work in agricultural research and resulting publications such as *Statistical Methods for Research Workers*, is regarded as a pioneer in the use of data, associated research methods, and statistical tests in biostatistics.
- **Rear Admiral Grace Hopper** (1906–1992) was a pioneer using programming languages as a tool for analytics, leading to development of COBOL. Her early leadership on the use of computers and associated high-level programming languages led to the eventual development by others of languages such as S and R.
- **Evelyn Berezin** (1925–2018) pioneered the first computer-based word processor as well as an early airline reservation system that bridged change from vacuum tubes to transistors.

Although Fisher may be the immediate answer if anyone were asked to identify a famous biostatistician, Snow should also be noted. To put the many individuals who contributed to our current view of biostatistics into context, consider Snow's work during the mid-1850s London cholera (e.g., *Vibrio cholerae*) outbreak and his, then, innovative use of mapping techniques based on data gained through exhaustive empirical methods. Far from being an academic who dealt only in theory, Snow put his own life at risk to obtain the data needed to validate that cholera was a waterborne pathogen. Using these data, he used persuasive

argumentation with government officials, based on scientific outcomes, to confront the problem and take appropriate actions.

1.3 Development of R

R was first developed in the early-to-mid 1990s, based on programming features previously used with S and Scheme:

- S was developed in the mid-1970s at a leading telecommunications research and development center, largely as a tool used to solve problems that the few existing statistical analysis packages and programming languages at that time could not solve.
- S was bundled with the UNIX operating system and made available to universities. Initially, S was offered at no additional charge.
- By the mid-1980s there was a change in organizational structure at the telecommunications company where S was developed and soon after there was an attempt to commoditize S.
- Reacting to this change in the status of S and how S was now no longer freely available to researchers in the educational community, in the early-1990s the heuristics of S were reimaged and eventually R grew out of these efforts. R followed along with the structure of S to the point where many prior S programs could be used with R, often with few, if any, modifications.
- In 1995 R was offered as freely available open source software under the GNU Public License.

R provides an excellent environment for the organization, statistical analysis, and graphical presentation of data. As opposed to well-known proprietary statistical analysis software programs, R is both open source and free to download. R is available through the Comprehensive R Archive Network (CRAN, <http://cran.us.r-project.org/>) and supports all major operating systems.

The R environment is based on a fairly large set of functions, which are found in packages 10–15 min, available when R is downloaded. The download takes about 10–15 min, depending on the speed of Internet connectivity. As needed, additional functions are available in external packages that are also freely available as convenient downloads. It is estimated that there are 15,000 or more external packages hosted through CRAN. More complete details on all packages are available through CRAN at <https://cran.r-project.org/web/packages/>. Some external packages have only a few functions whereas other external packages have hundreds of functions available for use.

In the more than 25 years since R grew out of S, the R user community has grown substantially. R is among the most frequently used programming languages

and is consistently one of the leading languages for use by data scientists. To support this large user community, R has active Internet discussion groups. The R community also supports an annual international conference, typically rotating between Europe, North America, and Oceania.

It cannot be overstated that R is gaining international recognition as a preferred tool for data organization, statistical analysis, and graphical presentation. Quite simply, the free nature of open source software is appealing, and the far-reaching availability of R is evident by viewing the many CRAN mirror sites that host R, currently ranging in alphabetical order from Algeria to Uruguay.

1.4 How R is Used in This Text

Each biologically-oriented problem addressed in this text is approached in a manner that promotes consistency, modularity, and ease of syntax reuse for later analyses. The structure of how R is used for each problem follows:

- **Background:** Information is usually provided, at least in part, about the data.
 - **Description of the Data:** Information about the research process, methods, etc., is provided to give context for the data, how the data were obtained, what the data represent, expected minimum and maximum values for the data, etc. There are examples in this text, however, where there is purposely only a limited amount of information about the data, reflecting how some biostatisticians are tasked with *black-box* analyses, where they are purposely given only limited detail so as to minimize the possible introduction of bias.
 - **Null Hypothesis (Ho):** When the structure of the research process and the resulting data allow, a Null Hypothesis (Ho) is provided to give a sense of the expected analyses (e.g., a Null Hypothesis that addresses difference, a Null Hypothesis that addresses association, etc.) and how to interpret the meaning of resulting analyses.
- **Import Data in Comma-Separated Values (.csv) File Format:** With only a few exceptions for demonstration purposes, most data used in this text are prepared in .csv (comma-separated values) file format.⁵ Various R functions are then used to import the .csv dataset into R.⁶ The

⁵Multiple terms are used along with comma-separated values to signify the .csv file format, including: Comma-Separated Values, Comma Separated Values, Comma-separated values, etc. There is some variance in this text for the term used to identify the .csv file format, recognizing these many terms for the same construct.

⁶The .csv (comma-separated values) file format is nearly universal for the way data are put into electronic format and then shared with others. The data for each record (e.g., case, subject, etc.) are typically placed on one line (e.g., row), with commas used to separate the fields (e.g., columns). Open a .csv file in a text editor, not a spreadsheet, to see how commas

various .csv datasets are made available at the publisher's Web-based resource associated with this text.

- **Organize the Data and Display the Code Book:** A Code Book is created by the programmer and is used to communicate data organization, both for personal use and to accommodate others with whom the data are shared. Ideally, a Code Book provides an adequate description of all data, including data contents, data layout, data structure, data types (e.g., nominal, ordinal, interval, numeric, logical, text, etc.). It is also common to provide expected minimum and maximum values for numeric data, to give a sense of data that are in-range as opposed to data that are either outliers or incorrect data entries. Ideally, the Code Book should be of sufficient detail so that personal acquaintance and recall are not needed to use the data effectively.⁷
- **Conduct a Visual Data Check Using Graphics (e.g., Figures):** Most analyses in this text are reviewed first by using simple graphics (e.g., figures), to present a visual data check. Graphics, in this manner, serve many purposes but their main use early in the statistical analyses process is that they provide a sense of the data, range of values, and ultimately a glimpse of direction and possible outcomes. Later, as needed and judged appropriate, the simple graphics are embellished to prepare highly-detailed figures—figures that in many cases are appropriate for professional publication.
- **Descriptive Statistics for Initial Analysis of the Data:** A beginning activity for all analyses involving numeric data (e.g., weight—Kg or Lb, systolic blood pressure—mmHg, low-density lipoproteins—mg/dL), from simple to complex, is to provide a full understanding of the data. For numeric data, this activity is achieved by preparing simple descriptive statistics that reflect measures of central tendency (e.g., mode, mean, standard deviation, median, minimum, maximum, etc.). For data that involve headcounts, it is common to prepare simple descriptive statistics that reflect frequency distributions (e.g., N and percent of total by break-out groups, Female v Male, Alive v Dead, etc.). Descriptive statistics are, collectively, one of many tools used to gain a complete understanding of the data—an essential task before any attempt is made for more complex statistical analyses.

appear as field separators. A .csv file is easily managed and can be opened with most, if not all, text editors or spreadsheets, whether the software is proprietary or freeware. Because of this simplicity, .csv files can be easily shared with others, across multiple platforms, operating systems, and data management software. R is by no means restricted to the use of .csv file format datasets, but this is more than likely the most common file format.

⁷It is common to provide sufficient detail in a Code Book so that a dataset can be revisited 1 year or more after last use and the Code Book will provide all prompts needed to quickly understand the data.

- **Quality Assurance, Data Distribution, and Tests for Normality:** A variety of quality assurance approaches are used in this text, with the selected process based on need and opportunity. In some cases, when warranted, a precise statistical test is used to provide a sense of quality assurance. In other cases, when the data do not allow such precision, simple descriptive statistics alone are used to provide a gauge of data quality. Graphics, from simple throwaway black and white figures to highly-detailed embellished figures, are also frequently used to serve the quality assurance process. As an adjunct activity closely linked to the quality assurance process, it is common to examine data for distribution patterns. Do the data follow or at least approximate patterns of normal distribution (e.g., bell-shaped curve)? Or, do the data instead fail to show a discernible pattern of normal distribution? This issue is important for selection of the many available statistical tests and the underlying assumptions associated with each test. For some statistical analysis tests, it is assumed that the data for a variable in question follow a normal distribution pattern—tests based on parametric data. If this pattern of normal distribution is not met, then other tests may be more appropriate—tests based on nonparametric data. There are more than a few tests for normality, and they are demonstrated in various lessons throughout this text. The key thing to remember about quality assurance is that it should be pervasive throughout the research and statistical analysis process. Quality assurance is *never* a *one-and-done* process..
- **Statistical Test(s):** A few of the leading statistical tests are demonstrated in this text, both tests that use data where this is an assumption of normal distribution (e.g., parametric tests) and tests that use data that fail to meet normal distribution (e.g., nonparametric tests). The statistical tests are demonstrated where it is assumed that the data either meet or at least approximate normal distribution. However, in many cases an addendum is provided that challenges the assumption of normality and, instead, a nonparametric approach is used for statistical analysis.
- **Summary of Outcomes:** It is only too common for many students and beginning researchers to have some degree of difficulty understanding output from the many available statistical tests, whether R or some other software package is used. Recognizing this concern, a summary of outcomes for each statistical test is provided as a guide for interpretation as the previously stated Null Hypothesis is addressed.
- **Addenda:** Each chapter ends with multiple addenda, providing an opportunity to either introduce or reinforce important topics and associated functions that go far beyond the basics of what could be otherwise found in a standard R-based documentation page:

- In many cases, the addenda either introduce or reinforce concepts, packages, functions, function arguments, etc., in greater detail than what was presented earlier, often using different data—all to give a new perspective.
 - There is a great deal of attention to the concept of parametric data, as well as attention to the converse of nonparametric data—issues related to normality. Too many texts give scrutiny about data distribution patterns short shrift, or worse, totally ignore this assumption inherent to the correct use of many inferential tests.
 - Practice data sets are also part of the addenda, often demonstrating one dataset addressing data with a normal distribution pattern and another dataset addressing data that do not exhibit normal distribution patterns. Students and beginning researchers benefit when reminded that data are not always *neat and pretty* and multiple approaches to statistical analysis have merit.
- **Prepare to Exit, Save, and Later Retrieve this R Session:** As a good programming practice, it is best to prepare all R syntax in a separate file, using a text editor. This practice makes it easy to save and reuse the prepared syntax for later use or as a template for other future R-based analyses. It is also a good programming practice to save active R sessions, again for the purpose of facilitating later reuse. Clear instructions are provided on how to execute a *graceful* exit from the R session and to save the session for later reuse.

Again, small and easy-to-follow confidence-building examples are used at the beginning of this text. Greater complexity is gradually introduced until the final lessons in this text. In these last lessons R is used in a fairly robust manner, where large and complex datasets are used to introduce and reinforce skills needed for independent statistical analyses that support research efforts.

1.5 Import Data Into R

Comma-separated values (.csv) files are typically used to import data into R. However, there are other ways to import data into R, as demonstrated in this introductory lesson. The many data-import examples in this lesson use small datasets (small in terms of the number of subjects and small in terms of the number of variables) and are purposely provided early in this text to demonstrate the flexibility of R in relation to data management. There is no attempt to use these simple datasets for any analyses, now. This section is only provided to demonstrate the many different ways data can be imported into R.

To follow along with these examples, first download R from the Comprehensive R Archive Network (<https://www.cran.r-project.org/>). Make selections based

on operating system. Open R after it has been downloaded and experiment with the interface, recognizing that it is a plain and stark compared to more common Graphical User Interface (GUI) statistical analysis programs.

Use R in an interactive mode, typing syntax such as `2 * 2`, or `x <- c(1, 2, 3); print(x)`, or `print(rnorm(100, mean=120, sd=6))` at the R prompt. Do not forget to press the Enter key after each line of syntax.

However, for a more meaningful experience, download the many different datasets made available at the publisher's Web-based resource associated with this text. With R open, and with the datasets downloaded, follow along with the different examples used in this lesson to gain initial experience with R. To promote the best use of time-on-task, it may be better to prepare all R-based syntax using a text editor or some IDE-type middleware software, and then transfer the syntax to R. The ways by which this can be put into place are many and there is no one-and-only-one way to achieve this aim, ranging from a simple copy and paste action to a more complex action where a *chunk* of syntax in an external editor is blocked and sent to the R session.⁸

For now, observe the R syntax shown in the rest of this lesson, but do not give too much concern about the syntax. The many R-based functions, arguments, etc., that collectively represent R syntax will be explained in future lessons. Look at the initial Housekeeping syntax, below, to see a sample of what to expect whenever a new R session is started:

- The `date()` function is used to provide a marker for when the R session is saved and then reviewed again, later.
- `R.version.string` is used to verify which version of R was used for the session.
- The `rm()` function is used to remove objects.
- The `setwd()` function is used to change directories and have the R session in the desired directory or `F:/R_BiostatisticsIntroduction` in this session.
- The `getwd()` function is used to confirm the current working directory.
- The `ls()` function is used to list all files in the current directory.
- The `sessionInfo()` function is used to confirm the R version and to gain information on the locale and available packages.

⁸The term *chunk* was purposely selected to provide an advance organizer to terms used with the markdown process, where R syntax (a section or *chunk* of R code) and narrative text are integrated into one common document, which has become an increasingly popular way of preparing formal reports that include narrative text and R syntax. The markdown process is detailed in the last lesson in this text.

- The search() function is used to review the attached packages and to list all objects in the directory.

R Input

```
#####
# Housekeeping           Use for All Analyses #
#####
date()          # Current system time and date.
Sys.time()       # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls()             # List all objects in the working
                 # directory.
rm(list = ls())  # CAUTION: Remove all files in the
                 # working directory. If this
                 # action is not desired, use rm()
                 # one-by-one to remove the objects
                 # that are not needed.
ls.str()         # List all objects with finite detail.
getwd()          # Identify the current working directory.
setwd("F:/R_BiostatisticsIntroduction")
                 # Set to a new working directory.
                 # Note the single forward slash and double
                 # quotes.
                 # This new directory should be the directory
                 # where the data file is located, otherwise
                 # the data file will not be found.
getwd()          # Confirm the working directory.
list.files()     # List files at the PC directory.
.libPaths()      # Library pathname.
.Library         # Library pathname.
sessionInfo()   # R version, locale, and packages.
search()         # Attached packages and objects.
searchpaths()   # Attached packages and objects.
#####
```

When these actions are completed, the current R session should then begin, knowing that all files are in proper order, the working directory is set as desired, etc. These initial actions, known as Housekeeping in this text, provide assurance that the R session will begin correctly. Omitting any of this syntax may result

in either problems with output or output that does not match what shows throughout this lesson.^{9,10}

With this beginning Housekeeping activity completed, the different subsections that follow show how to import data into an active R session. Practice with these many examples to gain exposure and confidence before the later lessons are attempted. Remember, again, that these are all simple datasets and no attempt has been made to apply any analyses or to generate any highly-detailed figures against the data in the main body of this lesson.

1.5.1 Import a .csv File of Comma-Separated Values into R

Follow the syntax shown below to import `GenderEndurance.csv`, a .csv (comma-separated values) file saved in the declared working directory, which was cited in the Housekeeping section by using the `setwd()` function. The `read.table()` function, which is included in the `utils` package, will be used to direct this activity. The output of this action will be placed into an object called `GenEnd.df`.¹¹

R Input

```
GenEnd.df <- utils::read.table (file =
  "GenderEndurance.csv",
  header=TRUE, dec=". ", sep=",")
# Use the utils::read.table() function to import the
# .csv file GenderEndurance.csv into the current R
# session and place the contents into the object
```

⁹When writing R syntax, recall that functions are used by R to make things happen. The `mean()` function is used to determine the mean (e.g., arithmetic average) of a set of numbers included in an object variable. However, functions are contained in packages. As such, it is common to include the package name along with the function name, such that `base::mean()` is a more formal (and arguably, better) way to write R-based syntax that results in calculation of the mean, given that the `mean()` function is included in the base package. Both methods (e.g., `Function()` and the more formal `Package::Function()`) are used throughout this text, reflecting how syntax is written by others. Generally, it is common to avoid writing the package name for the packages that are included in the initial R download, but to write the package name when using functions included in external packages—but this is all a matter of choice in many cases. Demonstration of these two ways of writing function names is not meant to be confusing but is instead shown in this lesson and later lessons to reflect what should be expected when viewing syntax prepared by others, either colleagues or what may be seen using other resources.

¹⁰Throughout this text, notice how R Input and R Output are placed in different lightly-colored boxes. The syntax for all input is included in this text. However, to conserve space and to prevent an overflow of pages, all output is not shown. All output can be generated, however, by using the datasets associated with this text and the input shown throughout. Ideally, it would only be necessary to change the name of the working directory, declared at the local level, to replicate all examples in this and other lessons.

¹¹Throughout this text, the `.df` extension is used to reinforce that the object is a dataframe.

```
# GenEnd.df, which is a dataframe that: (1) has a
# header row, (2) uses a period for decimals, and (3)
# uses a comma to separate one field from another.
#
# Note how the utils package is available as one of the
# packages immediately put into use when a R session is
# first started.
```

With the object `GenEnd.df` imported into R, it is then necessary to perform a series of quality assurance actions to be sure that everything is correct and that the file is in proper order and ready for use.¹²

R Input

```
getwd()                      # Identify the working directory
ls()                          # List objects
attach(GenEnd.df)            # Attach the data, for later use
str(GenEnd.df)                # Identify structure
head(GenEnd.df, n=3)          # Show the head, 1st 3 cases
summary(GenEnd.df)             # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

```
> str(GenEnd.df)              # Identify structure
'data.frame':   20 obs. of  3 variables:
 $ Subject : Factor w/ 20 levels "S01","S02","S03",...: 1 2
 $ Gender   : Factor w/ 2 levels "Female","Male": 1 1 1 1 1
 $ Endurance: num  3.49 2.97 2.89 2.92 3.42 ...
> summary(GenEnd.df)          # Summary statistics
   Subject      Gender     Endurance
   S01       : 1    Female:10    Min.   :2.37
   S02       : 1    Male  :10    1st Qu.:2.91
   S03       : 1                  Median :2.95
   S04       : 1                  Mean   :3.01
```

¹²This lesson is focused on the many ways that data can be brought into a R session. The emphasis, now, is not on functions such as `str()`, `summary()`, etc. These functions are detailed completely in later lessons. As such, the text input and text output is presented in a slightly different manner in this lesson than what is seen in all later lessons, so that focus remains on how data are brought into R.

S05 : 1	3rd Qu.: 3.15
S06 : 1	Max. : 3.49
(Other): 14	

The file called `GenderEndurance.csv` has been imported into R, with the data contained in the object `GenEnd.df`. The data seem to be in correct order, but a graphic will also serve a quality assurance role, to be certain that everything is in order. Again, do not be overly concerned about the syntax used to generate the graphic and, instead, observe how the `plot()` function is used an overlay on the `density()` function, to gain a sense of data distribution for the `GenEnd.df$Endurance` object variable (Fig. 1.1).¹³

R Input

```
par(ask=TRUE)
plot(density(GenEnd.df$Endurance, na.rm=TRUE),
     main="Endurance of Selected Female and Male Subjects:
           Quality Assurance Density Plot", col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable GenEnd.df$Endurance, with arguments
# resulting in a thick (lwd=5) red line.
```

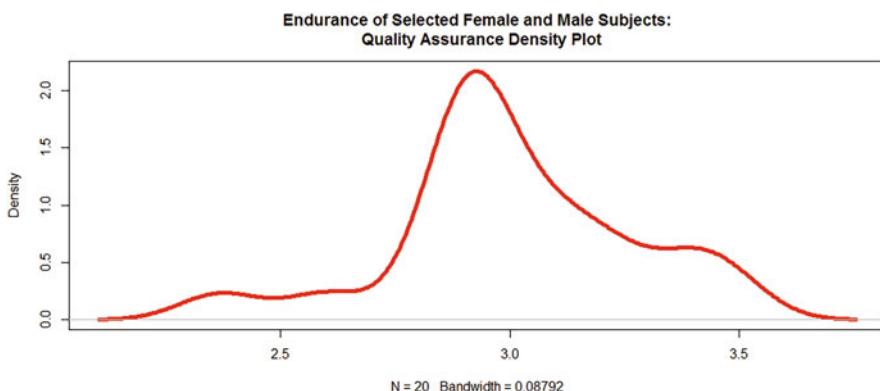


Figure 1.1: Quality assurance of endurance

¹³The `$` character is used to fully identify the `GenEnd.df$Endurance` object variable. In the same way that a formal, if long, naming process is used for packages and functions (e.g., `Package::Function`), the same concept applies to objects and variables associated with objects. The more formal `GenEnd.df$Endurance` is a better way to identify the object in question, instead of `Endurance` alone. Otherwise, there would be confusion if there were two objects in an active R session and each object contained a variable called `Endurance`. Explicit naming schemes such as `Object$Variable` and `Package::Function` have value as quality assurance measures.

1.5.2 Import a .txt File of Tab-Separated Values into R

Similar to what was used when importing data found in a .csv file, follow along with the syntax shown below to import a .txt file that consists of data separated by tab characters, not commas. Although the file `BreedMilkLb365.txt` uses a .txt (e.g., text) file extension, it is not uncommon to see files consisting of tab-separated values with either a .tsv file extension or a .tab file extension, although they are actually text files, regardless of the .tsv and .tab file extensions.

R Input

```
BreedMilk.df <- utils::read.table (file =
  "BreedMilkLb365.txt",
  header=TRUE, dec=".",
  sep="\t")
# Use the utils::read.table() function to import the
# .txt file BreedMilkLb365.txt into the current R
# session and place the contents into the object
# BreedMilk.df, which is a dataframe that: (1) has a
# header row, (2) uses a period for decimals, and (3)
# uses a tab to separate one field from another.
```

Once again, use standard quality assurance actions to confirm that the data are correct and acceptable for later use.

R Input

```
getwd()          # Identify the working directory
ls()            # List objects
attach(BreedMilk.df) # Attach the data, for later use
str(BreedMilk.df) # Identify structure
head(BreedMilk.df, n=3) # Show the head, 1st 3 cases
summary(BreedMilk.df) # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

```
> str(BreedMilk.df)      # Identify structure
'data.frame': 20 obs. of 3 variables:
 $ Cow       : Factor w/ 20 levels "C01","C02","C03",...: 1 2
 $ Breed     : Factor w/ 2 levels "Holstein","Jersey": 2 2 2
 $ MilkLb365: int 14514 15443 14963 15997 15653 15854 14361
> summary(BreedMilk.df) # Summary statistics
```

Cow	Breed	MilkLb365
C01 : 1	Holstein:10	Min. :14219
C02 : 1	Jersey :10	1st Qu.:15164
C03 : 1		Median :16987
C04 : 1		Mean :17449
C05 : 1		3rd Qu.:19691
C06 : 1		Max. :20810
(Other):14		

As a final quality assurance review, produce a graphic (specifically, a density plot) of the object variable `BreedMilk.df$MilkLb365` to further understand the nature of the data and to review if the data follow along expected distribution patterns (Fig. 1.2).

R Input

```
par(ask=TRUE)
plot(density(BreedMilk.df$MilkLb365, na.rm=TRUE),
      main="Annual Milk Production (Pounds) of Holstein and
      Jersey Cows: Quality Assurance Density Plot",
      col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable BreedMilk.df$MilkLb365, with arguments
# resulting in a thick (lwd=5) red line.
```

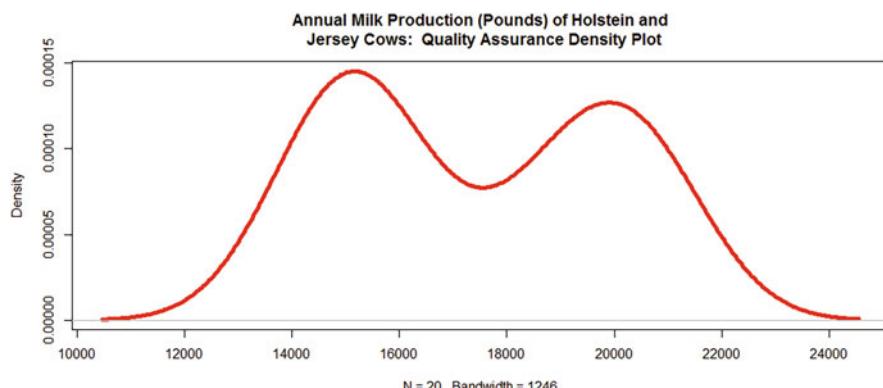


Figure 1.2: Quality assurance of MilkLb365

1.5.3 Import a .txt File of Fixed-Width Format Values into R

Data in a .txt (e.g., text) file that are placed into fixed-width format are placed in neatly organized rows and columns, usually with all data for each case showing on one row. In turn, each object has a set width, ranging from 1 to N columns wide. When viewing these rows and columns it is important to recall that spaces are used to separate one datum from another—neither commas nor tabs are used to separate the data. It is common to use the term *padding* when describing fixed-width data, especially if leading zeros are used to align data in columns.

Fixed-width format .txt files were once fairly common but admittedly comma-separated values (.csv) file format is now far more common. Thorough review and setup are needed to successfully import a fixed-width format file, which is again a reason for more frequent use of .csv files. Even so, most researchers will eventually encounter fixed-width format files so it is useful to gain some degree of experience with this row-by-column, space-delimited file format.¹⁴

The file `YearSoilTypeCropRainYieldBushelsPerAcreNoHeader.txt` is a fixed-width format .txt file, consisting of 60 cases (e.g., rows) and five object variables (e.g., columns). Carefully review the documentation to see more about the data, what the data represent, and how the data are organized. Give special attention to the `utils::read.fwf()` function and how it is used to import the data.

R Input

```
SoilYield.df <- utils::read.fwf(
  "YearSoilTypeCropRainYieldBushelsPerAcreNoHeader.txt",
  header=FALSE, # There is no header for column names.
  skip=0, # Skip no lines; read data from the 1st line.
  na.strings=" ",# Blank spaces in a character object are NA.
  width=c(-1,4, # Year (1997 to 2016)
          -1,4, # Predominant Soil Type (Sand, Silt, Clay)
          -1,4, # Crop
          -1,6, # Rain (Dry, Normal, Wet)
          -1,9)) # Yield (Bushels Per Acre, BUpерAcre)
# Use the utils::read.fwf() function to import the .txt file
# YearSoilTypeCropRainYieldBushelsPerAcreNoHeader.txt into
# the current R session and place the contents into the
```

¹⁴Although fixed-width format files are no longer common, they were once a standard format for dataset organization, where simple text editors were used to organize and manipulate data. Among many remaining comparative advantages of fixed-width format is that an incorrectly placed comma in a fixed-width format dataset cannot alter data structure beyond placement of the errant comma. However, when a comma is typed, by mistake, during manual data entry of a .csv text file, errors will likely show throughout—all due to a simple comma out-of-place.

```

# object SoilYield.df, which is a dataframe that: (1) in
# original format does not have a header row, (2) uses a
# somewhat complicated process to identify those fixed-width
# columns that do not contain data and therefore should be
# skipped and those columns that do contain data and should
# therefore be used to read data:
# Skip the first column (-1) and then read Year data in the
# next four columns.
# Skip the next column (-1) and then read Soil data in the
# next four columns.
# Skip the next column (-1) and then read Crop data in the
# next four columns.
# Skip the next column (-1) and then read Rain data in the
# next six columns.
# Skip the next column (-1) and then read BUperAcre data in
# the next nine columns.

```

R Input

```

names(SoilYield.df) <- c(
  "Year",           # Year
  "Soil",           # Soil
  "Crop",           # Crop
  "Rain",           # Rain
  "BUperAcre")     # Bushels per Acre
# Column names were not included in the original .txt
# dataset. Use the names() function to supply column
# names to the object SoilYield.df.

```

If there are no output errors with this seemingly complex scheme for declaring columns and data placement, use simple quality assurance actions such as `str()` and `summary()` to confirm that the data are correct.

R Input

```

getwd()                  # Identify the working directory
ls()                     # List objects
attach(SoilYield.df)    # Attach the data, for later use
str(SoilYield.df)       # Identify structure
head(SoilYield.df, n=3) # Show the head, 1st 3 cases
summary(SoilYield.df)   # Summary statistics

```

R Output

[Selected output is not shown, to save space.]

```
> str(SoilYield.df)      # Identify structure
'data.frame': 60 obs. of 5 variables:
 $ Year     : int 1997 1998 1999 2000 2001 2002 2003 2004
 $ Soil      : Factor w/ 3 levels "Clay","Sand",...: 1 1 1 1
 $ Crop      : Factor w/ 1 level "Corn": 1 1 1 1 1 1 1 1 1
 $ Rain      : Factor w/ 3 levels "Dry", "Normal", ...: 3 2
 $ BUperAcre: int 184 166 169 170 175 166 183 191 164 162
> summary(SoilYield.df)  # Summary statistics
   Year        Soil       Crop      Rain      BUperAcre
Min. :1997    Clay:20    Corn:60   Dry   : 9    Min.   :127
1st Qu.:2002   Sand:20          Normal:39  1st Qu.:149
Median :2006   Silt:20          Wet    :12    Median :162
Mean   :2006          NA's:21          Mean   :158
3rd Qu.:2011          NA's:10          3rd Qu.:168
Max.   :2016          NA's: 1          Max.   :192
```

Output from use of the `summary()` function provides evidence that all values are within expected ranges. A graphic of corn yields (e.g., `SoilYield.df$BUperAcre`) will provide further confirmation that the data were imported correctly into R (Fig. 1.3).¹⁵

R Input

```
par(ask=TRUE)
plot(density(SoilYield.df$BUperAcre, na.rm=TRUE),
      main="Corn Yield (Bushels per Acre) for Different Soils
            from 1997 to 2016 at a Selected Midwestern
            Region: Quality Assurance Density Plot", col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable SoilYield.df$BUperAcre, with arguments
# resulting in a thick (lwd=5) red line.
```

¹⁵Again, do not be concerned about completely understanding syntax such as `par(ask=TRUE)` or `lwd=5`. This lesson is focused on the processes needed to import data into R. Syntax used to generate figures and descriptive statistics is detailed in later lessons.

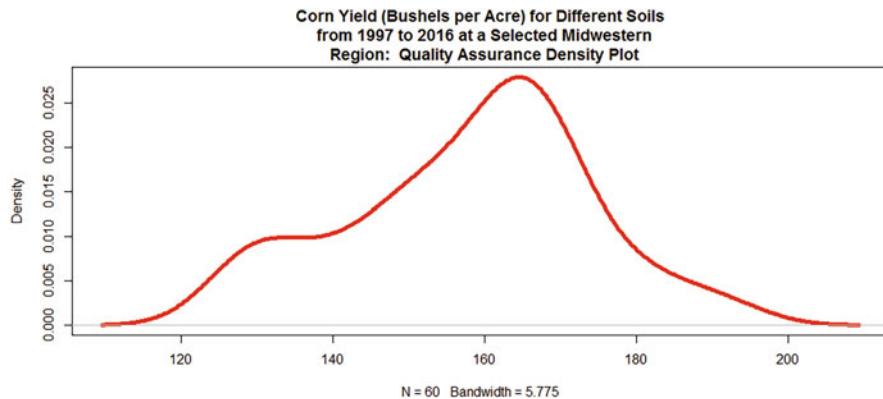


Figure 1.3: Quality assurance of corn yield

Knowing that the process for using the `utils::read.fwf()` function is at first challenging, given the way blank columns are marked with a negative number corresponding to the number of skipped columns, look at another fixed-width format dataset that is more comprehensive, with more objects and therefore more columns that need accounting. In this additional example, the fixed-width format .txt file `WellnessInventory.txt` will be imported into R and the data will be placed in the object `Wellness.df`. The process to import a fixed-width format .txt file into R remains the same, where skipped columns are identified by a negative number for the number of skipped columns, and columns that actually hold data are identified by the specific number of columns allocated for the individual object variable.¹⁶

R Input

```
Wellness.df <- utils::read.fwf("WellnessInventory.txt",
  header=FALSE, # There is no header for column names.
  skip=0,        # Skip no lines; read data from the first line.
  na.strings=" ",# Blank spaces in a character object are NA.
  width=c(-1,3, # Patient Identification Number
          -1,1,  # Gender
          -1,10, # Date-of-Birth (Month/Day/Year, XX/XX/XXXX)
          -1,3,  # Height in Inches
          -1,1,  # Race
          -1,3,  # Weight in Pounds
          -1,2,  # Waist in Inches
          -1,3,  # Systolic Blood Pressure (mmHg)
          -1,3,  # Diastolic Blood Pressure (mmHg)
```

¹⁶Compared to other file formats, there may be a slight delay in processing time when importing a fixed-width format dataset. This delay is due to the complexity of how data are put into final form. The delay is generally minimal and has no impact on data quality after data are imported—just be patient.

```

-1,3, # Resting Heart Rate
-1,3, # Total Blood Cholesterol
-1,2, # Sleep Hours Last Night
-1,1, # Smoke
-1,1, # Drink
-1,1, # Income
-1,1, # Exercise
-1,1)) # Food Frome Home

# Use the utils::read.fwf() function to import the
# .txt file WellnessInventory.txt into the current R
# session. Again, note how there is no header row of
# column names and review how blank columns and columns
# holding data are identified.

```

R Input

```

names(Wellness.df) <- c(
  "ID",           # Patient Identification Number
  "Gender",       # Gender
  "DOB",          # Date-of-Birth (Month/Day/Year, XX/XX/XXXX)
  "Height",       # Height in Inches
  "Race",          # Race
  "Weight",       # Weight in Pounds
  "Waist",        # Waist in Inches
  "SBP",          # Systolic Blood Pressure (mmHg)
  "DBP",          # Diastolic Blood Pressure (mmHg)
  "RHR",          # Resting Heart Rate
  "TBC",          # Total Blood Cholesterol
  "Sleep",         # Sleep Hours Last Night
  "Smoke",         # Smoke
  "Drink",         # Drink
  "Income",        # Income
  "Exercise",      # Exercise
  "Food")         # Food From Home

```

The need for quality assurance again calls for simple measures (e.g., use of the `str()` and `summary()` functions) to initially examine if the data were imported correctly into R.

R Input

```

getwd()           # Identify the working directory
ls()              # List objects

```

```
attach(Wellness.df)      # Attach the data, for later use
str(Wellness.df)        # Identify structure
head(Wellness.df, n=3)  # Show the head, 1st 3 cases
summary(Wellness.df)    # Summary statistics
```

Only a selected portion of the output for these different quality assurance measures is shown below. Give attention to the output from the str() function where each part of the dataframe is clearly identified. Determine if this structure is correct or if modifications (e.g., enumerations) are needed. Additionally, look at selected portions of output from the summary() function to see how NA (e.g., missing data) is automatically introduced into the dataframe when the original dataset has a blank value for an individual datum.

R Output

[Selected output is not shown, to save space.]

```
> str(Wellness.df)      # Identify structure
'data.frame':   120 obs. of  17 variables:
 $ ID      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Gender   : Factor w/ 2 levels "F","M": 1 1 2 2 1 NA 2 1 1
 $ DOB      : Factor w/ 117 levels "           ","01/01/1973"
 $ Height   : int  51 63 69 72 74 63 62 NA 60 68 ...
 $ Race     : Factor w/ 5 levels "A","B","H","W",...: 1 4 3 4
 $ Weight   : int  221 112 148 130 210 186 165 135 110 147 .
 $ Waist    : int  42 30 30 30 40 38 38 29 27 29 ...
 $ SBP      : int  130 96 124 110 140 132 128 118 92 132 ...
 $ DBP      : int  80 62 68 70 82 78 76 64 62 78 ...
 $ RHR      : int  80 74 72 69 83 76 72 84 81 76 ...
 $ TBC      : int  221 167 170 162 218 204 195 168 170 188 .
 $ Sleep    : int  5 6 9 4 8 9 6 10 6 7 ...
 $ Smoke    : int  2 2 1 4 4 1 1 4 2 1 ...
 $ Drink    : int  4 2 3 4 2 1 2 2 2 3 ...
 $ Income   : int  1 2 4 5 5 2 2 3 4 2 ...
 $ Exercise: int  0 1 2 3 1 2 3 0 3 1 ...
 $ Food     : int  1 NA 1 3 2 0 NA NA 3 3 ...
```

R Output

[Selected output is not shown, to save space.]

```
> summary(Wellness.df)    # Summary statistics
```

Gender	Height	Race	Weight
F :62	Min. :51.00	A : 7	Min. : 95.0
M :57	1st Qu.:64.00	B :13	1st Qu.:133.5
NA's: 1	Median :67.00	H :25	Median :165.0
	Mean :66.85	W :69	Mean :173.7
	3rd Qu.:70.00	X : 3	3rd Qu.:210.0
	Max. :79.00	NA's: 3	Max. :310.0
	NA's :2		NA's :2

Finally, prepare a simple density plot of `Wellness.df$Waist`, a selected numerical object variable, to examine the dataframe for quality purposes. Ideally, the diligent researcher will prepare quality assurance graphics of all numerical object variables, going far beyond what is shown in this introductory lesson (Fig. 1.4).

R Input

```
par(ask=TRUE)
plot(density(Wellness.df$Waist, na.rm=TRUE),
      main="Waist (Inches) of Selected Wellness Inventory
Subjects: Quality Assurance Density Plot",
      col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable Wellness.df$Waist, with arguments
# resulting in a thick (lwd=5) red line.
```

1.5.4 Import a .xlsx Spreadsheet File into R

It is generally common to import .csv (comma-separated values) files into R: the format is simple, the files are easy-to-import, and .csv files are considered by some researchers as the *de facto* file format for external datasets. However, there are functions in R-based packages that can be used to import a spreadsheet into R, with the `readxl` package possibly the easiest to use, although other R-based packages (i.e., `gdata`, `gnumeric`, `XLConnect`, `xlsx`) have value and should be considered.¹⁷

The `readxl::read_excel()` function will be used in this lesson to import data from a spreadsheet.¹⁸ Otherwise, the process to obtain and then use functions

¹⁷There are currently 15,000 or more external packages available to the R community, and the list of available packages grows daily. Use available resources to learn more about these packages and how functions in these packages can greatly facilitate the research process.

¹⁸Be sure to note the formal naming scheme: `Package::Function()`. This formal process is often used throughout this text whenever external packages are used, as opposed to the use

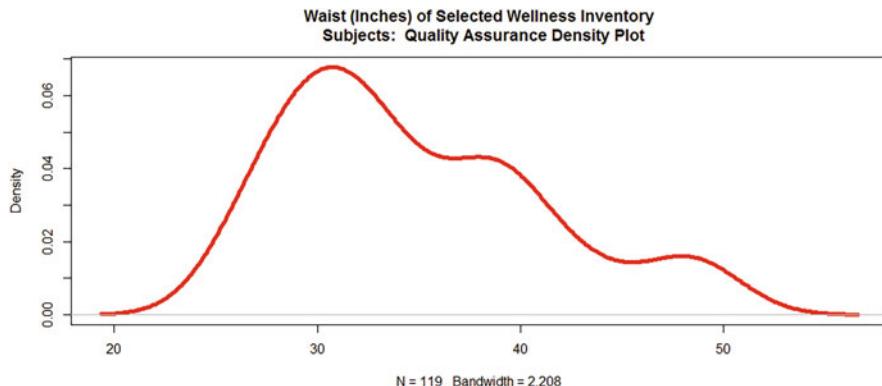


Figure 1.4: Quality assurance of waist

from an external package is quite simple:

- Use the `install.packages()` function to obtain and install the external package.¹⁹
- Use the `library()` function to load the package.
- The `help()` function and the `sessionInfo()` function should also be used to learn more about the package and how it fits in the list of packages available in the current R session.
- Use the desired function from the external package to perform the required action. In this example, import a .xlsx spreadsheet.

R Input

```
install.packages("readxl", dependencies=TRUE)
library(readxl)                      # Load the readxl package.
help(package=readxl)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.
```

With the `readxl` package made available, apply the `readxl::read_excel()` function against the desired .xlsx spreadsheet, to import the data. Direct the imported data into the named dataframe, `Sorghum.df` in this example.

of the less formal `Function()`, only, naming scheme. By using this formal process, there is no doubt about the package name from which the function was obtained.

¹⁹This action will result in a pop-up menu that calls for the selection of a CRAN mirror site, where packages are located. Select the most local geographic site.

R Input

```
Sorghum.df <- readxl::read_excel("Sorghum2012to2016.xlsx", 1)
# Use the readxl::read_excel() function to import the .xlsx
# spreadsheet Sorghum2012to2016.xlsx into the current R
# session and place the contents into the object
# Sorghum.df.
#
# The number 1 that shows after the .xlsx filename is used
# to declare that only the 1st sheet in the spreadsheet
# should be read into the intended object, Sorghum.df in
# this example.
```

Confirm that the data are imported and that they are in correct format by applying selected quality assurance actions. Notice a slight difference in how the first few lines of data show when the head() function is applied against the spreadsheet-originated data, with the data type for each object showing under the header row. Review the term **tibble** or the symbol `tbl_df` to see how .xlsx data, when imported into R, are slightly different than the expected dataframe format.

R Input

```
getwd()                      # Identify the working directory
ls()                          # List objects
attach(Sorghum.df)           # Attach the data, for later use
str(Sorghum.df)               # Identify structure
head(Sorghum.df, n=3)         # Show the head, 1st 3 cases
summary(Sorghum.df)           # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

```
> str(Sorghum.df)              # Identify structure
Classes 'tbl_df', 'tbl' and 'data.frame':      496 obs. of
 $ TrackingID: chr  "Sorghum0060CY2012" "Sorghum0021CY2012"
 $ Year       : num  2012 2012 2012 2012 2012 ...
 $ Management: chr  "Conventional" "Conventional" "Conventi
 $ BUperAcre  : num  28 35 42 43 44 45 46 47 47 48 ...
> head(Sorghum.df, n=3)        # Show the head, 1st 3 cases
# A tibble: 3 x 4
  TrackingID   Year Management  BUperAcre
  <chr>       <dbl> <chr>        <dbl>
1 "Sorghum0060CY2012" 2012 Conventional 28
2 "Sorghum0021CY2012" 2012 Conventional 35
3 "Sorghum0060CY2012" 2012 Conventional 42
```

	<chr>	<dbl>	<chr>	<dbl>
1	Sorghum0060CY2012	2012	Conventional	28
2	Sorghum0021CY2012	2012	Conventional	35
3	Sorghum0063CY2012	2012	Conventional	42

Given that the data seem acceptable, even though the dataframe is somewhat different, apply a graphical quality assurance action against a selected numeric object variable, to gain a sense of data quality (Fig. 1.5).

R Input

```
par(ask=TRUE)
plot(density(Sorghum.df$BUperAcre, na.rm=TRUE),
      main="Sorghum Yield (Bushels per Acre) for Different Management
Practices from 2012 to 2016 at a Selected Midwestern
Region: Quality Assurance Density Plot", col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable Sorghum.df$BUperAcre, with arguments
# resulting in a thick (lwd=5) red line.
```

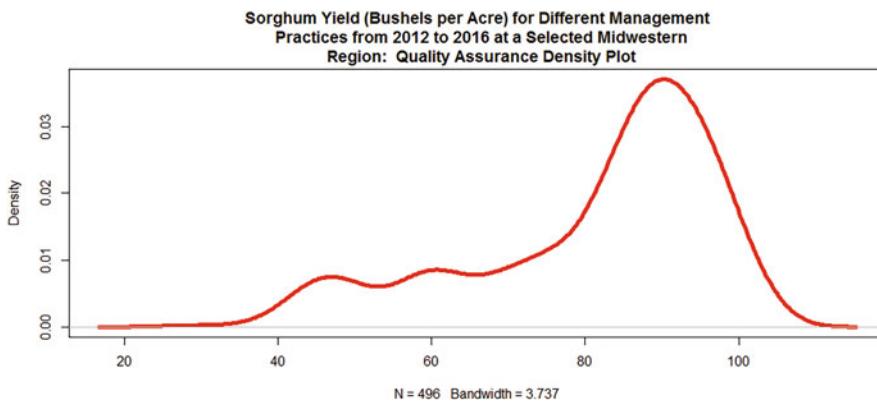


Figure 1.5: Quality assurance of sorghum yield

1.5.5 Import a .csv File of Comma-Separated Values from an Online Source into R

As the Internet has developed, it is now common to find valuable data related to biostatistics at a variety of online resources, both from private sources and sources maintained by different government agencies. In many cases, data offered by government agencies are in the public domain and the data are available to anyone, with no password requirement or other restrictions:

- Some government-supplied data are organized as one large dataset relating to a common topic, typically as a .csv file. After the .csv file is downloaded it is then often necessary to use different offline organizational strategies to focus on data of immediate concern and to either discard or ignore data that are not needed.
- Some government-supplied data are available at an interface that allows search-and-organize queries, where data can be arranged to meet specific search parameters. The data can then be downloaded, often in multiple file formats, with .csv files perhaps the most common format.

Consider an example where it is possible to import an appropriately-organized online file, such as a .csv file, directly into a R session, making it unnecessary to first download the file to the working directory. For this example, focus on a health-related survey of children and adolescents and their consumption of fruit and vegetables, which of course has biostatistics-related implications for those who work in both agriculture and the health sciences. The data are posted at a Centers for Disease Control and Prevention Web location (<https://chronicdata.cdc.gov>), presented as a .csv file.²⁰

Use the `install.packages()` function to obtain the `data.table::fread()` function, which is used to obtain Internet-based datasets. Notice how the data are imported into R directly from an Internet-based file. By knowing the needed URL, it is not necessary to implement any selection-type activities.

R Input

```
install.packages("data.table", dependencies=TRUE)
library(data.table)                      # Load the data.table package.
help(package=data.table)                 # Show the information page.
sessionInfo()                           # Confirm all attached packages.
```

R Input

```
ChildHealth.df <- data.table::fread(
  'https://chronicdata.cdc.gov/api/views/vba9-s8jp/rows.csv'
)
```

R Input

```
getwd()                                # Working directory
ls()                                    # List objects
```

²⁰As a reminder, URL refers to the term *Uniform Resource Locator*, or what many simply call a Web address.

```
attach(ChildHealth.df)      # Attach the data
str(ChildHealth.df)        # Identify structure
head(ChildHealth.df, n=3)  # Show the head
summary(ChildHealth.df)    # Summary statistics
```

A review of this exceptionally large file (currently more than 35,000 rows and more than 30 columns, with new data being added over time) is needed, but for this lesson focus on data from two specific columns:

- The columns labeled Topic and Question focus on specific behaviors that may impact the health of children and adolescents. For this example all analyses and graphics will be specific to:
 - Topic, Fruits and Vegetables—Behavior
 - Question, Percent of students in grades 9–12 who consume fruit less than one time daily
- The column labeled Data_Value is the location of the actual percentage value related to the question about daily consumption of fruit, along with all other questions in the dataset.

Because the dataset `ChildHealth.df` is so large, it may be helpful to construct a new dataset that relates only to this example. Do not give undue attention to the Boolean-based syntax presented below, but for now only notice how R syntax is used so that a new smaller dataset is constructed from the original larger dataset.²¹ The somewhat complex R-based syntax shown below is detailed in future lessons. For this introductory lesson, focus on process, not syntax.

R Input

```
ChildHealthFruitVegetable.df <- ChildHealth.df[ which(
  ChildHealth.df$Topic=="Fruits and Vegetables - Behavior"), ]
```

If executed correctly, the dataset `ChildHealthFruitVegetable.df` should consist of only those cases where the string Fruits and Vegetables—Behavior shows for the object variable `ChildHealthFruitVegetable.df$Topic`.

²¹R is similar to many other programming languages that use two equal signs in the Boolean comparison, `ChildHealth.df$Topic=="Fruits and Vegetables - Behavior"`. The use of `==` (two equal signs with no space between the two) instead of `=` (one equal sign) is not a mistake.

R Input

```
getwd()                      # Working directory
ls()                         # List objects
attach(ChildHealthFruitVegetable.df) # Attach the data
str(ChildHealthFruitVegetable.df)    # Identify structure
head(ChildHealthFruitVegetable.df, n=3) # Show the head
summary(ChildHealthFruitVegetable.df) # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

```
> summary(ChildHealthFruitVegetable.df)      # Summary statistics
   Topic          Data_Value
Length:9184      Min.    :13.5
Class :character 1st Qu.:36.7
Mode  :character Median :40.4
                  Mean   :40.7
                  3rd Qu.:44.7
                  Max.   :67.5
                  NA's   :2605
```

Immediately, notice that `ChildHealth.df` consisted of 35,042 cases as of the time data were obtained. The smaller dataset `ChildHealthFruitVegetable.df` consists of 9184 cases.²²

Although it may be unnecessary, as a standard quality assurance action confirm that the number associated with `ChildHealthFruitVegetable.df$Data_Value` are numeric.

R Input

```
ChildHealthFruitVegetable.df$Data_Value <-
  as.numeric(ChildHealthFruitVegetable.df$Data_Value)
```

R Input

```
length(ChildHealthFruitVegetable.df$Data_Value)
table(is.na(ChildHealthFruitVegetable.df$Data_Value))
```

²²It is often desirable to work with smaller subsets of larger datasets, but this topic is covered in later lessons.

R Output

```
> length(ChildHealthFruitVegetable.df$Data_Value)
[1] 9184
> table(is.na(ChildHealthFruitVegetable.df$Data_Value))

FALSE  TRUE
6579  2605
```

R Input

```
mean(ChildHealthFruitVegetable.df$Data_Value, na.rm=TRUE)
sd(ChildHealthFruitVegetable.df$Data_Value, na.rm=TRUE)
median(ChildHealthFruitVegetable.df$Data_Value, na.rm=TRUE)
summary(ChildHealthFruitVegetable.df$Data_Value)
```

R Output

```
> mean(ChildHealthFruitVegetable.df$Data_Value, na.rm=TRUE)
[1] 40.6619
> sd(ChildHealthFruitVegetable.df$Data_Value, na.rm=TRUE)
[1] 6.40717
> median(ChildHealthFruitVegetable.df$Data_Value, na.rm=TRUE)
[1] 40.4
> summary(ChildHealthFruitVegetable.df$Data_Value)
   Min. 1st Qu. Median      Mean 3rd Qu.      Max.    NA's 
 13.5     36.7    40.4    40.7     44.7    67.5    2605
```

The mean for *Percent of students in grades 9–12 who consume fruit less than 1 time daily* was 40.6619 percent. A density plot will also confirm the distribution pattern for ChildHealthFruitVegetable.df\$Data_Value from among the students who responded to the question about daily consumption of fruit (Fig. 1.6).

R Input

```
par(ask=TRUE)
plot(density(ChildHealthFruitVegetable.df$Data_Value,
na.rm=TRUE),
main="Percent of Students in Grades 9-12 Who Consume Fruit
Less Than 1 Time Daily: Quality Assurance
Density Plot", col="red", lwd=5)
```

```
# This graphic is a quality assurance density plot of the
# object variable ChildHealthFruitVegetable.df$Data_Value,
# with arguments resulting in a thick (lwd=5) red line.
```

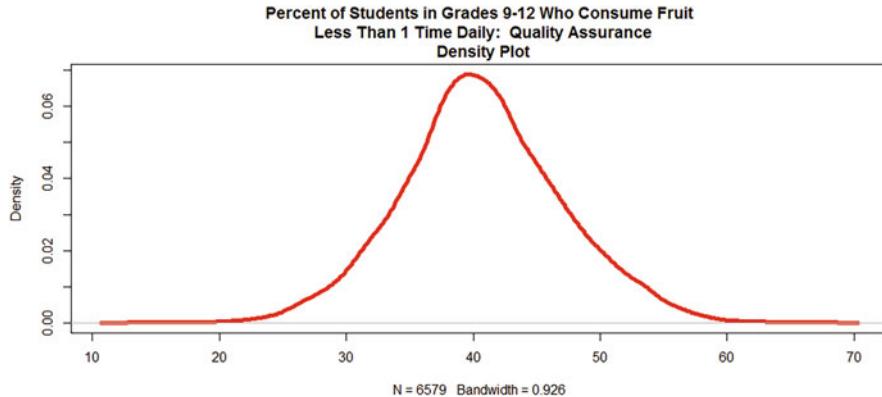


Figure 1.6: Quality assurance of fruit consumption by grade 9–12 students

For those who work in areas of biostatistics, this figure can be interpreted in many ways, given that approximately 40% of all youth in grades 9–12 consume fruit less than one time daily:

- From a perspective specific to promotion of the fruit industry, it is obvious that there is opportunity to increase product sales to a major market segment of children and adolescents.
- From a public health perspective, it is obvious that a vast number of children and adolescents are not enjoying the health benefits of daily fruit consumption.

Regardless of specific perspective, note how online data in the public domain were easily gained, imported into R, put into a more manageable dataset, quickly reviewed for descriptive statistics, and then put into graphical format as a vibrant figure. It should be mentioned, however, that when data are obtained from an Internet-based resource, it is often more convenient in the long term to obtain the data and then download it, importing it into R only after careful offline review. When data are imported directly into R from an online resource, it is too easy to give minimal attention to quality assurance review practices that are better accomplished when working offline.

Saying that offline review of online data has advantages, consider another Internet-based dataset, where the data are selected from a Web site and then downloaded to the declared working directory. For this example, look below at the online data source made available by the United States Department of Energy and the United States Environmental Protection Agency, focusing on fuel economy and related air quality issues resulting from vehicular emissions, as

found in the file Fuel Economy Data for 2000–2017 (<http://www.fueleconomy.gov/feg/download.shtml>).

From this starting point, select the icon showing data for 2008. The year 2008 was purposely selected to provide historical perspective of vehicular emissions data and other related fuel efficiency data before more stringent fuel economy and emissions control regulations were promulgated. Download the file and place it in the appropriate working directory, saving it in .csv file format. While offline, note how this .csv file may only require light editing, if any manipulation is required. Then, use regular R-based functions such as the `utils::read.table()` function to import the .csv file into the current R session.

R Input

```
FuelMPG2008.df <- utils::read.table (file =
  "SmartWayVehicleListMY2008.csv",
  header=TRUE, dec=". ", sep=",")
# Use the utils::read.table() function to import the
# .csv file SmartWayVehicleListMY2008.csv into the
# current R session and place the contents into the
# object FuelMPG2008.df, which is a dataframe that:
# (1) has a header row, (2) uses a period for decimals,
# and (3) uses a comma to separate one field from
# another field.
#
# Look at the header row in the original .csv file and
# note how some column names have a space between two
# separate words (e.g., Hwy MPG). Then, note in the
# dataset FuelMPG2008.df how the space has been
# replaced by a period. The space in the original file
# was automatically replaced by a period as part of the
# R data-import process.
```

R Input

```
getwd()                      # Identify the working directory
ls()                          # List objects
attach(FuelMPG2008.df)       # Attach the data, for later use
str(FuelMPG2008.df)          # Identify structure
head(FuelMPG2008.df, n=3)    # Show the head, 1st 3 cases
summary(FuelMPG2008.df)      # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

```
> str(FuelMPG2008.df)      # Identify structure
'data.frame':   2404 obs. of  16 variables:
 $ Model           : Factor w/ 171 levels "", "ACURA TL", ...
 $ Displ            : num  3.2 3.5 3.5 2.4 2.4 2 2 2 2 2 ...
 $ Cyl              : Factor w/ 5 levels "", "(4 cyl)", "(5 ...
 $ Trans            : Factor w/ 13 levels "", "Auto-4", "Aut ...
 $ Drive             : Factor w/ 3 levels "", "2WD", "4WD": 2 ...
 $ Fuel              : Factor w/ 3 levels "", "CNG", "Gasolin ...
 $ Sales.Area        : Factor w/ 4 levels "", "CA", "FA", "FC" ...
 $ Stnd              : Factor w/ 9 levels "", "B2", "B3", "B4" ...
 $ Underhood.ID      : Factor w/ 129 levels "", "8ADXV02.035 ...
 $ Veh.Class         : Factor w/ 8 levels "", "large car", ...
 $ Air.Pollution.Score: num  7 7 7 6 6 7 7 6 6 7 ...
 $ City.MPG          : int  18 17 18 20 20 21 22 21 22 21 ...
 $ Hwy.MPG           : int  26 26 27 28 28 29 29 29 29 30 ...
 $ Cmb.MPG           : int  21 20 21 23 23 24 25 24 25 24 ...
 $ Greenhouse.Gas.Score: int  6 6 6 7 7 7 7 7 7 7 ...
 $ SmartWay          : Factor w/ 2 levels "", "yes": 2 2 2 2
```

R Output

[Selected output is not shown, to save space.]

```
> summary(FuelMPG2008.df)  # Summary statistics
   City.MPG          Hwy.MPG          Cmb.MPG
Min.   :17.00    Min.   :22.00    Min.   :20.00
1st Qu.:20.00   1st Qu.:27.00   1st Qu.:22.00
Median :21.00   Median :29.00   Median :24.00
Mean   :21.44   Mean   :28.87   Mean   :24.19
3rd Qu.:22.00   3rd Qu.:31.00   3rd Qu.:25.00
Max.   :48.00   Max.   :45.00   Max.   :46.00
NA's    :1766    NA's    :1766    NA's    :1766
```

The data seem to be in proper order and within expected ranges. Generate a simple figure as a graphical quality assurance tool, to further confirm that the dataset is acceptable for later analyses (Fig. 1.7).

R Input

```
par(ask=TRUE)
plot(density(FuelMPG2008.df$Hwy.MPG, na.rm=TRUE),
     main="Highway Miles per Gallon of 2008 Vehicles:
           Quality Assurance Density Plot", col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable FuelMPG2008.df$Hwy.MPG, with arguments
# resulting in a thick (lwd=5) red line.
```

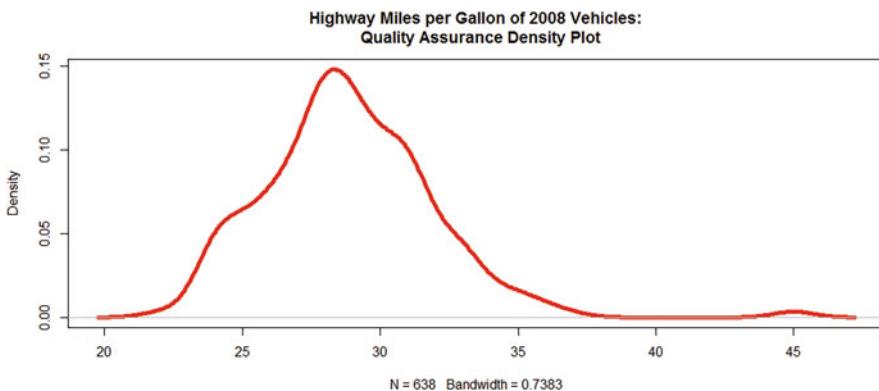


Figure 1.7: Quality assurance of highway MPG

The above example demonstrated how a .csv (comma-separated values) file was downloaded from a public domain online resource, saved into the R session working directory, and then used for R-based analysis and graphics. This example complements the prior demonstration of how R can be used to link directly to an online .csv file and import the data into the current R session. For both situations, online resources are not always available when desired. It is a good management practice to download each .csv file so that a copy of the data is available, offline, when needed.

1.5.6 Import a .csv File of Comma-Separated Values into R by Using Graphical User Interface (GUI) Selections

This lesson and all following lessons in this text focus on the use of syntax when using R. Among the many advantages of this approach, there are three immediate reasons for the use of syntax over any other approach to the use of R:

- The use of written (and saved) R-based syntax allows for documentation of all actions, as there is a historical record of what was attempted.
- The use of written (and saved) R-based syntax allows for replication and reuse of all actions, as syntax that works well for one set of analyses can be used for future analyses, often with only minor alteration.

- The use of written (and saved) R-based syntax allows for debugging, as syntax can be carefully examined for errors or other issues when there are problems that need attention.

When a Graphical User Interface (GUI) menu-type process is used for programming or any other application, there is always the concern that proper documentation and recorded sequence-of-events may be missing if the activity needs to be revisited. Even with this concern about use of a GUI and how it can impact documentation, replication, and debugging, it is necessary to demonstrate that there is an easy way to import data into R by using a quasi-GUI menu selection approach—even though this approach is not recommended. To achieve this aim and use a GUI-type approach to the task of importing data into R:

- Use the `read.csv()` function, from the `utils` package, which is obtained and made available when R is downloaded.
- Use the `file.choose()` function, from the `base` package, which is obtained and made available when R is downloaded.
- Create a temporary holding object (called `X`), that merely holds the data as the GUI process is executed.
- After the selected `.csv` file has been brought into the R session, give the temporary holding object a descriptive name (called `DoeMiniLopRabbit.df`).

Recall that as this GUI-based process is used, visual inspection and selection are involved and, as such, a full set of actions is not recorded. Use care and be sure to make the right selection(s). This selection process may not be an issue for a simple file that has a unique name, but for files that are similar in name (often with only a sequential code differentiating one file from another) this process requires care.

R Input

```
X <- utils::read.csv(base::file.choose())
# Use the mouse/keypad to select the file
# RabbitWeightKg.csv, which is in the
# working directory established at the
# start of this session. Notice how there
# is no record of this action, using syntax.
# Only visual selections are involved.
```

R Input

```
DoeMiniLopRabbit.df <- X
# Create an object with a descriptive name
# (DoeMiniLopRabbit.df) to hold data that
# were first imported into and placed in
# the temporary object (X).
```

R Input

```
getwd()                      # Identify the working directory
ls()                          # List objects
attach(DoeMiniLopRabbit.df)   # Attach the data, for later use
str(DoeMiniLopRabbit.df)      # Identify structure
head(DoeMiniLopRabbit.df, n=3) # Show the head, 1st 3 cases
summary(DoeMiniLopRabbit.df)  # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

```
> str(DoeMiniLopRabbit.df)      # Identify structure
'data.frame': 20 obs. of 2 variables:
 $ Rabbit : Factor w/ 20 levels "DoeMiniLop01",...: 1 2 3 4 5 ...
 $ WeightKg: num 1.96 1.88 2.07 1.87 2.02 ...
> summary(DoeMiniLopRabbit.df)  # Summary statistics
    Rabbit      WeightKg
DoeMiniLop01: 1   Min.   :1.84
DoeMiniLop02: 1   1st Qu.:1.95
DoeMiniLop03: 1   Median  :2.05
DoeMiniLop04: 1   Mean    :2.08
DoeMiniLop05: 1   3rd Qu.:2.22
DoeMiniLop06: 1   Max.    :2.41
(Other)        :14
```

To repeat, the desired .csv file (`RabbitWeightKg.csv`) was selected by using a physical interface, such as a mouse or keypad. The data were originally put into a temporary holding object (`X`). The data in the temporary holding object were then redirected into an object with a far more descriptive name (`DoeMiniLopRabbit.df`).

Finally, create a figure of the numerical object variable (`DoeMiniLopRabbit.df$WeightKg`) to confirm, graphically, that the data are in expected ranges and show an expected distribution pattern. These graphical quality assurance

actions are always necessary and should never be viewed as being redundant (Fig. 1.8).

R Input

```
par(aspect=TRUE)
plot(density(DoeMiniLopRabbit.df$WeightKg, na.rm=TRUE),
      main="Weight (Kg) of Selected Mini Lop Doe Rabbits:
Quality Assurance Density Plot", col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable DoeMiniLopRabbit.df$WeightKg, with
# arguments resulting in a thick (lwd=5) red line.
```

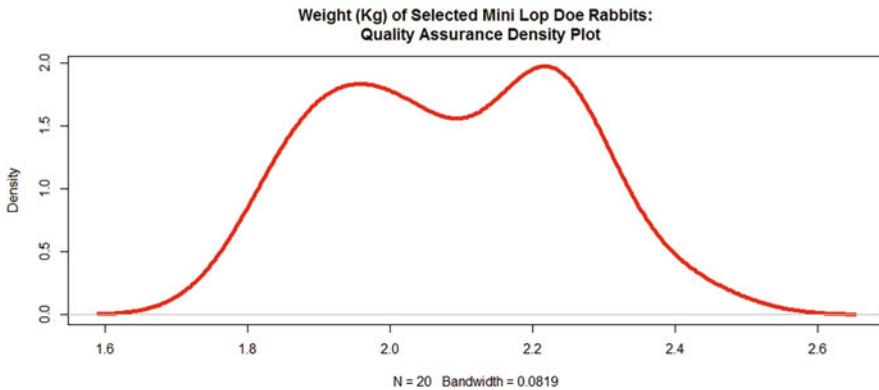


Figure 1.8: Quality assurance of rabbit weights

Once again, it is cautioned that using a GUI selection process does not leave a complete trail of actions, which would be the case if syntax were used. Though it would be remiss to *not* demonstrate this process, avoid using a GUI selection process for data import into R.

1.5.7 Import by Direct Keyboard *On the Fly* Data Entry into R

It is also possible to enter data directly into an active R session, avoiding any attempt to import data saved in an external file. When this action is attempted, data are typed at the R prompt. As always, however, it is best to avoid direct typing at the R prompt and to instead save all syntax (including data entry) in an external text file.

Look below at the way the `read.table(textConnection())` functions are used to enter and collect data for a dataset. Note how there is a header row (as confirmed by using the `header=TRUE` argument) and 20 separate rows of data for two object variables, Subject and SBP (e.g., Systolic Blood Pressure).

Given the difficulty of correct data entry actions, use this process only for small

datasets, if at all. There are other tools, such as spreadsheets, that are easier to use for data entry.

R Input

```
SBPAdultMale.df <- read.table(textConnection("Subject      SBP
SBPAM01      106
SBPAM02      104
SBPAM03      114
SBPAM04      118
SBPAM05      102
SBPAM06      112
SBPAM07      100
SBPAM08      108
SBPAM09      116
SBPAM10      108
SBPAM11      112
SBPAM12      106
SBPAM13      114
SBPAM14      116
SBPAM15      108
SBPAM16      118
SBPAM17      100
SBPAM18      102
SBPAM19      120
SBPAM20      114"), header=TRUE)
```

R Input

```
getwd()                      # Identify the working directory
ls()                          # List objects
attach(SBPAdultMale.df)       # Attach the data, for later use
str(SBPAdultMale.df)          # Identify structure
head(SBPAdultMale.df, n=3)    # Show the head, 1st 3 cases
summary(SBPAdultMale.df)      # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

```
> str(SBPAdultMale.df)        # Identify structure
'data.frame':   20 obs. of  2 variables:
```

```
$ Subject: Factor w/ 20 levels "SBPAM01", "SBPAM02", ... : 1 2 3 4
$ SBP      : int  106 104 114 118 102 112 100 108 116 108 ...
> summary(SBPAdultMale.df)    # Summary statistics
  Subject        SBP
  SBPAM01: 1   Min.   :100
  SBPAM02: 1   1st Qu.:106
  SBPAM03: 1   Median :110
  SBPAM04: 1   Mean    :110
  SBPAM05: 1   3rd Qu.:114
  SBPAM06: 1   Max.    :120
  (Other):14
```

If data entry actions were correct, then the SBP (Systolic Blood Pressure) values should show in the density plot with a mean SBP of about 110. Further, the data distribution pattern should show a minimum SBP of about 100 and a maximum SBP of about 120 (Fig. 1.9).

R Input

```
par(ask=TRUE)
plot(density(SBPAdultMale.df$SBP, na.rm=TRUE),
      main="Systolic Blood Pressure of Selected Adult Males:
Quality Assurance Density Plot", col="red", lwd=5)
# This graphic is a quality assurance density plot of the
# object variable SBPAdultMale.df$SBP, with arguments
# resulting in a thick (lwd=5) red line.
```

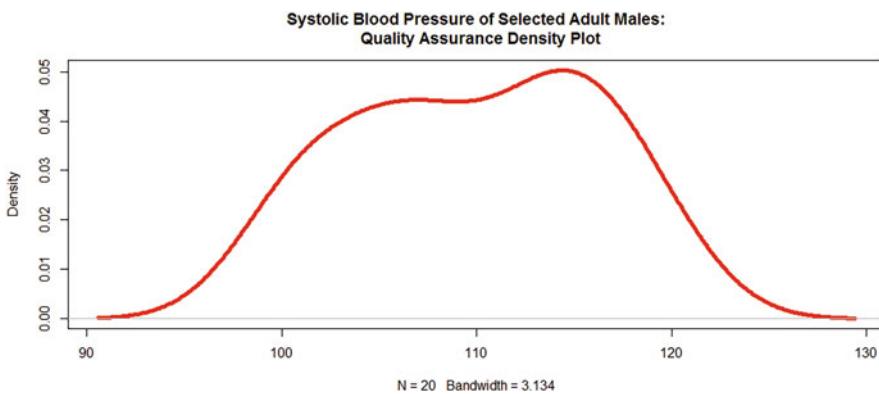


Figure 1.9: Quality assurance of systolic blood pressure for adult males

Spreadsheets are perhaps the most frequently used tool to organize and collect data, whether the data are entered at the keyboard by an assigned data entry specialist or automatically entered by user-initiated completion of a Web-based

form. Yet, it is possible to enter data directly into a R session, in the following two ways:

- Enter data in an external text file and then use some type of copy and paste or redirect action to move the data from the external text file to the R session.
- Enter data directly at the R prompt.

However data are put into electronic format and then imported into R, quality assurance needs to be a constant activity. There are simply far too many opportunities for errors with data entry, data import, and data use. Analyses and graphics will only be as useful as the data are correct, prompting the need for constant attention to quality assurance actions.

1.6 Addendum 1: Efficient Programming with R, Project Workflow, and Good Programming Practices (gpp)

An Internet search of efficient programming, project workflow, and good programming practices will generate a nearly endless list of useful resources. It is far beyond the purpose of this addendum and text to redundantly summarize what others make readily available. Instead, consider a few helpful hints that are mostly general in nature, with a few specifics to the use of R.

Efficient Programming With R Is it anticipated that the current area of inquiry will require 100 lines of syntax, 1000 lines, 10,000 lines, etc.? Is it anticipated that the current area of inquiry will result in text-based outcomes only, text-based outcomes and only a few graphics, text-based outcomes and a large and expanding set of graphics? Even with the power of current computers and disk storage that at times seems nearly limitless (Who would ever thought that a terabyte external storage disk could be purchased for less than \$USD 50.00?), it is still prudent to consider programming efficiency so that the final output demonstrates not only computing efficiency but also human efficiency. If an inquiry results in 1000 lines of syntax that takes many hours to prepare, consider what has been learned and ask if the next similar attempt can be made simpler, requiring fewer lines of syntax and, more importantly, less time for coding. Of course, efficiency is gained with close attention to detail and experience.

Project Workflow When first learning R, it is common to open a R session and to then work interactively at the R prompt, keying something as simple as `X <- c(10, 20, 30)` to create the object X. Then, perhaps the syntax `mean(X)` is keyed at the R prompt to obtain the mean. These simple actions, with analyses attempted in an interactive fashion by keying at the R prompt, are fine at the beginning when R is first attempted. Yet, this is not a desirable approach for any meaningful attempt at statistical analyses using R. Instead, consider

the R-based syntax shown throughout this text and know that this syntax was planned in advance, keyed into some type of text editor, regularly saved, sent to R using selected actions, and tested for expected outcomes.²³ It is well beyond the purpose of this text to offer explicit details on the use of Command Line Interface (CLI) v Integrated Development Environment (IDE) in support of project workflow—again, we all have our preferred method for preparing and executing syntax. The key issue here is that R-based actions do not happen in a haphazard fashion. Instead, project workflow demands preplanning, syntax that is saved and easily reproduced, and regular testing for expected outcomes.

Good Programming Practices (gpp) Look at the R-based syntax presented throughout this text to see examples of good programming practices (gpp):

- A first example is the Housekeeping section that starts each R session. Notice the use of functions such as date(), ls.str(), getwd(), setwd(), sessionInfo(), and other functions in the Housekeeping session. When placed in a text-based file that is regularly saved and available for use and reuse, these many initial functions promote a transparent and reproducible approach to the use of R. Before any data are brought into R and analyses are attempted, a researcher has full knowledge of the R environment.
- Look at the use of descriptive names for dataframes, such as the consistent use of .df to indicate that the object in question is a dataframe: GenEnd.df, BreedMilk.df, SoilYield.df, etc. The use of a .df extension is not required, but by using this convention there is no question but that the object is indeed a dataframe.
- Notice the consistent use of a Object\$Variable naming scheme for objects GenEnd.df\$Endurance, BreedMilk.df\$MilkLb365, SoilYield.df\$BUpperAcre, etc. Although an explicit naming scheme may not be needed, this approach promotes quality assurance.²⁴
- As a final comment somewhat specific to R, notice the Package::Function naming scheme used in this lesson for the use of nearly all functions gained from external R packages. If the ggplot2 package were downloaded and put into action correctly, then it may only be necessary to key ggplot() to use this function. However, the explicit expression ggplot2::ggplot() is a far more efficient, transparent, and reproducible way to invoke this function.

²³It would be meaningless to suggest the *best* text editor—there are many from which to select and we all have our favorite.

²⁴Review the term CamelCase (e.g., CamelCase, camel caps, Pascal case, Dromedary case, medial capitals), a common naming scheme in computer science. This practice of using letters in uppercase and lowercase serves a useful purpose of easily assigning a descriptive name, possibly a long descriptive name, to R-based objects.

There may seem to be no discernible limit to the number and quality of resources associated with this addendum. The key issues here are that R syntax should be efficient, transparent, and reproducible. These three concepts are the central focus of good programming and the desire for continuous quality improvement.

1.7 Addendum 2: Preview of Descriptive Statistics and Graphics Using R

The purpose of this addendum is to provide a limited introduction to the power of R functions as they are used with simple descriptive statistics and accompanying graphics (e.g., figures). To achieve this aim, consider this early addendum as a preview of how R can:

- Generate a theoretical dataset of infant birth weights, with the data consisting of 10,000 random numbers exhibiting a collective mean of 3500 g and a standard deviation of 450 g, or an avoirdupois weight (a system of weights in pounds and ounces still widely used in some English-speaking countries) of approximately mean equals 7-pounds and 11-ounces and standard deviation equals 1-pound and 0-ounces.
- Apply functions used to display data and generate descriptive statistics such as minimum, first quartile, median, mean, third quartile, and maximum.
- Display increasingly detailed and compelling graphics (e.g., figures), used to enhance understanding of the relationship between and among data.

Follow along with the syntax shown below and give special attention to the many comments that follow along after the R-based function and function arguments are deployed. If there is text on the screen from prior use of R, it may be helpful to clear the screen using one of two methods:

- Method 1: Press the Control key and the letter l (lowercase l, not the number 1) at the same time.
- Method 2: Use a mouse or keypad to click the R menu selections Edit and then Clear console.

With the R screen clear of all unnecessary text, follow along with the R-based syntax shown below:

R Input

```
ls()  
# List all objects in the active R session  
  
set.seed(8)
```

```

# Allow reproducible random numbers

BirthWeightGrams <- round(rnorm(10000, mean=3500, sd=450))
# Make an object variable called BirthWeightGrams. Note
# how the name BirthWeightGrams is an example of internal
# self-documentation, providing a ready reference to what
# was measured (e.g., BirthWeight) and the unit of
# measure (e.g., Grams).
#
# R supports the use of = and <- as assignment operators.
# For this text, <- is used for assignment, to avoid any
# possible confusion and incorrect use of == (two equal
# signs with no space between them) since == is used to
# express comparative equivalency, similar to many other
# programming languages.
#
# For this demonstration, the rnorm() function will be
# used to generate a dataset of:
#
#   N ..... 10,000 random numbers
#   Mean ..... 3,500 grams
#   Standard Deviation (sd) .. 450 grams
#
# These values approximate expected birth weights of
# infants at an otherwise unidentified metropolitan
# area, but a few values at either end of the continuum
# may certainly seem extreme for this theoretical dataset
# of randomly-generated numbers.
#
# Note how the round() function was wrapped around the
# rnorm() function so that output would show as whole
# numbers.

```

Far more discussion on descriptive statistics and measures of central tendency appears in later lessons throughout this text. For now, note how English-like terms are used to deploy selected functions. Look also at the way the # symbol is used to express a comment, shown immediately below as a brief description of each function:

R Input

```

head(BirthWeightGrams)      # Show the head, first values
tail(BirthWeightGrams)      # Show the tail, last values

```

```
mean(BirthWeightGrams)      # Mean, arithmetic average
sd(BirthWeightGrams)        # SD, standard deviation
median(BirthWeightGrams)    # Median, midpoint
min(BirthWeightGrams)       # Minimum value
max(BirthWeightGrams)       # Maximum value
summary(BirthWeightGrams)   # Summary descriptive statistics
```

R Output

[Selected output is not shown, to save space.]

```
> summary(BirthWeightGrams) # Summary descriptive statistics
   Min. 1st Qu. Median     Mean 3rd Qu. Max.
   1619    3195    3507    3502    3804    5232
```

Study on the output of these above functions before the graphics (e.g., figures) shown below are attempted. Collectively, these functions should provide a sense of the data, the average birth weight, dispersion of birth weights, etc.²⁵

However, to gain a more complete understanding of the object `BirthWeightGrams`, focus on the different graphics shown below, all based on the use of the `hist()` function. The graphics start out as a simple histogram, with few if any embellishments. By using different R-based arguments, note how increasingly interesting and useful figures are generated, adding color, labels, bold text, larger fonts, etc. In later lessons even greater complexity will be added to the figures, but for now merely observe the potentials that can be achieved by using R for biostatistics (Fig. 1.10).

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
hist(BirthWeightGrams)
hist(BirthWeightGrams,
  breaks=25,           # Granularity of output
  main="Distribution of Infant Birth Weight (g)",
  col="red",
  xlab="Infant Birth Weight (g)",
```

²⁵By itself, the term *average* is somewhat ambiguous given that there are multiple ways to view centrality. Does average refer to *mode*, the most frequently occurring value? Does average refer to *median*, the mid-point? Does average refer to *mean*, which is generally viewed as the arithmetic expression of Sum divided by N? Fortunately, R supports functions and function arguments that easily accommodate these multiple views of central distribution.

```

ylab="Count")
hist(BirthWeightGrams,
  breaks=50,           # Granularity of output
  main="Distribution of Infant Birth Weight (g)",
  col="red",            # Color(s)
  xlab="Infant Birth Weight (g)",
  ylab="Count",
  font.axis=2,          # Make the axis bold
  font.lab=2)           # Make the labels bold

hist(BirthWeightGrams,
  breaks=50,           # Granularity of output
  main="Distribution of Infant Birth Weight (g)",
  col="red",            # Color(s)
  xlab="Infant Birth Weight (g)",
  ylab="Count",
  font.axis=2,          # Make the axis bold
  font.lab=2,           # Make the labels bold
  cex.main=1.75,         # Size - main (title)
  cex.lab=1.25,          # Size - labels
  xlim=c(0,6000),        # Adjust X axis limits
  ylim=c(0,1000))        # Adjust Y axis limits

```

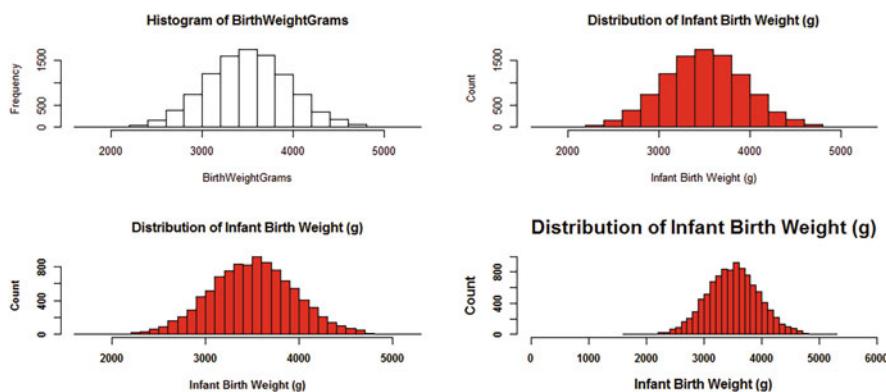


Figure 1.10: Multiple histograms of birth weight

1.8 Addendum 3: R and *Beautiful Graphics*

Simple graphical tools have been used in this lesson to provide some degree of quality assurance that the data for each set of analyses are in expected range and that the measures of central tendency are approximately correct. There are far more sophisticated graphical tools, however, that are supported by R and these tools are used to create *Beautiful Graphics*—a term for graphics that are frequently used to produce figures suitable for professional publication.

Going beyond the immediate purpose of this lesson, look below at the way R, and more specifically the R packages ggplot2, ggthemes, ggmosaic, gridExtra, grid, and scales are used to produce figures of the highest quality. As previously mentioned at the beginning of this lesson, for now merely give attention to process. The actual R-based syntax will be highlighted in far more detail in later lessons.

There are more than a few datasets currently available in this active R session, as can be demonstrated by using either the `ls()` function or the `objects()` function, with the datasets ranging in alphabetical order from `BreedMilk.df` to the temporary object called X.

R Input

```
ls()  
objects()
```

In this addendum, the package `ggplot2` and supporting packages are used to produce a variety of figures using the dataframes currently available. Recall that these packages are external to what is available when R is first downloaded and that it is necessary to actively download these packages, to take advantage of their specialized functionality.

R Input

```
install.packages("ggplot2", dependencies=TRUE)  
library(ggplot2) # Load the ggplot2 package.  
help(package=ggplot2) # Show the information page.  
sessionInfo() # Confirm all attached packages.  
  
install.packages("ggthemes", dependencies=TRUE)  
library(ggthemes) # Load the ggthemes package.  
help(package=ggthemes) # Show the information page.  
sessionInfo() # Confirm all attached packages.  
  
install.packages("ggmosaic", dependencies=TRUE)  
library(ggmosaic) # Load the ggmosaic package.  
help(package=ggmosaic) # Show the information page.  
sessionInfo() # Confirm all attached packages.  
  
install.packages("gridExtra", dependencies=TRUE)  
library(gridExtra) # Load the gridExtra package.  
help(package=gridExtra) # Show the information page.  
sessionInfo() # Confirm all attached packages.
```

```

install.packages("grid", dependencies=TRUE)
library(grid)                      # Load the grid package.
help(package=grid)                 # Show the information page.
sessionInfo()                      # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)                     # Load the scales package.
help(package=scales)                # Show the information page.
sessionInfo()                      # Confirm all attached packages.

```

Using these specialized packages and the R-based functions available from these packages, look at the way increasingly complex figures are prepared, each adding more detail than the preceding figure. To achieve the aim of *Beautiful Graphics*, use the `Sorghum.df` dataset and in progression focus on how sorghum yield (Bushels per Acre) is presented, addressing different factor-type object variables, different graphical displays (e.g., density plot, boxplot), and different ggthemes-based presentations (e.g., `theme_few`, `theme_stata`, etc.). Once again, the syntax will be explained in detail in later lessons. Focus now on the many possibilities available with R (Fig. 1.11).

R Input

```

ggplotSorghum1 <-
ggplot2::ggplot(Sorghum.df,
aes(x=BUpperAcre)) +
geom_density(alpha = 0.5) +
ggtitle("Sorghum Yield -- ggplot Simple") +
xlab("Yield") +
ylab("Density\n") +
theme_few() +           # Review the ggthemes package
theme(legend.position="none")   # No legend
# \n is used to force a line break

ggplotSorghum2 <-
ggplot2::ggplot(Sorghum.df,
aes(x=Management, y=BUpperAcre, fill=Management)) +
geom_boxplot() +
stat_summary(fun.y=mean, geom="point", shape=1, size=6,
col="black") +
# Add a circle to represent the mean, along with the median
# which shows as the solid line
facet_grid(~ as.factor(Year)) +
ggtitle("Sorghum Yield -- ggplot Complex") +

```

```

xlab("Management") +
ylab("Yield") +
scale_y_continuous(labels=scales::comma, limits=c(0,104),
  breaks=scales::pretty_breaks(n = 3)) +
theme_stata() # Review the ggthemes package
theme(legend.position="none") # No legend
# \n is used to force a line break

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotSorghum1,
  ggplotSorghum2, ncol=1)

```

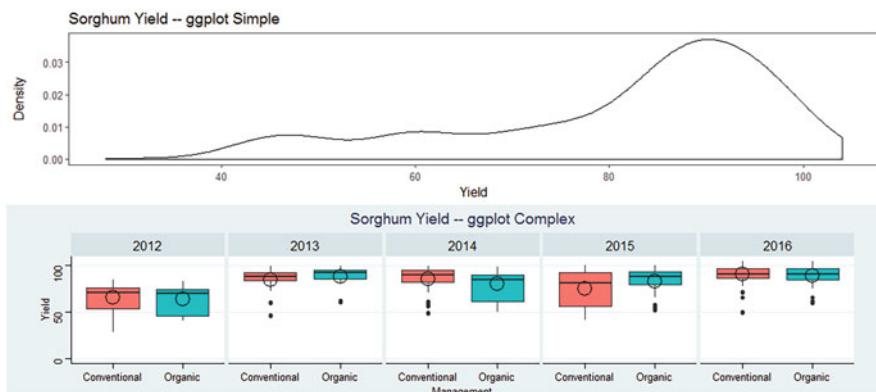


Figure 1.11: ggplot2 demonstration 1—simple to complex

As a final demonstration of the ggplot2 package and how it is used to produce figures acceptable for professional publication, revisit an earlier section in this lesson and review Fuel Economy Data for 2000–2017 (<http://www.fueleconomy.gov/feg/download.shtml>) and the prior resource `FuelMPG2008.df`, which focused on fuel consumption of different cars in 2008, when new regulations about fuel economy were considered. For this last figure, look at the difference in fuel economy (e.g., MPG—Miles per Gallon) by the number of cylinders.

Using this figure only, is it reasonable to say that fuel economy decreases as the number of cylinders increases? In a later lesson on correlation and association, this type of question will be explored more closely, where different statistical tests will be used to provide a more finite answer. Different statistical tests will also be used to address issues such as, from this dataset, fuel economy between city driving v highway driving (e.g., `City.MPG` v `Hwy.MPG`). For now, however, merely focus on the graphical presentation gained by using the `ggplot2::ggplot()` function (Fig. 1.12).

R Input

```

par(ask=TRUE)
ggplot2::ggplot(FuelMPG2008.df,
  aes(x=Cyl, y=Hwy.MPG, fill=Cyl)) +
  geom_boxplot() +
  stat_summary(fun.y=mean, geom="point", shape=1, size=6,
    col="black") +
  # Add a circle to represent the mean, along with the median
  # which shows as the solid line
  ggttitle(
    "Highway Miles per Gallon of 2008 Vehicles by the Number of
    Cylinders: Side-by-Side Box Plots With a Superimposed
    Mean as a Circle\n") +
  xlab("\nCylinders") +
  ylab("Highway Miles per Gallon (MPG)\n") +
  scale_y_continuous(labels=scales::comma, limits=c(20,46),
    breaks=scales::pretty_breaks(n = 5)) +
  theme_economist_white(base_size=12, base_family="sans",
    gray_bg=FALSE, horizontal=TRUE) +
  theme(legend.position="none")    # No legend
# \n is used to force a line break

```

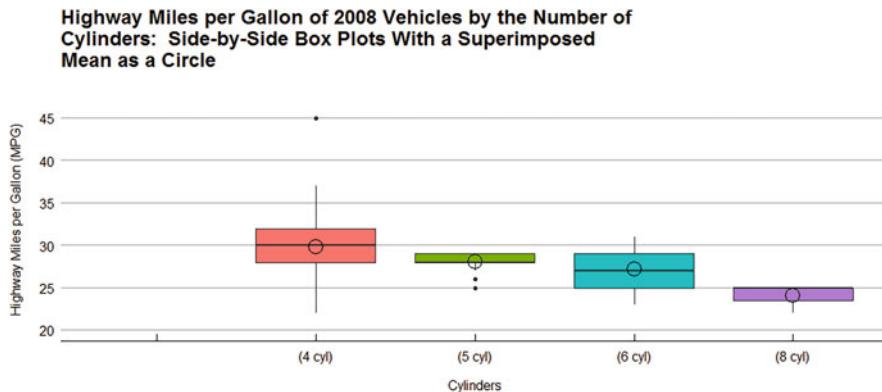


Figure 1.12: ggplot2 demonstration 2—complex

Never underestimate the persuasive power of a well-prepared figure and how graphics gain immediate attention. A properly designed figure will receive far more attention than the attention given to what many see as sterile numeric-based statistical output. Statistical tests are certainly important and are emphasized throughout this text, but again do not overlook the value of quality graphics.

1.9 Addendum 4: Research Designs Used in Biostatistics

Before any attempt is made at listing and briefly describing a few of the many different research designs used in biostatistics, it is perhaps best to first reach common agreement on terms typically used when engaging in the research process. There may be slight degree of nuance in the way each term is applied across various disciplines associated with the biological sciences, but there are still common elements, with the following terms frequently used in biostatistics:

- **Research**—the art and science of seeking answers to questions.
- **Evaluate**—to place worth or value on a phenomenon.
- **Design**—the systematic organization of resources so that a researchable problem is addressed in an efficient manner.
- **Statistic**—the summary description of a variable.
- **Datum and Data**—a value or a fact that can be used for later interpretative purposes. A datum can be numeric and of different scales (e.g., nominal, ordinal, interval, ratio), text (e.g., character), logical (e.g., True or False), etc.²⁶
- **Population**—a broadly defined group.
- **Sample**—a representation (e.g., subset) of a population.
- **Reliability**—the *consistency* (e.g., steadiness) of a test, process, protocol, or other type of attempt used to obtain measurements.
- **Validity**—the finding that a test, process, protocol, or other type attempt used to obtain measurements yields results that are *truthful* (e.g., accurate) and meet identified needs.
- **Representation**—the sample is a true portrayal of the population. A question commonly used for representation is *Are sample statistics in parity with population parameters?*
- **Transparency**—through external publication, internal comments, descriptive variable naming schemes, use of a Code Book, etc., all actions associated with the research process are explicitly detailed in clear language so that a peer with comparable skills has a clear sense of the research process.
- **Replication**—initial results deriving from a research study can be obtained again with a high degree of parity, either by the initial researcher or by others. A replicated study that results in inconsistent results will

²⁶Going back to the Latin derivation of datum and data, bear in mind that the term datum represents singular and the term data represents plural, although it is now common to see the term data used for both a singular and plural context.

receive close scrutiny of the original study as well as the replicated study. The term replication is closely associated with the terms representation, transparency, reliability, and validity.

With this definition of terms and acceptance that these terms apply across the many different areas associated with research in the biological sciences, consider a few of the most frequently used research designs for biostatistics. Some research designs are simple and are for exploratory purposes only. Some research designs are organized to investigate research questions that are complex and address multifaceted issues.

1.9.1 Case Study and Clinical Trial

The Case Study and Clinical Trial is a research process that is often associated with early-on or preliminary investigations, often when the number of participants (e.g., subjects, patients, plots, etc.) is limited in number and representation. Then, based on results, the case study or clinical trial may be expanded in either scope and complexity, sample size, and greater representation of the sample to the population. It is common for a case study or clinical trial to focus on investigations involving prototype devices, feed or drugs in the product development pipeline, or exploratory protocols.

1.9.2 Pretest–Posttest for One Group

The Pretest–Posttest for One Group is a fairly simple approach to the research process where one identified group of subjects receives a pretest for a specific area of interest, a treatment of some type is applied, and a posttest is used as an ending activity. It is assumed, to some degree of acceptance, that the change between pretest and posttest, if any, is attributed to the treatment. This research process is often viewed as an early-on research endeavor that is used for exploratory purposes, to look for general trends, or to practice with different treatment and measurement protocols.

1.9.3 Pretest–Posttest for Control Group

The Pretest–Posttest for Control Group is a somewhat more ambitious approach, where there are typically two groups, a control group and an experimental group. The experimental group receives a pretest, a treatment, and a posttest. The control group receives a pretest, **no** treatment, and a posttest. Change between pretest and posttest can now be differentiated between, and among, those subjects who received the treatment and those who did not receive the treatment, allowing for some degree of interpretation of impact due to the treatment.

1.9.4 Posttest Only for Control Group

The Posttest Only for Control Group is a research process where there are typically two groups, a control group and an experimental group. There is **no** administration of a pretest. The experimental group receives a treatment and a posttest. The control group receives a posttest, only. This approach is used to address the possibility that the pretest, as an experience, possibly impacts posttest results. Differences between the control group and the experimental group, if any, are attributed to the treatment.

1.9.5 Fixed Group Comparative Analysis of a Single Factor

The Fixed Group Comparative Analysis of a Single Factor is a research process where one factor (e.g., Breed: Holstein or Guernsey; Region: North, South, East, or West; Soil Type: Clay, Sand, or Silt) and differences between and among breakout groups is the focus of inquiry for a defined measurement (e.g., dairy cattle milk production per lactation, Labrador Retrievers and weight at weaning, bushel per acre yields of grain sorghum plots, etc.).

1.9.6 Factorial Data Organization of Multiple Independent Variables

The Factorial Data Organization of Multiple Independent Variables is a research process where there are two or more factors (e.g., Gender and Race, Breed and Added Mineral Supplements, etc.) and one or more measurable variables associated with the subjects (e.g., Systolic Blood Pressure, dressing percentage after slaughter, etc.). Factorial designs are used in an attempt to promote the efficient use of resources, in that multiple analyses are possible as opposed to singular-by-singular attempts at experimentation. A factorial approach to the research process allows not only for inquiries at the univariate level of comparison, but it is now possible to investigate the influence of multiple factors and possible interactions between and among the factors.

1.9.6.1 Goodness of Fit (e.g., Chi-Square)

With a Goodness of Fit or Chi-Square (e.g., X^2) approach to factorial inquiries, it is possible to determine if sample data are in parity with the expected distribution from a certain population. A Goodness of Fit research approach might include something as simple as investigations into Rhode Island Red poultry flocks of approximately equal size and the monthly production of eggs of each egg size category (e.g., Peewee, Small, Medium, Large, Extra Large, Jumbo) by confinement type (e.g., battery cage confinement, cage-free with access to fenced-in pasture, cage-free but contained inside a structure). Are the observed

counts (e.g., number of eggs of each egg size by confinement type) in parity with expected counts, with expectations often based on a theoretical framework or set of assumptions?

1.9.6.2 Comparison of Group Means (e.g., Analysis of Variance)

With a Comparison of Group Means or Analysis of Variance (ANOVA), it is possible to compare group means using a mean comparison procedure. This concept is addressed in later lessons, but for now it is best to merely mention that Tukey's HSD (Honestly Significant Difference) test and Scheffé's *post-hoc* test are used to determine if differences in means between and among breakout groups are statistically significant at a declared level of significance (e.g., 0.05, 0.01, 0.001) or if the differences are due only to chance.

1.9.7 Correlation, Association, Regression, Likelihood, and Prediction

The Correlation, Association, Regression, Likelihood, and Prediction research process is based on the common phrase *Past behavior is the best predictor of future behavior*. In simple form, Variable X is compared to Variable Y to see if there is any degree of correlation (e.g., association) between the two. If an association is found, then ultimately constructs such as regression, likelihood, probability, etc., are used to build a prediction equation. By knowing a value for X, it is possible to predict with some degree of assurance the value for Y. The important thing to recall with correlation designs is that there may be an association between Variable X and Variable Y, but that does not in any way suggest that Variable X caused Variable Y or that Variable Y caused Variable X. Correlation does not imply or suggest causation—only association.

Whether a selected design is simple or complex, give special attention to the notion that good research in the biological sciences promotes the discovery of outcomes that should ultimately improve the human condition. Following along with that thought, recall that there are many other resources that go into intricate depth on research design and the type of questions addressed by those designs. Also, consider how the efficient use of resources and unavoidable impediments often demand real-world research design variations that may impact outcomes, in contrast to theoretical research designs where the influence of extraneous phenomena are controlled. With this concern, always consider the need for transparency and how transparency promotes replication, and from replication—acceptance of results by the professional community.

1.10 Prepare to Exit, Save, and Later Retrieve This R Session

Use the following set of actions to exit from the current R session.

R Input

```
getwd()           # Identify the current working directory.  
ls()              # List all objects in the working  
                  # directory.  
ls.str()          # List all objects, with finite detail.  
list.files()      # List files at the PC directory.  
  
save.image("RLessonIntroductionBiostatistics.rdata")  
  
getwd()           # Identify the current working directory.  
ls()              # List all objects in the working  
                  # directory.  
ls.str()          # List all objects, with finite detail.  
list.files()      # List files at the PC directory.  
  
alarm()           # Alarm, notice of upcoming action.  
q()               # Quit this session.  
                  # Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: **File** and then **Load Workspace**. Otherwise, use the load() function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use a R script file (typically saved as a .txt text file) and recreate the analyses and graphics, provided the data files remain available.

1.11 External Data and/or Data Resources Used in This Lesson

The publisher's Web site associated with this text includes the following files, presented in .csv, .txt, and .xlsx file formats.

- GenderEndurance.csv
- BreedMilkLb365.txt
- YearSoilTypeCropRainYieldBushelsPerAcreNoHeader.txt
- WellnessInventory.txt

- Sorghum2012to2016.xlsx
- SmartWayVehicleListMY2008.csv
- RabbitWeightKg.csv

Use these files to practice and replicate the outcomes used in this lesson.

The following external resources should also be accessed, to obtain data referenced and used in this lesson.

- <https://chronicdata.cdc.gov/api/views/vba9-s8jp/rows.csv>
- <http://www.fueleconomy.gov/feg/download.shtml>

Use the data gained from these external resources to practice and replicate the outcomes used in this lesson.

All other data were enumerated directly in the R session, using functions such as `round(rnorm())` and `read.table(textConnection())`.



Chapter 2

Data Exploration, Descriptive Statistics, and Measures of Central Tendency

Abstract

This lesson provides a demonstration of descriptive statistics, measures of central tendency, and graphical presentation of data, which are essential before any inferential statistical analyses are conducted. Initial efforts should be placed on data exploration, and specifically, the use of descriptive statistics and measures of central tendency (e.g., mode, median, mean, standard deviation, etc.). A complete summary of descriptive statistics is presented in this lesson, both for factor-type object variables and numeric object variables of an interval or continuous nature. An initial summary of graphical presentations available through R is provided, with emphasis on fully-embellished, publishable quality graphics deferred until later lessons.

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_2) contains supplementary material, which is available to authorized users.

Keywords: Barplot, Boxplot (box-and-whiskers plot), Boxplot statistics, Data exploration, Density plot, Descriptive statistics, Dotchart, Histogram, Interquartile range (IQR), Length, Maximum, Maximum location, Mean, Measures of central tendency, Median, Minimum, Minimum location, Mode, Quantile-quantile (Q-Q, QQ) plot, Quartiles, Range, Scatter plot, Sort, Standard deviation, Stripchart, Sum, Summary, Trimmed mean, Tukey's five number summary, Variance, Violin plot, Winsor mean

2.1 Background

The purpose of this lesson is to demonstrate how R is used to explore data, to provide descriptive statistics, and to generate measures of central tendency. It can be said with confidence that all statistical analyses, regardless of whether the data are parametric or nonparametric, whether the research design is simple or whether the analyses are for initial use or for eventual publication, etc., call for the topics covered in this lesson. Researchers must know their data. The ability to explore data, the ability to generate descriptive statistics, and the ability to produce measures of central tendency are essential skills when working with data, with data, and there are exceptionally good resources for these tasks, using R.

2.1.1 Description of the Data

There is one dataset used in this lesson, and the data are organized in long (e.g., stacked) format. The data, in long format, generally follow the expected format used among researchers in biostatistics.

- This lesson on descriptive statistics and measures of central tendency is taken from a study that was conducted at a large high school in Florida, as part of a general investigation of wellness and student health. The dataset for this lesson is fairly small ($N = 61$ subjects and five object variables) and represents only a small part of a much larger dataset, larger in terms of more subjects and larger in terms of more variables.
- For this lesson, the data were gained by a school nurse who weighed the 61 students enrolled in Computer Programming III (Course Number 0201320), a Computer Science Education course offered to Grade 12 (e.g., high school seniors, usually 17–18 years old) students. Other biometric measures were also obtained, such as blood pressure and height, but weight is the focus of this lesson. All other biometric measures are excluded from the dataset associated with this lesson.
- Weight was measured using a reliable and valid medical scale and recorded in pounds, using standard rules for rounding. As the principal investiga-

tor, the school nurse is naturally concerned with overall trends as well as individual measures. What was the average weight? What was the lowest weight? What was the highest weight? What was the variance in weight? Were there any trends that need attention, either for immediate purposes or in the future? With proper analysis, this information could be used, in part, as the basis for informed decision-making on wellness, food selections in the cafeteria, recommendations for physical education classes and individual exercise programs, etc.

Going back to the purpose of this lesson, follow along with the use of a small sample of only 61 subjects and how data exploration, descriptive statistics, and measures of central tendency have value on their own and also as indicators for the use of other statistical tests. Quite often, when examining data and between and among data, it is useful to offer a general view of the data. Saying this, consider the data associated with this lesson.^{1,2} It would be more than somewhat useful to know:

- How many students were enrolled in the class and are eligible to have their weights measured?
- How many students had their weights measured?
- How many students were not weighed, either because they were unavailable at the time of data collection or because either they or their parents declined participation in the study?
- Are the data representative of the overall population? Are the data representative of all Grade 12 students at this school, are the data representative of all Grade 12 students in the general community, etc.?
- What is the average weight and are there multiple definitions of the term average? If there are multiple definitions for the term average, when is it appropriate to use one view of the term average (e.g., mean, median, or mode) but not the other(s)?
- Did most weights cluster around the average weight or was there a wide degree of variance (e.g., spread, dispersion, etc.) in weights?
- Were there any weights that seem to be exceptionally out-of-range (e.g., outliers), demanding specific attention for these observed weights?

¹When reviewing these measures of central tendency, know that with a perfect (e.g., theoretical) distribution of values showing in what is commonly called a bell-shaped curve, all three measures for average (e.g., mode, median, and mean) would be equivalent, but this level of theoretical perfection is rarely, if ever, achieved.

²In advance, consider how an oddity of R is that the mode() function has nothing to do with measures of central tendency, but there are from external packages that provide mode as an average.

- Were there any weights that seem to be illogical, perhaps due to accidental data entry of alphabetical characters, perhaps due to wildly unexpected values, or perhaps due to similar errors in the construction of an object variable that has otherwise been declared as a vector of numeric values?
- What was the range of weights, from the lowest (e.g., minimum) weight to the highest (e.g., maximum) weight?
- Do the weights display normal distribution, approximating a bell-shaped curve, or is the distribution skewed and, if so, how? Are weights skewed to the left or are weights skewed to the right?

The following listing identifies a general series of descriptive statistics that are often the first focus for data exploration:

- Measures of central tendency, or representation of the average:
 - Mode: most frequent measure.
 - Median: mid-point of an array of measures.
 - Mean: arithmetic average (Sum/N).
- Measures of dispersion, spread, or values away from the average:
 - Variance: the sum of squared deviations from the mean.
 - SD or sd: the standard deviation or the square root of the variance.
 - Range: the spread from the lowest measure to the highest measure.

It is common to present these statistics early in the research process to give the reader a general view of the data. It is also highly desirable to provide graphical figures of these statistics that visually represent trends.

This lesson has been designed as a demonstration of how R can be used to explore data, and from this initial inquiry provide descriptive statistics and measures of central tendency. The emphasis will be on the use of functions found in the R packages obtained when R is first downloaded. There is also a brief demonstration on the use of functions gained from external R packages. Complementary graphical representations are also provided that add additional insight into the data.

Accordingly, this lesson should provide a fairly detailed introduction to the practice of data exploration using R, and from this practice generate a full set of descriptive statistics and measures of central tendency. This topic is of special importance since nearly each statistical analysis associated with parametric data (e.g., the use of interval or ratio data for tests such as Student's t-Test for Independent Samples, Oneway Analysis of Variance, etc.) and nonparametric data (e.g., the use of nominal and ordinal data for tests such as Mann-Whitney

U Test, Kruskal-Wallis Analysis of Variance, etc.) begins with data exploration, descriptive statistics, and measures of central tendency.

2.1.2 Null Hypothesis

There are no inferential analyses associated with this lesson. Therefore, a Null Hypothesis is not provided. All analyses are descriptive (e.g., describe the data), not inferential (e.g., allow an inference or judgment about differences between groups, association between object variables, etc.).

2.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

The data for this lesson are from a much larger dataset. The complete dataset was originally prepared in Gnumeric, an open source spreadsheet. The data are now in ASCII format and they are separated by commas. The data are not separated by tabs or spaces.

Eventually, the data were placed on an external hard drive (the F drive, for this lesson) in a directory marked as `R_BiostatisticsIntroduction`. All analyses and presentations start here. From this starting point, note how R is set to work in the appropriate directory and then how the `read.table()` function is used to read in the .csv (e.g., comma-separated values) file that contains the data.

R Input

```
#####
# Housekeeping                                     Use for All Analyses #
#####
date()          # Current system time and date.
Sys.time()      # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls()            # List all objects in the working
                # directory.
rm(list = ls()) # CAUTION: Remove all files in the
                # working directory. If this
                # action is not desired, use rm()
                # one-by-one to remove the objects
                # that are not needed.
ls.str()        # List all objects with finite detail.
getwd()         # Identify the current working directory.
```

```

setwd("F:/R_BiostatisticsIntroduction")
  # Set to a new working directory.
  # Note the single forward slash and double
  # quotes.
  # This new directory should be the directory
  # where the data file is located, otherwise
  # the data file will not be found.
getwd()          # Confirm the working directory.
list.files()     # List files at the PC directory.
.libPaths()      # Library pathname.
.Library         # Library pathname.
sessionInfo()   # R version, locale, and packages.
search()         # Attached packages and objects.
searchpaths()    # Attached packages and objects.
#####
#####
```

The Housekeeping template shown above, with occasional changes, is used throughout this text. The purpose of this collection of R functions in one convenient beginning section is to be sure that the working environment, selected directory, etc., are as desired.

Create an object called `CPIIIsecLbsGen.df`. The object `CPIIIsecLbsGen.df` will be a dataframe, as indicated by the enumerated `.df` extension to the object name. This object will represent the output of applying the `read.table()` function against the comma-separated values file called `CPIIISectionLbsGender.csv`. Note the arguments used with the `read.table()` function, showing that there is a header with descriptive variable names (`header = TRUE`) and that the separator between fields is a comma (`sep = ","`).

R Input

```

CPIIIsecLbsGen.df <- read.table (file =
  "CPIIISectionLbsGender.csv",
  header = TRUE, sep = ",") # Import the .csv file.

getwd()          # Identify the working directory
ls()             # List objects
attach(CPIIIsecLbsGen.df) # Attach the data, for later use
str(CPIIIsecLbsGen.df)   # Identify structure
nrow(CPIIIsecLbsGen.df) # List the number of rows
ncol(CPIIIsecLbsGen.df) # List the number of columns
dim(CPIIIsecLbsGen.df)  # Dimensions of the data frame
names(CPIIIsecLbsGen.df) # Identify names
```

```
colnames(CPIIIsecLbsGen.df)      # Show column names  
rownames(CPIIIsecLbsGen.df)      # Show row names  
head(CPIIIsecLbsGen.df)          # Show the head  
tail(CPIIIsecLbsGen.df)          # Show the tail  
CPIIIsecLbsGen.df                # Show the entire dataframe  
summary(CPIIIsecLbsGen.df)       # Summary statistics
```

2.3 Organize the Data and Display the Code Book

The dataframe `CPIIIsecLbsGen.df` is fairly simple and very little, if anything, needs to be done to organize the data. This lesson was designed to serve as an easy-to-follow confidence-building introduction to R, so a simple dataset was selected for this lesson.

For this simple lesson, the `class()`, `str()`, and `duplicated()` functions will be sufficient first steps to be sure that data are organized as desired, going along with the syntax immediately above.³

R Input

```
class(CPIIIsecLbsGen.df)          # Class
```

R Output

```
[1] "data.frame"
```

R Input

```
str(CPIIIsecLbsGen.df)            # Structure
```

R Output

```
'data.frame': 61 obs. of 5 variables:  
 $ StudentID: Factor w/ 61 levels "N2224","N3374",...: 35 24 58  
 $ Course    : Factor w/ 1 level "CPIII0201320": 1 1 1 1 1 ...
```

³All syntax required for replication of this lesson is presented. Most screen prints generated by the syntax in the main body of the lesson are also presented, with only a few screen prints excluded from presentation. This practice also applies to the figures. The syntax for all figures is presented throughout this lesson, but the output of this graphically-focused syntax is sometimes excluded to keep this lesson at a reasonable length.

```
$ Section   : Factor w/ 2 levels "AM","PM": 1 1 1 1 1 1 ...
$ Lbs       : int 123 120 137 127 129 124 114 114 112 110 ...
$ Gender    : Factor w/ 2 levels "Female","Male": 1 1 1 1 ...
```

R Input

```
duplicated(CPIISecLbsGen.df$StudentID) # Duplicates
# DataFrame$ObjectName notation
```

R Output

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[10] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[19] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[28] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[46] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[55] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

The class and structure for each object seems to be correct and there are no duplicate subjects in the dataset. Saying this, a Code Book will help with future understanding of this dataset, even if the data currently seem simple and obvious.

R Input

```
#####
# Code Book for CPIISecLbsGen.df
#####
#
# StudentID ..... Factor (e.g., nominal)
#           A unique ID assigned at the county-level #
#
# Course ..... Factor (e.g., nominal)
#           Computer Programming III - Code 0201320 #
#
# Section ..... Factor (e.g., nominal)
#           AM (Morning) or PM (Afternoon) class #
#
# Lbs ..... Numeric (e.g., interval)
#           Weight (Lbs) of Grade 12 students, most are #
#                           17 to 18 years #
```

```
#  
# Gender .....Factor (e.g., nominal) #  
#                                         Female and Male #  
#####
```

Labels and recoding of individual object variables are not needed for this simple dataset. However, these actions will be seen in future lessons that include more complex datasets. Again, small confidence-building activities with easy-to-follow examples are used at the beginning of this set of lessons, with more complexity introduced gradually.

2.4 Conduct a Visual Data Check Using Graphics (e.g., Figures)

As desirable as numeric descriptive statistics and measures of central tendency may be, and are therefore often our first thought, to have a full understanding of the data it is necessary to generate graphics to actually see how data are organized. Graphics provide an essential complement to our understanding of the data. In later lessons, other graphics will be demonstrated, but for initial purposes the graphical functions of primary interest are `hist()`, `plot()` and `plot(density())`, `stripchart()`, `dotchart()`, and `qqnorm()`. Many arguments are available to embellish these graphical figures, but for now the figures will be prepared in simple simple format.

The `par(ask=TRUE)` function and argument are used to freeze the presentation on the screen, one figure at a time. Note how the top line of the figure, under File—Save as, provides a variety of graphical formats to save each figure, presented in the following order: Metafile, Postscript, PDF, PNG, BMP, TIFF, and JPEG.⁴

To save space and to also provide a convenient side-by-side view for common figures, look at the way multiple figures are put into one common figure using the `par(mfrow=c())` functions. This technique is especially useful and should be considered not only for exploratory figures but also for final output, when appropriate.

R Input

```
par(ask=TRUE)  
par(mfrow=c(2,3)) # 6 figures into a 2 row by 3 column grid
```

⁴It is also possible to perform a simple copy and paste against each graphical image or to use R syntax to save a graphical image by using R syntax. These actions are shown in later lessons.

```
hist(CPIIIsecLbsGen.df$Lbs, main="Weight (Lbs): Histogram")
plot(CPIIIsecLbsGen.df$Lbs, main="Weight (Lbs): Plot")
plot(density(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE), # na.rm=TRUE
     main="Weight (Lbs): Density Plot") # missing data
boxplot(CPIIIsecLbsGen.df$Lbs,
        main="Weight (Lbs): Box Plot")
stripchart(CPIIIsecLbsGen.df$Lbs,
           main="Weight (Lbs): Stripchart") # Stripchart
qqnorm(CPIIIsecLbsGen.df$Lbs, main="Weight (Lbs): Q-Q Plot")
# Common figures for a numeric-type object variable
```

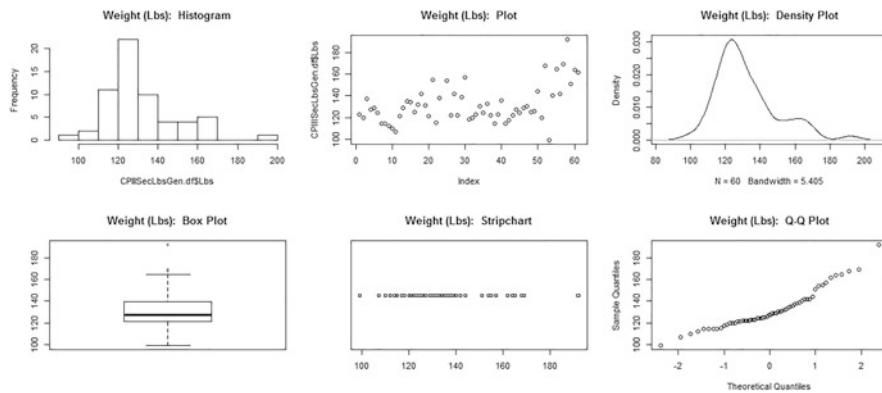


Figure 2.1: Multiple visualization of weight

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
barplot(table(CPIIIsecLbsGen.df$Section),
        main="Section: Barplot Frequency Distribution",
        col=c("black", "red"), ylim=c(0,40)) # Alter color - Y scale
barplot(table(CPIIIsecLbsGen.df$Gender),
        main="Gender: Barplot Frequency Distribution",
        col=c("black", "red"), ylim=c(0,40)) # Alter color - Y scale
# Common figures for a factor-type object variable
```

As an early introduction to the `ggplot2::ggplot()` function and how it is used to produce breakout details in graphical format, using facets, look at the density of the numeric object variable Lbs for each of the two factor-type object variables (e.g., Section and Gender). Remember that use of the `ggplot2::ggplot()` function for access to multiple external packages. Typically, select the most local site for where the external packages are located and be patient, waiting for all packages to download before attempting their use (Figs. 2.1 and 2.2).

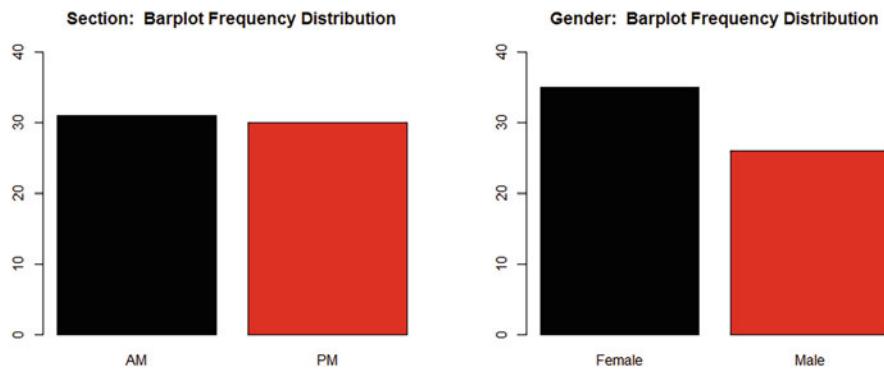


Figure 2.2: Bar plots of section and gender

R Input

```
install.packages("ggplot2", dependencies=TRUE)
library(ggplot2)                      # Load the ggplot2 package.
help(package=ggplot2)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

install.packages("ggthemes", dependencies=TRUE)
library(ggthemes)                     # Load the ggthemes package.
help(package=ggthemes)                # Show the information page.
sessionInfo()                         # Confirm all attached packages.

install.packages("ggmosaic", dependencies=TRUE)
library(ggmosaic)                    # Load the ggmosaic package.
help(package=ggmosaic)               # Show the information page.
sessionInfo()                         # Confirm all attached packages.

install.packages("gridExtra", dependencies=TRUE)
library(gridExtra)                   # Load the gridExtra package.
help(package=gridExtra)              # Show the information page.
sessionInfo()                         # Confirm all attached packages.

install.packages("grid", dependencies=TRUE)
library(grid)                          # Load the grid package.
help(package=grid)                   # Show the information page.
sessionInfo()                         # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)                        # Load the scales package.
help(package=scales)                 # Show the information page.
```

```
sessionInfo() # Confirm all attached packages.
```

R Input

```
# Section (two breakouts) by Lbs
DensityFacetSectionLbs <-
ggplot2::ggplot(CPIIIsecLbsGen.df,
aes(x=Lbs)) +
geom_density(col="red", lwd=2) +
facet_grid(. ~ Section) +
ggtitle("Section by Weight (Lbs): Facet - ggplot\n") +
labs(x = "\nWeight (Lbs)", y = "Density\n") +
scale_x_continuous(labels=scales::comma, limits=c(0,200),
breaks=seq(0,200, by=25)) +
theme_bw()
# Generate the figure DensityFacetSectionLbs, but it will
# not show until using the gridExtra::grid.arrange()
# function.

# Gender (two breakouts) by Lbs
DensityFacetGenderLbs <-
ggplot2::ggplot(CPIIIsecLbsGen.df,
aes(x=Lbs)) +
geom_density(col="red", lwd=2) +
facet_grid(. ~ Gender) +
ggtitle("Gender by Weight (Lbs): Facet - ggplot\n") +
labs(x = "\nWeight (Lbs)", y = "Density\n") +
scale_x_continuous(labels=scales::comma, limits=c(0,200),
breaks=seq(0,200, by=25)) +
theme_bw()

# Section (two breakouts) by Lbs
BoxplotSectionLbs <-
ggplot(CPIIIsecLbsGen.df,
aes(x=Section, y=Lbs, fill=Section)) +
geom_boxplot() +
ggtitle("Section by Weight (Lbs): Boxplot\n") +
labs(x = "\nSection", y = "Weight (Lbs)\n") +
scale_y_continuous(labels=scales::comma, limits=c(100,200),
breaks=seq(100,200, by=25)) +
theme_bw()
```

```
# Gender (two breakouts) by Lbs
BoxplotGenderLbs <-
ggplot(CPIISecLbsGen.df,
aes(x=Gender, y=Lbs, fill=Gender)) +
geom_boxplot() +
ggttitle("Gender by Weight (Lbs): Boxplot\n") +
labs(x = "\nGender", y = "Weight (Lbs)\n") +
scale_y_continuous(labels=scales::comma, limits=c(100,200),
breaks=seq(100,200, by=25)) +
theme_bw()
```

Now that the figures (e.g., `DensityFacetSectionLbs`, `DensityFacetGenderLbs`, `BoxplotSectionLbs`, and `BoxplotGenderLbs`) have been generated as separate objects, put these four objects into one common figure, using the `gridExtra::grid.arrange()` function (Fig. 2.3).

R Input

```
gridExtra::grid.arrange(
  DensityFacetSectionLbs,
  DensityFacetGenderLbs,
  BoxplotSectionLbs,
  BoxplotGenderLbs, ncol=2)
```

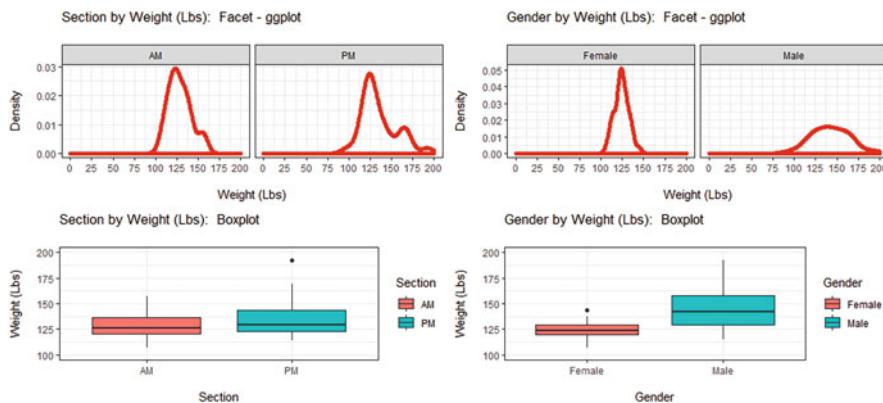


Figure 2.3: Multiple visualizations of weight by section and gender

By using the `gridExtra::grid.arrange()` function to put selected ggplot-based objects into one common figure, it is fairly easy to make meaningful side-by-side comparisons of graphical output. This data exploration technique increases the ability to make meaningful comparisons and supports better decision-making on how to continue with analyses.

These initial graphics are simple and currently have only a few embellishments. They only serve as a first guide to general trends in data organization. Embellishments to the graphics will be introduced in later lessons by demonstrating the many arguments used to present titles, prepare text and lines in bold and color, etc.

2.5 Descriptive Statistics for Initial Analysis of the Data

When R is first downloaded, before any external packages are obtained, there is a large set of functions that can be used to calculate a variety of descriptive statistics and measures of central tendency, such as `length()`, `is.na()`, `complete.cases()`, `summary()`, `mean()`, `sd()`, `var()`, `median()`, etc. A glaring omission is that the `mode()` function does not determine the most frequently occurring value (e.g., mode) but instead provides information on the storage mode for a R-based object. A specialized function, found in an external R-based package will be used to calculate mode, when mode is viewed as one of three representations of average: mode, median, and mean.

Be sure to notice the `DataFrame$Object` notation or `CPIISecLbsGen.df$Lbs` in this case. This type of specificity may at first seem verbose, but it is a desirable quality assurance practice and provides protection against unintended naming outcomes.^{5,6}

Accommodations are needed for `CPIISecLbsGen.df$Lbs`, due to missing data. This concern, and the best way to work with missing data, needs to be addressed early on in the research process, which is why the dataset used for this lesson was selected, given that there is one missing datum for the object variable `CPIISecLbsGen.df$Lbs`.

R Input

```
length(CPIISecLbsGen.df$Lbs)
# Length or N of a vector
```

R Output

```
[1] 61
```

⁵Multiple terms are used for this construct—Quality Assurance, quality assurance, and QA are common and are used interchangeably throughout this text.

⁶To learn more about the nature of each R function, use the built-in help features found in R. At the R prompt key `help(function.name)`, such as `help(length)`, `help(summary)`, `help(mean)`, etc., to learn the exact nature of each R function.

R Input

```
table(is.na(CPIISecLbsGen.df$Lbs))
# Returns TRUE if indexed value is missing (e.g., NA) and
# FALSE if indexed value is not missing
```

R Output

FALSE	TRUE
60	1

R Input

```
table(complete.cases(CPIISecLbsGen.df$Lbs))
# Returns TRUE if indexed value is not missing (e.g., NA)
# and FALSE if indexed value is missing
```

R Output

FALSE	TRUE
1	60

R Input

```
summary(CPIISecLbsGen.df$Lbs)
# Descriptive statistics, including NAs if any
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
99	121	127	132	139	192	1

Although there are many functions associated with descriptive statistics and measures of central tendency, output from the `summary()` function is often a first choice in an effort to make judgment on data organization and quality assurance issues.

Other functions for descriptive statistics and measures of central tendency are demonstrated below. Again, notice how missing values needed to be accommodated.

R Input

```
mean(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)
# Mean or arithmetic average
```

R Output

```
[1] 131.733
```

R Input

```
sd(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)
# Standard Deviation
```

R Output

```
[1] 17.5894
```

R Input

```
var(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)
# Variance
```

R Output

```
[1] 309.385
```

R Input

```
median(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)
# Median or midpoint
```

R Output

```
[1] 127
```

R Input

```
install.packages("modes", dependencies=TRUE)
library(modes) # Load the modes package.
```

```
help(package=modes)          # Show the information page.  
sessionInfo()                # Confirm all attached packages.
```

R Input

```
modes::modes(CPIIIsecLbsGen.df$Lbs, type=1)  
# Mode, or the most frequent value  
# Note how this distribution is bimodal
```

R Output

```
[,1] [,2]  
Value 114 122  
Length 4 4
```

R Input

```
range(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)  
# Range, minimum and maximum
```

R Output

```
[1] 99 192
```

R Input

```
min(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)  
# Minimum
```

R Output

```
[1] 99
```

R Input

```
which.min(CPIIIsecLbsGen.df$Lbs)  
# Location (e.g., index) of the first occurrence of the  
# minimum value
```

R Output

```
[1] 53
```

R Input

```
max(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)  
# Maximum
```

R Output

```
[1] 192
```

R Input

```
which.max(CPIIIsecLbsGen.df$Lbs)  
# Location (e.g., index) of the first occurrence of the  
# maximum value
```

R Output

```
[1] 58
```

R Input

```
quantile(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)  
# Quantiles, or values at: 0%, 25%, 50% 75%, and 100%
```

R Output

```
0%    25%    50%    75%    100%  
99.00 121.00 127.00 139.25 192.00
```

R Input

```
sum(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)  
# Arithmetic sum of all values in a vector
```

R Output

```
[1] 7904
```

R Input

```
boxplot.stats(CPIIIsecLbsGen.df$Lbs)
# Produce values for a vector related to a boxplot:
# lower whisker, lower hinge, median, upper hinge, upper
# whisker, N, and outliers
```

R Output

```
$stats
[1] 99.0 121.0 127.0 139.5 165.0

$n
[1] 60

$conf
[1] 123.226 130.774

$out
[1] 168 169 192
```

R Input

```
fivenum(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)
# Tukey's five number summary for a vector: minimum,
# lower-hinge, median, upper-hinge, and maximum
```

R Output

```
[1] 99.0 121.0 127.0 139.5 192.0
```

R Input

```
IQR(CPIIIsecLbsGen.df$Lbs, na.rm=TRUE)
# Interquartile range of a vector (e.g., a measure of
# dispersion that is equal to the difference between the
# upper quartile and the lower quartile
```

R Output

```
[1] 18.25
```

Although descriptive statistics and measures of central tendency are needed to understand the data, they are presented at the singular level—for all values of `CPIISecLbsGen.df$Lbs`. That is to say, the above statistics offer no details by breakouts of weight (e.g., Lbs) by either Section or by Gender, which are the two factor-type object variables found in the dataset.

Prepare frequency distributions by Section and by Gender, overall. Then, knowing the degree of representation of subjects by these two factor-type object variables, it will be possible to have a better understanding of descriptive statistics for each group, overall (e.g., Section and Gender), and then by group breakouts (e.g., Section, AM and PM; Gender, Female and Male).⁷

R Input

```
table(CPIISecLbsGen.df$Lbs, CPIISecLbsGen.df$Section)
# Value-by-value contingency table (e.g., crosstab) of
# counts for each combination of numeric object variable
# values v factor levels (e.g., Weight (rows) by Section
# (columns)
```

R Input

```
table(CPIISecLbsGen.df$Lbs, CPIISecLbsGen.df$Gender)
# Value-by-value contingency table (e.g., crosstab) of
# counts for each combination of numeric object variable
# values v factor levels (e.g., Weight (rows) by Gender
# (columns)
```

R Input

```
table(CPIISecLbsGen.df$Gender, CPIISecLbsGen.df$Section,
useNA=c("always"))
# Contingency table of cell sums, not individual values
# Observe Row (e.g., Gender) by Column (e.g., Section)
# placement and also note how there are no missing data
# for either Gender or Section (only for Lbs).
```

⁷The screen print of output from the `table()` function, both for Section and Gender when applied against Lbs, is quite long and is not shown. A few other screen prints, when overly long, are also excluded in an attempt to keep this lesson a reasonable length.

R Output

```
AM PM <NA>
Female 15 20 0
Male    16 10 0
<NA>    0  0  0
```

R Input

```
table(CPIIIsecLbsGen.df$Section, CPIIIsecLbsGen.df$Gender,
useNA=c("always"))
# Contingency table of cell sums, not individual values
# Observe Row (e.g., Section) by Column (e.g., Gender)
# placement and also note how there are no missing data
# for either Section or Gender (only for Lbs).
```

R Output

```
Female Male <NA>
AM      15   16   0
PM      20   10   0
<NA>    0    0   0
```

R Input

```
prop.table(table(CPIIIsecLbsGen.df$Section,
CPIIIsecLbsGen.df$Gender, useNA=c("always")))
# Proportions for each breakout group, Row by Column,
# which cell-by-cell adds to 100 percent
```

R Output

```
Female     Male     <NA>
AM  0.245902 0.262295 0.000000
PM  0.327869 0.163934 0.000000
<NA> 0.000000 0.000000 0.000000
```

R Input

```
prop.table(table(CPIIIsecLbsGen.df$Gender,
CPIIIsecLbsGen.df$Section, useNA=c("always")))
# Proportions for each breakout group, Row by Column,
```

```
# which cell-by-cell adds to 100 percent
```

R Output

	AM	PM	<NA>
Female	0.245902	0.327869	0.000000
Male	0.262295	0.163934	0.000000
<NA>	0.000000	0.000000	0.000000

The xtabs() function and the ftable() function should also be considered for preparation of frequency distributions in the form of contingency tables. The output from these functions is generally the same as what is seen with use of the table() function, but the presentation and placement of row and column headers may show differently.

R Input

```
xtabs(~Section+Gender, data=CPIISecLbsGen.df)
```

R Output

Gender		
Section	Female	Male
AM	15	16
PM	20	10

R Input

```
ftable(xtabs(~Section+Gender, data=CPIISecLbsGen.df))
# Common to many uses with R, note how the ftable()
# function is wrapped around the xtabs() function.
```

R Output

Gender		
Section	Female	Male
AM	15	16
PM	20	10

Use of the table() function and similar functions provides frequency distributions of factor-type object variables. Observe how descriptive statistics are also provided, overall, and more importantly by breakout classifications for factor-

type object variables. Although there are many R functions from which to choose, the `RcmdrMisc::numSummary()` function is a good first selection in terms of utility and appearance of presentation.

R Input

```
install.packages("RcmdrMisc", dependencies=TRUE)
library(RcmdrMisc)           # Load the RcmdrMisc package.
help(package=RcmdrMisc)      # Show the information page.
sessionInfo()                # Confirm all attached packages.
```

R Input

```
RcmdrMisc::numSummary(CPIIISeclbsGen.df$Lbs)
# Descriptive statistics overall, no breakout groupings
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
131.733	17.5894	18.25	99	121	127	139.25	192	60	1

The `RcmdrMisc::numSummary()` can also be used to display measures of central tendency at the breakout group level for numeric-type object variables:

R Input

```
RcmdrMisc::numSummary(CPIIISeclbsGen.df[,c("Lbs")],
groups=Section) # Default printout, breakouts by Section
```

R Output

[Selected output is not shown, to save space.]

	mean	sd	0%	25%	50%	75%	100%	data:n	data:NA
AM	128.300	13.1520	107	120.25	126	136.5	157	30	1
PM	135.167	20.7864	99	122.25	128	143.5	192	30	0

R Input

```
RcmdrMisc::numSummary(CPIIISeclbsGen.df[,c("Lbs")],
groups=Gender) # Default printout, breakouts by Gender
```

R Output

	mean	sd	IQR	0%	25%	50%	75%	100%	data:n	data:NA
Female	123.971	8.27997	9	107	120	124	129	144	35	0
Male	142.600	21.27401	32	99	125	142	157	192	25	1

2.6 Quality Assurance, Data Distribution, and Tests for Normality

Quality assurance (QA) is by no means a simple activity where one and only one attempt is made to examine the data for quality issues, distribution patterns, and examination of normality. Quality assurance needs to be inclusive throughout the entire research process, from the first ideas about the research project to the last proofreading of the final report. To that end, efforts have gone into the production of figures that offer a graphical sense of the data. Equally, descriptive statistics have been explored from many perspectives. Summative descriptive statistics were provided for the numeric object variable Lbs (e.g., weight), overall and by breakouts of the two factor-type object variables Gender and Section.

Following along with the notion that multiple R-based functions serve similar purposes, there are many statistical tests that provide sound judgment on data distribution and the degree to which there is adherence to normality. With each normality test offering a slightly different perspective to distribution patterns, the following tests are all supported by R:

- Anderson–Darling test for normality.
- Cramer–von Mises test for normality.
- Lilliefors (Kolmogorov–Smirnov) test for normality.
- Pearson chi-square test for normality.
- Shapiro–Francia test for normality.

From among these selections, the Shapiro test is demonstrated in this lesson, using the `shapiro.test()` function, which is included in the stats package. The stats package is one of the many packages obtained when R is first downloaded.

As a reminder, for nearly all statistical tests, the Null Hypothesis is worded in a negative fashion and is typically stated as *There is no statistically significant difference between A and B in terms of C* ($p \leq 0.05$). Somewhat different, the Null Hypothesis for a Shapiro Test of Normality is instead worded in the affirmative and is typically stated as *The data follow a normal distribution* ($p \leq 0.05$). Give attention to this different approach and the way the Shapiro

Test for Normality is worded when interpreting the p-values, significance levels, and outcomes from a Shapiro Test for Normality.

It is common to focus on a p-value of either $p \leq 0.05$ or $p \leq 0.01$ when considering normality tests, such as the Shapiro Test for Normality. Remember that due to the way the Shapiro Test for Normality is worded in the affirmative:

- Lower calculated p-values provide evidence against the null hypothesis. If the calculated p-value is less than or equal to the criterion p-value (e.g., significance level), the rules-based decision is to reject the null hypothesis and to declare that the data do not follow a pattern of normal distribution.
- Higher calculated p-values fail to provide evidence against the null hypothesis. If the calculated p-value is greater than the criterion p-value (e.g., significance level), the rules-based decision is to accept (e.g., fail to reject) the null hypothesis and to declare that the data seem to follow a pattern of normal distribution.

As demonstrated throughout this text, many different types of figures and text-based output are used to offer a sense of the data, overall. A key interest when viewing the many figures and statistics on central tendency is the concern over normal distribution for numeric object variables or Lbs in this lesson. Normal distribution is important in that many inferential tests are only used, appropriately, if the underlying data exhibit normal distribution. Of course, compromised decisions are often made and it is not uncommon to see the application of statistical tests that call for the use of normally-distributed data with data that do not follow a normal distribution pattern—certainly not a perfect normal distribution pattern. The question, then, is how much deviation away from normal distribution is acceptable before a nonparametric approach to statistical analysis is necessary. This is always a judgment call, but the Shapiro Test for Normality will at least provide a sense of data distribution. From that outcome, it can be determined whether it is accepted that the data exhibit a reasonable pattern of normal distribution or if the normality of the data should be questioned.

Apply the `shapiro.test()` function against `CPIISecLbsGen.df$Lbs`, overall, to address these first-level concerns. For now, there is no focus on factor-type object variable breakouts: `CPIISecLbsGen.df$Section` (e.g., AM or PM) or `CPIISecLbsGen.df$Gender` (e.g., Female and Male).

R Input

```
shapiro.test(CPIISecLbsGen.df$Lbs)
```

R Output

p-value = 0.000578

The calculated p-value (0.000578) for `CPIIIsecLbsGen.df$Lbs` is less than 0.05, providing evidence that normal distribution is not evident for the object variable `CPIIIsecLbsGen.df$Lbs`, overall. This finding needs to be considered in the future when deciding on whether to use a parametric test or a test when using `CPIIIsecLbsGen.df$Lbs`. However, going beyond the use of `CPIIIsecLbsGen.df$Lbs` at the overall level of comparison is deviation away from deviation away from normal distribution for `CPIIIsecLbsGen.df$Lbs` a concern for the two levels of Section (e.g., AM and PM) and for the two levels of Gender Male)? It is now known that `CPIIIsecLbsGen.df$Lbs` (e.g., weight) does not follow a normal distribution pattern, but:

- Is it possible that the weight of AM students follows a normal distribution pattern, regardless of the distribution pattern of weight for PM students?
- Is it possible that the weight of PM students follows a normal distribution pattern, regardless of the distribution pattern of weight for AM students?
- Is it possible that the weight of Female students follows a normal distribution pattern, regardless of the distribution pattern of weight for Male students?
- Is it possible that the weight of Male students follows a normal distribution pattern, regardless of the distribution pattern of weight for Female students?

The `RVAideMemoire::byf.shapiro()` function is an excellent choice to examine these questions on normality by breakouts of factor-type object variables. Observe below how the `RVAideMemoire::byf.shapiro()` function provides a simple and convenient summary of normality p-values by individual breakouts of a factor-type object variable.⁸

⁸Normality tests examine distribution patterns, not difference and not association. The Shapiro Test for Normality makes no attempt to determine if there is a statistically significant difference (e.g., $p \leq 0.05$ or $p \leq 0.01$) between AM and PM subjects in terms of weight. The Shapiro Test for Normality makes no attempt to determine if there is a statistically significant difference (e.g., $p \leq 0.05$ or $p \leq 0.01$) between Female and Male subjects in terms of weight. The Shapiro Test for Normality and more specifically for this lesson, the `RVAideMemoire::byf.shapiro()` function, is only used to examine normality by declared breakout groups.

R Input

```
install.packages("RVAideMemoire", dependencies=TRUE)
library(RVAideMemoire)          # Load the RVAideMemoire package.
help(package=RVAideMemoire)    # Show the information page.
sessionInfo()                  # Confirm all attached packages.
```

R Input

```
# Normality test of Lbs overall

RVAideMemoire::mshapiro.test(CPIISecLbsGen.df$Lbs)
```

R Output

```
p-value = 0.000578
# Compare to use of the shapiro.test() function.
```

R Input

```
# Normality test of Lbs by Section

RVAideMemoire::byf.shapiro(Lbs ~ Section,
  data=CPIISecLbsGen.df)
```

R Output

	W	p-value
AM	0.9560	0.244497
PM	0.9024	0.009616 **

R Input

```
# Normality test of Lbs by Gender

RVAideMemoire::byf.shapiro(Lbs ~ Gender,
  data=CPIISecLbsGen.df)
```

R Output

	W	p-value
Female	0.9865	0.9367
Male	0.9823	0.9261

Use the `ggplot2::ggplot()` function to produce Q-Q plots of the numeric object variable Lbs (e.g., weight), overall, and then by breakouts of each of the two factor-type object variables (e.g., Section and Gender).⁹ Use these figures as a visual reinforcement of outcomes from application of the Shapiro tests (Fig. 2.4).

R Input

```
# Weight (Lbs), Overall
QQLbs <-
ggplot2::ggplot(CPIIISSecLbsGen.df,
aes(sample=Lbs)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  ggtitle("Weight (Lbs) QQ-Plot and QQ-Line,\nOverall\n") +
  labs(x = "\nTheoretical", y = "Weight (Lbs)\n") +
  theme_bw()

# Weight (Lbs) by Section (two breakouts)
QQFacetLbsSection <-
ggplot2::ggplot(CPIIISSecLbsGen.df,
aes(sample=Lbs)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ Section) +
  ggtitle("Weight (Lbs) QQ-Plot and QQ-Line,\nby Course Section\n") +
  labs(x = "\nTheoretical", y = "Weight (Lbs)\n") +
  theme_bw()

# Weight (Lbs) by Gender (two breakouts)
QQFacetLbsGender <-
ggplot2::ggplot(CPIIISSecLbsGen.df,
aes(sample=Lbs)) +
```

⁹Similar to other terms that have multiple means of expression, throughout this text expect to see the terms Quantile-Quantile, Q-Q, and QQ to describe this graphical image.

```

stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ Gender) +
  ggtitle("Weight (Lbs) QQ-Plot and QQ-Line,\nby Gender\n") +
  labs(x = "\nTheoretical", y = "Weight (Lbs)\n") +
  theme_bw()

```

R Input

```

gridExtra:::grid.arrange(
  QQLbs,
  QQFacetLbsSection,
  QQFacetLbsGender, ncol=3)

```

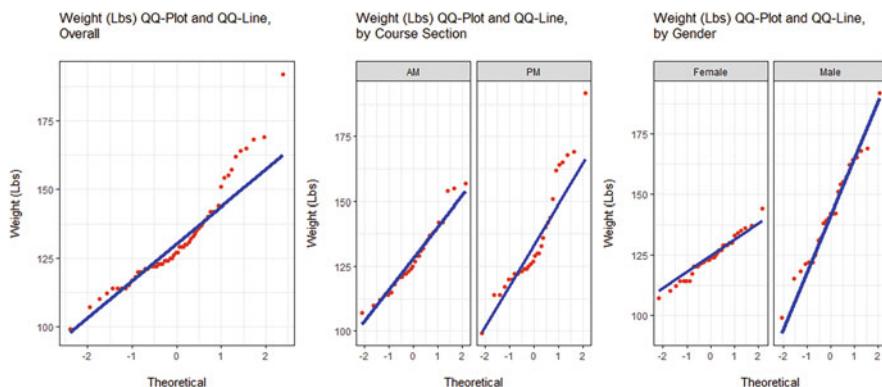


Figure 2.4: Weight Q-Q plot breakouts by section and gender

Overall, the Q-Q plots seem to reinforce the statistical outcome that there is overall deviation away from normal distribution. At a more finite level of observation for the breakouts, notice how the greatest degree of deviation away from normal distribution seems to be the weights above 150 pounds in the afternoon (e.g., PM) course section. Here is where the red circles, indicating weights of specific subjects, go farthest away from the straight fit line.

Quality Assurance is attempted throughout this lesson, by generating graphical images of the relationship between and among data and also by providing descriptive statistics and measures of central tendency between and among data. Further, normality has been addressed by examining distribution patterns of the measured object variable, overall (e.g., Lbs), and then by breakouts of the factor-type object variables (e.g., Section, AM and PM; Gender, Female and Male).

As a general statement, it was observed that normal distribution of the numeric object variable was not evident in the dataset associated with this lesson, at the overall level of comparison. Using $p \leq 0.05$ as the criterion value of acceptance or rejection of each normal distribution Null Hypothesis, overall, and by breakouts:

- Overall, the data for `CPIISecLbsGen.df$Lbs` do not follow a pattern of normal distribution (overall calculated p-value = 0.000578).
- When looking at normality by Section, students in the morning (e.g., AM calculated p-value = 0.244497) section had weights that followed normal distribution, whereas students in the afternoon (e.g., PM calculated p-value = 0.009616) section had weights that did not follow normal distribution.
- When looking at normality by Gender, both Female students (calculated p-value = 0.9367) and Male students (calculated p-value = 0.9261) had weights that followed normal distribution.

These findings, overall and by breakouts, must be considered when deciding which test(s) to use for either tests of difference or tests of association. Forward looking, should a useful test of difference, such as Student's t-Test for Independent Samples be avoided merely because a few students in the afternoon course section possibly impacted normality? There is no one and only one answer to this question. Careful thought and consultation with others (e.g., colleagues, editors, peers, etc.), as well as discovery from the published literature, will provide insight into the best course of action regarding findings of normality and eventual use of these findings on later statistical testing.

2.7 Statistical Test(s)

This lesson is focused on data exploration, descriptive statistics, and measures of central tendency. R-based functions have been used to provide text-based statistics and graphical images that address these three areas. Experienced researchers know to always challenge the data before great efforts are put into analyses and preparation of ostensibly publishable graphics. To achieve this aim, time spent looking at the data, no matter how simple the dataset, is time well-spent. A simple one-off mistake, early on, can compromise days and even weeks of work if data are not examined at the beginning of the research process and throughout. Quality assurance is a constant activity.

The object variable `CPIISecLbsGen.df$Lbs`, as presented in this lesson, is merely one measured variable from a much larger dataset. The same statement applies to the variables `CPIISecLbsGen.df$Section` and `CPIISecLbsGen.df$Gender` in that they are merely two factor-type object variables from a much larger dataset. The dataset used in this lesson is from a dataset that is larger

in terms of the number of subjects and larger in terms of the number of other variables, both measured interval-type object variables and grouped factor-type object variables.

This lesson does not include any statistical tests, neither parametric nor non-parametric tests, that examine differences and commonalities or associations:

Tests associated with parametric data typically include the following:

- Student's t-Test for Independent Samples.
- Student's t-Test for Matched Pairs.
- Oneway Analysis of Variance (ANOVA).
- Twoway Analysis of Variance (ANOVA).
- Pearson's r Coefficient of Correlation.

Tests associated with nonparametric data typically include the following:

- Mann–Whitney U Test.
- Wilcoxon Matched-Pairs Signed-Ranks Test.
- Kruskal–Wallis H-Test for Oneway Analysis of Variance (ANOVA) by Ranks.
- Friedman Twoway Analysis of Variance (ANOVA) by Ranks.
- Spearman's rho Rank-Difference Coefficient of Correlation.

The remaining lessons in this text address many of these tests. The emphasis on this lesson has been on data exploration, descriptive statistics, and measures of central tendency. These three constructs are essential first tasks if the many tests typically associated with statistics are to have best effect.

2.8 Summary

For the limited sample presented in this lesson (e.g., `CPIISecLbsGen.df$Lbs`), observe the broad array of statistics that provides far more detail, but again—only for this limited sample:

R Output

Minimum value:	99
Maximum value:	192
Mean value:	131.73
Median:	127
Upper quartile:	139.25

Lower quartile:	121.00
Variance:	309.39
Standard deviation:	17.59
Midspread (IQR):	18.25
Skewness:	1.08
Number of data values:	61
Number of missing values:	1

Of course, there was no background information provided in the front part of this lesson to know if the sample, taken from two computer programming course sections taught at one specific Florida-based high school, was representative of peer students at the high school, the county school district, the state, etc. Accordingly, the statistics and figures in this lesson are useful, but the production of these statistics and figures should be viewed as only a beginning in the research process. Issues related to the pervasive research process must take into account representation, but the research process must also be balanced against the cost of data acquisition. These issues will be addressed to some degree in later lessons found in this text.

2.9 Addendum 1: Specialized External Packages and Functions

There are more than 15,000 external R-based packages available for immediate download, without required permissions and without direct cost. From among these there are thousands of specialized functions to supplement the set of functions available when R is initially downloaded. A few specialized functions specific to descriptive statistics and measures of central tendency are demonstrated below.

Some specialized functions provide not only numerical statistics of immediate use, but they also concurrently provide a graphical image, to further reinforce the organization of data in question. Function arguments are typically used to embellish graphical output, but in this lesson function arguments have been kept to a minimum. More details, available through the use of function arguments, are demonstrated in later lessons.

R Input

R Input

```
asbio::Mode(CPIISecLbsGen.df$Lbs)
# The modes::modes() function was demonstrated previously.
# The asbio::Mode() function is merely another way to obtain
# this measure of central tendency.
```

R Output

```
[1] 114 122
```

The epiDisplay package is quite versatile, with many functions that support frequency distributions, descriptive statistics, and measures of central tendency. Some epiDisplay functions provide not only attractive graphics but also useful statistics in text format printed to the screen (Figs. 2.5, 2.6 and 2.7) .

R Input

```
install.packages("epiDisplay")
library(epiDisplay)           # Load the epiDisplay package.
help(package=epiDisplay)      # Show the information page.
sessionInfo()                 # Confirm all attached packages.
```

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
epiDisplay::tab1(CPIISecLbsGen.df$Section,
  main="Frequency Distribution of Section",
  col=c("red", "blue"), font.lab=2, font.axis=2)
epiDisplay::tab1(CPIISecLbsGen.df$Gender,
  main="Frequency Distribution of Gender",
  col=c("red", "blue"), font.lab=2, font.axis=2)
# Generate frequency distributions and percentages
# of by breakouts. Generate a figure of frequency
# distributions.
```

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
epiDisplay::tabpct(CPIISecLbsGen.df$Section,
```

```
CPIISecLbsGen.df$Gender, graph=TRUE, decimal=1,
main="Frequency Distribution of Section by Gender",
xlab = "Section", ylab = "Gender", cex.axis=1,
percent= "both", las=1, col=c("red", "blue"))

epiDisplay::tabpct(CPIISecLbsGen.df$Gender,
CPIISecLbsGen.df$Section, graph=TRUE, decimal=1,
main="Frequency Distribution of Gender by Section",
xlab = "Gender", ylab = "Section", cex.axis=1,
percent= "both", las=1, col=c("red", "blue"))

# Along with the figure that appears as a stacked bar
# chart, look at the detailed frequency distribution
# statistics printed to the screen, which can be
# copied and pasted into a word-processed document.
```

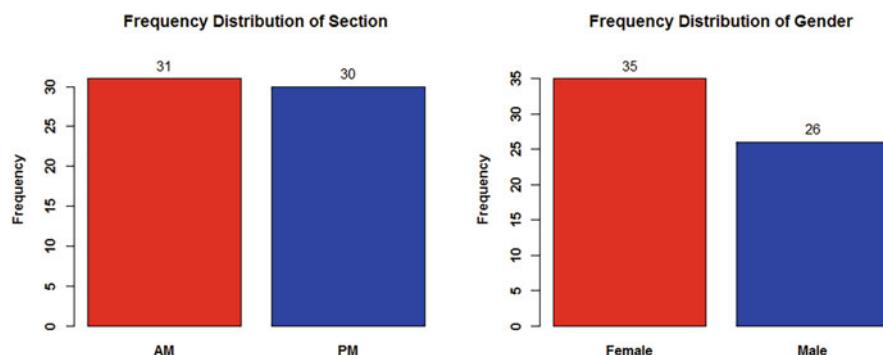


Figure 2.5: Section and gender: frequency distribution overall

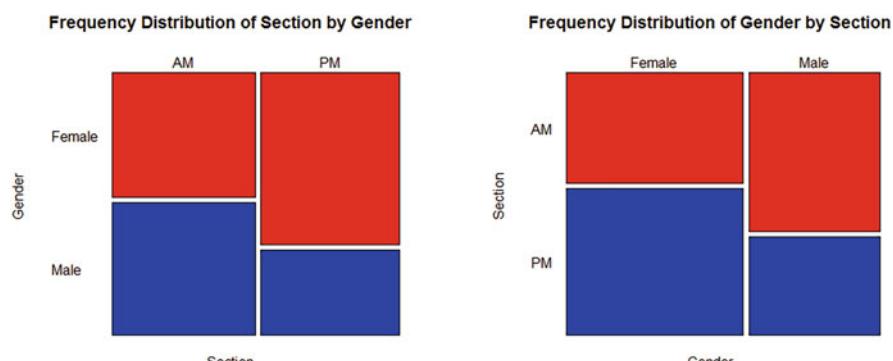


Figure 2.6: Section and gender: frequency distribution breakouts

R Input

```

par.ask=TRUE)
par(mfrow=c(1,3)) # 3 figures into a 1 row by 3 column grid
epiDisplay::summ(CPIIIsecLbsGen.df$Lbs,
  by=NULL,           # No breakout statistics
  graph=TRUE, box=TRUE, # Dotplot and boxplot
  pch=20, ylab="auto",
  main="Sorted Dotplot of Weight (Lbs), Overall",
  cex.X.axis=1.25, # Note X axis label size.
  cex.Y.axis=1.25, # Note Y axis label size.
  font.lab=2, dot.col="auto")
epiDisplay::summ(CPIIIsecLbsGen.df$Lbs,
  by=Section,        # Breakout statistics
  graph=TRUE,         # Dotplot
  pch=20, ylab="auto",
  main="Sorted Dotplot of Weight (Lbs) by Section",
  cex.X.axis=1.25, # Note X axis label size.
  cex.Y.axis=1.25, # Note Y axis label size.
  font.lab=2, dot.col="auto")
epiDisplay::summ(CPIIIsecLbsGen.df$Lbs,
  by=Gender,          # Breakout statistics
  graph=TRUE,          # Dotplot
  pch=20, ylab="auto",
  main="Sorted Dotplot of Weight (Lbs) by Gender",
  cex.X.axis=1.25, # Note X axis label size.
  cex.Y.axis=1.25, # Note Y axis label size.
  font.lab=2, dot.col="auto")
# Produce a sorted dotplot and accompanying descriptive
# statistics: by=NULL (e.g., overall), by=Section, and
# by=Gender. Do not confuse a sorted dotplot with a QQ plot.
# The two figures represent different constructs.

```

The s20x package is also quite good for generating detailed output of descriptive statistics, overall, and by breakout groups of factor-type object variables.

R Input

```

install.packages("s20x", dependencies=TRUE)
library(s20x)           # Load the s20x package.
help(package=s20x)       # Show the information page.
sessionInfo()            # Confirm all attached packages.

```

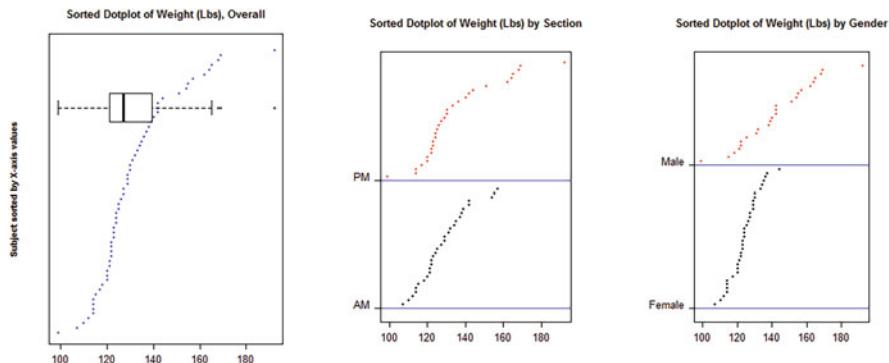


Figure 2.7: Weight by section and gender breakouts

R Input

```
s20x::summaryStats(CPIISecLbsGen.df$Lbs,
na.rm=TRUE) # Accommodate missing values.
```

R Output

Minimum value:	99
Maximum value:	192
Mean value:	131.73
Median:	127
Upper quartile:	139.25
Lower quartile:	121
Variance:	309.39
Standard deviation:	17.59
Midspread (IQR):	18.25
Skewness:	1.08
Number of data values:	61
Number of missing values:	1

R Input

```
s20x::summaryStats(Lbs ~ Gender, CPIISecLbsGen.df,
na.rm=TRUE) # Accommodate missing values
```

R Output

	Sample Size	No. Miss.	Mean	Median	Std Dev	Midspread
Female	35	0	123.971	124	8.27997	9

Male	25	1 142.600	142 21.27401	32
------	----	-----------	--------------	----

R Input

```
s20x::summaryStats(Lbs ~ Section, CPIISecLbsGen.df,
na.rm=TRUE) # Accommodate missing values
```

R Output

	Sample Size	No. Miss.	Mean	Median	Std Dev	Midspread
AM	30	1	128.300	126	13.1520	16.25
PM	30	0	135.167	128	20.7864	21.25

The s20x::rowdistr() function, when wrapped around the crosstabs() function, produces an attractive frequency distribution barchart of factor-type object variables. It also generates text-based descriptive statistics to the screen that add further insight into the data (Figs. 2.8 and 2.9) .

R Input

```
par(ask=TRUE); s20x::rowdistr(crosstabs(~ Gender + Lbs,
  data=CPIISecLbsGen.df), plot=TRUE, suppressText=FALSE,
  comp='basic')
```

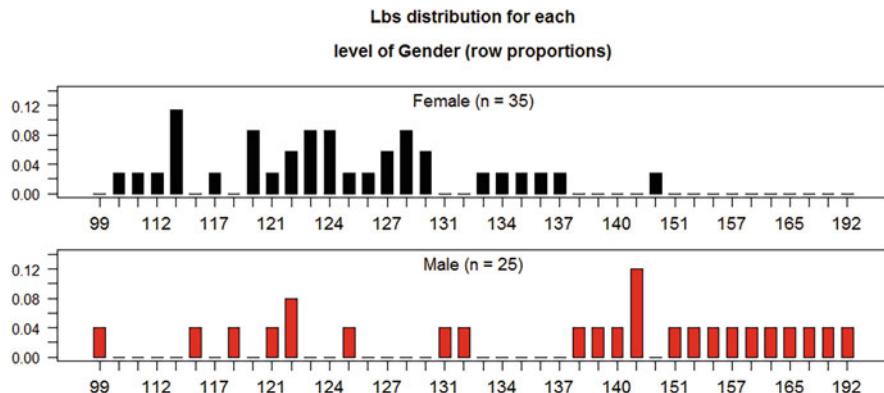


Figure 2.8: Graphical representation of weight by gender

R Input

```
par(ask=TRUE); s20x::rowdistr(crosstabs(~ Section + Lbs,
  data=CPIISecLbsGen.df), plot=TRUE, suppressText=FALSE,
  comp='basic')
```

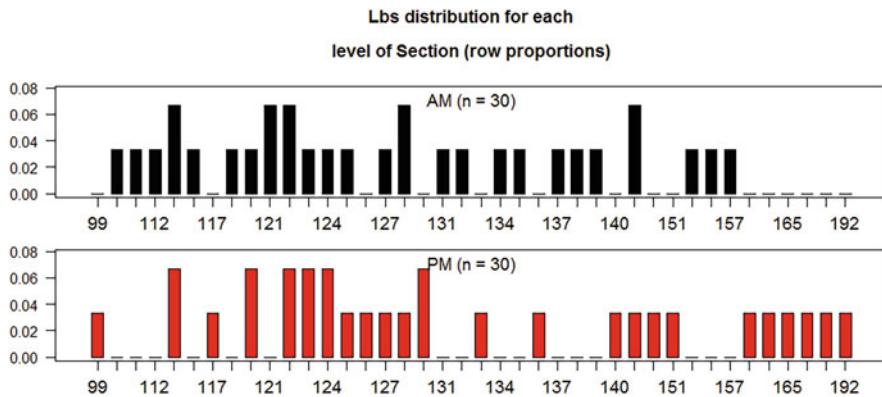


Figure 2.9: Graphical representation of weight by section

The `arsenal::tableby()` function, with the syntax shown below (but not the output, due to overly long length), builds on the presentation of breakout descriptive statistics. With this function, all of the breakouts for two or more factor-type object variables can be presented in one convenient table. A p-value is included in the printout for both breakouts of Gender and breakouts of Section. For now, ignore the p-value statistics but know that it will be explained in far more detail in later lessons. The focus in this beginning lesson, now, is merely on descriptive statistics and measures of central tendency, not inferential analyses.

R Input

```
install.packages("arsenal", dependencies=TRUE)
library(arsenal)                      # Load the arsenal package.
help(package=arsenal)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.
```

R Input

```
summary(arsenal::tableby(list(Section, Gender) ~ Lbs,
  data = CPIISecLbsGen.df), text=TRUE, total=TRUE)
# Produce a highly-detailed table of descriptive
# statistics, especially: mean, sd, and range.
```

The function `pivottabler::qpvt()` puts output into text format, as shown below. If desired, copy the syntax shown below but use the `pivottabler::qhpvt()` function to put output into HTML format. For those with advanced skills, the `pivottabler::qlpvt()` function generates output suitable for a \LaTeX document.

R Input

```
install.packages("pivotabler", dependencies=TRUE)
library(pivotabler)           # Load the pivotabler package.
help(package=pivotabler)      # Show the information page.
sessionInfo()                 # Confirm all attached packages.
```

R Input

```
pivotabler::qpvt(CPIISecLbsGen.df, "Gender", "Section",
  c("Mean Lbs"="mean(Lbs, na.rm=TRUE)",
    "SD Lbs"="sd(Lbs, na.rm=TRUE)"),
  formats=list("%.0f", "%.1f"))
# Row (Gender) by Column (Section) in text format
```

R Output

	AM			PM			Total				
	Mean	Lbs	SD	Mean	Lbs	SD	Lbs	Mean	Lbs	SD	Lbs
Female	122		9.5	125		7.3		124		8.3	
Male	134		13.9	155		24.7		143		21.3	
Total	128		13.2	135		20.8		132		17.6	

R Input

```
pivotabler::qpvt(CPIISecLbsGen.df, "Section", "Gender",
  c("Median Lbs"="median(Lbs, na.rm=TRUE)"),
  formats=list("%.0f", "%.1f"))
# Row (Section) by Column (Gender) in text format
# Note how the header Median Lbs does not show in output.
```

R Output

	Female	Male	Total
AM	123	132	126
PM	124	163	128
Total	124	142	127

Again, there are more than 15,000 external packages available to the R community. Many more packages and associated functions relating to data exploration, descriptive statistics, and measures of central tendency are available and will be demonstrated in future lessons.

2.10 Addendum 2: Parametric v Nonparametric

Throughout this lesson and looking forward to other lessons in this text, there has been reference to the terms parametric and nonparametric. There is no one and only one clear-cut definition of these terms, but for immediate purposes consider how data can be visualized as either parametric or nonparametric:

- Parametric data are often viewed as measured data on an interval scale where there is some degree of semblance to normal distribution. Perfect normal distribution is rarely, if ever, achieved under real conditions, but the key here is that there is at least a reasonable degree of normal distribution bell-shaped curve) for parametric data. The Systolic Blood Pressure (SBP) of one million subjects selected at random from an even larger population of over of over 335 million subjects should be expected to exhibit normal distribution for the measured datum (e.g., SBP, measured as mmHg) and under regular from such a study of Systolic Blood Pressure should be viewed as meeting the assumptions associated with tests that are viewed from a parametric perspective. The stats::rnorm() function is used in R to simulate a set of random numbers that exhibits normal distribution as is demonstrated in Addendum 3.
- Nonparametric data are often viewed as data that: (1) do not meet the declared precision of an interval scale, (2) bring to mind concerns about extreme deviation from normal distribution, and (3) show large differences in the number of subjects for each breakout group in datasets with factor-type object variables. Nonparametric data are often associated with data where measurement is from an ordinal scale instead of an interval scale. The heights of adults adults from a study where there are only a few subjects and where data were measured as Short, Medium, and Tall instead of an interval measurement such as centimeters would most likely be viewed as meeting the assumptions associated with tests that are viewed from a nonparametric perspective. The stats::runif() function is used in R to simulate a set of random numbers that fails to exhibit normal distribution, as is demonstrated in Addendum 3.

The appropriate selection of an inferential test is a key issue when examining data and then deciding if it is best to judge the data as being parametric or nonparametric. This issue is clearly addressed in future lessons.

2.11 Addendum 3: Additional Practice Datasets for Data with Normal Distribution Patterns and Data That Do Not Exhibit Normal Distribution Patterns

2.11.1 Purpose of This Addendum

The purpose of this addendum is to use the R environment to provide additional guidance on data exploration, descriptive statistics, and measures of central tendency. Mastery of these topics is essential for anyone who regularly engages in empirically-based research, either as a consumer or producer of research. Fortunately, the R environment provides many excellent tools for the core topics associated with this addendum.¹⁰

Note In the front matter to this lesson, the syntax is presented and is then immediately followed in most cases by either a copy of screen output or an accompanying figure. That approach is not used in this addendum. View this addendum as practice homework-type bonus materials. Syntax is presented and descriptive text goes along with the syntax. However, screen output and figures are generally excluded and when they show it is only to offer mid-point guidance that analyses follow along in a correct manner. Either key the syntax in this addendum or copy and paste it into an editor—whatever is feasible and most convenient. And then, to use the common expression, Practice—Practice—Practice!

2.11.2 Background

There are two datasets used in the first part of this addendum. The two datasets are self-generated, each created using R-based tools. The first dataset is created using the `stats::rnorm()` function and the data follow a generally normal distribution pattern. Then, to offer an interesting contrast, the second dataset is created using the `stats::runif()` function and the data do not follow a normal distribution pattern. Be sure to closely examine how R-based functions are used against both datasets, but then notice how there are widely different results between the data that exhibit normal distribution and the data that do not exhibit normal distribution.

In an effort to provide consistency and reproducible results, the `base::set.seed()` function is used at the start of this demonstration. Self-generated datasets, such as the two datasets used in the first part of this addendum (if prepared correctly using the `base::set.seed()` function) will then allow for equivalent replication of

¹⁰Syntax is provided throughout this addendum, but of course this syntax is only a suggestion. Experiment and take other approaches to how the data can be analyzed and outcomes presented by using other functions and other arguments. Use this addendum as a confidence-building resource on how R is used with increasingly complex analyses.

the data if generated again and/or if generated by other researchers.

There are no inferential analyses associated with this addendum. Therefore, a Null Hypothesis is not provided. All analyses are descriptive (e.g., describe the data) in nature, not inferential (e.g., allow an inference or judgment about differences between groups, association between object variables, etc.).

2.11.3 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

Instead of importing .csv files, the data in the first part of this addendum are self-generated, using R-based tools. After the `base::set.seed()` function is used, two R-based functions are used to self-generate the two datasets:

- The `stats::rnorm()` function is used to generate a dataset (e.g., called `SBPNormal`) that follows a pattern of normal distribution.
- The `stats::runif()` function is used to generate a dataset (e.g., called `SBPNotNormal`) that does not follow a pattern of normal distribution.

R Input

```
# Set the seed
base::set.seed(8)
  # The base::set.seed() function is commonly used
  # immediately before any attempt to generate random
  # numbers. Some numerical value is then used along
  # with this function in an effort to set the seed
  # and in turn produce a specific sequence of random
  # numbers. For this addendum, the number 8 was used
  # to set the seed. There is nothing special about
  # the number 8 being used to set the seed and the
  # sequence of random numbers generated because of
  # this selection. It would have been possible to
  # generate another set of random numbers by using
  # 1234 or any other number to set the seed. Some
  # number had to be selected and for this addendum
  # the number 8 was used to set the seed. If this
  # number were used by others the same set of random
  # numbers would be generated, allowing reproduction
  # of results in the future and/or by others.
```

R Input

```
# Data with Normal Distribution Patterns

SBPNormal <- stats::rnorm(100000, mean=120, sd=06)
# Approximately +3 and -3 SDs for SBP with 120 mean,
# when data show normal distribution.
# Use the stats::rnorm() function to generate random
# numbers.
# Create a set of 100,000 random numbers that
# exhibits normal distribution, with mean = 120 and
# standard deviation = 06. To allow for some degree
# of familiarity, this distribution (mean = 120 and
# sd = 06) is equivalent to common metrics for
# Systolic Blood Pressure, SBP.
```

R Input

```
# Data That Do Not Exhibit Normal Distribution Patterns

SBPNotNormal <- stats::runif(100000, min=102, max=138)
# Approximately +3 and -3 SDs for SBP with 120 mean,
# when data show normal distribution.
# Use the stats::runif() function to generate random
# numbers.
# Create a set of 100,000 random numbers that does
# not exhibit normal distribution. For this set of
# random numbers, the minimum value is set to 102
# and the maximum value is set to 138, which model
# to a large degree the two extreme values for SBP
# readings (mean = 120 and standard deviation = 06)
# at 3 SDs (standard deviations).
```

2.11.4 Organize the Data and Display the Code Book

The data for the first part of this addendum consist of two object variables, named `SBPNormal` and `SBPNotNormal`. The two object variables are self-generated using R-based functions. They are not imported from an external source.

R Input

```
#####
# Code Book for SBPNormal and SBPNotNormal #
#####
#
# SBPNormal ..... Numeric #
#           100,000 SBP readings with mean =
#                           120 and sd = 06 #
#
# SBPNotNormal ..... Numeric #
#           100,000 SBP readings with minimum =
#                           102 and maximum = 138 #
#####
#####
```

Recall that object variables `SBPNormal` and `SBPNotNormal` are separate object variables and they are not part of a common dataframe.

Additional R-based functions are now used to be sure that the data are of the expected type, appear correct and ready for use, and are within expected ranges. Follow along with the syntax and reproduce it to obtain equivalent output. Recall that output is only shown occasionally, with the expectation being that these bonus materials are geared toward self-initiated practice.

R Input

```
base::getwd()          # Confirm working directory
base::ls()             # Confirm available objects

utils::str(SBPNormal) # Identify structure
utils::head(SBPNormal, n=10) # Show the head, first 10 cases
utils::tail(SBPNormal, n=10) # Show the tail, last 10 cases
base::summary(SBPNormal) # Summary statistics

utils::str(SBPNotNormal) # Identify structure
utils::head(SBPNotNormal, n=10) # Show the head, first 10 cases
utils::tail(SBPNotNormal, n=10) # Show the tail, last 10 cases
base::summary(SBPNotNormal) # Summary statistics
```

There are many more R-based functions available for further diagnostics about singular object variables and dataframes with multiple object variables, but the functions demonstrated immediately above should be more than sufficient to confirm that the data are in good form.

2.11.5 Conduct a Visual Data Check Using Graphics (e.g., Figures)

For immediate use as a data check, produce throw-away graphics and avoid all embellishments. This approach, using graphics as a quality assurance measure, provides a convenient way to review the data and have confidence that the data are acceptable for later analyses.

For each numeric object variable, it is a good practice to put a histogram, a density plot, a boxplot, and a Q-Q plot all in the same figure, as shown below. If desired, or if there are any concerns about the data, it is also useful to use a violin plot and a dot plot to graphically examine numeric data. Each figure provides a different perspective of the data, regardless of whether the figures are ever shared with others.

R Input

```
par(ask=TRUE)                  # Pause
par(mfrow=c(2,2))              # 4 figures - 2 rows by 2 column grid
graphics::hist(SBPNormal)
graphics::plot(stats::density(SBPNormal, na.rm=TRUE))
graphics::boxplot(SBPNormal)
stats::qqnorm(SBPNormal); stats::qqline(SBPNormal)
# Place four separate figures (e.g., histogram, density
# plot, boxplot, and a Q-Q plot with an accompanying Q-Q
# line) into one common figure.
#
# Note how one function can be wrapped around another
# function. In this example, the graphics::plot() function
# has been wrapped around the stats::density() function. A
# requirement for the stats::density() function is that the
# na.rm=TRUE argument must be used.

par(ask=TRUE)                  # Pause
par(mfrow=c(2,2))              # 4 figures - 2 rows by 2 column grid
graphics::hist(SBPNNotNormal)
graphics::plot(stats::density(SBPNNotNormal, na.rm=TRUE))
graphics::boxplot(SBPNNotNormal)
stats::qqnorm(SBPNNotNormal); stats::qqline(SBPNNotNormal)
# Place four separate figures (e.g., histogram, density
# plot, boxplot, and a Q-Q plot with an accompanying Q-Q
# line) into one common figure.
```

These simple black-and-white throw-away graphics provided an initial view of the data, which for this addendum details differences in expected visualization for data that exhibit normality (e.g., SBPNormal) and data that do not exhibit

normality (e.g., `SBPNotNormal`). To allow for more comparative visualization between the two object variables (e.g., `SBPNormal` and `SBPNotNormal`), generate four separate figures: one figure for two side-by-side histograms, one figure for two side-by-side density plots, one figure for two side-by-side boxplots, and one figure for two side-by-side Q-Q plots and Q-Q lines. Arrange each figure into a 1 by 2 grid (1 row by 2 columns) and adjust the X axis and the Y axis for a common scale to allow meaningful side-by-side comparisons.

R Input

```
# Histogram

par(ask=TRUE)      # Pause
par(mfrow=c(1,2))  # 2 figures - 1 row by 2 column grid
graphics::hist(SBPNormal,
  main="SBP - Normal Distribution",
  col="red",        # Add color
  breaks=50,         # Increase granularity of histogram
  font.lab=2,        # Bold labels
  xlim=c(0,200),    # X axis scale
  ylim=c(0,7000))   # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
graphics::hist(SBPNotNormal,
  main="SBP - Not Normal Distribution",
  col="red",        # Add color
  breaks=50,         # Increase granularity of histogram
  font.lab=2,        # Bold labels
  xlim=c(0,200),    # X axis scale
  ylim=c(0,7000))   # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
# Notice how both histograms have the same X axis scale
# and Y axis scale, allowing meaningful side-by-side
# comparisons.

# Density Plot

par(ask=TRUE)      # Pause
par(mfrow=c(1,2))  # 2 figures - 1 row by 2 column grid
graphics::plot(stats::density(SBPNormal, na.rm=TRUE),
  main="SBP - Normal Distribution",
  col="red",          # Add color
  lwd=5,              # Thick line
```

```
font.lab=2,           # Bold labels
xlim=c(0,200),       # X axis scale
ylim=c(0,0.08))      # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
graphics::plot(stats::density(SBPNotNormal, na.rm=TRUE),
  main="SBP - Not Normal Distribution",
  col="red",          # Add color
  lwd=5,              # Thick line
  font.lab=2,          # Bold labels
  xlim=c(0,200),       # X axis scale
  ylim=c(0,0.08))      # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
# Notice how both density plots have the same X axis
# scale and Y axis scale, allowing meaningful side-by-
# side comparisons.

# Boxplot

par(ask=TRUE)        # Pause
par(mfrow=c(1,2))   # 2 figures - 1 row by 2 column grid
graphics::boxplot(SBPNormal,
  main="SBP - Normal Distribution",
  xlab="Boxplot",      # X axis label
  ylab="SBP",          # Y axis label
  cex.axis=1.15,        # Axis size
  cex.lab=1.15,         # Label size
  col="red",            # Box color
  lwd=2,                # Line thickness
  font.lab=2,            # Bold labels
  font=2,                # Bold font
  ylim=c(0,200))        # Y axis scale
graphics::boxplot(SBPNotNormal,
  main="SBP - Not Normal Distribution",
  xlab="Boxplot",      # X axis label
  ylab="SBP",          # Y axis label
  cex.axis=1.15,        # Axis size
  cex.lab=1.15,         # Label size
  col="red",            # Box color
  lwd=2,                # Line thickness
  font.lab=2,            # Bold labels
  font=2,                # Bold font
```

```

ylim=c(0,200))           # Y axis scale
# Notice how both boxplots have the same Y axis scale,
# allowing meaningful side-by-side comparisons.

# Q-Q Plot

par(ask=TRUE)          # Pause
par(mfrow=c(1,2))    # 2 figures - 1 row by 2 column grid
stats::qqnorm(SBPNormal,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP -
  Normal Distribution",
  col="blue", xlim=c(-4,4), ylim=c(0,200), font.axis=2,
  font.lab=2)
stats::qqline(SBPNormal,   # Add a Q-Q Line to the Q-Q Plot
  col="red", lwd=4, lty=2)
stats::qqnorm(SBPNotNormal,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP -
  Not Normal Distribution",
  col="blue", xlim=c(-4,4), ylim=c(0,200), font.axis=2,
  font.lab=2)
stats::qqline(SBPNotNormal,# Add a Q-Q Line to the Q-Q Plot
  col="red", lwd=4, lty=2)
# Notice how both Q-Q plots have the same X axis scale
# and Y axis scale, allowing meaningful side-by-side
# comparisons.

```

This addendum has not yet included categorical data similar to: (1) nominal objects, such as headcounts of female or male (gender) subjects or (2) ordinal objects, such as small, medium, or large (size) rankings of non-interval measurements. The visual representation of categorical data, whether nominal or ordinal, are often shown as barcharts and mosaic plots, as demonstrated in a later part of this addendum.

2.11.6 Descriptive Statistics for Initial Analysis of the Data

Descriptive statistics, as the name suggests, describe the data. There are many R-based functions that serve this purpose, using the packages that are obtained when R is first downloaded and also by using external packages that are downloaded from external sites that host the CRAN package repository:

- Some functions provide the desired statistic, only.
- Some functions provide the desired statistic, at the summary level and also by breakouts of factor-type object variables.

- Some functions not only provide the desired statistic, but they also provide a graphic that reinforces the outcome.

The data for this addendum are fairly simple and for the first part of this addendum the two self-created datasets (e.g., `SBPNormal` and `SBPNotNormal`) do not include factor-type object variables. Descriptive statistics by factor-type object variable breakouts are certainly important, and they are demonstrated in later parts of this addendum.

The important issue to recall for descriptive statistics of measured object variables is that the focus is usually on a central measure or average (e.g., mode, median, mean, etc.) and dispersion (e.g., variance, standard deviation, minimum, maximum, etc.). When describing a collection of numbers, it is common to hear reference to the term *average*, but this statement is incomplete. It is equally important to know something about the dispersion of the numbers to have a more complete understanding of the data.

Again, most statistics presented immediately below use packages made available from when R is first downloaded. However, as needed, external packages are obtained selected functions from these packages are used to provide either singular or multiple descriptive statistics and measures of central tendency. As a general reminder reminder about the functions presented immediately below:

- Depending on how the R session has been organized in the Housekeeping section, object variables with large Ns often produce output using exponential notation (e.g., e-notation, scientific notation, standard index form). If exponential notation is a problem, consider using the `base::format()` function or the `base::options()` function and adjust the arguments to achieve desired output.
- When obtaining external packages made available by CRAN, some packages may be unavailable from specific sites at specific times. Select another CRAN mirror site if a package does not download.

R text-based output of calculated statistics should always be correct, but consider some type of redundant quality assurance check just to be sure that output is indeed correct. The only thing worse than using a function that does not produce an outcome is a function that produces an outcome, but the outcome is not correct due to logic problems or other errors—not the function. Quality assurance should be pervasive. Planned redundancy is not a burden if it avoids a later error.

As practiced throughout this addendum, package names are generally used along with function names (e.g., `Package::Function`) to be precise even if this at first seems overly-formal—even for common functions found in packages that are obtained when R is first downloaded. This practice is used in this addendum as a reminder of where these many functions reside. If needed, summarize many packages gained from when R is first downloaded by keying:

R Input

```
base::getOption("defaultPackages")
```

For all other R-based functions found in external packages, be sure to download the appropriate package.

R Input

```
# Descriptive Statistics of SBPNormal

base::summary(SBPNormal)
# Summary of all object variables

modes::modes(SBPNormal)
# Mode, or the most frequent value

stats::median(SBPNormal)
# Median, or the mid-point

base::mean(SBPNormal)
# Mean, or the arithmetic average
# The argument na.rm was not used since there are no
# missing data for the object variable SBPNormal. If
# there were missing data then this argument would be
# needed.
```

Observe how functions from three separate packages were used to determine the three separate views toward *average*—mode, median, and mean. This observation serves as an example of why it is best to either use, or at least consider, a Package::Function naming system to keep current with functions and their use—certainly for functions gained from external packages.

Along with use of the base::mean() function, other R-based functions are available to investigate the concept of mean (e.g., arithmetic average) as a descriptive statistic of central tendency. A few examples immediately below should be more than sufficient to demonstrate the possible value of geometric mean, harmonic mean, trimmed mean, and winsor mean.

R Input

```
base::mean(SBPNormal, trim=0.05)
# Trimmed mean, or the arithmetic average after
# removing 5 percent (i.e., trim=0.05) of the
# highest and lowest values

install.packages("psych", dependencies=TRUE)
library(psych)                      # Load the psych package.
help(package=psych)                 # Show the information page.
sessionInfo()                       # Confirm all attached packages.

psych::geometric.mean(SBPNormal)
# Geometric mean, used to limit the impact of
# extreme values

psych::harmonic.mean(SBPNormal)
# Harmonic mean, used to address the impact of
# outliers

psych::winsor.mean(SBPNormal, trim=0.05)
# Winsorized mean, where data at the ends are not
# so much trimmed as they are replaced with
# values that provide a robust estimate of
# central tendency, accommodating the potential
# undue influence of outliers

stats::var(SBPNormal)
# Variance

stats::sd(SBPNormal)
# Standard deviation

base::min(SBPNormal)
# Minimum value

base::which.min(SBPNormal)
# Minimum value location (e.g., row number)

base::max(SBPNormal)
# Maximum value

base::which.max(SBPNormal)
```

```
# Maximum value location (i.e., row number)

base::range(SBPNormal)
# Range of values, minimum to maximum

base::length(SBPNormal)
# Number of occurrences (e.g., N, datapoints)

utils::head(base::sort(SBPNormal))
# First few datapoints of SBPNormal, sorted
# The head() function is wrapped around the
# sort() function.

utils::tail(base::sort(SBPNormal))
# Last few datapoints of SBPNormal, sorted

base::sum(SBPNormal)
# Sum of all values

stats::quantile(SBPNormal)
# Quantile scores, 0% 25% 50% 75% 100%

stats::quantile(SBPNormal,
prob=seq(0, 1, length=11), type=5)
# Use of stats::quantile() function to produce
# deciles, 0% 10% 20% 30% 40% 50% 60% 70% 80% 90%
# 100%

stats::IQR(SBPNormal)
# Interquartile range, or a measure of dispersion
# between the 3rd quartile and the 1st quartile

stats::mad(SBPNormal)
# Median Absolute Deviation (MAD), or the median
# of the absolute deviations from the median
# (compare the MAD statistic to the sd statistic)

grDevices::boxplot.stats(SBPNormal)
# Boxplot Statistics: Lower-Whisker, Lower-Hinge,
# Median, Upper-Hinge, and Upper-Whisker, N, and
# Outliers
# The boxplot.stats() function is included in the
# grDevices package, which is available when R is
```

```
# first downloaded.  
# Outlies and descriptive statistics are printed.  
  
stats::fivenum(SBPNormal)  
# Tukey's Five-Number Summary: Minimum,  
# Lower-Hinge, Median, Upper-Hinge, and Maximum
```

Many functions print to the screen one and only one measure of descriptive statistics and central tendency. However, there are external R packages that include functions where many different descriptive statistics are generated, all at the same time. From among the many possible selections, the following functions will be used to provide multiple statistics that may provide a broader understanding of the data:

- doBy::descStat()
- tables::tabular()
- pastecs::stat.desc()
- psych::describe()
- furniture::table1()
- RcmdrMisc::numSummary()
- epiDisplay::summ()

As a brief warning, the default screen output for some of these functions is quite verbose, and the screen output may show in horizontal (e.g., wide) format instead of vertical (e.g., long) format, making it difficult to copy and paste the output into an external word-processed document. Experiment with these functions and their many arguments to develop personal preferences, both for content and later manipulation of presentation.

R Input

```
install.packages("doBy", dependencies=TRUE)  
library(doBy)                      # Load the doBy package.  
help(package=doBy)                  # Show the information page.  
sessionInfo()                      # Confirm all attached packages.  
  
doBy::descStat(SBPNormal)  
  
install.packages("tables", dependencies=TRUE)  
library(tables)                     # Load the tables package.
```

```

help(package=tables)                      # Show the information page.
sessionInfo()                            # Confirm all attached packages.

tables::tabular((SBPNormal) ~ (n=1) + Format(digits=2)*
  (mean + sd + median))
# It is also possible to generate descriptive
# statistics for breakouts of factor-type object
# variables.

install.packages("pastecs", dependencies=TRUE)
library(pastecs)                         # Load the pastecs package.
help(package=pastecs)                    # Show the information page.
sessionInfo()                            # Confirm all attached packages.
# Select the most local mirror site using Set CRAN mirror.

pastecs::stat.desc(SBPNormal,
  basic=FALSE, desc=FALSE, norm=FALSE)
# Additional descriptive statistics are available

psych::describe(SBPNormal, fast=TRUE)
# Additional descriptive statistics are available

install.packages("furniture", dependencies=TRUE)
library(furniture)                        # Load the furniture package.
help(package=furniture)                  # Show the information page.
sessionInfo()                            # Confirm all attached packages.

furniture::table1((base::data.frame(SBPNormal)))
# Wrap the base::data.frame() function around the
# numeric object variable SBPNormal to temporarily
# coerce it into format as a dataframe to
# accommodate requirements for use of the
# furniture::table1() function.

RcmdrMisc::numSummary(SBPNormal,
  statistics=c("mean", "sd", "quantiles"))
# Additional descriptive statistics are available

epiDisplay::summ(SBPNormal, graph=TRUE)
# Along with descriptive statistics, the graph=TRUE
# argument produces a sorted dot chart.

```

Now, contrast the statistics and any accompanying graphics of parametric data in `SBPNormal` to nonparametric data in `SBPNotNormal`. Recall the similarity in mean for the two datasets, but then consider the otherwise wide variance in outcomes between the two sets of data, `SBPNormal` and `SBPNotNormal`.

R Input

```
# Descriptive Statistics of SBPNotNormal  
  
base::summary(SBPNotNormal)  
  
modes::modes(SBPNotNormal)  
  
stats::median(SBPNotNormal)  
  
base::mean(SBPNotNormal)  
  
base::mean(SBPNotNormal, trim=0.05)  
  
psych::geometric.mean(SBPNotNormal)  
  
psych::harmonic.mean(SBPNotNormal)  
  
psych::winsor.mean(SBPNotNormal, trim=0.05)  
  
stats::var(SBPNotNormal)  
  
stats::sd(SBPNotNormal)  
  
base::min(SBPNotNormal)  
  
base::which.min(SBPNotNormal)  
  
base::max(SBPNotNormal)  
  
base::which.max(SBPNotNormal)  
  
base::range(SBPNotNormal)  
  
base::length(SBPNotNormal)  
  
utils::head(base::sort(SBPNotNormal))  
  
utils::tail(base::sort(SBPNotNormal))
```

```
base::sum(SBPNotNormal)

stats::quantile(SBPNotNormal)

stats::quantile(SBPNotNormal,
  prob = seq(0, 1, length = 11), type = 5)

base::range(SBPNotNormal)

stats::IQR(SBPNotNormal)

stats::mad(SBPNotNormal)

grDevices::boxplot.stats(SBPNotNormal)

stats::fivenum(SBPNotNormal)

doBy::descStat(SBPNotNormal)

tables::tabular((SBPNotNormal) ~ (n=1) + Format(digits=2)*
  (mean + sd + median))

pastecs::stat.desc(SBPNotNormal,
  basic=FALSE, desc=FALSE, norm=FALSE)

psych::describe(SBPNotNormal, fast=TRUE)

furniture::table1(base::data.frame(SBPNotNormal))

RcmdrMisc::numSummary(SBPNotNormal,
  statistics=c("mean", "sd", "quantiles"))

epiDisplay::summ(SBPNotNormal, graph=TRUE)
```

2.11.7 Quality Assurance, Data Distribution, and Tests for Normality

The need for quality assurance and subsequent attention to data distribution patterns is stressed through this entire text. Quality assurance should be a planned, continuous, and pervasive activity where multiple approaches are used to address quality issues. The best-designed analyses are of little value if the data are of questionable value. And, given the nature of large-scale projects using distributed work groups and how those charged with responsibility for

analyses do not always (in fact, rarely) engage in all parts of the research process, it is simply not possible to assume that a dataset that successfully imports into R is acceptable. Data need to be examined throughout the entire production process, using functions that produce text-based statistics and functions that produce graphical output.

As an example of quality assurance, consider the constant attention to functions that produce text-based statistics and other forms of text-based output. If use a function that produces highly-detailed output, even if verbose, to allow a thorough examination of the data. The s20x::summaryStats() function is quite good for generating detailed output of descriptive statistics.

R Input

```
s20x::summaryStats(SBPNormal)
s20x::summaryStats(SBPNotNormal)

utils::head(base::sort(SBPNormal))
utils::head(base::sort(SBPNotNormal))

utils::tail(base::sort(SBPNormal))
utils::tail(base::sort(SBPNotNormal))
```

These simple actions are critical to quality assurance. To be sure that data are as expected and within anticipated ranges. Widely unexpected statistics and extreme outliers may be correct and therefore accepted, but at least attention to unexpected values demands attention. These actions will help identify those concerns.

Graphical tools, such as the previously demonstrated use of histogram, density plot, boxplot, and Q-Q plot, serve a purpose for quality assurance and attention to data distribution patterns. By using all four figures it should be possible to further identify possible problems with unexpected values and unexpected trends. If possible, generate a violin plot and a dotchart of each numeric variable requiring attention to have additional attention to quality assurance and data distribution.

R Input

```
install.packages("UsingR", dependencies=TRUE)
library(UsingR)          # Load the UsingR package.
help(package=UsingR)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

par(ask=TRUE)              # Pause
```

```
par(mfrow=c(2,2))      # 4 figures - 2 rows by 2 column grid
UsingR::simple.violinplot(SBPNormal,
  lty=4, lwd=2, col="red", ylim=c(50,160))
title("SBPNormal")
UsingR::simple.violinplot(SBPNotNormal,
  lty=4, lwd=2, col="red", ylim=c(50,160))
title("SBPNotNormal")
epiDisplay::dotplot(SBPNormal, dot.col="red",
  main="SBPNormal")
epiDisplay::dotplot(SBPNotNormal, dot.col="red",
  main="SBPNotNormal")
```

There are also a few statistical tests that are specifically designed to offer judgment on adherence to normality. The Anderson–Darling test is demonstrated in this addendum and other statistical tests that examine normal distribution will be demonstrated in other addenda throughout this text.¹¹ Look at application of the Anderson–Darling test against SBPNormal and SBPNotNormal and then consider how the outcomes are interpreted.

R Input

```
install.packages("nortest", dependencies=TRUE)
library(nortest)          # Load the nortest package.
help(package=nortest)     # Show the information page.
sessionInfo()             # Confirm all attached packages.

nortest::ad.test(SBPNormal)
```

R Output

```
p-value = 0.969
```

The key finding for SBPNormal and application of the Anderson–Darling test is that the p-value greatly exceeds $p \leq 0.05$. Accordingly, there is statistical confirmation that the data in SBPNormal exhibit normal distribution.

R Input

```
nortest::ad.test(SBPNotNormal)
```

¹¹When using the Anderson–Darling test, it is important to know that the Null Hypothesis for this test is based against any assumption of normality.

R Output

```
p-value < 0.0000000000000002
```

The key finding for `SBPNormal` and application of the Anderson–Darling test is that the p-value is far less than $p \leq 0.05$. Accordingly, there is statistical confirmation that the data in `SBPNormal` do not exhibit normal distribution.

Many figures (e.g., histogram, density plot, boxplot, Q-Q plot, violin plot, and dotplot) used in this addendum provided a visual inspection of the data and are offered as an initial idea about data distribution. Going beyond the use of visual inspection, the Anderson–Darling test is additionally used to provide statistical confirmation on normality. For this addendum, it is now confirmed that `SBPNormal` consists of data that exhibit normality and `SBPNotNormal` consists of data that fail to exhibit normality.

2.11.8 Statistical Test(s)

R-based functions applied against the object variables `SBPNormal` and `SBPNotNormal` were focused only on descriptive statistics. Inferential tests are demonstrated in future lessons.

2.11.9 Summary of Outcomes for `SBPNormal` and `SBPNotNormal`

It is far too common to hear someone say something such as *The average Systolic Blood Pressure for subjects was 120 mmHg*. A statement such as this is only the beginning of what needs to be identified if data are to be used to best effect. An experienced researcher would ask many questions to challenge this beginning statement, such as:

- How many subjects were included in the study?
- Were the data gained from all subjects? Were the data gained from a representative sample of the population? Were there any missing data in the dataset?
- Does the term average refer to mode, median, or mean?
- How are the data dispersed, addressing issues such as minimum and maximum (e.g., range), variance and standard deviation, outliers, extreme values, illogical values, etc.?
- Do the data exhibit a normal distribution pattern or do they fail to show any semblance of normality?

The values for `SBPNormal` and `SBPNotNormal` were purposely selected to emphasize how it is not sufficient to merely know the mean value for an object variable

and to then think that this statistic is sufficient. In this addendum, consider how:

- The mean for `SBPNormal` and `SBPNotNormal` are both about 120:
 - Mean `SBPNormal` 120.016
 - Mean `SBPNotNormal` 119.992
- The median for `SBPNormal` and `SBPNotNormal` are both about 120:
 - Median `SBPNormal` 120.030
 - Median `SBPNotNormal` ... 119.986

Going no further, it would be easy to think that the two object variables are similar in other ways—but that would be a wrong assumption. The variables `SBPNormal` and `SBPNotNormal` are both numeric object variables with the same length (100,000 datapoints) and approximately the same means (`SBPNormal` Mean = 120.016 and `SBPNotNormal` Mean = 119.992) and medians (`SBPNormal` Median = 120.030 and `SBPNotNormal` Median = 119.986).

Yet, the two object variables are widely divergent and only at first glance do they seem similar. How is it possible that the standard deviation of each object variable is widely divergent (`SBPNormal` SD = 6.00677 and `SBPNotNormal` SD = 10.39540)? When considering this question, go back to the prior statement that it was not enough to know only either the mean or median of a set of numbers. Far more information is needed to make an informed judgment, including normality, and that is the purpose of analyses related to data distribution.

This addendum provides some degree of guidance on how R is used to examine data for a wide variety of issues. This level of attention to detail cannot be ignored if later inferential tests are to have full value. Saying that, it is important to know more than the mean or median for a collection of numeric values. Look at the additional materials in this addendum. These materials reinforce how attention to detail simply cannot be ignored and equally how broad sweeping statements must be backed up by further analyses, whether those analyses are ever shared with others or published.

2.11.10 Additional Bonus Materials

Descriptive Statistics of Singular Numeric Object Variables

Data in the additional bonus materials section, below, are designed for teaching purposes only. Although the data may be inspired by real-world studies, the data are purposely organized to achieve desired outcomes—all to demonstrate how R is used for data exploration, descriptive statistics, and measures of central tendency.

It is not enough just to say that *The average value of X is Y*. Equally, it is not enough to say that *The mean of X is Y*. Look at the self-created data immediately below and from these data consider why it is necessary to, at a minimum, know length (e.g., N), mean, and standard deviation (Fig. 2.10) :

R Input

```
# Create the Data

SBPN100000Mean120SD05 <- round(rnorm(100000, mean=120, sd=05))
SBPN100000Mean120SD10 <- round(rnorm(100000, mean=120, sd=10))
SBPN100000Mean120SD15 <- round(rnorm(100000, mean=120, sd=15))
SBPN100000Mean120SD20 <- round(rnorm(100000, mean=120, sd=20))

# Create four objects. Each object consists of 100,000
# datapoints, each object has a mean of 120, and each object
# has a standard deviation that is unique: 05, 10, 15, 20.
# Note how the name for each object variable identifies the
# topic (SBP, Systolic BP, N (100,000), Mean, and
# Standard Deviation (SD). This type of naming scheme is an
# example of internal documentation, whereas a Code Book is
# an example of external documentation.

attach(SBPN100000Mean120SD05); attach(SBPN100000Mean120SD10)
attach(SBPN100000Mean120SD15); attach(SBPN100000Mean120SD20)

# Descriptive Statistics

summary(SBPN100000Mean120SD05); sd(SBPN100000Mean120SD05)
summary(SBPN100000Mean120SD10); sd(SBPN100000Mean120SD10)
summary(SBPN100000Mean120SD15); sd(SBPN100000Mean120SD15)
summary(SBPN100000Mean120SD20); sd(SBPN100000Mean120SD20)

# Create Individual Histograms Placed in a Common Figure

par(ask=TRUE)      # Pause
par(mfrow=c(2,2))  # 4 figures - 2 row by 2 column grid
hist(
  SBPN100000Mean120SD05,
  col="red",         # Add color
  breaks=50,         # Increase granularity of histogram
  xaxt="n",          # Suppress X axis for later manipulation
  xlab="SBP",         # X axis label
  yaxt="n",          # Suppress Y axis for later manipulation
  ylab="N",           # Y axis label
```

```
font.lab=2,      # Bold labels
xlim=c(0,200),   # X axis scale
ylim=c(0,10000)  # Y axis scale
)
axis(side=1, at=seq(0,200,10), font=2)          # X axis
axis(side=2, at=seq(0,10000,1000), font=2, las=3) # Y axis
# Notice how the X axis and the Y axis were suppressed
# using the xaxt="n" and yaxt="n" arguments. The X axis
# and Y axis were then brought back with desired changes
# to the scale and presentation:
# side=1                  X axis
# side=2                  Y axis
# at=seq(min,max,break)   Placement of tick marks
# font=2                  Bold
# las                      Orientation
hist(
SBPN100000Mean120SD10,
col="red",      # Add color
breaks=50,       # Increase granularity of histogram
xaxt="n",        # Suppress X axis for later manipulation
xlab="SBP",       # X axis label
yaxt="n",        # Suppress Y axis for later manipulation
ylab="N",         # Y axis label
font.lab=2,      # Bold labels
xlim=c(0,200),   # X axis scale
ylim=c(0,10000)  # Y axis scale
)
axis(side=1, at=seq(0,200,10), font=2)          # X axis
axis(side=2, at=seq(0,10000,1000), font=2, las=3) # Y axis
hist(
SBPN100000Mean120SD15,
col="red",      # Add color
breaks=50,       # Increase granularity of histogram
xaxt="n",        # Suppress X axis for later manipulation
xlab="SBP",       # X axis label
yaxt="n",        # Suppress Y axis for later manipulation
ylab="N",         # Y axis label
font.lab=2,      # Bold labels
xlim=c(0,200),   # X axis scale
ylim=c(0,10000)  # Y axis scale
)
axis(side=1, at=seq(0,200,10), font=2)          # X axis
axis(side=2, at=seq(0,10000,1000), font=2, las=3) # Y axis
```

```

hist(
  SBPN100000Mean120SD20,
  col="red",           # Add color
  breaks=50,          # Increase granularity of histogram
  xaxt="n",            # Suppress X axis for later manipulation
  xlab="SBP",           # X axis label
  yaxt="n",            # Suppress Y axis for later manipulation
  ylab="N",             # Y axis label
  font.lab=2,          # Bold labels
  xlim=c(0,200),       # X axis scale
  ylim=c(0,10000)      # Y axis scale
)
axis(side=1, at=seq(0,200,10), font=2)           # X axis
axis(side=2, at=seq(0,10000,1000), font=2, las=3) # Y axis

```

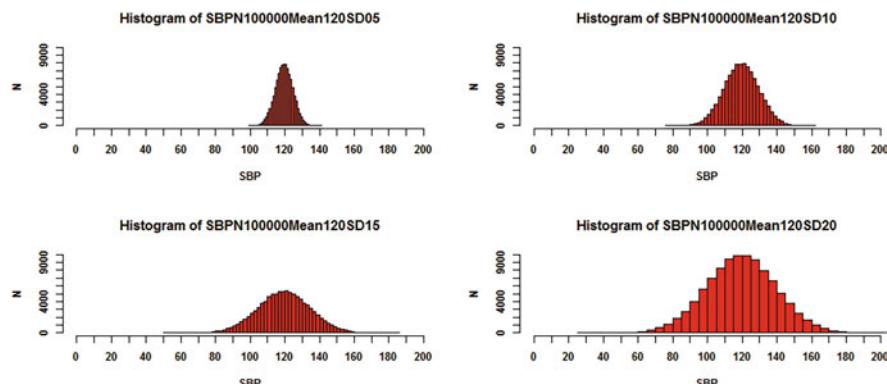


Figure 2.10: Multiple standard deviations with the same mean

From this simple demonstration, notice the extreme difference in the peak and width of each histogram, even though each numeric object variable has the same N and the same mean. There is an infinite number of normal distributions even when N and mean are the same given how there is also an infinite number of potential standard deviations.

Descriptive Statistics of Multiple Numeric and Factor Object Variables in a Dataframe

The previously self-created object variables relating to Systolic Blood Pressure are all singular and are not part of a dataframe. Most analyses, however, involve data organized in a dataframe that includes both numeric-type data and factor-type data. This addendum would be deficient if this reality were not addressed when considering data exploration and other topics associated with this addendum.¹²

¹²Similar to the first part of this addendum, rekey and/or copy and paste the syntax

R Input

```
# Create a Dataframe with Multiple Objects,  
# Factor-Type and Numeric-Type  
  
SoilCornYield.df <- read.table(textConnection("Year Soil Crop Rain BUpAcres  
1997 Clay Corn Wet 184  
1998 Clay Corn Normal 166  
1999 Clay Corn Normal 169  
2000 Clay Corn Normal 170  
2001 Clay Corn Normal 175  
2002 Clay Corn Normal 166  
2003 Clay Corn Wet 183  
2004 Clay Corn Wet 191  
2005 Clay Corn Normal 164  
2006 Clay Corn Normal 162  
2007 Clay Corn Normal 159  
2008 Clay Corn Dry 159  
2009 Clay Corn Normal 155  
2010 Clay Corn Normal 170  
2011 Clay Corn Dry 167  
2012 Clay Corn Normal 163  
2013 Clay Corn Normal 169  
2014 Clay Corn Dry 168  
2015 Clay Corn Normal 162  
2016 Clay Corn Wet 192  
1997 Sand Corn Wet 164  
1998 Sand Corn Normal 152  
1999 Sand Corn Normal 147  
2000 Sand Corn Normal 139  
2001 Sand Corn Normal 131  
2002 Sand Corn Normal 127  
2003 Sand Corn Wet 166  
2004 Sand Corn Wet 158  
2005 Sand Corn Normal 143  
2006 Sand Corn Normal 151  
2007 Sand Corn Normal 161  
2008 Sand Corn Dry 132  
2009 Sand Corn Normal 152"))
```

to recreate these analyses and graphics. Output, for inclusion in this lesson, is purposely minimal—so as to reinforce the concepts associated with data exploration, descriptive statistics, and supporting graphics.

```

2010 Sand Corn Normal      143
2011 Sand Corn Dry        130
2012 Sand Corn Normal     137
2013 Sand Corn Normal     131
2014 Sand Corn Dry        128
2015 Sand Corn Normal     143
2016 Sand Corn Wet        166
1997 Silt Corn Wet        175
1998 Silt Corn Normal     163
1999 Silt Corn Normal     168
2000 Silt Corn Normal     159
2001 Silt Corn Normal     148
2002 Silt Corn Normal     169
2003 Silt Corn Wet        177
2004 Silt Corn Wet        169
2005 Silt Corn Normal     173
2006 Silt Corn Normal     159
2007 Silt Corn Normal     163
2008 Silt Corn Dry         150
2009 Silt Corn Normal     155
2010 Silt Corn Normal     149
2011 Silt Corn Dry         137
2012 Silt Corn Normal     154
2013 Silt Corn Normal     150
2014 Silt Corn Dry         127
2015 Silt Corn Normal     165
2016 Silt Corn Wet        181"), header=TRUE)

getwd()                      # Identify the working directory
ls()                          # List objects
attach(SoilCornYield.df)       # Attach the data, for later use
str(SoilCornYield.df)          # Identify structure
head(SoilCornYield.df, n=3)    # Show the head, 1st 3 cases
summary(SoilCornYield.df)      # Summary statistics

#####
# Code Book for SoilCornYield.df          #
#####

#
# Year ..... Factor #
#                   1997 to 2016 #
#                               #
# Soil .(e.g., Predominant Soil Type) . . Factor #

```

```

#                                     Clay, Sand, Silt #
#                                     #
# Rain ..... Factor #
#           Ordinal progression of rainfall for the #
#           growing season; Dry, Normal, Wet #
#                                     #
# BUperAcre ..... Numeric #
#           Corn yield in Bushels per Acre, ranging #
#           up to 200 Bu/Acre or more #
#####
# Prepare Descriptive Statistics at the Summary Level
# and by Breakouts

furniture::table1((base::data.frame(
  SoilCornYield.df$BUperAcre)))
# Wrap the base::data.frame() function around the
# numeric object variable SoilCornYield.df$BUperAcre
# to temporarily coerce it into format as a dataframe
# to accommodate requirements for use of the
# furniture::table1() function.

# Breakout Statistics by Factor-Type Object Variables

furniture::table1(SoilCornYield.df, BUperAcre, splitby=~Soil,
  digits=2, test=TRUE)
# Examine BUperAcre (corn yield) by Soil (Clay, Sand, Silt)
# and calculate p-value to examine statistical significance.
# Note use of the tilde (e.g., ~) character.

furniture::table1(SoilCornYield.df, BUperAcre, splitby=~Rain,
  digits=2, test=TRUE)
# Examine BUperAcre (corn yield) by Rain (Dry, Normal, Wet)
# and calculate p-value to examine statistical significance.
# Note use of the tilde (e.g., ~) character.

```

A visual reinforcement of corn yields by breakout variables will provide another level of understanding. There are two special items of interest in the three associated figures:

- The object variable `SoilCornYield.df$Year` is an integer in current form. Notice how the `as.factor()` function wraps around `SoilCornYield.df$Year` to temporarily treat it as a factor-type object variable so that it can be used in the Year by BUperAcre boxplot.

- In an attempt to make the figures bold and vibrant, `par()` function settings are adjusted before syntax for the three boxplots and then observe how `par` settings were set back to their original form (Fig. 2.11).

R Input

```
savelwd      <- par(lwd=1)           # Line thickness
savefont     <- par(font=2)          # Bold text
savefontaxis <- par(font.axis=2)      # Bold axis
savecexlab   <- par(cex.lab=1.15)    # Label size
savecexaxis  <- par(cex.axis=0.95)   # Axis size
par(ask=TRUE)
plot(as.factor(Year), BUperAcre, data=SoilCornYield.df,
     main="Corn Yield (Bushels per Acre, Bu/Acre) by Year",
     col="gold", xlab="Year", ylim=c(100,210),
     ylab="Corn Yield (Bushels per Acre, Bu/Acre)")
```

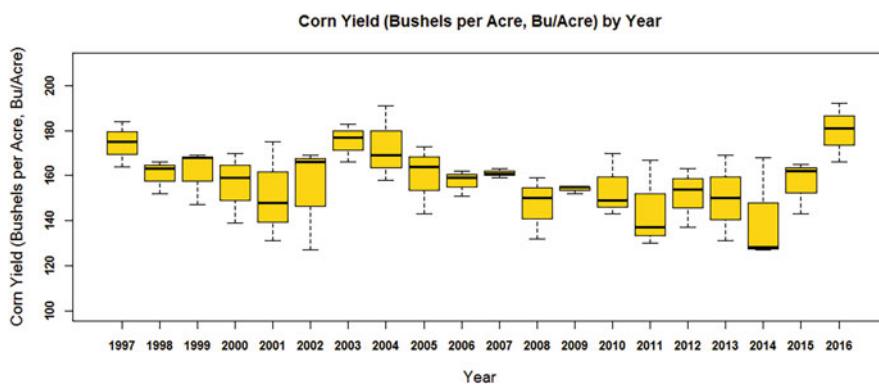


Figure 2.11: Corn yield by year

R Input

```
par(ask=TRUE)
plot(Soil, BUperAcre, data=SoilCornYield.df,
     main="Corn Yield (Bushels per Acre, Bu/Acre) by Soil",
     col="gold", xlab="Soil-Type", ylim=c(100,210),
     ylab="Corn Yield (Bushels per Acre, Bu/Acre)")

par(ask=TRUE)
plot(Rain, BUperAcre, data=SoilCornYield.df,
     main="Corn Yield (Bushels per Acre, Bu/Acre) by Rain",
     col="gold", xlab="Rain", ylim=c(100,210),
     ylab="Corn Yield (Bushels per Acre, Bu/Acre)")
```

```

par(savelwd)      # Return to default setting
par(savefont)     # Return to default setting
par(savefontaxis) # Return to default setting
par(savecexlab)   # Return to default setting
par(savecexaxis)  # Return to default setting
# Notice how global settings were altered by using the
# par() function and then set back to default (e.g.,
# original) format by using the par() function again.

```

This addendum does not focus on extensive discussion about parametric statistics or nonparametric statistics. Even within this limited context about data patterns, it is still useful to: (1) address normal distribution tests when considering descriptive statistics and measures of central tendency, (2) look at the data graphically to more fully understand descriptive statistics and measures of central tendency, and (3) question the distribution pattern of selected object variables. These three actions will help guide decision-making when selecting the appropriate use of one test over another.

Look at another self-generated example of numeric and factor object variables organized in a dataframe. In original format, the dataframe will consist of three variables, referencing: (1) Systolic Blood Pressure, (2) Gender, and (3) Race-Ethnicity. Observe how the data are self-generated by using the stats::rnorm() function and the base::sample() function. The data are not from a field-based study and are for teaching and demonstration purposes only.

R Input

```

# set.seed(8)
# The set.seed() function was used previously. Use it
# here only if needed, thus the comment character in
# front of the set.seed() function, since the set.seed()
# function is still in use for this lesson.

SBP <- base::round(stats::rnorm(1000, mean=120, sd=05))
# Create a numeric object variable of 1,000 data points
# called SBP, with mean = 120 and sd = 05.
base::summary(SBP)
# The base::summary() function is usually the best first
# choice to generate descriptive statistics of a numeric
# object variable.

Gender1 <- c("Female", "Male")
Gender2 <- base::sample(Gender1, 1000, replace=TRUE,
                       prob=c(0.35,0.65))
# Create a factor-type object variable of 1,000 data

```

```
# points, where approximately 35 percent of all data read
# as Female and 65 percent read as Male.
base::table(Gender2)
# The base::table() function is a typical first selection
# to gain a sense of frequency distributions of a factor
# object variable.

RaceEthnic1 <- c("African-American", "Hispanic", "Other",
                 "White")
RaceEthnic2 <- base::sample(RaceEthnic1, 1000,
                           replace=TRUE,
                           prob=c(0.20, 0.25, 0.05, 0.50))
# Create a factor-type object variables of 1,000 data
# points, where 20 percent of all data read as African-
# American, 25 percent read as Hispanic, 5 percent read
# as Other, and 50 percent read as White.
base::table(RaceEthnic2)

SBPGenderRaceEthnic.df <- base::data.frame(
  SBP, Gender2, RaceEthnic2)
# Place all three separate object variables (SBP,
# Gender2, and RaceEthnic2) into a common dataframe,
# using the base::data.frame() function.

base::getwd()                      # ID working directory
base::ls()                          # List objects
base::attach(SBPGenderRaceEthnic.df) # Attach the data
utils::str(SBPGenderRaceEthnic.df)   # Identify structure
utils::head(SBPGenderRaceEthnic.df, n=3) # Show the head
base::summary(SBPGenderRaceEthnic.df) # Summary statistics
```

The `base::summary()` function provides descriptive statistics and measures of central tendency for the numeric object variable `SBPGenderRaceEthnic.df$SBP`. The `base::summary()` function also provides a sense of frequency distribution for the two factor-type object variables, both `SBPRaceEthnic.df$Gender2` and `SBPRaceEthnic.df$RaceEthnic2`.

The `base::table()` function can also be used to generate another view of frequency distributions of factor object variables, as follows:

R Input

```
base::table(SBPGenderRaceEthnic.df$Gender2)

base::table(SBPGenderRaceEthnic.df$RaceEthnic2)
```

Frequency distributions are put into graphical format by using the graphics::barplot() function. As the factor-type object variables are currently found in the dataset, it is necessary to wrap the graphics::barplot() function around the base::table() function to generate the desired figure.

R Input

```
graphics::barplot(base::table(
  SBPGenderRaceEthnic.df$Gender2),
  main="Frequency Distribution of Gender",
  horiz=FALSE,                      # Vertical bars
  las=1,                            # Horizontal axis labels
  col="red",                          # Color
  xlab="Gender",                     # X axis label
  ylab="Number of Cases",           # X axis label
  ylim=c(0,650),                     # X axis scale
  cex.axis=1.25,                     # Axis size
  cex.lab=1.25)                      # Label size

# Adjust the axis values to accommodate the figure.
# Place the bars in either horizontal or vertical
# orientation, depending on the number of letters needed
# to display labels.
```

The barplot for Race-Ethnicity will need some accommodation, due to the long string (16 characters) for the expression African-American. From among the many ways this issue could be approached, it may be best to approach this issue by temporarily adjusting margins:

R Input

```
# Put the bar plot into horizontal orientation.

# Temporarily adjust the margins, generate the figure,
# and then toggle back to original margins.

par("mar")                           # Confirm default margin (BLTR)
# BLTR - Bottom, Left, Top, Right
```

```
# [1] 5.1 4.1 4.1 2.1

par(mar=c(5, 8, 4, 2) + 0.1) # Set a new margin
par(ask=TRUE) # Control the screen
graphics::barplot(base::table(
  SBPGenderRaceEthnic.df$RaceEthnic2),
  main="Frequency Distribution of Race-Ethnicity",
  horiz=TRUE, # Horizontal bars
  las=1, # Horizontal axis labels
  col="red", # Color
  xlab="Number of Cases", # X axis label
  xlim=c(0,550), # X axis scale
  cex.axis=1.25, # Axis size
  cex.lab=1.25) # Label size
par(mar=c(5.1, 4.1, 4.1, 2.1))# Toggle back to default margin

par("mar") # Confirm default margin (BLTR)
# BLTR - Bottom, Left, Top, Right
```

With R there is hardly ever one and only one way to obtain desired statistics or generate an attractive figure. Often there are functions in external packages that accommodate both—text-based statistics and graphical figures. A few functions from the very versatile epiDisplay package provide text output and graphics. When using the epiDisplay package, it is often best to start with the epiDisplay::tableStack() function, to generate text-based statistics that give a sense of frequency distribution and percentage distribution for one or more variables. Notice how 2:3 was used to declare analyses for Gender2 (Column 2) and RaceEthnic2 (Column 3), avoiding SBP (Column 1).

R Input

```
epiDisplay::tableStack(2:3,
  dataFrame=SBPGenderRaceEthnic.df,
  by="none", count=TRUE, decimal=2,
  percent=c("column", "row"))
# Column 2 = Gender2
# Column 3 = RaceEthnic2
```

The epiDisplay::tableStack() function can also be used to create a crosstab of two object variables. Gender2 is positioned as the column, RaceEthnic2 is positioned as the row, and along with descriptive statistics note the p-value and Chi-Square test for determining proportional representation, which is not an immediate concern in this lesson.

R Input

```
epiDisplay::tableStack(
  vars=RaceEthnic2,                                     # Rows
  dataFrame=SBPGenderRaceEthnic.df,
  by=Gender2, count=TRUE, decimal=2,      # Columns
  percent=c("column", "row"),
  frequency=TRUE, name.test=TRUE,
  total.column=TRUE, test=TRUE)
```

The epiDisplay::tableStack() function output is sufficiently detailed, and it could be easily copied and placed into a word-processed document, requiring only minor editing to serve as an attractive and informative table.

To not only gain text-based statistics, but to also generate an attractive bar chart, use the epiDisplay::tab1() function. Carefully review how the arguments are used to best advantage.

R Input

```
par(ask=TRUE)
epiDisplay::tab1(SBPGenderRaceEthnic.df$Gender2,
  main="Frequency Distribution of Gender", # Title
  col=c("red", "blue"),                      # Color
  font.lab=2,                                # Bold
  font.axis=2)                               # Bold
# Generate frequency distributions and percentages
# of Gender2 breakouts: Female and Male. Then,
# generate an accompanying figure of frequency
# distributions.

par(ask=TRUE)
epiDisplay::tab1(SBPGenderRaceEthnic.df$RaceEthnic2,
  main="Percentage Distribution of Race-Ethnicity",
  col=c("red", "blue", "seagreen", "cyan"),# Color
  font.lab=2,                                # Bold
  font.axis=2,                                # Bold
  decimal=1,                                 # Decimals
  bar.values="percent")                      # Percent
# Generate frequency distributions and percentages
# of RaceEthnic2 breakouts: African-American,
# Hispanic, Other, and White. Then, generate an
# accompanying figure of frequency distributions.
# Note how the bar.values argument was used to show
```

```
# percentage distribution as opposed to N.
```

Although the attractive figure associated with the epiDisplay::tab1() function will likely receive first attention, be sure to observe the detailed frequency distribution table, showing N and percentages (e.g., both Percent and Cumulative Percent).

Along with use of the graphics::barplot() function, the mosaic plot should also be considered as a graphical tool for displaying proportional representation of categories for factor-type object variables. Review documentation for the graphics::mosaicplot() function and consider how it compares to use of the graphics::barplot() function.

R Input

```
par.ask=TRUE)
graphics::mosaicplot(~Gender2 + RaceEthnic2,
  data=SBPGenderRaceEthnic.df,
  main="Race-Ethnicity (Rows) by Gender (Column) Breakouts",
  col=c("red", "blue", "seagreen", "cyan"),
  xlab="Gender", ylab="Race-Ethnicity",
  type="pearson")                                # Pearson's Chi-square
```

The waffle chart (e.g., squared pie chart) is also used to show the distribution of breakout groups compared to total counts. The waffles of a waffle chart show as squares, which many consider easier to understand than the wedges of a pie chart.

It is perhaps best to step back and review once again the proportional representation of Gender2 and Race-Ethnicity2 in the dataframe SBPGenderRaceEthnic.df. The base::table() function will serve this purpose at first, but look at the way other simple R-based functions are used.

R Input

```
RaceEthnicGender <- base::table(
  SBPGenderRaceEthnic.df$RaceEthnic2,      # Row
  SBPGenderRaceEthnic.df$Gender2)           # Column
# Use the base::table() function to create an enumerated
# object variable (e.g., RaceEthnicGender) that represents a
# table of summed frequencies of rows (e.g., RaceEthnic)
# by columns (e.g., Gender).
```

```
RaceEthnicGender
```

```
# Display the object variable RaceEthnicGender
```

Use the `base::margin.table()` and `base::prop.table()` functions to produce more detailed information about the frequencies shown in the newly enumerated dataset `RaceEthnicGender`: total count, by cell breakout counts, and by cell breakout percentages.

R Input

```
base::margin.table(RaceEthnicGender)      # Total count
base::margin.table(RaceEthnicGender, 1) # 1 represents rows
base::margin.table(RaceEthnicGender, 2) # 2 represents columns
base::prop.table(RaceEthnicGender)        # Sum to 100%, all cells
# Proportions, by individual cell
base::prop.table(RaceEthnicGender, 1) # Sum to 100%, by rows
# Proportions, by row
base::prop.table(RaceEthnicGender, 2) # Sum to 100%, by columns
# Proportions, by column
```

To add more detail, use the `stats::addmargins()` function to add totals to both rows and columns.

R Input

```
stats::addmargins(base::prop.table(RaceEthnicGender))
```

These many individual actions provide a sense of frequency distribution for `RaceEthnic2` (rows) and `Gender2` (column), individually and as these two object variables intersect. The `base::margin.table()` function is used to create enumerated object variables, which are then used to create the waffle chart:

Create the object `RacEth`, which becomes an object representing the names and values for `SBPGenderRaceEthnic.df$RaceEthnic2`:

R Input

```
base::margin.table(RaceEthnicGender, 1) # 1 represents rows

RacEth <- c('African-American (187)' = 187,
           'Hispanic (256)'          = 256,
           'Other (049)'            = 049,
           'White (508)'             = 508)

# Note the single quote character used for this
```

```
# syntax. It is the single quote character that
# occurs on the keyboard immediately below the
# tilde (e.g., ~) character.
```

```
RacEth
str(RacEth)
```

Create the object `Gen`, which represents a named number object of the names and values for `SBPGenderRaceEthnic.df$Gender2` (Fig. 2.12) :

R Input

```
base::margin.table(RaceEthnicGender, 2) # 2 represents columns

Gen <- c('Female (343)' =343,
        'Male (657)' =657)

Gen
str(Gen)

install.packages("waffle")
library(waffle)                      # Load the waffle package.
help(package=waffle)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

par(ask=TRUE)
waffle::waffle(RacEth/10,
               rows=5, size=0.1, col=c("red", "blue", "seagreen", "cyan"),
               title="Waffle Chart of Race-Ethnicity",
               xlab="With 1,000 subjects each square =~ 10 subjects")
# Experiment, as needed, to determine the best number of
# rows, cell size, and the number of subjects for each cell
# in the waffle chart, or 1,000 overall subjects in this
# example.
```

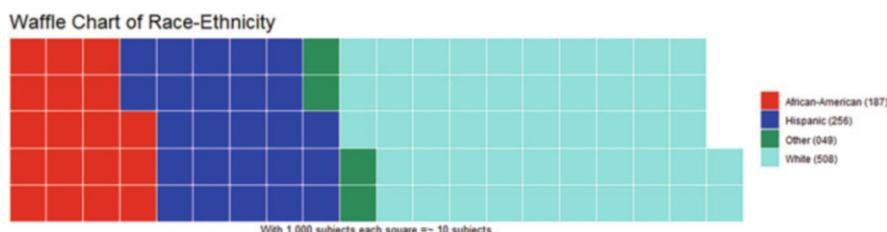


Figure 2.12: Waffle chart of race-ethnicity

R Input

```
par(asleep=TRUE)
waffle::waffle(Gen/10,
  rows=5, size=0.1, col=c("red", "blue"),
  title="Waffle Chart of Gender",
  xlab="With 1,000 subjects each square =~ 10 subjects")
# Experiment, as needed, to determine the best number of
# subjects for each cell in the waffle chart, or 500 for
# this example.
```

Notice how a pie chart is not displayed in this addendum. A pie chart is not the best choice to show the representation of breakout groups for factor-type (e.g., categorical) object variables. It is more than difficult to discern relationships and proportions of slices of a pie when they are anything other than half-slices (50%) or quarter-slices (25%).

Descriptive Statistics and Use of the ggplot2 Package for *Beautiful Graphics*

In an effort to gradually introduce the ggplot2 package (and the many packages associated with use of the ggplot2 package), this addendum also includes a demonstration of a few ways this highly-popular package is used to generate *Beautiful Graphics* involving descriptive statistics. As the ggplot2 package is demonstrated in this addendum and throughout the text, notice how the presentation of each figure is easily changed by using a wide selection of attractive themes associated with the ggthemes package.¹³

A demonstration of the ggplot2::ggplot() function is used to create a simple barplot, similar to what was previously shown using the graphics::barplot() function.

R Input

```
par(asleep=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df, aes(x=Gender2)) +
  geom_bar()
# A simple ggplot2 figure, showing a drab and generally
# colorless bar plot, absent any embellishments. Even
```

¹³As a general practice throughout these lessons, when the ggplot2 package is downloaded it is common to also download a full set of associated packages which were previously downloaded in an earlier section of this lesson: ggplot2, ggthemes, ggmosaic, gridExtra, grid, and scales. As a reminder, the ggplot2 package is also included among the many packages obtained when the tidyverse package is downloaded.

```
# so, look at the basics of this figure: (1) the
# ggplot2::ggplot() function is applied against the
# SBPGenderRaceEthnic.df dataset, (2) aes() (e.g.,
# aesthetic mapping) was used to layer the figure with
# additional detail(s), which for this figure was the
# declaration that the X axis should focus on the
# Gender2 object variable, and (3) the resulting
# figure should be a bar plot, gained by using the geom
# geom_bar().
```

Different themes and embellishments are used with the ggplot2::ggplot() function to produce more visually appealing figures. Immediately below, see how fill was used to add color to the bars and how theme_bw() was used to make a slight improvement over the prior presentation, changing the background color from gray to white.

R Input

```
par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df, aes(x=Gender2)) +
  geom_bar(stat="count",
  fill=c("red", "blue")) +
  ggtitle("Gender: Counts") +
  xlab("Gender") +
  ylab("Count") +
  theme_bw()
# Use stat="count" to count the number of cases for each bar.
```

Now, notice how additional syntax is added to make an even more appealing presentation.¹⁴

R Input

```
par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df, aes(x=RaceEthnic2)) +
  geom_bar(stat="count",
  fill=c("red", "darkgreen", "orange4", "dodgerblue")) +
  ggtitle("Race-Ethnicity: Counts") +
  xlab("Race-Ethnicity") +
```

¹⁴In later lessons, a user-created theme will be demonstrated, to obtain desired presentation while reducing the many lines of syntax presented in this demonstration of the ggplot2::ggplot() function.

```

ylab("Count") +
# The many options available with ggplot2 are used to create a
# figure that is based on bold (e.g., face="bold") and large
# (e.g., size=14) fonts. This selection is purposely used to
# create figures where details are easily seen, especially
# for large-group presentations where the image is projected
# to a screen. Research the literature and experiment with
# different colors, font types, and font sizes to see what
# works best, based on local conditions and needs.
theme(plot.title=element_text(face="bold", size=14)) +
theme(axis.title.x=element_text(face="bold", size=14)) +
theme(axis.text.x=element_text(face="bold", size=14)) +
theme(axis.title.y=element_text(face="bold", size=14)) +
theme(axis.text.y=element_text(face="bold", size=14)) +
# Following along with large bold fonts, notice how the axis
# is also easy to view, with large tick marks that are thick
# and extend away beyond regular placement.
theme(axis.ticks.x=element_line(size=2)) +
theme(axis.ticks.y=element_line(size=2)) +
theme(axis.ticks.length=unit(0.5,"cm")) +
theme(panel.background=element_rect(fill="grey95"))

```

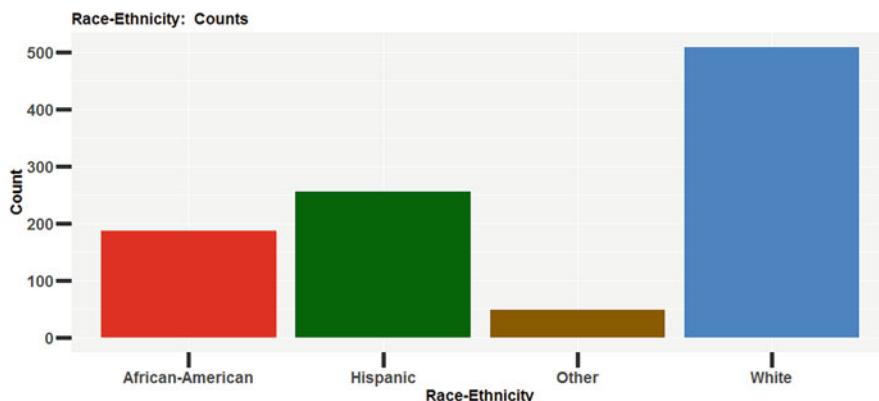


Figure 2.13: Bar plot of race-ethnicity

In view of descriptive statistics and how data are organized, data can be transformed from a continuous nature to a collapsed factor-type nature, which is needed to organize a continuous object variable into meaningful breakout groups with the `ggplot2` package for bar plots and similar graphics. Consider continuous object `SBPGenderRaceEthnic.df$SBP`, when treated as a factor-type object variable by using the `base::factor()` function, which has 30 or more breakouts. It is simply too difficult to understand with desired granularity the exact Systolic Blood Pressure values in the mosaic chart. A way to approach this issue is to use the `base::cut()` function and collapse (e.g., reduce) the 30 or more SBP values into a more manageable collection of SBP values (Fig. 2.13).

R Input

```
SBPGenderRaceEthnic.df$SBP.factor <-
  base::cut(SBPGenderRaceEthnic.df$SBP,
            breaks=c(-Inf, 100, 105, 110, 115, 120, 125, 130, 135,
            140, Inf),
            labels=c("<= 099", "100-104", "105-109", "110-114",
            "115-119", "120-124", "125-129", "130-134", "135-139",
            ">= 140"),
            right=FALSE)

table(SBPGenderRaceEthnic.df$SBP.factor)
```

As a new object variable (e.g., `SBPGenderRaceEthnic.df$SBP.factor`) has been created, take steps to properly incorporate this new object variable into the original dataframe.

R Input

```
base::getwd()
base::ls()
base::attach(SBPGenderRaceEthnic.df)
utils::str(SBPGenderRaceEthnic.df)
utils::head(SBPGenderRaceEthnic.df, n=3)
base::summary(SBPGenderRaceEthnic.df)
```

A simple barplot of `SBPGenderRaceEthnic.df$SBP.factor` breakouts using the `ggplot2` package is a reasonable way to approach best use of this newly enumerated object variable.

R Input

```
par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df) +
  geom_bar(aes(x=SBP.factor, fill=Gender2)) +
  facet_grid(. ~ Gender2) +
  theme(legend.position="none")
# Add embellishments, labels, and themes as needed.

par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df) +
  geom_bar(aes(x=SBP.factor, fill=RaceEthnic2)) +
  facet_grid(. ~ RaceEthnic2) +
```

```
theme(axis.text.x = element_text(angle = 45)) +
  theme(legend.position="none")
# Add embellishments, labels, and themes as needed.
```

Then, consider the use of a histogram to visualize Systolic Blood Pressure as a continuous numeric value by Gender2 and by RaceEthnic2. Review the descriptive statistics for these breakouts. Look at the way these descriptive statistics show in graphical format, using the ggplot2 package. Again, R supports many ways to present outcomes. Select the approach that best meets project requirements.

R Input

```
s20x::summaryStats(SBP ~ Gender2,
  data=SBPGenderRaceEthnic.df)

par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df) +
  geom_histogram(aes(x=SBP, fill=Gender2)) +
  facet_grid(. ~ Gender2) +
  theme(legend.position="none")

s20x::summaryStats(SBP ~ RaceEthnic2,
  data=SBPGenderRaceEthnic.df)

par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df) +
  geom_histogram(aes(x=SBP, fill=RaceEthnic2)) +
  facet_grid(. ~ RaceEthnic2) +
  theme(legend.position="none")
```

Even more complex presentations are possible using the ggplot2 package. These figures should offer even more insight into the data, Systolic Blood Pressure, Gender, and Race-Ethnicity in this example.

R Input

```
par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df) +
  geom_histogram(aes(x=SBP, fill=RaceEthnic2)) +
  facet_grid(RaceEthnic2 ~ Gender2)

par(ask=TRUE)
```

```
ggplot2::ggplot(SBPGenderRaceEthnic.df) +
  geom_histogram(aes(x=SBP, fill=Gender2)) +
  facet_grid(Gender2 ~ RaceEthnic2)
```

Finally, to put closure to what may seem like an endless array of possibilities, use a dot chart to produce another visualization of how factor-type object variables can be presented against values for a continuous numeric object variable. This process and degree of granularity enhances a complete understand of the data (Fig. 2.14).

R Input

```
par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df,
  aes(x=SBP, y=Gender2, color=RaceEthnic2)) +
  geom_point(size=2.5) +
  facet_wrap(~ RaceEthnic2, ncol=1) +
  labs(x="\nSystolic Blood Pressure", y=" ",
       title='SBE by Gender and by Race-Ethnicity\n') +
  theme_bw() +
  theme(legend.position="none")

par(ask=TRUE)
ggplot2::ggplot(SBPGenderRaceEthnic.df,
  aes(x=SBP, y=RaceEthnic2, color=Gender2)) +
  geom_point(size=2.5) +
  facet_wrap(~ Gender2, ncol=1) +
  labs(
    x="\nSystolic Blood Pressure", y =" ",
    title='SBP by Race-Ethnicity and by Gender\n') +
  theme_classic() +
  theme(plot.title=element_text(face="bold", size=14)) +
  theme(axis.title.x=element_text(face="bold", size=14)) +
  theme(axis.text.x=element_text(face="bold", size=14)) +
  theme(axis.title.y=element_text(face="bold", size=14)) +
  theme(axis.text.y=element_text(face="bold", size=14)) +
  theme(plot.background = element_rect(fill="aliceblue")) +
  theme(strip.background=element_rect(color="black",
    fill="linen", size=2, linetype=1)) +
  theme(legend.position="none")
```

Summary to the Added Bonus Materials

It may seem that there is an infinite set of R-based packages, functions, argu-

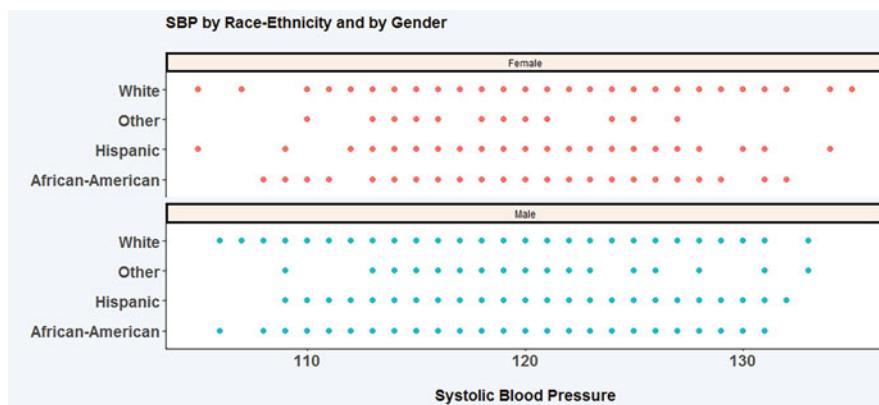


Figure 2.14: Dot chart of SBP by race-ethnicity and gender breakouts

ments, etc., that in one way or another support a basic understanding of data exploration, descriptive statistics, and measures of central tendency. This addendum focused on the two main data types frequently used in biostatistics: (1) numeric-type data of a continuous nature and (2) factor-type data.

- A typical numeric-type datum encountered in biostatistics would be either weight, height, length, systolic blood pressure, diastolic blood pressure, crop yield, etc. For numeric-type data, the main focus will be measures of central tendency (e.g., mode, median, and mean) and dispersion (e.g., variance, standard deviation, minimum, maximum, and range).
- A typical factor-type datum encountered in biostatistics would be either gender (e.g., breakouts of either Female or Male), race-ethnicity (e.g., of either Asian, Black or African-American, Hispanic, Other, or White) or locations (e.g., breakouts of Upper Township, Middle Township, or Lower Township). For factor-type data, the main focus will be frequency distributions (e.g., headcount or N, percent of total).

Experiment with the many R-based functions and arguments shown earlier, but be sure to also look at the appended materials for additional ideas on this lesson. R supports a wide selection of tools that relate to data exploration, descriptive statistics, and measures of central tendency. Every research project should begin with this foundation—as text-based statistics and graphical figures.

2.12 Prepare to Exit, Save, and Later Retrieve This R Session

R Input

```
getwd()           # Identify the current working directory.  
ls()              # List all objects in the working  
                  # directory.  
ls.str()          # List all objects, with finite detail.  
list.files()      # List files at the PC directory.  
  
save.image("R_Lesson_DescriptiveStatistics.rdata")  
  
getwd()           # Identify the current working directory.  
ls()              # List all objects in the working  
                  # directory.  
ls.str()          # List all objects, with finite detail.  
list.files()      # List files at the PC directory.  
  
alarm()           # Alarm, notice of upcoming action.  
q()               # Quit this session.  
                  # Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: File -> Load Workspace. Otherwise, use the load() function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use the .R script file (typically saved as a .txt ASCII-type file) and recreate the analyses and graphics, provided the data files remain available.

2.13 External Data and/or Data Resources Used in This Lesson

The publisher's Web site associated with this text includes **CPIIISectionLbsGender.csv**, which was the only external file directly imported into this lesson. Use this file to practice and replicate the outcomes used in this lesson.

All other data in this lesson were enumerated (e.g., user-created) directly in the R session, using a variety of R-based functions. The user-created data in this lesson are **not** included at the publisher's Web site associated with this text and they are instead easily created in R by following the syntax presented in this lesson.



Chapter 3

Student's t-Test for Independent Samples

Abstract

This lesson provides a demonstration of inquiries into differences between groups, specifically by using Student's t-Test for Independent Samples. Overall, Student's t-Test is a very common test for determining differences when a singular measured variable (e.g., Systolic Blood Pressure, weight of dairy cow milk production per lactation, length of shark dorsal fin, etc.) is compared to differences between a grouping variable with two breakout groups (e.g., Female v Male humans, Guernsey v Jersey cows, Mako v Great White sharks). The t-Test was developed more than 100 years ago, as part of quality assurance work for a beverage company, but published under the pen name *Student*. Student's t-Test is the appropriate test for comparing differences between small samples, typically 30 or fewer. However, it is also common to see Student's t-Test for Independent Samples used with larger samples.

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_3) contains supplementary material, which is available to authorized users.

Keywords: Barplot, boxplot (box-and-whiskers plot), Density plot, Dotchart, Gosset (William Sealy), Histogram, Independent samples, Matched pairs, Student's t-Test, t-statistic, z-statistic

3.1 Background

The purpose of this lesson is to show how the R environment supports Student's t-Test for Independent Samples, the empirical test used to determine if there are statistically significant differences between two samples of continuous data (e.g., Is there a statistically significant ($p \leq 0.05$) difference between females and males regarding Systolic Blood Pressure? Is there a statistically significant ($p \leq 0.05$) difference between Chantecler chickens and Leghorn chickens regarding annual egg production? Is there a statistically significant ($p \leq 0.05$) difference between Timothy grass (*Phleum pratense*) and Orchard grass (*Dactylis glomerata*) regarding protein content?) Student's t-Test for Independent Samples is one of the most commonly used inferential tests and is a common first choice for when two groups are examined against a continuous (e.g., interval) variable and mastery of this test is essential for those who regularly conduct research in the biological sciences.

3.1.1 Description of the Data

This lesson on Student's t-Test for Independent Samples is based on data involving milk production from two specific dairy breeds: Holstein and Jersey. Holstein females in production typically weigh about 1300 pounds or more and may produce up to 25,000 pounds of milk per lactation. In contrast, Jersey females in production often weight up to 1000 pounds and may produce up to 18,000 pounds of milk per lactation.

- Instead of a focus on pounds of milk per lactation, the data for this lesson are geared toward quality factors, specifically percent butterfat and percent protein. Holstein dairy cows may produce more milk per lactation than Jersey dairy cows, but Jersey dairy cows usually produce milk that has higher percentages of butterfat and protein than Holstein dairy cows.
- The data for this lesson are taken from a sample of 20 Holstein dairy cows and 22 Jersey dairy cows, with the percentage of butterfat and protein provided to six places to the right of the decimal point. There is one missing datum in the dataset, specifically PctProtein for subject SH10.
- By no means is suggested that the data are representative of the herds where the subjects are housed and it is equally not suggested that the data are representative of the two breeds. Be sure to notice that no data are provided on either efficiency or production standards: pounds of milk per

lactation, feeding costs, operational costs, etc. Those data are excluded from the dataset, purposely, to maintain focus on the main purpose of this lesson.

When using Student's t-Test for Independent Samples to examine if there is a statistically significant difference ($p \leq 0.05$) between two groups against a measured (e.g., interval) object variable, recall that:

- Student's t-Test is still an appropriate and commonly used test with greater than 30 observations.
- When N goes beyond 30 or so observations, the t-statistic begins to approximate the z-statistic (e.g., z score).

A few parameters should be observed when using Student's t-Test for Independent Samples, in an attempt to determine if the difference between two groups is indeed a true difference or if the difference between the two groups is due only to chance:

- The data from both groups should approximate normal distribution.
- It is best if random selection were used for all members of the two groups.

However, Student's t-Test for Independent Samples is sufficiently robust such that these two assumptions are often not met or perhaps not met as rigorously as desired.

As a reminder, the data are from a one-time collection of milk samples from 42 dairy cattle and the degree of representation of either the herds or breeds is unknown. When applying the Student's t-Test for Independent Samples, it simply cannot be ignored that replication and representation are two key terms to consider when determining practical applications of statistical analysis of biological events.

The dataset is fairly small ($N = 42$) and there is one missing datum.

3.1.2 Null Hypothesis

There are two Null Hypotheses associated with this part of the lesson, with $p \leq 0.05$ the criterion level of significance for both hypotheses.

- Ho: There is no statistically significant difference ($p \leq 0.05$) in Percent Butterfat of milk by Breed (Holstein v Jersey).
- Ho: There is no statistically significant difference ($p \leq 0.05$) in Percent Protein of milk by Breed (Holstein v Jersey).

Notice how each Null Hypothesis (Ho) uses $p \leq 0.05$. The expression $p \leq 0.05$ is used to identify the declared probability level specific to the Null Hypothesis. As a general statement, after calculation of the t-statistic and subsequent calculated p-value, through the use of R, with correct interpretation of

these statistics there will be a 5% or less probability of an incorrect inference related to differences associated with this test.

Many exploratory inferential analyses in the biological and social sciences are conducted at $p \leq 0.05$. However, it is also common to see many published studies set at the more restrictive $p \leq 0.01$ and even $p \leq 0.001$. And, there is a trend by some researchers to publish the calculated p-value, whether above or below $p \leq 0.05$, and to let the reader decide on how to interpret the practical applications of the p-value.¹

Numerical codes have been purposely used for some data (e.g., Breed) in this lesson. Numerical codes are quite common and as such, the creation of a later Code Book is essential so that there is agreement on what each code represents.

3.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

The data for this lesson are from a much larger dataset. The full dataset incorporates data on feed types and feeding regimes as well as other production inputs that ultimately impact profitability. Those data are excluded from this lesson and the focus is exclusively on differences in percent butterfat and percent protein by breed.

For this lesson the dataset is in .csv (e.g., comma-separated values) file format, after being originally prepared in a spreadsheet. The data are separated by commas. The data are neither separated by tabs nor by spaces.

R Input

```
#####
# Housekeeping                                     Use for All Analyses #
#####
date()           # Current system time and date.
Sys.time()       # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls(all.names=TRUE) # List all objects in the working
                   # directory, including hidden files.
rm(list=ls())     # CAUTION: Remove all files in the
                   # working directory. If this
```

¹Along with the use of p, you will also see the term alpha in any discussions about the level of probability, but p will be used in this lesson.

```

#           action is not desired, use rm()
#           one-by-one to remove the objects
#           that are not needed.
ls.str()      # List all objects, with finite detail.
getwd()        # Identify the current working directory.
setwd("F:/R_BiostatisticsIntroduction")
               # Set to a new working directory.
               # Note the single forward slash and double
               # quotes.
               # This new directory should be the directory
               # where the data file is located, otherwise
               # the data file will not be found.
getwd()        # Confirm the working directory.
list.files()   # List files at the PC directory.
.libPaths()    # Library pathname.
.Library       # Library pathname.
sessionInfo() # R version, locale, and packages.
search()       # Attached packages and objects.
searchpaths()  # Attached packages and objects.
#####

```

The Housekeeping template shown above, with occasional changes, is used throughout this text.² The purpose of this collection of R functions, all in one convenient beginning section, is to be sure that the working environment, selected directory, etc., is as desired.

Create an object called `MilkBreedFatProt.df` which is a dataframe, as indicated by the enumerated `.df` extension to the object name. This object will represent the output of applying the `read.table()` function against the comma-separated values file called `MilkBreedButterfatProtein.csv`.³ Note the arguments used with the `read.table()` function, showing that there is a header with descriptive variable names (`header = TRUE`), the `.` character is used for decimals (`dec = ".")`, and the separator between fields is a comma (`sep = ","`).

R Input

```
MilkBreedFatProt.df <- read.table (file =
  "MilkBreedButterfatProtein.csv",
```

²As an addition to the Housekeeping syntax in prior lessons, note the addition of the `ls(all.names=TRUE)` function and argument, which will list hidden files (e.g., files that start with a `.` character).

³The data are in four separate columns, Subject, Breed, PctButterfat, and PctProtein. The data are in stacked (e.g., long) format, as opposed to structuring data in unstacked (e.g., wide) format. The difference between the two data formats, stacked and unstacked, is detailed in later lessons. Once again, this lesson starts with a simple confidence-building approach to data organization, with more detail added as skills with R increase.

```

header = TRUE, dec = ".", sep = ",") # Import the .csv file.

getwd()                      # Identify the working directory
ls()                          # List objects
attach(MilkBreedFatProt.df)   # Attach the data, for later use
str(MilkBreedFatProt.df)      # Identify structure
nrow(MilkBreedFatProt.df)     # List the number of rows
ncol(MilkBreedFatProt.df)     # List the number of columns
dim(MilkBreedFatProt.df)      # Dimensions of the data frame
names(MilkBreedFatProt.df)    # Identify names
colnames(MilkBreedFatProt.df) # Show column names
rownames(MilkBreedFatProt.df) # Show row names
head(MilkBreedFatProt.df)     # Show the head
tail(MilkBreedFatProt.df)     # Show the tail
MilkBreedFatProt.df          # Show the entire dataframe
summary(MilkBreedFatProt.df)  # Summary statistics

```

An object called `MilkBreedFatProt.df` has been created by completing this action. This R-based object is a dataframe and it consists of the data originally included in the file `MilkBreedButterfatProtein.csv`, a comma-separated .csv file. To avoid possible conflicts, make sure that there are no prior R-based objects called `MilkBreedFatProt.df`. The prior use of the `rm(list = ls())` functions accommodates this concern, removing all prior objects in the current R session.

It was only necessary to key the filename for the .csv file and not the full pathname since the R working directory is currently set to the directory and/or subdirectory where this .csv file is located (see the Housekeeping section at the beginning of this lesson).

3.3 Organize the Data and Display the Code Book

The dataframe `MilkBreedFatProt.df` is fairly simple and very little, if anything, needs to be done to organize the data. This will not be the case in later lessons, but this lesson was designed to continue with early on easy to follow confidence-building exercises on the use of R, so a simple dataset was selected for this lesson.

For this simple lesson, the `class()`, `str()`, and `duplicated()` functions will be sufficient first steps to be sure that data are organized as desired.⁴

⁴All syntax required for replication of this lesson is presented. Most screen prints generated by the syntax in the main body of the lesson are also presented, with only a few screen prints excluded from presentation. This practice also applies to the figures. The syntax for all figures is presented throughout this lesson, but the output of this graphically-focused syntax is occasionally excluded to keep this lesson at a reasonable length.

R Input

```
class(MilkBreedFatProt.df) # Class
```

R Output

```
[1] "data.frame"
```

R Input

```
str(MilkBreedFatProt.df) # Structure
```

R Output

```
'data.frame': 42 obs. of 4 variables:  
 $ Subject      : Factor w/ 42 levels "SH01","SH02",...: 1 2 ...  
 $ Breed        : int  1 1 1 1 1 1 1 1 1 1 ...  
 $ PctButterfat: num  3.03 4.04 3.14 4.08 3.15 ...  
 $ PctProtein   : num  3.28 3.6 3.36 2.98 3.11 ...
```

R Input

```
duplicated(MilkBreedFatProt.df$Subject) # Duplicates  
# DataFrame$Object notation
```

R Output

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[10] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[19] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[28] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[37] FALSE FALSE FALSE FALSE FALSE FALSE
```

The class and structure for each object seems to be correct and there are no duplicate subjects in the sample. Saying this, a Code Book will help with future understanding of this dataset, even if the data currently seem simple and obvious.

R Input

```
#####
# Code Book for MilkBreedFatProt.df          #
#####
# Subject ..... Factor (e.g., nominal) ##
#           A unique ID assigned to each cow #
##
# Breed ..... Factor (e.g., nominal) ##
#           1 = Holstein and 2 = Jersey (alpha order) #
##
# PctButterfat ..... Numeric (e.g., interval) ##
#           Percent Butterfat that can reach #
#           5.000000 or more #
##
# PctProtein ..... Numeric (e.g., interval) ##
#           Percent Protein that can reach #
#           4.000000 or more #
#####
```

Although this dataset is fairly simple, labels and recoding of individual object variables are used in this lesson. The process calls for attention to detail while the actions are put into place. Recall that the Code Book shows data in their desired formats, which often requires some degree of recoding which has not yet occurred.

Once there is agreement that the data were brought into R in correct format, it is usually necessary to organize the data to some degree:

- The object variable `Subject` is currently viewed as a factor, with each code beginning with S (e.g., `Subject`) and then either H (e.g., `Holstein`) or J (e.g., `Jersey`).
- Integer numeric codes (e.g., 1 and 2) have been used in the original file to identify groups for the factor object variable `Breed`. A set of simple R-based actions can easily: (1) transform (e.g., recode) the object variable `MilkBreedFatProt.df$Breed` into a new object variable, (2) change the recoded object variable from original integer format to enumerated factor format, and (3) apply English text labels for the otherwise cryptic numeric codes (e.g., 1 and 2).
- Values for `PctButterfat` and `PctProtein` are in decimal format and are treated in R as numeric data.

A transformation (typically called a recode action) is needed and the process, using R-based syntax, follows. There may be some unnecessary (perhaps redundant) actions with the following recode activities, but these are purposely done to provide assurance that each variable is in desired format, both original variables and well as the newly-created (e.g., enumerated) variables such as Breed.recode:

R Input

```
MilkBreedFatProt.df$Subject      <- as.factor(
  MilkBreedFatProt.df$Subject)

MilkBreedFatProt.df$Breed.recode <- factor(
  MilkBreedFatProt.df$Breed,
  labels=c("Holstein", "Jersey"))
# Use factor() and not as.factor().

MilkBreedFatProt.df$PctButterfat <- as.numeric(
  MilkBreedFatProt.df$PctButterfat)

MilkBreedFatProt.df$PctProtein   <- as.numeric(
  MilkBreedFatProt.df$PctProtein)
```

Use a wide selection of R-based factors to confirm that the dataset is in good order and that all object variables are organized as desired. Give special notice to the summary() function, comparing output from when this function was previously applied.

R Input

```
getwd()                      # Identify the working directory
ls()                          # List objects
attach(MilkBreedFatProt.df)    # Attach the data, for later use
str(MilkBreedFatProt.df)       # Identify structure
nrow(MilkBreedFatProt.df)      # List the number of rows
ncol(MilkBreedFatProt.df)      # List the number of columns
dim(MilkBreedFatProt.df)       # Dimensions of the data frame
names(MilkBreedFatProt.df)     # Identify names
colnames(MilkBreedFatProt.df)  # Show column names
rownames(MilkBreedFatProt.df)  # Show row names
head(MilkBreedFatProt.df)      # Show the head
tail(MilkBreedFatProt.df)       # Show the tail
MilkBreedFatProt.df            # Show the entire dataframe
summary(MilkBreedFatProt.df)   # Summary statistics
```

R Output

```

  Subject      Breed     PctButterfat    PctProtein
SH01   : 1   Min.   :1.00   Min.   :2.86   Min.   :2.91
SH02   : 1   1st Qu.:1.00  1st Qu.:3.56  1st Qu.:3.34
SH03   : 1   Median  :2.00  Median  :4.54  Median  :3.47
SH04   : 1   Mean    :1.52  Mean    :4.24  Mean    :3.48
SH05   : 1   3rd Qu.:2.00  3rd Qu.:4.82  3rd Qu.:3.66
SH06   : 1   Max.    :2.00  Max.    :5.24  Max.    :4.01
(Other):36
Breed.recode
Holstein:20
Jersey   :22

```

The object variable MilkBreedFatProt.df\$Breed has been retained in original format. However, the object variable MilkBreedFatProt.df\$Breed.recode was created by putting the object variable MilkBreedFatProt.df\$Breed into factor format, in contrast to the original integer-type use of 1 and 2 codes. Labels were then applied in sequential order for this new object, with Holstein used to represent every occurrence of the code 1 and Jersey used to represent every occurrence of the code 2.

Note the formal use of `DataFrame$Object` notation when working with object variables that are part of a dataframe. Note also how the \$ symbol is used to separate the name of the dataframe from the name of the object: `DataFrame$Object`.

3.4 Conduct a Visual Data Check Using Graphics (e.g., Figures)

A complete understanding of the data requires the use of graphics. As expected, different functions are used to generate graphics of factor-type object variables (e.g., `Breed.recode`) and numeric-type object variables (e.g., `PctButterfat` and `PctProtein`):

- For factor-type object variables, the `barplot()` function is usually the best first choice for displaying group membership. The rectangular bars provide a good view of the number of subjects for each breakout group (e.g., `Breed.recode`; Holstein v Jersey).⁵
- For numeric-type object variables, the graphical functions of primary interest are `hist()`, `plot()` and `plot(density())`, `boxplot()`, `stripchart()`,

⁵Avoid the use of pie charts. Pie charts are of questionable value for communicating the concept of varying degrees of membership by breakout groups and they are not well-received by the professional community, even though their use is common in the mass media.

dotchart(), qqnorm(), and qqnorm() and an accompanying qqline(). Many arguments are available to embellish these graphical figures, but for now the figures will be prepared in simple format.

The par(ask=TRUE) function and argument are used to freeze the presentation on the screen, one figure at a time. Note how the top line of the figure, under File—Save as, provides a variety of graphical formats to save each figure, presented in the following order: Metafile, Postscript, PDF, PNG, BMP, TIFF, and JPEG.⁶

Factor-Type Barchart of Breed.recode

Wrap the barplot() function around the table() function to produce a simple barchart of Breed.recode (Fig. 3.1).

R Input

```
par(ask=TRUE)
barplot(table(MilkBreedFatProt.df$Breed.recode),
        main="Breed: Barplot Frequency Distribution",
        col=c("black", "burlywood4"), ylim=c(0,25))
```

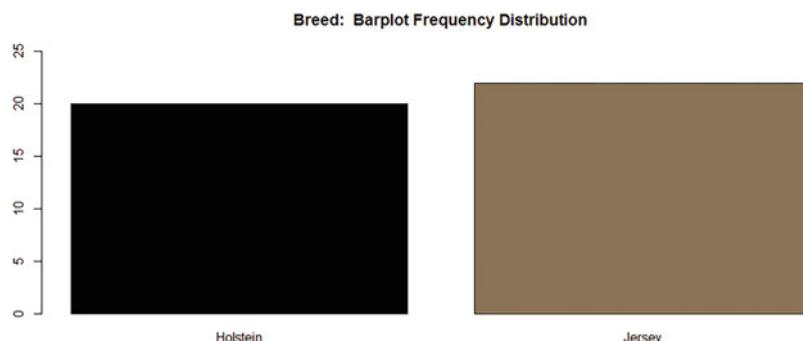


Figure 3.1: Distribution of breed by count—1

Numeric-Type Graphics of PctButterfat and PctProtein

To save space and to also provide a convenient side-by-side view for common figures, look at the way multiple figures are put into one common figure, using the par(mfrow=c()) functions. This technique is especially useful and should be considered not only for exploratory figures but also for final output, when appropriate (Figs. 3.2 and 3.3).

⁶It is also possible to perform a simple copy and paste against each graphical image or to use R syntax to save a graphical image by using R syntax.

R Input

```

par(ask=TRUE)
par(mfrow=c(2,4)) # 8 figures into a 2 row by 4 column grid
hist(MilkBreedFatProt.df$PctButterfat,
     main="Percent Butterfat: Histogram")
plot(MilkBreedFatProt.df$PctButterfat,
      main="Percent Butterfat: Plot")
plot(density(MilkBreedFatProt.df$PctButterfat,
             na.rm=TRUE),           # Required: na.rm=TRUE for missing data
      main="Percent Butterfat: Density Plot")
boxplot(MilkBreedFatProt.df$PctButterfat,
        main="Percent Butterfat: Box Plot")
stripchart(MilkBreedFatProt.df$PctButterfat,
           main="Percent Butterfat: Stripchart")
dotchart(MilkBreedFatProt.df$PctButterfat,
         main="Percent Butterfat: Dotchart")
qqnorm(MilkBreedFatProt.df$PctButterfat,
       main="Percent Butterfat: Q-Q Plot")
qqnorm(MilkBreedFatProt.df$PctButterfat,
       main="Percent Butterfat: Q-Q Plot\nand Q-Q Line")
qqline(MilkBreedFatProt.df$PctButterfat)
# Common figures for a numeric-type object variable
# The \n characters force a new line.

```

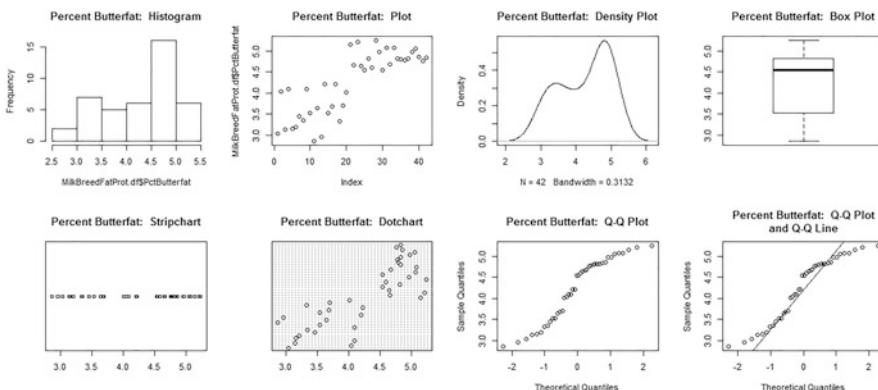


Figure 3.2: Distribution of percent butterfat—1

R Input

```

par(ask=TRUE)
par(mfrow=c(2,4)) # 8 figures into a 2 row by 4 column grid
hist(MilkBreedFatProt.df$PctProtein,

```

```

main="Percent Protein: Histogram")
plot(MilkBreedFatProt.df$PctProtein,
     main="Percent Protein: Plot")
plot(density(MilkBreedFatProt.df$PctProtein,
             na.rm=TRUE),           # Required: na.rm=TRUE for missing data
     main="Percent Protein: Density Plot")
boxplot(MilkBreedFatProt.df$PctProtein,
        main="Percent Protein: Box Plot")
stripchart(MilkBreedFatProt.df$PctProtein,
           main="Percent Protein: Stripchart")
dotchart(MilkBreedFatProt.df$PctProtein,
         main="Percent Protein: Dotchart")
qqnorm(MilkBreedFatProt.df$PctProtein,
       main="Percent Protein: Q-Q Plot")
qqnorm(MilkBreedFatProt.df$PctProtein,
       main="Percent Protein: Q-Q Plot\nand Q-Q Line")
qqline(MilkBreedFatProt.df$PctProtein)
# Common figures for a numeric-type object variable

```

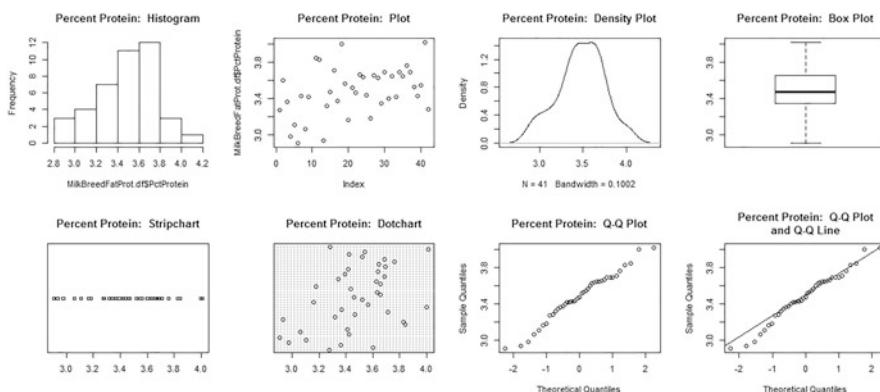


Figure 3.3: Distribution of percent protein—1

After these initial figures are reviewed and when there is agreement that data are correct and the general approach for graphics is acceptable, prepare a more embellished figure if desired. Remember to make colors vibrant and use print that is large and dark, whenever possible, to support future public display of the figure.

From among the many potential packages and functions available to the R community, for factor-type object variables consider use of the epiDisplay package and associated functions. Many functions in the epiDisplay package are used not only to prepare visually appealing graphics, but they also generate useful statistics printed to the screen, such as frequency distributions and percentage representation about the factor object variables in question, Breed.recode for

this lesson. The additional statistics gained by using these specialized functions are information-rich and give reason why specialized functions, such as epiDisplay::tableStack() and epiDisplay::tab1() demand attention Fig. 3.4.

R Input

```
install.packages("epiDisplay", dependencies=TRUE)
library(epiDisplay)           # Load the epiDisplay package.
help(package=epiDisplay)     # Show the information page.
sessionInfo()                # Confirm all attached packages.

epiDisplay::tableStack(Breed.recode,
  dataFrame=MilkBreedFatProt.df,
  by="none", count=TRUE, decimal=2,
  percent=c("column", "row"),
  frequency=TRUE, name.test=TRUE,
  total.column=TRUE, test=TRUE)
# Descriptive statistics, only
```

R Output

	Total
Total	42
Breed.recode	
Holstein	20 (47.62)
Jersey	22 (52.38)

R Input

```
par(ask=TRUE)
epiDisplay::tab1(MilkBreedFatProt.df$Breed.recode,
  decimal=2,                                # Use the tab1() function
  sort.group=FALSE,                          # from the epiDisplay
  cum.percent=TRUE,                          # package to see details
  graph=TRUE,                               # about the selected
  missing=TRUE,                             # object variable. (The
  bar.values=c("frequency"),               # 1 of tab1 is the one
  horiz=TRUE,                               # numeric character and
  cex=1.15,                                 # it is not the letter
  cex.names=1.15,                           # lowercase l).
  cex.lab=1.15, cex.axis=1.15,
  main="Dairy Cow Breeds (Holstein v Jersey) N Values",
```

```
col= c("black", "burlywood4"))
# Prepare a publishable quality barplot of Breed.recode
# and have descriptive statistics printed to the screen.
```

R Output

```
MilkBreedFatProt.df$Breed.recode :
      Frequency Percent Cum. percent
Holstein          20    47.62     47.62
Jersey            22    52.38    100.00
Total             42   100.00    100.00
```

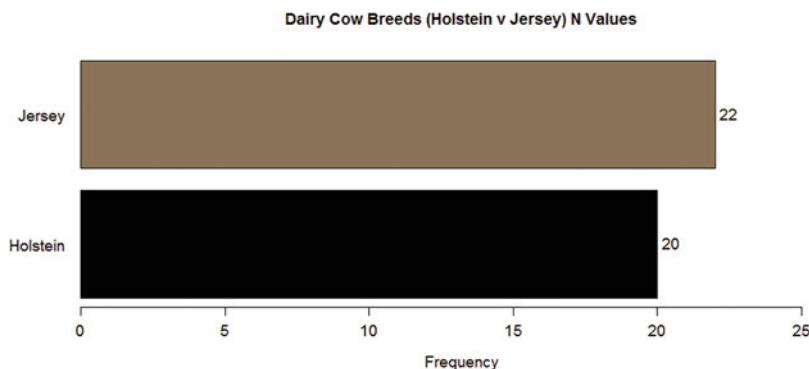


Figure 3.4: Distribution of breed by count—2

There are also many ways to show relationships between and among the numeric variables PctButterfat and PctProtein, individually and by breakout groups. From among the many possible selections, functions found in the ggplot2 package and associated packages are quite popular and for many researchers the functions in these packages are often the first choice for graphical presentations. Look at the way the figures are prepared individually and then put them into one common figure, to enhance easy comparisons of graphical presentations.⁷

R Input

```
install.packages("ggplot2", dependencies=TRUE)
library(ggplot2)           # Load the ggplot2 package.
help(package=ggplot2)       # Show the information page.
sessionInfo()              # Confirm all attached packages.
```

⁷The ggplot2 package and supporting packages are used to produce a variety of figures associated with the concept of *Beautiful Graphics*. These packages are external to what is available when R is first downloaded and it is necessary to actively download these packages to take advantage of their specialized functionality.

```

install.packages("ggthemes", dependencies=TRUE)
library(ggthemes)                      # Load the ggthemes package.
help(package=ggthemes)                  # Show the information page.
sessionInfo()                          # Confirm all attached packages.

install.packages("ggmosaic", dependencies=TRUE)
library(ggmosaic)                      # Load the ggmosaic package.
help(package=ggmosaic)                  # Show the information page.
sessionInfo()                          # Confirm all attached packages.

install.packages("gridExtra", dependencies=TRUE)
library(gridExtra)                     # Load the gridExtra package.
help(package=gridExtra)                # Show the information page.
sessionInfo()                          # Confirm all attached packages.

install.packages("grid", dependencies=TRUE)
library(grid)                           # Load the grid package.
help(package=grid)                     # Show the information page.
sessionInfo()                          # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)                         # Load the scales package.
help(package=scales)                   # Show the information page.
sessionInfo()                          # Confirm all attached packages.

```

In an attempt to make the figures bold and vibrant, but also in an attempt to reduce redundant keying, look at the self-created theme_MacYates, a theme for use with the ggplot2::ggplot() function. The use of additional themes is valuable in an attempt to alter font appearance and size, axis and tick mark presentation, title, appearance and size, etc. However, the use of these additional themes requires many lines of syntax that could easily be placed into syntax that can be reproduced, constructing a user-created theme which is called theme_MacYates().

R Input

```

theme_MacYates <- function(base_size=12, base_family="sans"){
  theme(
    plot.title=element_text(face="bold", size=14, hjust=0),
    axis.title.x=element_text(face="bold", size=14,
      hjust=0.5),
    axis.text.x=element_text(face="bold", size=10),

```

```
axis.title.y=element_text(face="bold", size=14, vjust=1,
angle=90),
axis.text.y=element_text(face="bold", size=12, hjust=1),
legend.title=element_text(face="bold", size=14),
legend.text=element_text(face="bold", size=14),
axis.ticks.x=element_line(size=1.2),
axis.ticks.y=element_line(size=1.2),
axis.ticks.length=unit(0.25,"cm"),
panel.background=element_rect(fill="whitesmoke")
)
}

# hjust - horizontal justification; 0 = left edge to 1 = right
# edge, with 0.5 the default
# vjust - vertical justification; 0 = bottom edge to 1 = top
# edge, with 0.5 the default
# angle - rotation; generally 1 to 90 degrees, with 0 the
# default
```

R Input

```
class(theme_MacYates)
# Confirm the class of the enumerated theme.
```

R Output

```
[1] "function"
```

Self-created themes are quite flexible. Explore many options for size, bold, etc. See what type of theme is best for project requirements.

Now that the theme theme_MacYates() has been created, prepare a set of figures using the ggplot2::ggplot() function to examine the numeric-type object variables, PctButterfat and PctProtein. Note the use of self-documentation and how naming schemes are descriptive and clearly detail the nature of each graphic (Figs. 3.5, 3.6, and 3.7).

R Input

```
HistogramPctButterfatOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
aes(x=PctButterfat)) +
geom_histogram(binwidth=0.30, color="black", lwd=1.25,
fill="cornsilk2") +
```

```

geom_vline(aes(xintercept=mean(PctButterfat, na.rm=TRUE)),
  color="darkred", linetype="dashed", size=1.25) +
geom_vline(aes(xintercept=median(PctButterfat, na.rm=TRUE)),
  color="dodgerblue", linetype="dotted", size=1.25) +
ggttitle(
"Percent Butterfat Produced by Holstein and Jersey
Dairy Cows, Mean - Red and Median - Blue") +
scale_x_continuous(name="Percent Butterfat", limits=c(0,6),
  breaks=seq(0,6,1.0)) +
scale_y_continuous(name="Count", limits=c(0,10),
  breaks=seq(0,10,1)) +
theme_MacYates()
# Generate the figure, but it will not show until using the
# gridExtra::grid.arrange() function.
# Regarding the scales used for this and other figures, it is
# often best to first generate the figure with no attention
# to the later axis scales, to see what the default shows.
# Then, determine individual needs for presentation and
# practice with scale_x_continuous(), scale_y_continuous(),
# and associated options such as name, limits, and breaks.
# Using name, the axis label can be placed within
# scale_x_continuous() as well as scale_y_continuous().
# Notice how limits is used to set the scale that shows
# on the axis, from minimum and maximum.
# See how breaks is used to determine the placement of
# tick marks.

```

```

HistogramPctButterfatBreed.recode <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(x=PctButterfat)) +
geom_histogram(binwidth=0.30, color="black", lwd=1.25,
  fill="cornsilk2") +
facet_grid(. ~ Breed.recode) +
ggttitle(
"Percent Butterfat Produced by Holstein and Jersey
Dairy Cows by Breed: Holstein v Jersey") +
scale_x_continuous(name="Percent Butterfat", limits=c(0,6),
  breaks=seq(0,6,1.0)) +
scale_y_continuous(name="Count", limits=c(0,10),
  breaks=seq(0,10,1)) +
theme(strip.text.x=element_text(face="bold", size=12,
color="navyblue")) +
theme(strip.background=element_rect(fill="wheat1")) +

```

```

theme_MacYates()
# Note how theme(strip.text) and theme(strip.background) are
# placed before theme_MacYates()

par.ask=TRUE)
gridExtra::grid.arrange(
  HistogramPctButterfatOverall,
  HistogramPctButterfatBreed.recode, ncol=2)

```

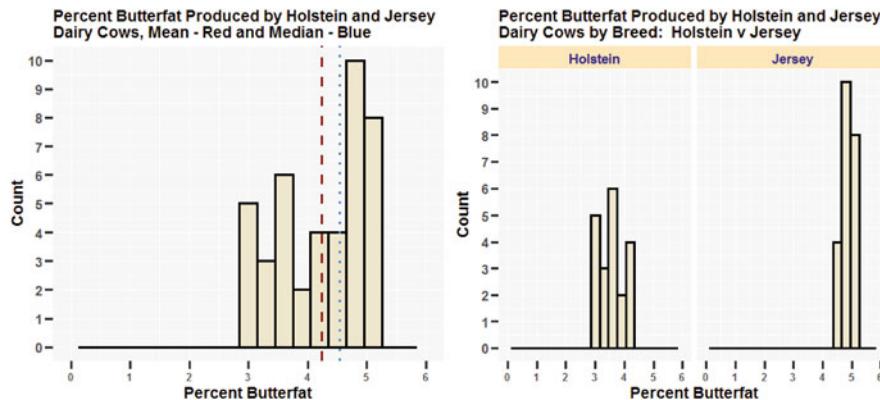


Figure 3.5: Distribution of percent butterfat—2

R Input

```

BoxplotPctButterfatOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(x=factor(0), y=PctButterfat)) +
  geom_boxplot() +
  stat_summary(fun.y=mean, geom="point", shape=1, size=12,
  col="red") +
  ggtitle(
  "Percent Butterfat Produced by Holstein and Jersey
  Dairy Cows, Mean - Red Circle") +
  xlab("Both Breeds: Holstein and Jersey") +
  scale_x_discrete(breaks=NULL) +
  scale_y_continuous(name="Percent Butterfat",
  limits=c(3.0,6.0), breaks=seq(3,6,0.5)) +
  theme_MacYates()
# Note the creation of a dummy variable, x=factor(0).

BoxplotPctButterfatBreed.recode <-
ggplot2::ggplot(MilkBreedFatProt.df,

```

```

aes(x=factor(0), y=PctButterfat)) +
geom_boxplot() +
facet_grid(. ~ Breed.recode) +
ggtitle(
"Percent Butterfat Produced by Holstein and Jersey
Dairy Cows by Breed: Holstein v Jersey") +
scale_x_discrete(name="Breed", breaks=NULL) +
scale_y_continuous(name="Percent Butterfat",
limits=c(3.0,6.0), breaks=seq(3,6,0.5)) +
theme(strip.text.x=element_text(face="bold", size=12,
color="navyblue")) +
theme(strip.background=element_rect(fill="wheat1")) +
theme_MacYates()

par(ask=TRUE)
gridExtra::grid.arrange(
BoxplotPctButterfatOverall,
BoxplotPctButterfatBreed.recode, ncol=2)

```

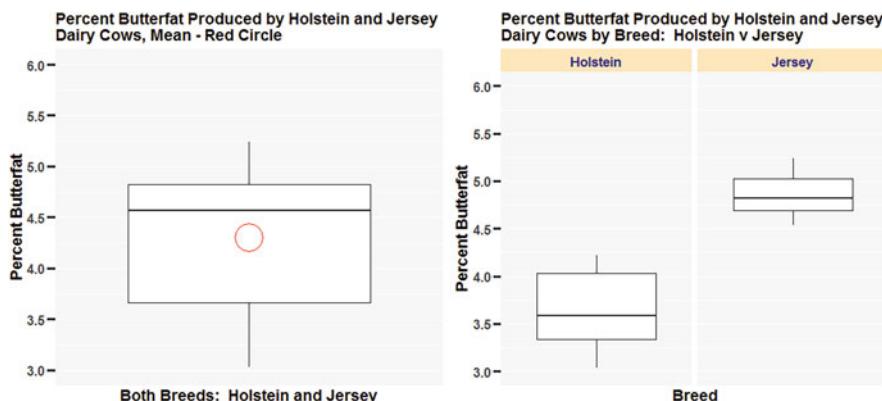


Figure 3.6: Distribution of percent butterfat—3

R Input

```

DensityCurvePctButterfatOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
aes(x=PctButterfat)) +
geom_density(size=1.5, col=("red")) +
ggtitle(
"Percent Butterfat Produced by Holstein and
Jersey Dairy Cows") +
scale_x_continuous(name="Percent Butterfat", limits=c(0,6),

```

```
breaks=seq(0,6,1.0)) +
scale_y_continuous(name="Density", limits=c(0,0.60),
  breaks=seq(0,1,0.1)) +
theme_MacYates()
# There is no easy way to know the best Y axis scale to use
# with a density curve. It is usually best to generate the
# density curve using default settings first, with no use of
# arguments with scale_y_continuous. Then, experiment to see
# the best values for limits and breaks.

DensityCurvePctButterfatBreed.recode <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(x=PctButterfat)) +
  geom_density(size=1.5, col="red") +
  facet_grid(. ~ Breed.recode) +
  ggtitle(
  "Percent Butterfat Produced by Holstein and Jersey
Dairy Cows by Breed: Holstein v Jersey") +
  scale_x_continuous(name="Percent Butterfat", limits=c(0,6),
    breaks=seq(0,6,1.0)) +
  scale_y_continuous(name="Density", limits=c(0,2),
    breaks=seq(0,2,0.5)) +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme_MacYates()
# Generate the figure, but it will not show until using the
# gridExtra::grid.arrange() function.

par(ask=TRUE)
gridExtra::grid.arrange(
  DensityCurvePctButterfatOverall,
  DensityCurvePctButterfatBreed.recode, ncol=2)
```

Use of a histogram, boxplot, and density curve, both overall and side-by-side with breakouts of Percent Butterfat, should give a reasonable sense of data distribution, overall and by breakouts. As spacing and appearance permit, practice with the `gridExtra::grid.arrange()` function to help others understand the data. Comparative side-by-side figures are always helpful to gain a full sense of the data (Figs. 3.8, 3.9, and 3.10).

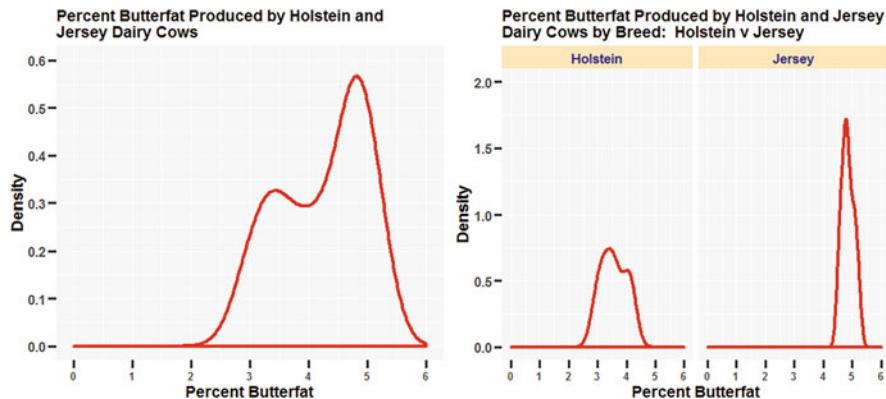


Figure 3.7: Distribution of percent butterfat—4

R Input

```
HistogramPctProteinOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
aes(x=PctProtein)) +
geom_histogram(binwidth=0.30, color="black", lwd=1.25,
fill="cornsilk2") +
geom_vline(aes(xintercept=mean(PctProtein, na.rm=TRUE)),
color="darkred", linetype="dashed", size=0.75) +
geom_vline(aes(xintercept=median(PctProtein, na.rm=TRUE)),
color="dodgerblue", linetype="dotted", size=1.25) +
ggttitle(
"Percent Protein Produced by Holstein and Jersey
Dairy Cows, Mean - Red and Median - Blue") +
scale_x_continuous(name="Percent Protein", limits=c(0,5),
breaks=seq(0,5,1.0)) +
scale_y_continuous(name="Count", limits=c(0,20),
breaks=seq(0,20,5)) +
theme_MacYates()
# Generate the figure, but it will not show until using the
# gridExtra::grid.arrange() function.
# Practice with scale_x_continuous and scale_y_continuous to
# determine the best selections for limits and breaks.
# The red dashed line (PctProtein Mean = 3.48) and the blue
# dotted line (PctProtein Median = 3.47) are quite close to
# each other, given the similar values for mean and median.
```

```
HistogramPctProteinBreed.recode <-
ggplot2::ggplot(MilkBreedFatProt.df,
aes(x=PctProtein)) +
```

```

geom_histogram(binwidth=0.30, color="black", lwd=1.25,
  fill="cornsilk2") +
  facet_grid(. ~ Breed.recode) +
  ggtitle(
    "Percent Protein Produced by Holstein and Jersey
    Dairy Cows by Breed: Holstein v Jersey") +
  scale_x_continuous(name="Percent Protein", limits=c(0,5),
    breaks=seq(0,5,1.0)) +
  scale_y_continuous(name="Count", limits=c(0,20),
    breaks=seq(0,20,5)) +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme_MacYates()
# Note how theme(strip.text) and theme(strip.background) are
# placed before theme_MacYates()

par(ask=TRUE)
gridExtra::grid.arrange(
  HistogramPctProteinOverall,
  HistogramPctProteinBreed.recode, ncol=2)

```

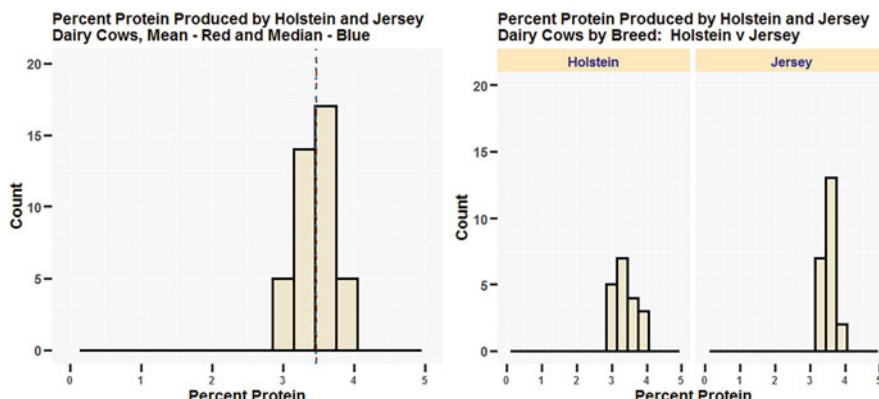


Figure 3.8: Distribution of percent protein—2

R Input

```

BoxplotPctProteinOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(x=factor(0), y=PctProtein)) +
  geom_boxplot() +
  stat_summary(fun.y=mean, geom="point", shape=1, size=12,

```

```

    col="red") +
  ggtitle(
  "Percent Protein Produced by Holstein and Jersey
Dairy Cows, Mean - Red Circle") +
  scale_x_discrete(name="Both Breeds: Holstein and Jersey",
    breaks=NULL) +
  scale_y_continuous(name="Percent Protein",
    limits=c(3.0,4.25), breaks=seq(3.0,4.25,0.5)) +
  theme_MacYates()
# Note the creation of a dummy variable, x=factor(0). Give
# attention, also, to scale_x_discrete() and how it is set to
# present the X axis.

BoxplotPctProteinBreed.recode <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(x=factor(0), y=PctProtein)) +
  geom_boxplot() +
  facet_grid(. ~ Breed.recode) +
  ggtitle(
  "Percent Protein Produced by Holstein and Jersey
Dairy Cows by Breed: Holstein v Jersey") +
  scale_x_discrete(name="Breed", breaks=NULL) +
  scale_y_continuous(name="Percent Protein",
    limits=c(3.0,4.25), breaks=seq(3.0,4.25,0.5)) +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme_MacYates()

par(ask=TRUE)
gridExtra::grid.arrange(
  BoxplotPctProteinOverall,
  BoxplotPctProteinBreed.recode, ncol=2)

```

R Input

```

DensityCurvePctProteinOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(x=PctProtein)) +
  geom_density(size=1.5, col=("red")) +
  ggtitle(
  "Percent Protein Produced by Holstein and"

```

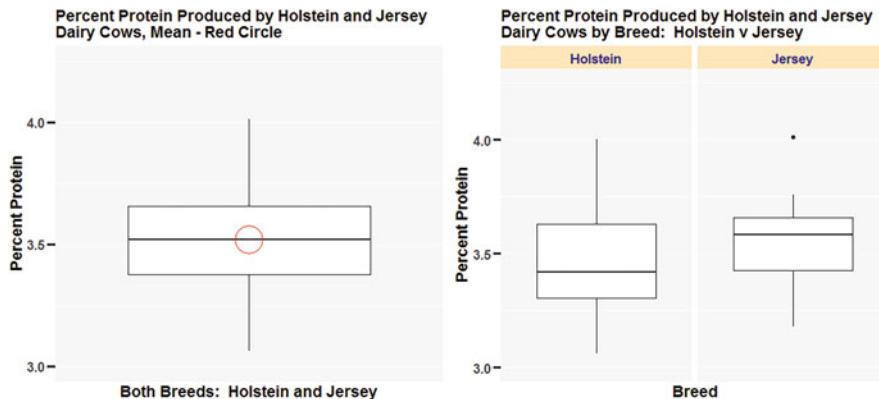


Figure 3.9: Distribution of percent protein—3

```

Jersey Dairy Cows") +
  scale_x_continuous(name="Percent Protein", limits=c(0,6),
    breaks=seq(0,6,1.0)) +
  scale_y_continuous(name="Density", limits=c(0.0,2.5),
    breaks=seq(0.0,2.5,0.50)) +
  theme_MacYates()
# There is no easy way to know the best Y axis scale to use
# with a density curve. It is usually best to generate the
# density curve using default settings first, with no use of
# arguments with scale_y_continuous. Then, experiment to see
# the best values for limits and breaks.

DensityCurvePctProteinBreed.recode <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(x=PctProtein)) +
  geom_density(size=1.5, col="red") +
  facet_grid(. ~ Breed.recode) +
  ggtitle(
  "Percent Protein Produced by Holstein and Jersey
Dairy Cows by Breed: Holstein v Jersey") +
  scale_x_continuous(name="Percent Protein", limits=c(0,6),
    breaks=seq(0,6,1.0)) +
  scale_y_continuous(name="Density", limits=c(0.0,2.5),
    breaks=seq(0.0,2.5,0.50)) +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme_MacYates()
# Generate the figure, but it will not show until using the

```

```
# gridExtra::grid.arrange() function.

par(aspect=TRUE)
gridExtra::grid.arrange(
  DensityCurvePctProteinOverall,
  DensityCurvePctProteinBreed.recode, ncol=2)
```

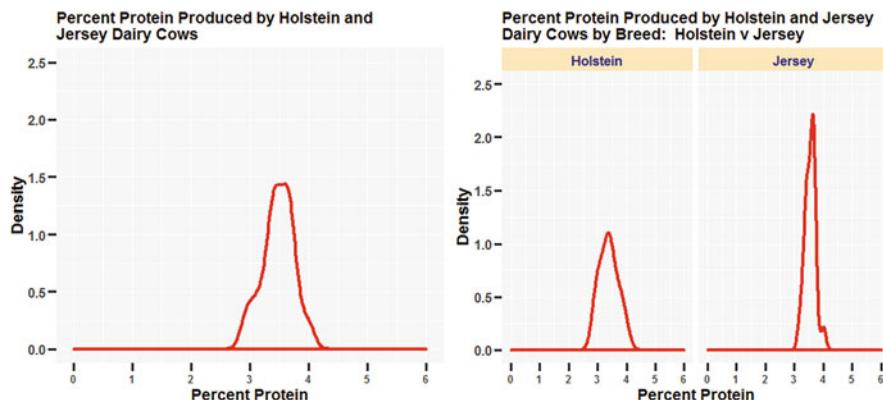


Figure 3.10: Distribution of percent protein—4

Similar to the above comment, side-by-side comparative figures of overall and breakouts for Percent Protein give a good sense of the data and how the data are distributed, for both breeds (e.g., overall) and by breakouts of each breed (e.g., Holstein v Jersey). The `gridExtra::grid.arrange()` function is a useful tool for this aim.

When viewing these figures, remember that the syntax used in this lesson can (and should) be used in future analyses. The modularity of syntax use and reuse is a comparative advantage of R, as opposed to a more trackpointer or mouse-driven approach toward statistical analyses, common to other software packages. In many cases, the syntax for a histogram, boxplot, or density plot used for one set of data can be easily adjusted for a new histogram, boxplot, or density plot—simply alter the syntax, typically changing the dataframe name and object names, and then adjust margins and scales, as needed.

As an ending comment to this section on the use of graphics produced using R, never forget that although inferential tests (e.g., Student's t-Test for Independent Samples, in this lesson) are needed to make final judgment as to whether there are or are not statistically significant ($p \leq 0.05$) differences between Holstein dairy cows and Jersey dairy cows regarding percent butterfat and percent protein, the figures provide a fairly good idea (but still—just an idea) of general trends and how the data compare to each other, individually and by group breakouts. Of course, even when judgment is finally made, always remember the key concepts of replication and representation. This lesson applies

only to two small collections of dairy cows and the data are by no means presented as being representative of the herd or breeds. Again, there will be more discussion on replication and representation in future lessons.

3.5 Descriptive Statistics for Initial Analysis of the Data

The dataframe `MilkBreedFatProt.df` has at least one missing datum. Whether or not it is known in advance that missing data are a concern, it is fairly common to use the `is.na()` function and the `complete.cases()` function against the entire dataframe. Both functions return a TRUE or FALSE response, depending on the function and the outcome of whether data are missing or data are not missing. To make the presentation to the screen compact, wrap the `table()` function around the `is.na()` function and the `complete.cases()` function.

R Input

```
table(is.na(MilkBreedFatProt.df))  
# Check for missing data
```

R Output

FALSE	TRUE
209	1

R Input

```
table(complete.cases(MilkBreedFatProt.df))  
# Check for complete cases
```

R Output

FALSE	TRUE
1	41

For this simple dataset of butterfat and protein for two dairy breeds, the `summary()` function may be all that is necessary to gain a sense of the data. Note how the `summary()` function is applied against selected columns in the dataframe. Recall that the object `Breed` was later transformed into the new object `Breed.recode`.

R Input

```
summary(MilkBreedFatProt.df[, 3:5])
# Apply the summary() function for columns 3 to 5, or
# PctButterfat - PctProtein - Breed.recode.
```

R Output

PctButterfat	PctProtein	Breed.recode
Min. :2.86	Min. :2.91	Holstein:20
1st Qu.:3.56	1st Qu.:3.34	Jersey :22
Median :4.54	Median :3.47	
Mean :4.24	Mean :3.48	
3rd Qu.:4.82	3rd Qu.:3.66	
Max. :5.24	Max. :4.01	
NA's :1		

Although the `summary()` function is often quite sufficient, a full set of descriptive statistics for individual object variables may be desired. To achieve this aim, review the prior lesson *Data Exploration, Descriptive Statistics, and Measures of Central Tendency* for a comprehensive review of the functions used for descriptive statistics, especially: `length()`, `asbio::Mode()`, `median()`, `mean()`, `sd()`, `table()`, and finally `summary()`.

The use of these many one-by-one functions can be tedious and time-consuming. Consider, instead, functions from packages that provide a variety of descriptive statistics all as one attractive and compact output to the screen, allowing easy copy and paste to a word-processed document if needed.

For the factor-type object variables `Breed.recode`, the best first choice to see frequency distribution of Holstein and Jersey dairy cows is likely the `table()` function:

R Input

```
table(MilkBreedFatProt.df$Breed.recode)
```

R Output

Holstein	Jersey
20	22

For more detail on the two breeds (e.g., Holstein v Jersey), revisit the previously used `epiDisplay::tab1()` function, but now using the `graph=FALSE` argument.

This option will generate frequency distribution statistics to the screen, but with no accompanying figure.

R Input

```
epiDisplay::tab1(MilkBreedFatProt.df$Breed.recode,
  decimal=2, sort.group=FALSE, cum.percent=TRUE,
  graph=FALSE, missing=TRUE)
# Frequency distribution shows at the screen.
```

R Output

	Frequency	Percent	Cum. percent
Holstein	20	47.62	47.62
Jersey	22	52.38	100.00
Total	42	100.00	100.00

Revisit the epiDisplay::summ() function as another early choice to fully understand the data, since this function provides descriptive statistics overall, descriptive statistics by breakout groups, and a dotplot of data distribution by breakout groups. Few functions are as useful as the epiDisplay::summ() function. Look at the way the with() function is wrapped around the syntax, so that everything shows in one simple neat display (Fig. 3.11).

R Input

```
par.ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
with(MilkBreedFatProt.df, epiDisplay::summ(PctButterfat,
  by=Breed.recode, main="Percent Butterfat by Breed",
  graph=TRUE))
with(MilkBreedFatProt.df, epiDisplay::summ(PctProtein,
  by=Breed.recode, main="Percent Protein by Breed",
  graph=TRUE))
# Display a dotplot of PctButterfat and then a side-by-side
# dotplot of PctProtein, Breed.recode.
# Look at descriptive statistics as screen output, too.
```

R Output

```
# Percent Butterfat by Breed
For Breed.recode = Holstein
  obs. mean   median   s.d.   min.   max.
  20   3.558  3.53     0.434  2.861  4.218

For Breed.recode = Jersey
  obs. mean   median   s.d.   min.   max.
  22   4.856  4.823    0.209  4.538  5.239

# Percent Protein by Breed
For Breed.recode = Holstein
  obs. mean   median   s.d.   min.   max.
  19   3.385  3.369    0.319  2.907  4

For Breed.recode = Jersey
  obs. mean   median   s.d.   min.   max.
  22   3.555  3.584    0.183  3.18   4.014
```

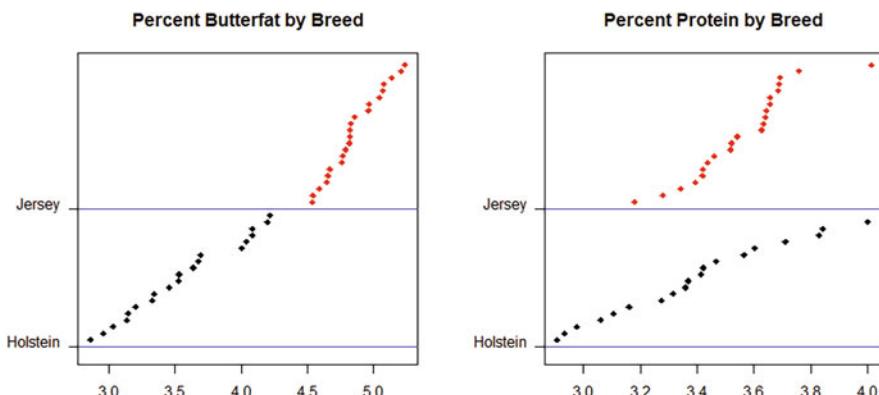


Figure 3.11: Percent butterfat by breed and percent protein by breed—1

The RcmdrMisc::numSummary() function also provides a very convenient and easy to use output of summary statistics, overall and by groups. By default the RcmdrMisc::numSummary() function displays Mean, Standard Deviation, IQR (Interquartile Range, or Q3 minus Q1) Quartiles (0% (Minimum), 25%, 50% (Median), 75%, 100% (Maximum)), N, and NAs (the number of missing values, if any).

R Input

```
install.packages("RcmdrMisc", dependencies=TRUE)
library(RcmdrMisc)                      # Load the RcmdrMisc package.
help(package=RcmdrMisc)                  # Show the information page.
sessionInfo()                           # Confirm all attached packages.

RcmdrMisc::numSummary(MilkBreedFatProt.df$PctButterfat,
                      statistics=c("mean", "sd", "quantiles"))
# Show descriptive statistics of Percent Butterfat, overall.
```

R Output

mean	sd	0%	25%	50%	75%	100%	n
4.23803	0.734791	2.8615	3.5587	4.54057	4.8232	5.23881	42

R Input

```
RcmdrMisc::numSummary(MilkBreedFatProt.df[,c("PctButterfat")],
                      groups=Breed.recode, statistics=c("mean", "sd", "quantiles"))
# Show descriptive statistics of Percent Butterfat, by Breed.
```

R Output

	mean	sd	0%	25%	50%	75%
Holstein	3.55834	0.434054	2.8615	3.18849	3.53034	4.01362
Jersey	4.85593	0.209067	4.5382	4.69403	4.82281	5.02702

	100% data:n
Holstein	4.21788 20
Jersey	5.23881 22

R Input

```
RcmdrMisc::numSummary(MilkBreedFatProt.df$PctProtein,
                      statistics=c("mean", "sd", "quantiles"))
# Show descriptive statistics of Percent Protein, overall.
```

R Output

mean	sd	0%	25%	50%	75%	100%	n	NA
3.47642	0.265861	2.90675	3.34287	3.46742	3.65648	4.01362	41	1

R Input

```
RcmdrMisc::numSummary(MilkBreedFatProt.df[,c("PctProtein")],  
groups=Breed.recode, statistics=c("mean", "sd", "quantiles"))  
# Show descriptive statistics of Percent Protein, by Breed.
```

R Output

	mean	sd	0%	25%	50%	75%
Holstein	3.38514	0.318855	2.90675	3.13364	3.36921	3.58374
Jersey	3.55525	0.182865	3.18029	3.42501	3.58438	3.65726

	100%	data:n	data:NA
Holstein	3.99997	19	1
Jersey	4.01362	22	0

Although the epiDisplay::summ() function may be sufficient for production of descriptive statistics by different groups, there are many other functions that serve the same purpose. The tables::tabular() function can be used to provide detail in a fairly attractive table format that can be easily copied or used in some other fashion in a summary report.

R Input

```
install.packages("tables")
library(tables)                      # Load the tables package.
help(package=tables)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.
```

By default, the tables::tabular() function cannot immediately accommodate missing values, such as the case for Subject SH10—PctProtein. A simple enumeration of a new set of functions for length, min, max, mean, sd, and median will take care of this concern.

R Input

```

Length <- function(x) base::length(x)
Min     <- function(x) base::min(x, na.rm=TRUE)
Max     <- function(x) base::max(x, na.rm=TRUE)
Mean    <- function(x) base::mean(x, na.rm=TRUE)
SD      <- function(x) stats::sd(x, na.rm=TRUE)
Median  <- function(x) stats::median(x, na.rm=TRUE)

# Create new functions that can accommodate missing
# data, using na.rm=TRUE in these examples.

tables::tabular((Breed.recode + 1) ~ (n=1) + Format(digits=2)*
  (PctButterfat)*(Length + Min + Max + Mean + SD + Median),
  data=MilkBreedFatProt.df)
# Breed.recode (rows) by PctButterfat (columns)

```

R Output

PctButterfat						
Breed.recode	n	Length	Min	Max	Mean	SD
Holstein	20	20.00	2.86	4.22	3.56	0.43
Jersey	22	22.00	4.54	5.24	4.86	0.21
All	42	42.00	2.86	5.24	4.24	0.73
						3.53
						4.82
						4.54

R Input

```

tables::tabular((Breed.recode + 1) ~ (n=1) + Format(digits=2)*
  (PctProtein)*(Length + Min + Max + Mean + SD + Median),
  data=MilkBreedFatProt.df)
# Breed.recode (rows) by PctProtein (columns)

```

R Output

PctProtein						
Breed.recode	n	Length	Min	Max	Mean	SD
Holstein	20	20.00	2.91	4.00	3.39	0.32
Jersey	22	22.00	3.18	4.01	3.56	0.18
All	42	42.00	2.91	4.01	3.48	0.27
						3.37
						3.58
						3.47

For those with specialized publishing purposes, the table output can also be passed through the Hmisc::latex() function to generate output suitable for L^AT_EX, a typesetting environment often used in the sciences. Use output gained by wrapping the Hmisc.latex() function around the tables::tabular() function.

Additional embellishments can be added to the table, if desired, by those with specialized knowledge of the L^AT_EXenvironment.

R Input

```
install.packages("Hmisc")
library(Hmisc)                      # Load the Hmisc package.
help(package=Hmisc)                  # Show the information page.
sessionInfo()                       # Confirm all attached packages.

Hmisc:::latex(
tables:::tabular((Breed.recode + 1) ~ (n=1) + Format(digits=2)*
  (PctButterfat)*(Length + Min + Max + Mean + SD + Median),
  data=MilkBreedFatProt.df)
)
# Wrap Hmisc:::latex() around tables:::tabular() to put the
# output into a LATEX-ready format, for those who use LATEX.
# Use LATEX syntax, as desired, to edit the table and include
# centering, color, altered alignment, etc.
```

Table 3.1: Percent butterfat by breed

		PctButterfat					
Breed.recode	n	Length	Min	Max	Mean	SD	Median
Holstein	20	20.00	2.86	4.22	3.56	0.43	3.53
Jersey	22	22.00	4.54	5.24	4.86	0.21	4.82
All	42	42.00	2.86	5.24	4.24	0.73	4.54

R Input

```
Hmisc:::latex(
tables:::tabular((Breed.recode + 1) ~ (n=1) + Format(digits=2)*
  (PctProtein)*(Length + Min + Max + Mean + SD + Median),
  data=MilkBreedFatProt.df)
# Breed.recode (rows) by PctProtein (columns)
)
```

Additional functions from a wide collection of external R packages could be demonstrated, but the above functions should provide a broad representation of how descriptive statistics and measures of central tendency are determined when using R. Keep current with the R literature as there is a constantly evolving set of external packages and functions in these packages to choose from for attractive presentation of descriptive statistics and measures of central tendency.

Table 3.2: Percent protein by breed

Breed.recode	n	Length	PctProtein				
			Min	Max	Mean	SD	Median
Holstein	20	20.00	2.91	4.00	3.39	0.32	3.37
Jersey	22	22.00	3.18	4.01	3.56	0.18	3.58
All	42	42.00	2.91	4.01	3.48	0.27	3.47

3.6 Quality Assurance, Data Distribution, and Tests for Normality

Quality assurance (QA) is by no means a one and done activity. Quality assurance needs to be inclusive throughout the entire research process. A great deal of attention has gone into the preparation of figures that offer a graphical sense of the data. Descriptive statistics have been explored from many perspectives. Summative descriptive statistics were provided for PctButterfat and PctProtein, overall and also by the two breakout groups—Holstein dairy cows and Jersey dairy cows.

As a first attempt at quality assurance and from among the many possible selections, use the epiDisplay::tab1() function to be assured that the factor-type object variable Breed.recode is represented in the dataframe (Fig. 3.12).

R Input

```
par(ask=TRUE)
epiDisplay::tab1(MilkBreedFatProt.df$Breed.recode,
  main="Representation of Dairy Cows by Breed",
  col= c("black", "burlywood4"), cex.main=1.5,
  cex.name=1.2, cex.axis=1.3, cex.lab=1.3)
# Redundant confirmation of subjects, dairy cows
```

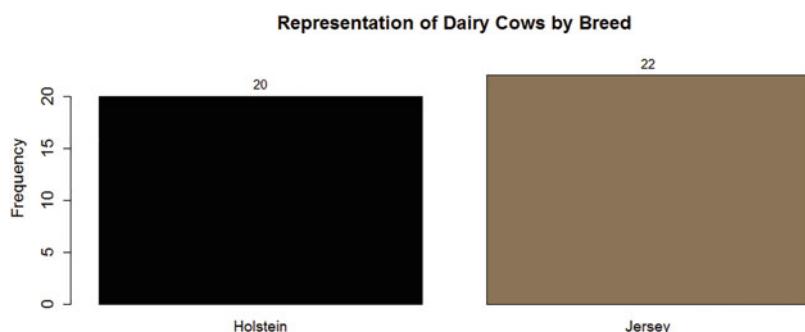


Figure 3.12: Breakouts of breed by count

Along with the vibrant and easily viewed figure, once again note how a summary table is also provided, showing the number of subjects in each breakout group and their percentage representation.

The epiDisplay::summ() function should then be considered as a good choice to review by breed breakout descriptive statistics for PctButterfat and PctProtein.

R Input

```
with(MilkBreedFatProt.df, epiDisplay::summ(PctButterfat,
    by=Breed.recode, graph=FALSE))
# Redundant confirmation of PctButterfat by breed
```

R Output

```
For Breed.recode = Holstein
  obs. mean   median   s.d.   min.   max.
  20   3.558  3.53     0.434  2.861  4.218
```

```
For Breed.recode = Jersey
  obs. mean   median   s.d.   min.   max.
  22   4.856  4.823    0.209  4.538  5.239
```

R Input

```
with(MilkBreedFatProt.df, epiDisplay::summ(PctProtein,
    by=Breed.recode, graph=FALSE))
# Redundant confirmation of PctProtein by breed
```

R Output

```
For Breed.recode = Holstein
  obs. mean   median   s.d.   min.   max.
  19   3.385  3.369    0.319  2.907  4
```

```
For Breed.recode = Jersey
  obs. mean   median   s.d.   min.   max.
  22   3.555  3.584    0.183  3.18   4.014
```

These statistics are provided, again, to renew a sense of the data, overall and by breakout groups. A key interest when viewing the many figures and statistics on central tendency is the concern over normal distribution for numeric object variables or PctButterfat and PctProtein in this lesson. Normal distribution is important in that many inferential tests are only used, appropriately, if the un-

derlying data exhibit normal distribution. Specifically to this lesson, Student's t-Test for Independent Samples is best used with numeric data that follow a pattern of normal distribution. Of course, that ideal is not always met and when that finding occurs a decision needs to be made on the degree of deviation away from normal distribution that can be accepted before another test is selected. The issue of test selection is discussed in more detail later in this lesson.

Not surprisingly, there are more than a few statistical tests that are specifically designed to offer judgment on adherence to normality, including:⁸

- Anderson–Darling test for normality.
- Cramer–von Mises test for normality.
- Lilliefors (Kolmogorov–Smirnov) test for normality.
- Pearson chi-square test for normality.
- Shapiro–Francia test for normality.

It is common to focus on a p-value of either $p \leq 0.05$ or $p \leq 0.01$ when considering normality tests, such as the Anderson–Darling Test for Normality or the Shapiro Test for Normality. Remember that due to the way normality tests are worded, in the affirmative:

- Lower calculated p-values provide evidence against the null hypothesis. That is to say, if the calculated p-value is less than or equal to the criterion p-value (e.g., significance level), the rules-based decision is to reject the null hypothesis and to declare that the data do not follow a pattern of normal distribution.
- Higher calculated p-values fail to provide evidence against the null hypothesis. That is to say, if the calculated p-value is greater than the criterion p-value (e.g., significance level), the rules-based decision is to accept (e.g., fail to reject) the null hypothesis and to declare that the data seem to follow a pattern of normal distribution.

Although the Anderson–Darling is possibly the most frequently used test for addressing normality, note below how each test is applied against PctButterfat and PctProtein. Observe if there are consistent outcomes.

⁸For nearly all statistical tests, the Null Hypothesis is worded in a negative fashion and is typically stated as *There is no statistically significant difference between A and B in terms of C*. Somewhat different, the Null Hypothesis for a normality test is instead worded in the affirmative and is typically stated as *The data follow a normal distribution*. Give attention to this different approach to the way the normality tests are worded when interpreting the p-values, significance levels, and outcomes.

R Input

```
install.packages("nortest", dependencies=TRUE)
library(nortest)           # Load the nortest package.
help(package=nortest)      # Show the information page.
sessionInfo()              # Confirm all attached packages.

# Normality test for PctButterfat

nortest::ad.test(MilkBreedFatProt.df$PctButterfat)
```

R Output

```
Anderson-Darling normality test

data: MilkBreedFatProt.df$PctButterfat
A = 1.4, p-value = 0.0011
```

R Input

```
nortest::cvm.test(MilkBreedFatProt.df$PctButterfat)
```

R Output

```
Cramer-von Mises normality test

data: MilkBreedFatProt.df$PctButterfat
W = 0.2419, p-value = 0.0015
```

R Input

```
nortest::lillie.test(MilkBreedFatProt.df$PctButterfat)
```

R Output

```
Lilliefors (Kolmogorov-Smirnov) normality test

data: MilkBreedFatProt.df$PctButterfat
D = 0.1824, p-value = 0.00121
```

R Input

```
nortest::pearson.test(MilkBreedFatProt.df$PctButterfat)
```

R Output

```
Pearson chi-square normality test

data: MilkBreedFatProt.df$PctButterfat
P = 13.71, p-value = 0.033
```

R Input

```
nortest::sf.test(MilkBreedFatProt.df$PctButterfat)
```

R Output

```
Shapiro-Francia normality test

data: MilkBreedFatProt.df$PctButterfat
W = 0.9223, p-value = 0.0091
```

For the Anderson–Darling test (and generally for the other tests, also), when applied against the numeric object variable `MilkBreedFatProt.df$PctButterfat`, the calculated p-value (`p-value = 0.0011`) is less than the criterion p-value of `p <= 0.05`. This finding brings to question if there is normal distribution for `PctButterfat`.

To address this concern, prepare a simple density curve of `PctButterfat` to once again gain a sense of the distribution pattern (Fig. 3.13):

R Input

```
plot(density(MilkBreedFatProt.df$PctButterfat, na.rm=TRUE),
      main="Percent Butterfat: Density Plot", col="red",
      lwd=3, xlim=c(0,6), font=2)
```

For the numeric object variable `PctButterfat`, overall, there are concerns about normal distribution:

- The calculated p-value using the Anderson–Darling Test for Normality is 0.0011, which is less than the criterion p-value of 0.05.
- The density plot does not begin to approximate the normal curve (e.g., bell-shaped curve).

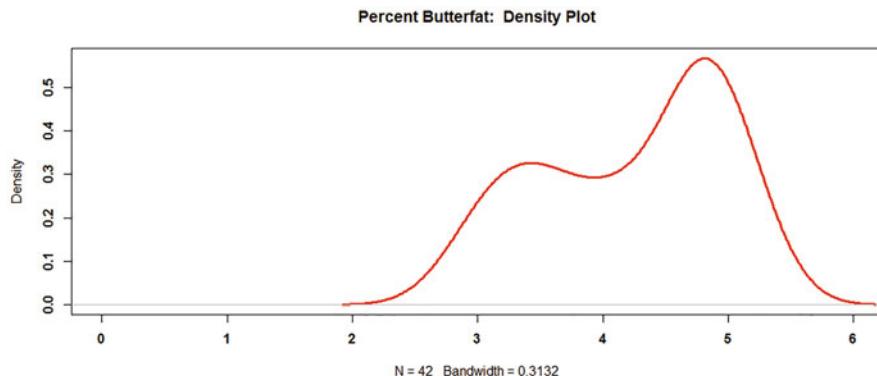


Figure 3.13: Distribution of percent butterfat—5

Can it be said, however, that normal distribution for PctButterfat is a concern by breed, Holstein dairy cows v Jersey cows? To address this concern, consider use of the RVAideMemoire::byf.shapiro() function, which conveniently provides normality statistics of a numeric object variable (e.g., PctButterfat) by breakout groups (e.g., Holstein v Jersey):

R Input

```
install.packages("RVAideMemoire", dependencies=TRUE)
library(RVAideMemoire)          # Load the RVAideMemoire package.
help(package=RVAideMemoire)    # Show the information page.
sessionInfo()                  # Confirm all attached packages.

RVAideMemoire::mshapiro.test(MilkBreedFatProt.df$PctButterfat)
# Normality test of Percent Butterfat, overall
```

R Output

Multivariate Shapiro--Wilk normality test

```
data: ()
W = 0.9095, p-value = 0.00281
```

R Input

```
RVAideMemoire::byf.shapiro(PctButterfat ~ Breed.recode,
data=MilkBreedFatProt.df)
# Normality test of Percent Butterfat by Dairy Breed
```

R Output

```
Shapiro-Wilk normality tests

data: PctButterfat by Breed.recode

      W p-value
Holstein 0.9405 0.2446
Jersey   0.9551 0.3973
```

Note how PctButterfat did not achieve normal distribution when tested against the RVAideMemoire::mshapiro.test() function. However, when the same function was applied by breed it became evident that normality was the case for both breeds. Using the RVAideMemoire::byf.shapiro() function, Multivariate Shapiro-Wilk Normality Test p-values were

```
PctButterfat p-value = 0.00281 Overall
PctButterfat p-value = 0.2446 Holstein
PctButterfat p-value = 0.3973 Jersey
```

Apply the same process against PctProtein, to determine if there is normal distribution overall and by breed:

R Input

```
# Normality tests for PctProtein

nortest::ad.test(MilkBreedFatProt.df$PctProtein)
```

R Output

```
Anderson-Darling normality test

data: MilkBreedFatProt.df$PctProtein
A = 0.3198, p-value = 0.521
```

R Input

```
nortest::cvm.test(MilkBreedFatProt.df$PctProtein)
```

R Output

```
Cramer-von Mises normality test
```

```
data: MilkBreedFatProt.df$PctProtein
W = 0.04798, p-value = 0.531
```

R Input

```
nortest::lillie.test(MilkBreedFatProt.df$PctProtein)
```

R Output

```
Lilliefors (Kolmogorov-Smirnov) normality test
```

```
data: MilkBreedFatProt.df$PctProtein
D = 0.07988, p-value = 0.734
```

R Input

```
nortest::pearson.test(MilkBreedFatProt.df$PctProtein)
```

R Output

```
Pearson chi-square normality test
```

```
data: MilkBreedFatProt.df$PctProtein
P = 5.756, p-value = 0.451
```

R Input

```
nortest::sf.test(MilkBreedFatProt.df$PctProtein)
```

R Output

```
Shapiro-Francia normality test
```

```
data: MilkBreedFatProt.df$PctProtein
W = 0.9808, p-value = 0.613
```

For the Anderson–Darling test (and generally for the other tests, also), when applied against the numeric object variable `PctProtein`, the calculated p-value ($p\text{-value} = 0.521$) is greater than the criterion p-value of $p \leq 0.05$. This finding provides a degree of assurance that there is normal distribution for `PctProtein`.

To add assurance to this finding, prepare a simple density curve of `PctProtein` to once again gain a sense of the distribution pattern for `PctProtein` (Fig. 3.14):

R Input

```
plot(density(MilkBreedFatProt.df$PctProtein, na.rm=TRUE),
      main="Percent Protein: Density Plot", col="red",
      lwd=3, xlim=c(0,6), font=2)
```

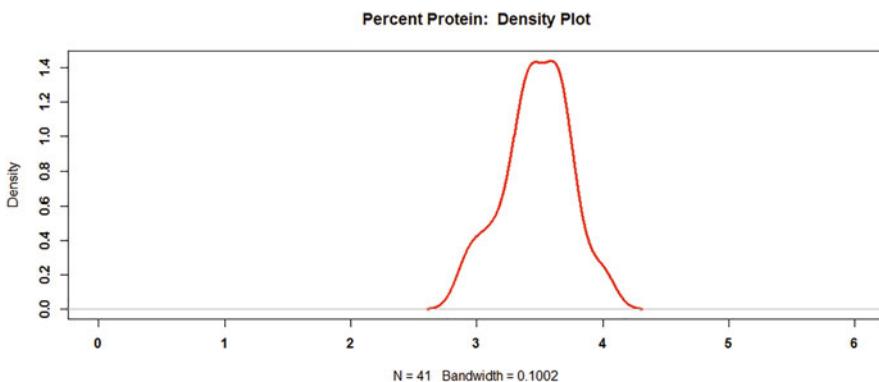


Figure 3.14: Distribution of percent protein—5

For the numeric object variable `MilkBreedFatProt.df$PctProtein`, overall, there is a fair degree of assurance about normal distribution:

- The calculated p-value using the Anderson–Darling Test for Normality is 0.521, which is greater than the criterion p-value of 0.05.
- The density plot seems to approximate the normal curve (e.g., bell-shaped curve).

Although normal distribution seems to be assured, overall, for `PctProtein`, does this finding hold by breed, Holstein dairy cows v Jersey cows? To address this concern, consider use of the `RVAideMemoire::byf.shapiro()` function, which conveniently provides normality statistics of a numeric object variable (e.g., `PctProtein`) by breakout groups (e.g., Holstein v Jersey):

R Input

```
RVAideMemoire::mshapiro.test(MilkBreedFatProt.df$PctProtein)
# Normality test of Percent Protein, overall
```

R Output

Multivariate Shapiro-Wilk normality test

```
data: ()
W = 0.9772, p-value = 0.569
```

R Input

```
RVAideMemoire::byf.shapiro(PctProtein ~ Breed.recode,
  data=MilkBreedFatProt.df)
# Normality test of Percent Protein by Dairy Breed
```

R Output

Shapiro-Wilk normality tests

```
data: PctProtein by Breed.recode

      W p-value
Holstein 0.9671 0.7173
Jersey    0.9617 0.5237
```

Note how PctProtein achieved normal distribution as tested against the RVAideMemoire::mshapiro.test() function. Further, when the same function was applied by breed there was evidence that normality was the case for both breeds. Using the RVAideMemoire::byf.shapiro() function, Multivariate Shapiro-Wilk Normality Test p-values were

```
PctProtein p-value = 0.569 Overall
PctProtein p-value = 0.7173 Holstein
PctProtein p-value = 0.5237 Jersey
```

In many ways these statistical outcomes, using the Anderson–Darling Test for Normality and the Multivariate Shapiro–Wilk Normality Test, reinforce what was seen in the breakout figures previously presented in this lesson. To further reinforce distribution findings about normal distribution, use the ggplot2::ggplot() function to produce Q-Q plots of the numeric object variables

PctButterfat and PctProtein, overall and then by breakouts of the factor-type object variable Breed.recode (e.g., Holstein v Jersey). Use these figures as a visual reinforcement of outcomes from application of the Anderson–Darling test and the Shapiro test (Fig. 3.15).

R Input

```
# Percent Butterfat, Overall
QQPctFatOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(sample=PctButterfat)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.25) +
  ggtitle("")

% Butterfat QQ-Plot and QQ-Line, Overall" ) +
  labs(x = "\nTheoretical", y = "Percent\n") +
  theme_MacYates()

# Percent Butterfat by Breed (two breakouts)
QQPctFatBreed.Recode <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(sample=PctButterfat)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.25) +
  facet_grid(. ~ Breed.recode) +
  ggtitle("")

% Butterfat QQ-Plot and QQ-Line, by Dairy Breed" ) +
  labs(x = "\nTheoretical", y = "Percent\n") +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme_MacYates()

# Percent Protein, Overall
QQPctProteinOverall <-
ggplot2::ggplot(MilkBreedFatProt.df,
  aes(sample=PctProtein)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.25) +
  ggtitle("")

% Protein QQ-Plot and QQ-Line, Overall" ) +
  labs(x = "\nTheoretical", y = "Percent\n") +
  theme_MacYates()
```

```
# Percent Protein by Breed (two breakouts)
QQPctProteinBreed.Recode <-
ggplot2::ggplot(MilkBreedFatProt.df,
aes(sample=PctProtein)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.25) +
facet_grid(. ~ Breed.recode) +
ggtitle("% Protein QQ-Plot and QQ-Line, by Dairy Breed") +
labs(x = "\nTheoretical", y = "Percent\n") +
theme(strip.text.x=element_text(face="bold", size=12,
color="navyblue")) +
theme(strip.background=element_rect(fill="wheat1")) +
theme_MacYates()

par(ask=TRUE)
gridExtra::grid.arrange(
QQPctFatOverall,
QQPctFatBreed.Recode,
QQPctProteinOverall,
QQPctProteinBreed.Recode, ncol=2)
```

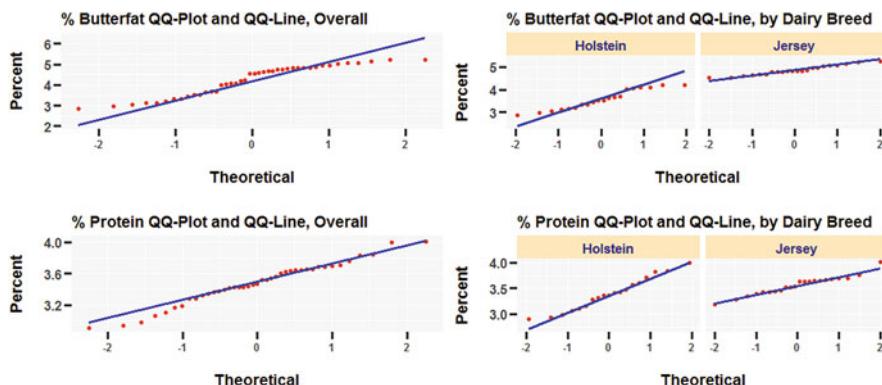


Figure 3.15: Percent butterfat and percent protein overall and by breed

3.7 Statistical Test(s)

Before the statistical test(s) for this lesson are attempted, it may be useful to provide a recap of the data, outcomes up to this point, and planned inferential testing:

- The data are from 42 dairy cows, organized by breed: Holstein ($N = 20$) and Jersey ($N = 22$).

- The data are from analyses of the milk provided by the dairy cows, with a focus specifically on the percent butterfat and percent protein.
- The data are not from a repeated process of data collection and instead the data are from a one-time sampling process. There is no claim that the data are representative of the herds or breeds given the small sample (e.g., 42 subjects, overall) associated with the dataset. Accordingly, any interpretation of outcomes that may be gained from statistical testing should take into account that replication and representation are not in play for this lesson.
- Normality was examined and it was determined, as well as graphically portrayed, that normal distribution was not evident for Percent Butterfat—Overall whereas normal distribution was evident for Percent Protein—Overall. It was interesting to note, however, that normal distribution was evident for both breeds regarding Percent Butterfat and normal distribution was also evident for both breeds regarding Percent Protein.
- Along with a variety of generated figures, descriptive statistics and measures of central tendency of the two measured object variables (e.g., PctButterfat and PctProtein) were completed and the general findings, overall and by breakouts are

% Butterfat Overall ... Mean = 4.24 and SD = 0.73
% Butterfat Holstein ... Mean = 3.56 and SD = 0.43
% Butterfat Jersey Mean = 4.86 and SD = 0.21

% Protein Overall Mean = 3.48 and SD = 0.27
% Protein Holstein Mean = 3.39 and SD = 0.32
% Protein Jersey Mean = 3.56 and SD = 0.18

The parametric Student's t-Test for Independent Samples will be used to determine if there is a statistically significant difference ($p \leq 0.05$) in the mean value of PctButterfat by breed (e.g., Holstein v Jersey) and also to examine the mean value of PctProtein by breed (e.g., Holstein v Jersey). Although normal distribution was not evident for Percent Butterfat—Overall, it is currently judged that Student's t-Test for Independent Samples is sufficiently robust and the concern over normal distribution should not preclude the use of this test.

It may be best to review the two Null Hypotheses (H_0) stated earlier in this lesson, all based on the eventual use of Student's t-Test for Independent Samples:

- H_0 : There is no statistically significant difference ($p \leq 0.05$) in Percent Butterfat of milk by Breed (Holstein v Jersey).

- Ho: There is no statistically significant difference ($p \leq 0.05$) in Percent Protein of milk by Breed (Holstein v Jersey).

From among the many R-based functions that could be used to examine if there is a statistically significant ($p \leq 0.05$) difference between Holstein dairy cows and Jersey dairy cows regarding PctButterfat and PctProtein, start by using the `t.test()` function, which is among the many functions available in the stats package—a package that is automatically made available when R is first downloaded.

Ho: There is no statistically significant difference ($p \leq 0.05$) in Percent Butterfat of milk by Breed (Holstein v Jersey).

R Input

```
t.test(MilkBreedFatProt.df$PctButterfat ~ # Measured variable
       MilkBreedFatProt.df$Breed.recode,          # Grouping variable
       alternative="two.sided",                  # Two-sided t-Test
       paired=FALSE,                            # Independent samples
       na.rm=TRUE,                             # Missing data
       var.equal=TRUE)                         # Equal variance
```

R Output

Two Sample t-test

```
MilkBreedFatProt.df$PctButterfat by
MilkBreedFatProt.df$Breed.recode
t = -12.53, df = 40, p-value = 0.0000000000000201
mean in group Holstein   mean in group Jersey
            3.55834           4.85593
```

Outcome: The calculated p-value (0.0000000000000201) is less than the criterion p-value (0.05). Therefore, there is a statistically significant ($p \leq 0.05$) difference in PctButterfat between the two breeds. For this specific dataset, Jersey cows have a significantly ($p \leq 0.05$) higher mean value for PctButterfat (Mean = 4.85593) than the mean value for PctButterfat of Holstein cows (Mean = 3.55834) and the difference between the two breeds is not due to chance.

Ho: There is no statistically significant difference ($p \leq 0.05$) in Percent Protein of milk by Breed (Holstein v Jersey).

R Input

```
t.test(MilkBreedFatProt.df$PctProtein ~ # Measured variable  
       MilkBreedFatProt.df$Breed.recode, # Grouping variable  
       alternative="two.sided", # Two-sided t-Test  
       paired=FALSE, # Independent samples  
       na.rm=TRUE, # Missing data  
       var.equal=TRUE) # Equal variance
```

R Output

Two Sample t-test

```
MilkBreedFatProt.df$PctProtein by  
MilkBreedFatProt.df$Breed.recode  
t = -2.132, df = 39, p-value = 0.0394  
mean in group Holstein   mean in group Jersey  
            3.38514           3.55525
```

Outcome: The calculated p-value (0.0394) is less than the criterion p-value (0.05). Therefore, there is a statistically significant ($p \leq 0.05$) difference in PctProtein between the two breeds. For this specific dataset, Jersey cows have a significantly ($p \leq 0.05$) higher mean value for PctProtein (Mean = 3.55525) than the mean value for PctProtein of Holstein cows (Mean = 3.38514) and the difference between the two breeds is not due to chance.

Note Imagine if the Null Hypothesis had been focused on a p-value of 0.01, which is not at all uncommon in biostatistics. With a calculated p-value of 0.0394, the difference in mean values for PctProtein between Holstein dairy cows and Jersey dairy cows was statistically significant at $p \leq 0.05$, since 0.0394 is less than 0.05. However, with a calculated p-value of 0.0394, the difference in mean values for PctProtein between Holstein dairy cows and Jersey dairy cows would not be statistically significant at $p \leq 0.01$, since 0.0394 is greater than 0.01. Use this example as a caution that it is always necessary to identify the criterion p-value (e.g., $p \leq 0.10$, $p \leq 0.05$, $p \leq 0.01$, $p \leq 0.001$) when making any inference (e.g., judgment) on statistical significance.

Based on the notion that redundant confirmation of outcomes by the use of multiple statistical testing is a desired activity, use the onewaytests::describe(), onewaytests::st.test(), and onewaytests::wt.test() functions to further examine the prior outcomes gained from use of the t.test() function. Although results when using multiple functions may not be equivalent there should be at least a close degree of parity.

R Input

```
install.packages("onewaytests", dependencies=TRUE)
library(onewaytests)           # Load the onewaytests package.
help(package=onewaytests)      # Show the information page.
sessionInfo()                  # Confirm all attached packages.

# Confirmation of outcomes for Percent Butterfat by Breed

onewaytests::describe(PctButterfat ~ Breed.recode,
                      data=MilkBreedFatProt.df)
# Generate descriptive statistics of PctButterfat by
# Breed.recode and confirm that the results are in close
# parity to prior outcomes.
```

R Output

	n	Mean	Std.Dev	Median	Min	Max	25th
Holstein	20	3.55834	0.434054	3.53034	2.8615	4.21788	3.18849
Jersey	22	4.85593	0.209067	4.82281	4.5382	5.23881	4.69403

	75th	Skewness	Kurtosis	NA
Holstein	4.01362	0.0900536	1.7646	0
Jersey	5.02702	0.2488077	2.0628	0

R Input

```
onewaytests::st.test(PctButterfat ~ Breed.recode,
                     data=MilkBreedFatProt.df, alpha=0.05, na.rm=TRUE,
                     verbose=TRUE)
# Perform a Student's t-Test for two samples and confirm
# that the results are in close parity to prior outcomes.
```

R Output

```
Student's t-Test (alpha = 0.05)

Groups : Holstein vs. Jersey

statistic : -12.187
parameter : 39
p.value   : 0.00000000000000712314
```

```
Result      : Difference is statistically significant.
```

R Input

```
onewaytests::wt.test(PctButterfat ~ Breed.recode,  
                      data=MilkBreedFatProt.df, alpha=0.05, na.rm=TRUE,  
                      verbose=TRUE)  
# Perform a Welch's Unequal Variances t-Test for two  
# samples and confirm that the results are in close  
# parity to prior outcomes.
```

R Output

```
Welch's t-Test (alpha = 0.05)  
-----  
Groups : Holstein vs. Jersey  
  
statistic : -11.6148  
parameter : 24.7202  
p.value   : 0.0000000000166865  
  
Result      : Difference is statistically significant.
```

Application of the stats::t.test() function and the onewaytests::st.test() function provides generally similar outcomes. The Welch's Unequal Variances t-Test is a good selection when there are possible concerns about variance of the two samples (e.g., PctButterfat for Holstein v PctButterfat for Jersey). The calculated p-value is slightly different, but the concluding outcome on statistical significance is the same. For both tests, observe the conveniently placed statement on whether or not there is statistical significance, given the declared p-value (e.g., alpha = 0.05).

R Input

```
# Confirmation of outcomes for Percent Protein by Breed  
  
onewaytests::describe(PctProtein ~ Breed.recode,  
                      data=MilkBreedFatProt.df)  
# Generate descriptive statistics of PctProtein by  
# Breed.recode and confirm that the results are in close
```

```
# parity to prior outcomes.
```

R Output

	n	Mean	Std.Dev	Median	Min	Max	25th
Holstein	19	3.38514	0.318855	3.36921	2.90675	3.99997	3.13364
Jersey	22	3.55525	0.182865	3.58438	3.18029	4.01362	3.42501
					75th	Skewness	Kurtosis
Holstein		3.58374	0.226249	2.16638		1	
Jersey		3.65726	0.169813	3.40730		0	

R Input

```
onewaytests::st.test(PctProtein ~ Breed.recode,
  data=MilkBreedFatProt.df, alpha=0.05, na.rm=TRUE,
  verbose=TRUE)
# Perform a Student's t-Test for two samples and confirm
# that the results are in close parity to prior outcomes.
```

R Output

```
Student's t-Test (alpha = 0.05)

Groups : Holstein vs. Jersey

statistic : -2.13163
parameter : 39
p.value   : 0.0393936

Result     : Difference is statistically significant.
```

R Input

```
onewaytests::wt.test(PctProtein ~ Breed.recode,
  data=MilkBreedFatProt.df, alpha=0.05, na.rm=TRUE,
  verbose=TRUE)
# Perform a Welch's Unequal Variances t-Test for two
# samples and confirm that the results are in close
# parity to prior outcomes.
```

R Output

```
Welch's t-Test (alpha = 0.05)
-----
Groups : Holstein vs. Jersey

statistic : -2.05224
parameter : 27.7586
p.value   : 0.0496846

Result     : Difference is statistically significant.
```

Application of the stats::t.test() function and the onewaytests::st.test() function provides generally similar outcomes. The Welch's Unequal Variances t-Test is a good selection when there are possible concerns about variance of the two samples (e.g., PctProtein for Holstein v PctProtein for Jersey). The calculated p-value is slightly different, but the concluding outcome on statistical significance is the same.

An interesting use of the onewaytests::st.test() function is the declaration about statistical significance, which prints to the screen when using the verbose=TRUE argument. Even with this aid, be sure to compare calculated p-value to criterion p-value, to confirm outcomes.

On the issue of statistical significance, go back to the observation that the comparison of PctProtein by Breed.recode resulted in a calculated p-value of 0.0394. This finding provides evidence that there is a statistically significant difference between Holstein dairy cows and Jersey dairy cows in PctProtein at a criterion p-value of 0.05 (e.g., calculated p-value 0.0394 is less than criterion p-value 0.05). Yet, using a rules-based decision-making process, a calculated p-value of 0.0394 provides evidence that there is no statistically significant difference between Holstein dairy cows and Jersey dairy cows in PctProtein at a criterion p-value of 0.01 (e.g., calculated p-value 0.0394 is greater than criterion p-value 0.01).

3.8 Summary of Outcomes

In this lesson the graphics and statistics provided a great deal of information. Of immediate importance, however, focus on the two Null Hypothesis statements and the outcomes of each.

Ideally, the dataset would have been more complete, with some degree of background and information provided on milk production (e.g., pounds of milk per lactation), age of each dairy cow, feeding practices, etc., It would have also been

helpful if there had been some degree of attention to replication and representation. Yet, that information is not available, being sequestered from the current dataset.

Therefore, for the immediate purpose of this lesson these additional details are not needed. What is needed is a review of the Code Book and that this lesson is focused on a determination of differences, if any, between Holstein dairy cows and Jersey dairy cows regarding the percentage of butterfat and protein in their milk. Go back to the beginning section and the Code Book and then review how:

- There is one factor-type object variable of importance in this lesson—Breed. The original numeric codes for Breed were recoded so that Holstein and Jersey would show in all output, instead of the original numeric codes of 1 and 2.
- There are two numeric-type object variables of importance in this lesson, PctButterfat and PctProtein.
- As a reflection of the realities of the research process for the biological sciences, there was a missing datum in the dataset. Many functions need the use of special arguments to accommodate missing data.

After all of the figures, descriptive statistics, and inferential analyses were attempted, it can be said that: (1) Jersey dairy cows have a higher percentage of butterfat in their milk than Holstein dairy cows ($p \leq 0.05$), and (2) Jersey dairy cows have a higher percentage of protein in their milk than Holstein dairy cows ($p \leq 0.05$).

```
% Butterfat Overall ... Mean = 4.24 and SD = 0.73
% Butterfat Holstein .. Mean = 3.56 and SD = 0.43
% Butterfat Jersey .... Mean = 4.86 and SD = 0.21

% Protein Overall ..... Mean = 3.48 and SD = 0.27
% Protein Holstein .... Mean = 3.39 and SD = 0.32
% Protein Jersey ..... Mean = 3.56 and SD = 0.18
```

A final attempt at visualizing statistically significant ($p \leq 0.05$) differences between the two breeds regarding production of butterfat and protein should put closure to these analyses. The lattice package will be used, largely to serve as an alternate selection to the now far more popular ggplot2 package.

R Input

```
install.packages("lattice", dependencies=TRUE)
library(lattice)           # Load the lattice package.
help(package=lattice)      # Show the information page.
```

```
sessionInfo()                      # Confirm all attached packages.

install.packages("gridExtra", dependencies=TRUE)
library(gridExtra)                 # Load the gridExtra package.
help(package=gridExtra)           # Show the information page.
sessionInfo()                      # Confirm all attached packages.
```

The task here is to use the lattice::densityplot() function to create two figures, one figure organized as a density plot of PctButterfat by Breed.recode and another figure organized as a density plot of PctProtein by Breed.recode. Then, use the gridExtra:: grid.arrange() function to put the two separate figures into one common figure, allowing for a convenient comparison of the main outcomes associated with this lesson (Fig. 3.16).

R Input

```
latticedensityButterfatBreed <-
lattice::densityplot(~ MilkBreedFatProt.df$PctButterfat |
  MilkBreedFatProt.df$Breed.recode, type="density", lwd=6,
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  main="Percent Butterfat by Dairy Cow Breed",
  xlab=list("Percent Butterfat by Dairy Cow Breed:
  Student's t-Test p-value = 0.0000000000000201",
  cex=1.15, font=2),
  xlim=c(0,6), ylim=c(0.00,2.75), # Note ranges
  ylab=list("Density", cex=1.15, font=2),
  aspect=0.25,
  layout = c(1,2), # Note: 1 Column by 2 Rows.
  col="dodgerblue")

latticedensityProteinBreed <-
lattice::densityplot(~ MilkBreedFatProt.df$PctProtein |
  MilkBreedFatProt.df$Breed.recode, type="density", lwd=6,
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  main="Percent Protein by Dairy Cow Breed",
  xlab=list("Percent Protein by Dairy Cow Breed:
  Student's t-Test p-value = 0.0394",
  cex=1.15, font=2),
  xlim=c(0,6), ylim=c(0.00,2.75), # Note ranges
  ylab=list("Density", cex=1.15, font=2),
  aspect=0.25,
```

```

layout = c(1,2), # Note: 1 Column by 2 Row.
col="dodgerblue")

par(ask=TRUE)
gridExtra::: grid.arrange(
  latticedensityButterfatBreed,
  latticedensityProteinBreed, ncol=2)
# Combine different lattice plots.

```

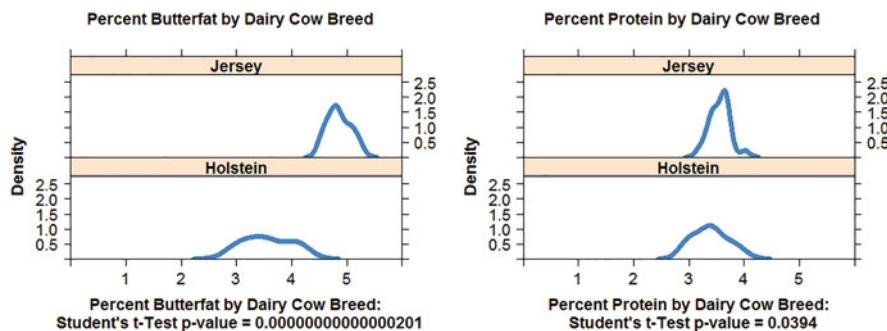


Figure 3.16: Percent butterfat by breed and percent protein by breed—2

The lattice-based density plots provide additional insight into differences between the two breeds (e.g., Holstein v Jersey) regarding Percent Butterfat and Percent Protein in this one-sample collection of milk. Again, replication and representation should be addressed before any management decisions would ever be used to make change to the dairy operation.

Graphical output and statistical output should always be used in tandem, regardless of future reporting requirements and required form and style. For scientific report writing, the statistical outcomes are typically of primary importance in terms of what is included in a publication. Yet, for presentation purposes, graphical images are desired. The ideal analysis includes both and is then tailored to meet specific presentation needs.

3.9 Addendum 1: t-Statistic v z-Statistic

The t-Statistic (e.g., t) begins to approximate the z-Statistic (e.g., z) when the number of subjects in a sample increases, especially after 30 subjects.⁹

⁹The z-Test is similar to the t-Test in that both tests are used to determine if there is a statistically significant difference in the means of two populations. The z-Test and the t-Test are also similar in that, ideally, there is normal distribution (or at least a reasonable semblance of normal distribution) for each population in question. However, there are a few issues where there are differences between the z-Test and the t-Test. An assumption associated with the t-Test is that the standard deviation of each population is unknown whereas for the z-Test the

Saying this, look at the enumerated dataset `Ad1BloodPressure.df` below and determine if the data (hypothetical Systolic Blood Pressure measurements by Gender) support this statement. The `coin::independence_test()` function will be used to conduct the z-Test.

For this demonstration, there are 40 subjects. Does the t-Statistic approximate the z-Statistic? Would the finding hold if the dataset were expanded to 400 subjects?

3.9.1 Create an Enumerated Dataset

R Input

```
Ad1BloodPressure.df <- read.table(textConnection("
```

Ad1Subject	Ad1Gender	Ad1Systolic
S01	Male	147
S02	Female	121
S03	Female	116
S04	Female	77
S05	Male	79
S06	Female	107
S07	Male	119
S08	Female	105
S09	Female	128
S10	Male	155
S11	Male	79
S12	Female	111
S13	Male	122
S14	Male	130
S15	Female	150
S16	Male	170
S17	Male	146
S18	Male	171
S19	Male	153
S20	Female	113
S21	Male	106
S22	Male	147
S23	Female	158
S24	Male	143

```
"))
```

standard deviation of each population should be known. Another key difference is that the z-Test is used for when samples are large while the t-Test is the preferred test when samples are small (typically 30 or fewer datapoints for each sample). Again, the t-statistic begins to closely approximate the z-statistic when sample sizes increases, justifying use of the t-Test for samples that exceed 30 or more subjects.

```

S25      Male          132
S26      Female        131
S27      Male          84
S28      Male          133
S29      Male          165
S30      Female        123
S31      Female        163
S32      Male          119
S33      Female        125
S34      Male          143
S35      Male          106
S36      Male          96
S37      Female        78
S38      Female        125
S39      Male          150
S40      Male          80"), header=TRUE)

getwd()                      # Identify the working directory
ls()                          # List objects
attach(Ad1BloodPressure.df)   # Attach the data, for later use
str(Ad1BloodPressure.df)     # Identify structure
head(Ad1BloodPressure.df, n=3) # Show the head, 1st 3 cases
summary(Ad1BloodPressure.df)  # Summary statistics

```

R Output

	Ad1Subject	Ad1Gender	Ad1Systolic
S01	: 1	Female:16	Min. : 77
S02	: 1	Male : 24	1st Qu.:107
S03	: 1		Median :125
S04	: 1		Mean :125
S05	: 1		3rd Qu.:147
S06	: 1		Max. :171
(Other)	:34		

3.9.2 Calculate the t-Statistic

R Input

```
t.test(Ad1BloodPressure.df$Ad1Systolic ~ # Measured variable
       Ad1BloodPressure.df$Ad1Gender,          # Grouping variable
       alternative="two.sided",                # Two-sided t-Test
       paired=FALSE,                          # Independent
       na.rm=TRUE,                            # Missing data
       var.equal=TRUE)                         # Equal variance
```

R Output

```
Two Sample t-test
Ad1BloodPressure.df$Ad1Systolic by
Ad1BloodPressure.df$Ad1Gender
t = -0.8463, df = 38, p-value = 0.403
mean in group Female   mean in group Male
120.688                 128.125
```

3.9.3 Calculate the z-Statistic

R Input

```
install.packages("coin")
library(coin)                                # Load the coin package.
help(package=coin)                            # Show the information page.
sessionInfo()                                # Confirm all attached packages.

coin::independence_test(Ad1Systolic ~ Ad1Gender,
                        data=Ad1BloodPressure.df)
```

R Output

```
Asymptotic General Independence Test
Ad1Systolic by Ad1Gender (Female, Male)
Z = -0.8494, p-value = 0.396
alternative hypothesis: two.sided
```

In this sample of 40 subjects, within the scope of rounding, t approximates z and the two p-values are also approximately equal. Student's t-Test for Independent Samples is certainly appropriate for small samples ($N \leq 30$ subjects), but the

z-statistic may be of more interest when the number of subjects is fairly large. Thinking of Quality Assurance, the diligent researcher may want to confirm outcomes from a t-Test by subjecting the data to a z-Test to determine if there is parity in the practical significance of outcomes and resulting conclusions and recommendations.

3.10 Addendum 2: Parametric v Nonparametric

The dataframe `MilkBreedFatProt.df` presented an interesting concern regarding normality. As found earlier in this lesson, normal distribution was not evident for Percent Butterfat—Overall whereas normal distribution was evident for Percent Protein—Overall. Ideally, the correct use of Student's t-Test for Independent Samples calls for data that exhibit normality. Student's t-Test for Independent Samples is robust and it is common to use this test when data fail to meet ideal requirements, including normality.

Recognizing this concern about normality, this addendum demonstrates use of the Mann–Whitney U test, which is viewed by some as the nonparametric counterpart to Student's t-Test for Independent Samples. With the Mann–Whitney U Test it is acceptable if data deviate to some degree from normal distribution. Accordingly, view the Mann–Whitney U Test, in this context, as a confirming nonparametric inferential test—a test used to confirm outcomes from the otherwise parametric Student's t-Test for Independent Samples.

R Input

```
onewaytests::mw.test(PctButterfat ~ Breed.recode,
  data=MilkBreedFatProt.df, alpha=0.05, na.rm=TRUE,
  verbose=TRUE)
# Perform a nonparametric Mann--Whitney U Test for two
# samples and confirm that the results are in close
# parity to prior outcomes from the parametric Student's
# t-Test for Independent Samples.
```

R Output

```
Mann--Whitney U Test (alpha = 0.05)
-----
Groups : Holstein vs. Jersey

statistic : 5.46417
p.value   : 0.000000046508

Result     : Difference is statistically significant.
```

When using `stats::t.test()`, the calculated p-value is 0.000000000000000201. When using `onewaytests::mw.test()`, the calculated p-value is 0.000000046508. The two functions provide p-values that although not equal are proximate and result in the same concluding outcome. Given the declared p-value (e.g., `alpha = 0.05`), notice the statement in the output that Difference is statistically significant.

R Input

```
onewaytests::mw.test(PctProtein ~ Breed.recode,
  data=MilkBreedFatProt.df, alpha=0.05, na.rm=TRUE,
  verbose=TRUE)
# Perform a nonparametric Mann--Whitney U Test for two
# samples and confirm that the results are in close
# parity to prior outcomes from the parametric Student's
# t-Test for Independent Samples.
```

R Output

```
Mann--Whitney U Test (alpha = 0.05)
-----
Groups : Holstein vs. Jersey
statistic : 2.01311
p.value   : 0.0441025
Result     : Difference is statistically significant.
```

When using the `stats::t.test()` function the calculated p-value is 0.0394. With the `onewaytests::mw.test()` function the calculated p-value is 0.0441025. The two functions provide p-values that although not equal are proximate and result in the same concluding outcome. Given the declared p-value (e.g., `alpha = 0.05`), notice the statement in the output that Difference is statistically significant.

3.11 Addendum 3: Additional Practice Datasets for Data with Normal Distribution Patterns and Data That Do Not Exhibit Normal Distribution Patterns

Purpose of this Addendum

The purpose of this addendum is to continue with demonstrations on how the R environment supports Student's t-Test for Independent Samples, the inferential test used to determine if there are statistically significant differences

between two samples of continuous data (e.g., Is there a statistically significant ($p \leq 0.05$) difference between females and males regarding calculated Body Mass Index? Is there a statistically significant ($p \leq 0.05$) difference between conventional pyrethroid pest control and organic pest control for Colorado Potato Beetle (*Leptinotarsa decemlineata*) infestations regarding yields (e.g., bushels per acre) of retail-to-consumer tomatoes (*Solanum lycopersicum*)? Is there a statistically significant ($p \leq 0.05$) difference in alfalfa (*Medicago sativa*) between first-cutting protein content and third-cutting protein content?) Student's t-Test for Independent Samples is one of the most commonly used inferential tests and is a common first choice for when two groups are examined against one continuous measured variable and mastery of this test is essential for those who regularly conduct research in the biological sciences.¹⁰

Note In the front matter to this lesson, the syntax is presented and is then immediately followed in most cases by either a copy of screen output or an accompanying figure. That approach is not used in this addendum. View this addendum as practice homework-type bonus materials. Syntax is presented and descriptive text goes along with the syntax. However, screen output and figures are generally excluded and when they show it is only to offer mid-point guidance that analyses follow along in a correct manner. Either key the syntax in this addendum or copy and paste it into an editor—whatever is feasible and most convenient. And then, to use the common expression, Practice—Practice—Practice!

3.11.1 Data with Normal Distribution Patterns

Background: Description of the Data

There are two datasets used in the first part of this addendum. The two datasets are self-generated, each created using R-based tools. The first dataset is created using the `stats::rnorm()` function and the data follow a generally normal distribution pattern. Then, to offer an interesting contrast, the second dataset is created using the `stats::runif()` function and the data do not follow a normal distribution pattern. Be sure to closely examine how R-based functions are used against both datasets, but then notice how there are widely different results between the data that exhibit normal distribution and data that do not exhibit normal distribution.

In an effort to provide consistency and outcomes that allow reproducible results, the `base::set.seed()` function is used at the start of this addendum. Self-generated datasets, such as the two datasets used in the first part of this adden-

¹⁰Syntax is provided throughout this addendum, but of course this syntax is only a suggestion. Experiment and take other approaches to how the data can be analyzed and outcomes presented by using other functions and other arguments. Use this addendum as a confidence-building resource on how R is used with increasingly complex analyses.

dum, if prepared correctly using the `base::set.seed()` function, will then allow for equivalent replication of the data if generated again or if generated by other researchers. This approach for the way practice datasets are self-generated is common when different approaches at data exploration are tried, where data with known values are first used to conceptualize and practice methods for later when real data are used and outcomes are not immediately certain.

Background: Null Hypotheses

Null Hypothesis for Gender (e.g., Female and Male) v Systolic Blood Pressure from the `SBPFM.df` dataset: There is no statistically significant difference ($p \leq 0.05$) difference in Systolic Blood Pressure between female subjects and male subjects. As a comment, due to the way the dataset was created, it is expected that data should ostensibly show normal distribution.

Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

Instead of importing .csv files, the data in the first part of this lesson are self-generated, using R-based tools. After the `base::set.seed()` function is used, two R-based functions are used to self-generate the two datasets:

- The `stats::rnorm()` function is used to generate a dataset that follows a pattern of normal distribution (e.g., `SBPFM.df`).
- The `stats::runif()` function is used to generate a dataset that does not follow a pattern of normal distribution (e.g., `DBPGTE21LTE49.df`).

The many lines of syntax associated with the Housekeeping section is still in use and it does not need to be repeated. If that were not the case, then apply Housekeeping syntax to be sure that data will be organized as desired.

R Input

```
base::set.seed(8)                      # Set the seed.  
# See prior lessons why the set.seed() function  
# is used to provide consistent outcomes that  
# can be reproduced when constructing data that  
# are user-created.
```

Create a dataframe that will eventually consist of data related to Systolic Blood Pressure for female subjects and male subjects. Using R, there are many ways to create this eventual dataframe. A three-step process is used in this lesson, purposely to show each part of how the eventual dataframe is prepared.

The first action is to create a dataframe of Systolic Blood Pressure data for 480 Females, `SBPF.df`.

R Input

```
SBPF.df <- data.frame(
  Gender <- replicate(480, "F"),
  SBP <- round(rnorm(480, mean=118, sd=02))
)
colnames(SBPF.df) <- c("Gender", "SBP")
str(SBPF.df)
names(SBPF.df)
```

Create a dataframe called `SBPF.df` (Systolic Blood Pressure—Female), consisting of separate object variables: (1) replicate F (e.g., Female) 480 times and place the output into the object `Gender`, (2) use the `colnames()` function to apply appropriate labels to the dataframe, (3) apply the `rnorm()` function to create an object vector (e.g., `SBP`, Systolic Blood Pressure) in this dataframe of length=480, mean=118, and `sd=02`, and (4) confirm the contents of the dataframe by applying the `str()` function and the `names()` function.

R Input

```
attach(SBPF.df)      # Attach the dataset
head(SBPF.df, 5)    # Look at the first 5 rows of data
tail(SBPF.df, 5)    # Look at the last 5 rows of data
summary(SBPF.df)    # Summary descriptive statistics
```

Remove the individual object variables `Gender` and `SBP` since they are no longer needed and then use the `ls()` function to confirm that they no longer show in the working directory:

R Input

```
ls()
rm(Gender)
rm(SBP)
ls()
```

The second action is to create a dataframe of Systolic Blood Pressure data for 500 Males, `SBPM.df`.

R Input

```
SBPM.df <- data.frame(
  Gender <- replicate(500, "M"),
```

```
SBP <- round(rnorm(500, mean=120, sd=16))
)
colnames(SBPM.df) <- c("Gender", "SBP")
str(SBPM.df)
names(SBPM.df)
```

Create a dataframe called **SBPM.df** (Systolic Blood Pressure—Male), consisting of separate object variables: (1) replicate M (e.g., Male) 500 times and place the output into the object Gender, (2) use the `colnames()` function to apply appropriate labels to the dataframe, (3) apply the `rnorm()` function to create an object vector (e.g., SBP, Systolic Blood Pressure) in this dataframe of length=500, mean=120, and sd=16, and (4) confirm the contents of the dataframe by applying the `str()` function and the `names()` function.

R Input

```
attach(SBPM.df)      # Attach the dataset
head(SBPM.df, 5)    # Look at the first 5 rows of data
tail(SBPM.df, 5)    # Look at the last 5 rows of data
summary(SBPM.df)    # Summary descriptive statistics
```

Remove the individual object variables `Gender` and `SBP` since they are no longer needed and then use the `ls()` function to confirm that they no longer show in the working directory:

R Input

```
ls()
rm(Gender)
rm(SBP)
ls()
```

The third action is to join the two separate dataframes (e.g., `SBPF.df` and `SBPM.df`) into a common dataframe of Systolic Blood Pressure data for all 980 subjects: 480 Female subjects and 500 Male subjects, `SBPFM.df`.¹¹

¹¹There are many different approaches to the construction of a self-generated dataframe when using R. This method was purposely selected to demonstrate a detailed process and to also demonstrate functions such as the `replicate()` function and the `rbind()` function.

R Input

```
SBPFM.df <- rbind(SBPF.df, SBPM.df)

attach(SBPFM.df) # Attach the dataset
head(SBPFM.df, 5) # Look at the first 5 rows of data
tail(SBPFM.df, 5) # Look at the last 5 rows of data
summary(SBPFM.df) # Summary descriptive statistics
```

Notice the differences in the two self-generated objects for Systolic Blood Pressure, now merged into one common dataset—a stacked dataset of both Female and Male Systolic Blood Pressure values, where the value for Gender (F and M) will be used to distinguish between Systolic Blood Pressure values for the two genders:

- SBPF.df\$SBP, Female Systolic Blood Pressure values.
- SBPM.df\$SBP, Male Systolic Blood Pressure values.
- SBPFM.df\$SBP, Systolic Blood Pressure values for Females and Males.

When viewing the data from the two original datasets (SBPF.df and SBPM.df), notice differences in length, mean, and sd:

R Input

```
length(SBPF.df$SBP); mean(SBPF.df$SBP); sd(SBPF.df$SBP)
length(SBPM.df$SBP); mean(SBPM.df$SBP); sd(SBPM.df$SBP)
```

The N for each object vector is not the same: Female = 480 and Male = 500. There is a difference in Mean SBP: Female approximates 118 and Male approximates 120. There is a difference in SD SBP: Female approximates 02 and Male approximates 16.¹²

Then, as a quality assurance check, look at the combined dataset of SBP values for both genders, to be sure that the descriptive statistics are in range:

R Input

```
length(SBPFM.df$SBP); mean(SBPFM.df$SBP); sd(SBPFM.df$SBP)
```

The dataframe SBPFM.df seems to be in good form. Given this assurance, a few R-based functions are used to be sure that the data are in the correct location.

¹²Again, this dataset was created for teaching purposes and it is not suggested that the data begin to model clinical measurements.

R Input

```
getwd()          # Identify the working directory
ls()            # List objects
```

Given that SBPFM.df consists of 980 subjects and is far too large to show on the screen at one time, the head() function and tail() function were used to examine the data and the entire dataset is purposely not shown.

Organize the Data and Display the Code Book

The dataframe of interest for the first part of this addendum was created by combining two separate dataframes. The Code Book details the nature of the data.

R Input

```
#####
# Code Book for SBPFM.df
#####
#
# Gender ..... Nominal #
#           F (Female) and M (Male) #
#
# SBP ..... Numeric #
#           Range from about 60 to 160, + or - #
#####
```

The base::getwd() function and the base::ls() function were previously used to confirm that each object variable was placed in the expected directory.¹³ Although it may seem redundant, one last check of the data in SBPFM.df is prudent to be sure that the data are as expected.

R Input

```
utils::str(SBPFM.df)      # Identify structure
utils::head(SBPFM.df, n=5) # Show the head, first 5 cases
utils::tail(SBPFM.df, n=5) # Show the tail, last 5 cases
base::summary(SBPFM.df)   # Summary statistics
```

¹³Note the use of Package::Function notation, in an attempt to be fully descriptive.

R Output

```
Gender      SBP
F:480   Min.   : 67
M:500   1st Qu.:115
          Median :118
          Mean   :119
          3rd Qu.:122
          Max.   :168
```

There are many more R-based functions available for further diagnostics about singular object variables and dataframes with multiple object variables, but the functions demonstrated immediately above should be more than sufficient to confirm that the data are in good form.

Conduct a Visual Data Check Using Graphics (e.g., Figures)

For immediate use as a data check, produce throw-away graphics and avoid all embellishments. This approach, using graphics as a quality assurance measure, provides a convenient way to review the data and have confidence that the data are acceptable for later analyses.

For each numeric object variable, it is a good practice to put a histogram, a density plot, a boxplot, and a Q-Q plot all in the same figure, as shown below. If desired or if there are any concerns about the data, it is also useful to use a violin plot and a dot plot to graphically examine numeric data. Each figure provides a different perspective of the data, regardless of whether the figures are ever shared with others.

When preparing comparative figures, such as graphical presentation of Female SBP and Male SBP in the same figure, be sure to use a common scale. Note below how the `xlim()` argument and the `ylim()` arguments were used to achieve this aim. Review outcomes from the `summary()` function to observe the minimum and maximum value for each variable and then use this information, perhaps after a few tries, to decide on the best scale (Fig. 3.17).

R Input

```
# Histogram

par(ask=TRUE)      # Pause
par(mfrow=c(1,2))  # 2 figures - 1 row by 2 column grid
hist(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "F")),
  main="Systolic Blood Pressure - Female:
```

```
N = 480, Mean = 118, and SD = 02",
  col="red",           # Add color
  breaks=15,          # Increase granularity of histogram
  font.lab=2,          # Bold labels
  xlab="SBP",          # X label
  xlim=c(55,185),     # X axis scale
  ylim=c(0,125))      # Y axis scale
  axis(side=1, font=2) # X axis bold
  axis(side=2, font=2) # Y axis bold
hist(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "M")),
  main="Systolic Blood Pressure - Male:
N = 500, Mean = 120, and SD = 16",
  col="red",           # Add color
  breaks=15,          # Increase granularity of histogram
  font.lab=2,          # Bold labels
  xlab="SBP",          # X label
  xlim=c(55,185),     # X axis scale
  ylim=c(0,125))      # Y axis scale
  axis(side=1, font=2) # X axis bold
  axis(side=2, font=2) # Y axis bold
# At the R prompt, key help(subset) to learn more about
# this valuable function. However, know that there are
# other functions and other methods that could have been
# used to select specific subjects from a dataframe.

# Density Plot

par(ask=TRUE)      # Pause
par(mfrow=c(1,2))  # 2 figures - 1 row by 2 column grid
plot(density(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "F",
  na.rm=TRUE))),
  main="Systolic Blood Pressure - Female:
N = 480, Mean = 118, and SD = 02",
  col="red",           # Add color
  lwd=3,               # Thick line
  font.lab=2,          # Bold labels
  xlab="Density",      # X axis label
  xlim=c(50,185),     # X axis scale
  ylim=c(0,0.25))     # Y axis scale
  axis(side=1, font=2) # X axis bold
  axis(side=2, font=2) # Y axis bold
```

```

plot(density(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "M",
  na.rm=TRUE))),
  main="Systolic Blood Pressure - Male:
N = 500, Mean = 120, and SD = 16",
  col="red",           # Add color
  lwd=3,              # Thick line
  font.lab=2,          # Bold labels
  xlab="Density",      # X axis label
  xlim=c(50,185),      # X axis scale
  ylim=c(0,0.25))     # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold

# Boxplot

par(ask=TRUE)      # Pause
par(mfrow=c(1,2))  # 2 figures - 1 row by 2 column grid
boxplot(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "F")),
  main="Systolic Blood Pressure - Female:
N = 480, Mean = 118, and SD = 02",
  xlab="Boxplot",       # X axis label
  ylab="SBP",           # Y axis label
  cex.axis=1.15,         # Axis size
  cex.lab=1.15,          # Label size
  col="red",             # Box color
  lwd=1,                 # Line thickness
  font.lab=2,            # Bold labels
  font=2,                # Bold font
  ylim=c(65,175))       # Y axis scale
boxplot(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "M")),
  main="Systolic Blood Pressure - Male:
N = 500, Mean = 120, and SD = 16",
  xlab="Boxplot",       # X axis label
  ylab="SBP",           # Y axis label
  cex.axis=1.15,         # Axis size
  cex.lab=1.15,          # Label size
  col="red",             # Box color
  lwd=1,                 # Line thickness
  font.lab=2,            # Bold labels
  font=2,                # Bold font
  
```

```
ylim=c(65,175))      # Y axis scale

# Q-Q Plot

par.ask=TRUE)      # Pause
par(mfrow=c(1,2))  # 2 figures - 1 row by 2 column grid
qqnorm(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "F")),
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Systolic Blood
Pressure - Female: N = 480, Mean = 118,
and SD = 02",
  col="blue", xlim=c(-4,4), ylim=c(0,175), font.axis=2,
  font.lab=2
)
qqline(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "F")),
  col="red", lwd=4, lty=2
)
qqnorm(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "M")),
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Systolic Blood
Pressure - Male: N = 500, Mean = 120,
and SD = 16",
  col="blue", xlim=c(-4,4), ylim=c(0,175), font.axis=2,
  font.lab=2
)
qqline(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "M")),
  col="red", lwd=4, lty=2
)

# Dotchart

par.ask=TRUE)      # Pause
par(mfrow=c(1,2))  # 2 figures - 1 row by 2 column grid
dotchart(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "F")),
  main="Systolic Blood Pressure - Female:
N = 480, Mean = 118, and SD = 02",
  xlab="SBP",           # X axis label
  cex.axis=1.15,        # Axis size
  cex.lab=1.15,         # Label size
  pch=16,               # Dot symbol
```

```

pt.cex=0.75,           # Dot size
lcolor="cornsilk",      # Line color
col="red",              # Dot color
lwd=1,                  # Line thickness
font.lab=2,             # Bold labels
font=2,                  # Bold font
xlim=c(60,175))        # X axis scale

dotchart(
  (subset(SBPFM.df$SBP, SBPFM.df$Gender == "M")),
  main="Systolic Blood Pressure - Male:
N = 500, Mean = 120, and SD = 16",
  xlab="SBP",            # X axis label
  cex.axis=1.15,          # Axis size
  cex.lab=1.15,           # Label size
  pch=16,                 # Dot symbol
  pt.cex=0.75,             # Dot size
  lcolor="cornsilk",        # Line color
  col="red",                # Dot color
  lwd=1,                  # Line thickness
  font.lab=2,               # Bold labels
  font=2,                  # Bold font
  xlim=c(60,175))        # X axis scale

```

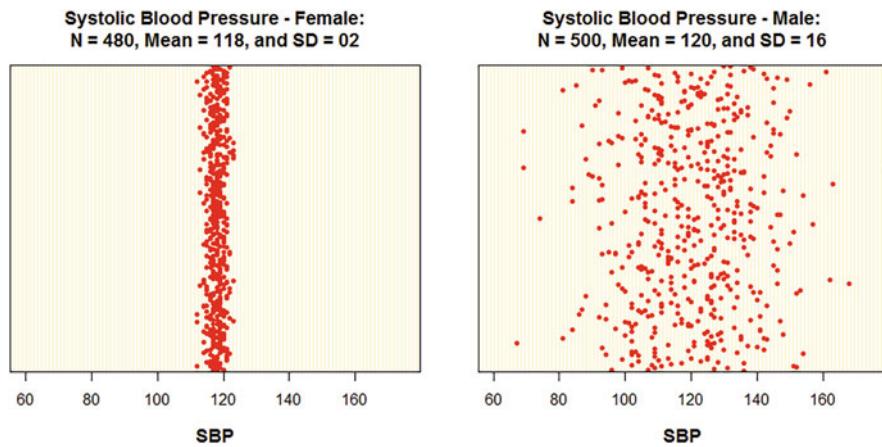


Figure 3.17: Distribution of systolic blood pressure by gender

For an alternate view of how a dotchart can be easily generated, look at the `epiDisplay::dotplot()` function. Only a few arguments were used to embellish the figure and the presentation otherwise follows along with default settings from the `epiDisplay` package.

R Input

```
epiDisplay::dotplot(SBPFM.df$SBP, by=SBPFM.df$Gender,  
                    ylab="Gender",  
                    cex.X.axis=1.25, cex.Y.axis=1.25)
```

For another approach to producing figures in R, look again at the way the ggplot2 package is used to produce graphics, a violin plot in this example. As time permits, explore how different options for presentation are available using themes.

Note Previously, the ggplot2 package was downloaded along with other packages often associated with use of the ggplot2 package. However, the ggplot2 package is also part of tidyverse, a collection of many R-based packages that is gaining increased attention and use. For this example, look how these many packages are downloaded all at one time. As a brief comment, with far more information available on the Internet, when tidyverse is installed the following core tidyverse packages are loaded and made available:

```
dplyr ..... data manipulation  
forcats .... factors  
ggplot2 .... data visualisation  
purrr ..... functional programming  
readr ..... data import  
stringr .... strings  
tibble ..... data frames  
tidyverse ..... organize and clean data
```

As well as installing and loading the core tidyverse packages, tidyverse, when installed, will also install a collection of less frequently used supporting packages. These packages (listed below) need to be individually loaded and address a wide range of functionality:

```
broom ..... convert models into tidy data  
feather ... interface with other languages  
haven ..... import proprietary statistics files  
hms ..... times  
httr ..... web usage  
jsonlite .. interface with JavaScript  
lubridate . dates and times  
modelr .... modelling  
readxl .... import xls and xlsx files  
rvest ..... Web scraping  
xml2 ..... XML markup
```

Because the `ggplot2` package and associated packages were previously installed, it is not necessary to install and load the `tidyverse` package. If that were the case, look at the syntax below on how the `tidyverse` package (a collection of packages) is downloaded. Because the `#` comment character is placed before the syntax, the syntax will not be put into action, but it is still helpful to show how the `tidyverse` package is easily downloaded.

```
# install.packages("tidyverse", dependencies=TRUE)
# library(tidyverse)           # Load the tidyverse package.
# help(package=tidyverse)      # Show the information page.
# sessionInfo()               # Confirm all attached packages.
# Obtain the ggplot2 package and other packages associated
# with the tidyverse package, core and auxillary.
# Note: Be patient. It may take ten minutes or more for the
# many packages associated with tidyverse to install and then
# load.
```

R Input

```
# Violin Plot

ggplot2::ggplot(SBPFM.df, aes(x=Gender, y=SBP,
group=Gender)) +
  geom_violin(aes(fill=Gender)) +           # Violin Plot geom
  labs(title =
    "Violin Plot of Systolic Blood Pressure by Gender") +
  theme_bw() +
  theme(plot.title=element_text(face="bold", size=14)) +
  theme(axis.title.x=element_text(face="bold", size=14)) +
  theme(axis.text.x=element_text(face="bold", size=14)) +
  theme(axis.title.y=element_text(face="bold", size=14)) +
  theme(axis.text.y=element_text(face="bold", size=14)) +
  coord_flip()      # Rotate (e.g., flip) the presentation
```

Descriptive Statistics for Initial Analysis of the Data

Going beyond the few descriptive statistics shown above, such as use of the `summary()` function, the first task is to obtain a fairly good understanding of the data by using the most common functions for descriptive statistics and measures of central tendency. Functions such as `mean()`, `median()`, `sd()`, and `range()` are all essential first choices.

However, the use of these many one-by-one functions can be tedious and time-consuming. Consider, instead, functions from packages that provide a variety of descriptive statistics all as one attractive and compact output to the screen, allowing easy copy and paste to a word-processed document, if needed.

From among the many packages with functions that provide such output, the epiDisplay::summ() function is often a first choice since this function provides descriptive statistics overall, descriptive statistics by breakout groups, a dotplot and box plot of overall data distribution, and a dotplot of data distribution by breakout groups. Few functions are as versatile as the epiDisplay::summ() function. Look at the way the with() function is wrapped around the algorithm, so that everything shows in one simple neat display.

R Input

```
par.ask=TRUE)
with(SBPFM.df, epiDisplay::summ(SBP, graph=TRUE, box=TRUE))
  # Display a dotplot and boxplot of SBP datapoints, overall.
  # Look at descriptive statistics as screen output, too.

par.ask=TRUE)
with(SBPFM.df, epiDisplay::summ(SBP, by=Gender, graph=TRUE))
  # Display a dotplot of SBP datapoints, by Gender breakouts.
  # Look at descriptive statistics as screen output, too.
```

The RcmdrMisc::numSummary() function also provides a very convenient and easy to use output of summary statistics, overall and by groups. By default the RcmdrMisc::numSummary() function displays Mean, Standard Deviation, IQR (Interquartile Range, or Q3 minus Q1) Quartiles (0% (Minimum), 25%, 50% (Median), 75%, 100% (Maximum)), N, and NAs (the number of missing values, if any).

R Input

```
RcmdrMisc::numSummary(SBPFM.df$SBP)
  # Show descriptive statistics, overall.
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n
118.789	11.839	7	67	115	118	122	168	980

R Input

```
RcmdrMisc::numSummary(SBPFM.df[,c("SBP")], groups=Gender)
  # Show descriptive statistics, by breakout groups.
```

R Output

```
mean      sd IQR  0% 25% 50% 75% 100% data:n
F 117.871 2.08984  2 112 117 118 119 123    480
M 119.670 16.40754 22 67 109 120 131 168    500
```

The `arsenal::tableby()` function is equally useful in that it provides breakout descriptive statistics and totals—all in a very attractive summary table. Look at the value of this function and how it fits into future plans for when R output is placed into a word-processed document.

R Input

```
install.packages("arsenal", dependencies=TRUE)
library(arsenal)                      # Load the arsenal package.
help(package=arsenal)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

SBPFMGenderSBPBreakouts <-
  arsenal::tableby(Gender ~ SBP, data=SBPFM.df)
# Create the arsenal::tableby() model.

summary(SBPFMGenderSBPBreakouts, text=TRUE)
# Show output from the arsenal::tableby() model.
```

The `s20x::summaryStats()` function is also quite good for generating detailed output of descriptive statistics.

R Input

```
install.packages("s20x", dependencies=TRUE)
library(s20x)                          # Load the s20x package.
help(package=s20x)                     # Show the information page.
sessionInfo()                         # Confirm all attached packages.

with(SBPFM.df, s20x::summaryStats(SBP))
# Show descriptive statistics, overall.

with(SBPFM.df, s20x::summaryStats(SBP ~ Gender))
# Show descriptive statistics, by breakout groups.
```

R Output

	Sample Size	Mean	Median	Std Dev	Midspread
F	480	117.871	118	2.08984	2
M	500	119.670	120	16.40754	22

Notice how the mode is not included in the output of these functions. If required, use the `asbio::Mode()` function since an oddity of the R environment is that a function to determine the most frequently occurring datum (e.g., mode) is not part of the set of packages and functions available when R is first downloaded.

R Input

```
install.packages("asbio", dependencies=TRUE)
library(asbio) # Load the asbio package.
help(package=asbio) # Show the information page.
sessionInfo() # Confirm all attached packages.

asbio::Mode((subset(SBPFM.df$SBP, SBPFM.df$Gender == "F")))
# F SBP Note == and not = immediately above.
asbio::Mode((subset(SBPFM.df$SBP, SBPFM.df$Gender == "M")))
# M SBP Note == and not = immediately above.
asbio::Mode(SBPFM.df$SBP) # F and M SBP
```

There is a nearly exhaustive selection of functions available for descriptive statistics. Search out other packages with functions that provide summary and breakout descriptive statistics, to learn more about the potential for this vital area of research and biostatistics.

Quality Assurance, Data Distribution, and Tests for Normality

The `epiDisplay::tab1()` function is a good first choice to be assured that the factor-type object variable `Gender` is represented in the dataframe correctly and as expected, where there are a few more Male subjects in the `SBPFM.df` dataframe than Female Subjects:

R Input

```
epiDisplay::tab1(SBPFM.df$Gender, col=c("blue","red"),
main="Representation of SBPFM.df Subjects by Gender",
cex.main=1.5, cex.name=1.2, cex.axis=1.3, cex.lab=1.3)
```

Along with the vibrant and easily viewed figure, note how a summary table is also printed to the screen, showing number of subjects in each breakout group and their percentage representation.

The `epiDisplay::summ()` function should then be considered as a good choice to provide a sense of Systolic Blood Pressure for each of the two Gender breakout groups.

R Input

```
with(SBPFM.df, epiDisplay::summ(SBP, by=Gender))
```

Along with the descriptive statistics by breakout groups printed to the screen, be sure to see how the above syntax generated the accompanying figure that is automatically displayed when using the `epiDisplay::summ()` function.

Many inferential tests are only used, appropriately, if the underlying data exhibit normal distribution. Although the Anderson–Darling is possibly the most frequently used test for addressing normality, note how each test is applied against the object variable `SBPFM.df$SBP` and then observe if there are consistent (but not necessarily equivalent) outcomes:

R Input

```
nortest::ad.test(SBPFM.df$SBP)
nortest::cvm.test(SBPFM.df$SBP)
nortest::lillie.test(SBPFM.df$SBP)
nortest::pearson.test(SBPFM.df$SBP)
nortest::sf.test(SBPFM.df$SBP)
```

For the Anderson–Darling test (and for the other tests also), when applied against the numeric object variable `SBPFM.df$SBP`, the calculated p-value is less than the criterion p-value of $p \leq 0.05$. This finding brings to question if there is normal distribution for Systolic Blood Pressure.

However, how is it possible that the object variable `SBPFM.df$SBP` fails to exhibit normal distribution, as evidenced by the above tests? After all, the `rnorm()` function was used to create the distribution pattern of Systolic Blood Pressure values for Female subjects and later for Male subjects. Perhaps a graphical image of `SBPFM.df$SBP` for all 980 subjects will offer confirmation of these above statistical outcomes of the Anderson–Darling test (and other tests, too):

R Input

```
par(ask=TRUE)      # Pause
qqnorm(SBPFM.df$SBP,
       main="Q-Q Plot (Blue) and Q-Q Line (Red) of Systolic Blood
```

```
Pressure - All Subjects, Female and Male",
col="blue", xlim=c(-4,4), ylim=c(0,175), font.axis=2,
font.lab=2)
qqline(SBPFM.df$SBP, col="red", lwd=4, lty=2)
```

The resulting image from the `qqnorm()` and `qqline()` functions provides confirming evidence that normal distribution has not been achieved. Yet again, how is this outcome possible given use of the `rnorm()` function to create the SBP values for Female and Male subjects? It is far beyond the purpose of this lesson to offer a detailed answer to that question other than to suggest (and only suggest) that even though the `rnorm()` function was used to create data for both Female and Male subjects, when SBP values for Female and Male subjects were joined into one common object variable, the extremely large difference in standard deviation for Female subjects (`mean=118, sd=02`) and Male subjects (`mean=120, sd=16`) is so great that when the data were combined into a common object variable normal distribution should not have been assumed. Variance is an issue that demands attention and that issue shows when figures reinforce provided statistics.¹⁴

Given this challenge, it may be useful to examine the normality of Systolic Blood Pressure for each of the two genders, Female and Male. The Anderson–Darling test will be used, but it will be deployed by using the `RcmdrMisc::normalityTest()` function. As a comment, this type of action is common, where the `RcmdrMisc::normalityTest()` function is linked to functions included in the `nortest` package, or specifically the `nortest::ad.test()` function.

R Input

```
RcmdrMisc::normalityTest(SBPFM.df$SBP,
test=c("ad.test"),
groups=SBPFM.df$Gender)
```

R Output

```
-----
SBPFM.df$Gender = F

Anderson-Darling normality test

data: SBPFM.df$SBP
```

¹⁴Recall that **SBPFM.df** was user-created for teaching purposes, largely to press the point of concerns about overall normal distribution for when breakout groups have such widely different standard deviations.

```
A = 4.561, p-value = 0.0000000000253
```

```
-----
SBPFM.df$Gender = M
```

```
Anderson-Darling normality test
```

```
data: SBPFM.df$SBP
A = 0.4542, p-value = 0.269
```

```
-----
p-values adjusted by the Holm method:
```

unadjusted	adjusted
------------	----------

F 0.00000000002529	0.00000000005058
--------------------	------------------

M 0.2686	0.2686
----------	--------

For Female subjects, the calculated p-value (0.0000000000253) does not exceed $p \leq 0.05$. Accordingly, there is a question if the SBP values for Female subjects exhibit normal distribution. Although the rnorm() function was used to create the SBP values for Female subjects, recall that the standard deviation was set to 02. It may be possible that this exceptionally low standard deviation and associated variance, given the set of SBP values, has an influence on the Anderson–Darling outcomes for Female subjects, but again this theoretical issue is far beyond the purpose of this lesson.

In contrast, the calculated Anderson–Darling normality p-value (0.269) for Male subjects exceeds $p \leq 0.05$. Saying this, there is a degree of assurance that SBP values for Male subjects (Male sd = 16) exhibit normal distribution.

In many ways these statistical outcomes, using the Anderson–Darling test, reinforce what was seen in the breakout figures. Again, the exceptionally low standard deviation for Female subjects may be problematic regarding normal distribution for Females SBP values. But of course, the values for use with the rnorm() function were purposely selected to make this teaching dataset interesting, reinforcing the need for Quality Assurance throughout the research process. Although the rnorm() function was used to create two separate object variables (SBP values for Female subjects and SBP values for Male subjects), attention to Quality Assurance needs to be considered throughout the entire process. To paraphrase the expression *Trust, but verify*. it is a good practice for biostatisticians to *Assume little and check everything*.

Statistical Test(s)

Student's t-Test for Independent Samples is possibly the most frequently used inferential statistical test when it is necessary to determine if there is a statistically significant difference between the means of two groups, however, the concept of group is defined. Student's t-Test for Independent Samples is focused on one measured object variable (e.g., Pounds of Milk Production for First Lactation Heifers, Systolic Blood Pressure, Weight, etc.) and differences in the mean of this measured object variable between two separate groups (e.g., Breed 1 v Breed 2, Female v Male, High Carbohydrate Diet v Low Carbohydrate Diet, etc.).

When using Student's t-Test for Independent Samples, it is assumed that:

- There should be a reasonable semblance of normal distribution of the measured object variable, overall and for each of the two breakout group.
- The two groups should have approximately the same number of subjects.
- Although Student's t-Test for Independent Samples was designed for use with sample groups of 30 subjects or fewer, it is common to apply this test with larger groups.

However, Student's t-Test for Independent Samples is considered a fairly robust test and deviation away from these assumptions is not uncommon to some degree, yet the test is still applied with a fair degree of confidence. If these assumptions are not met or are at least questioned, it may be prudent to apply the Mann–Whitney U Test as a confirming test from a nonparametric perspective.

Consider the previously stated Null Hypothesis for the dataset of female and male Systolic Blood Pressure values and then apply Student's t-Test for Independent Samples against the data: There is no statistically significant difference ($p \leq 0.05$) difference in Systolic Blood Pressure between female subjects and male subjects. As a comment, due to the way the dataset was created, it is expected that data should ostensibly show normal distribution.

Perhaps the most frequently used R function for application of Student's t-Test for Independent Samples is the `t.test()` function.

R Input

```
# Student's t-Test for Independent Samples Using the
# t.test() Function

t.test(SBP ~ Gender, data=SBPFM.df)
# Measured object variable ..... SBP
# Grouping object variable ..... Gender
```

R Output

```
Welch Two Sample t-test
SBP by Gender
t = -2.431, df = 515.9, p-value = 0.0154
mean in group F mean in group M
117.871      119.670
```

The data in the teaching dataset `SBPFM.df` were purposely organized to force an interesting decision as to statistical significance:

- Calculated p-value = 0.0154.
- The calculated p-value of 0.0154 is less than the criterion value of $p \leq 0.05$. There is a statistically significant difference between Female SBP values (Mean = 117.87) and Male SBP values (Mean = 119.67) at $p \leq 0.05$.
- Yet, the calculated p-value of 0.0154 is greater than the criterion value of $p \leq 0.01$, if that level of significance had been selected prior to analyses. There is no statistically significant difference between Female SBP values and Male SBP values at $p \leq 0.01$.
- That is to say, there is a statistically significant difference between Female SBP values (Mean = 117.87) and Male SBP values (Mean = 119.67) at $p \leq 0.05$, but there is no statistically significant difference between Female SBP values (Mean = 117.87) and Male SBP values (Mean = 119.67) at $p \leq 0.01$. Thus, it is incomplete to say either *There is no difference in Systolic Blood Pressure between Females and Males* or *There is a difference in Systolic Blood Pressure between Females and Males* without also stating the p-value associated with the statement (e.g., $p \leq 0.10$, $p \leq 0.05$, $p \leq 0.01$, $p \leq 0.001$).

As always, consider p-values and comparisons of calculated p-values (derived from inferential tests) to criterion p-values (derived from the Null Hypothesis, which is stated prior to any attempt at analyses):

- Determine in advance the level of significance for p-value comparisons. Will comparisons be made at $p \leq 0.10$, $p \leq 0.05$, $p \leq 0.01$, or $p \leq 0.001$?
- Observe the calculated p-value and the criterion p-value.
- Based on the comparison, declare if there is a statistically significant difference between the two groups or if there is no statistically significant difference between the two groups and be sure to state the associated p-value when making this declaration.

For this theoretical distribution of Systolic Blood Pressure values for the two genders in SBPFM.df:

- Calculated p-value was determined to be 0.0154.
- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure between Females (Mean = 117.8708 and SD = 2.0898) and Males (Mean = 119.6700 and SD = 16.4075) at $p \leq 0.05$. Male subjects have higher Systolic Blood Pressure than Female subjects ($p \leq 0.05$).
- It cannot be ignored, however, that there is no statistically significant difference ($p \leq 0.01$) in Systolic Blood Pressure between Females (Mean = 117.8708 and SD = 2.0898) and Males (Mean = 119.6700 and SD = 16.4075) at $p \leq 0.01$. There is parity in Systolic Blood Pressure between Female subjects and Male subjects ($p \leq 0.01$) and any difference in Systolic Blood Pressure between the two genders is due to chance.

As a sidebar demonstration, there are R-based packages that provide both descriptive statistics, measures of central tendency, and results of an inferential test. The furniture::table1() function generates Mean, SD, and a comparative p-value.

R Input

```
install.packages("furniture")
library(furniture)           # Load the furniture package.
help(package=furniture)      # Show the information page.
sessionInfo()                # Confirm all attached packages.

furniture::table1(SBPFM.df,
  SBP,                      # Measured object variable
  splitby=~Gender,          # Grouping object variable
  total=TRUE,                # Report total
  row_wise=TRUE,             # Report percentages across groups
  format_number=TRUE,         # Place commas for values above 999
  rounding_perc=2,           # Round to 2 digits after the decimal
  digits=2,                  # Number of significant digits
  test=TRUE)                 # Test of statistical significance

# Examine Systolic Blood Pressure (SBP) by Gender (Female
# and Male). By default, the furniture::table1() function
# will provide the following statistics: N, Mean, and SD.
# By adding the test argument, the furniture::table1()
# function will also provide a p-value for the appropriate
# test of significance when the splitby argument has more
# than 1 level.
```

R Output

Gender			
Total	F	M	P-Value
n = 980	n = 480	n = 500	
SBP			0.015
118.79 (11.84)	117.87 (2.09)	119.67 (16.41)	

This addendum demonstrated how to make two separate datasets and to then merge them into a unified dataset. The unified dataset had data as desired, in stacked format, which was achieved by using the rbind() function.

The data were then examined graphically (e.g., histogram, density plot, box-plot, and Q-Q plot) and also statistically. Assuming that the data exhibited appropriate normal distribution, Student's t-Test for Independent Samples was applied against the data, examining Systolic Blood Pressure by Gender.

Go back to the assumption of normal distribution and appropriate use of Student's t-Test for Independent Samples. Overall, there is a question about the normality of the entire set of data for the object variable SBPFM.df\$SBP. And, there was ample evidence to question the distribution pattern of SBP values for Female subjects. However, the SBP values for Male subjects seem to follow along with normal distribution.

Given this concern about normal distribution for Systolic Blood Pressure, apply the Mann–Whitney U Test against the data, to see if results for this nonparametric test confirm outcomes obtained from Student's t-Test for Independent Samples.

R Input

```
stats::wilcox.test(SBPFM.df$SBP ~ SBPFM.df$Gender,
                    alternative=c("two.sided"), paired=FALSE,
                    exact=TRUE, correct=TRUE)
# The stats::wilcox.test() function is used to
# perform an independent two-group Mann--Whitney
# U Test.
```

R Output

```
Wilcoxon rank sum test with continuity correction
SBPFM.df$SBP by SBPFM.df$Gender
W = 108508, p-value = 0.00934
```

As an interesting issue, complete a Mann–Whitney U Test analysis of the data again, using the `onewaytests::mw.test()` function instead of the `stats::wilcox.test()` function. Are the results consistent or at least in close parity?

R Input

```
onewaytests::mw.test(SBP ~ Gender,  
                      data=SBPFM.df, alpha=0.05, na.rm=TRUE,  
                      verbose=TRUE)
```

R Output

```
Mann--Whitney U Test (alpha = 0.05)  
-----  
Groups : F vs. M  
  
statistic : 2.59942  
p.value   : 0.00933825  
  
Result     : Difference is statistically significant.  
-----
```

Viewing the data from a nonparametric perspective, note how the calculated p-value is 0.00933825 whereas from a parametric perspective the calculated p-value is 0.0154. These two calculated p-value statistics are certainly not equivalent, but are they that different, given how 0.00933825 rounds to 0.01?

R Input

```
round(0.00933825, digits=2)
```

R Output

```
0.01
```

Based on the use of a nonparametric Mann–Whitney U Test approach, there is a statistically significant difference between Female SBP means and Male SBP means, both at the $p \leq 0.05$ level of significance and the $p \leq 0.01$ level of significance. Yet, the calculated p-value of 0.00934 has some degree of parity with the calculated p-value of 0.0154. Again, the teaching dataset for this addendum was purposely constructed to address these issues and to bring

to attention to concerns about later interpretation of outcomes.¹⁵

Similar to nearly all R applications, there are more than a few functions available for application of the Mann–Whitney U Test. The `exactRankTests::wilcox.exact()` function should also be considered.

R Input

```
install.packages("exactRankTests")
library(exactRankTests)      # Load the exactRankTests package.
help(package=exactRankTests) # Show the information page.
sessionInfo()               # Confirm all attached packages.

exactRankTests::wilcox.exact(SBP ~ Gender, data=SBPFM.df,
                           paired=FALSE, conf.int=TRUE, alternative="two.sided")
```

R Output

```
Asymptotic Wilcoxon rank sum test
SBP by Gender
W = 108508, p-value = 0.00934
```

Output from use of the `exactRankTests::wilcox.exact()` function and the previously used `onewaytests::mw.test()` function are nearly equivalent to what was obtained when using the `wilcox.test()`. Confirming redundancy practices are worth the time on task—with the reminder that quality assurance should be pervasive.

Summary of Outcome(s)

This lesson provided an opportunity to apply the inferential Student's t-Test for Independent Samples against a user-created teaching dataset, comparing Systolic Blood Pressure between females and males, with widely different standard deviations for the measured variable (e.g., Systolic Blood Pressure) for each breakout group (e.g., Female and Male).

For this dataset (`SBPFM.df`) and when the data were viewed from a parametric perspective:

- At $p \leq 0.05$ and using Student's t-Test for Independent Samples, there was a statistically significant difference in Systolic Blood Pressure between the two genders, where males had higher SBP means than females.

¹⁵Again, consider how in the social sciences there is a growing trend to report calculated p-values and to give less attention to a rules-based decision to either accept or fail to accept (e.g., reject) the Null Hypothesis.

- However, it cannot be ignored that at $p \leq 0.01$ and using Student's t-Test for Independent Samples, there was no statistically significant difference in Systolic Blood Pressure between the two genders, where ostensibly males and females were in parity regarding their SBP means.

Interestingly, for this dataset and when the data are viewed from a nonparametric perspective:

- Using the Mann–Whitney U Test, there was a statistically significant difference ($p \leq 0.05$ as well as $p \leq 0.01$) in Systolic Blood Pressure between the two genders, where males had higher SBP means than females.
- Yet, when viewing outcomes in finite detail, the calculated p-value associated with the nonparametric Mann–Whitney U Test was not all that off the mark from $p \leq 0.01$ —yet it did not quite rise to this level of significance.

Is this all a bit confusing, where this is a declaration of statistical difference in means between two breakout groups for one level of significance (e.g., $p \leq 0.05$) but not at a different level of significance (e.g., $p \leq 0.01$)? Is this all a bit confusing, where an inferential test associated with parametric data (e.g., Student's t-Test for Independent Samples) results in a p-value that is different than the p-value obtained when using a test associated with nonparametric data (e.g., Mann–Whitney U Test)? Careful attention to detail will help with understanding the statistical and practicaly significance of results and from this understanding, the construction of the narrative structure for interpreting the data.

3.11.2 Data That Do Not Exhibit Normal Distribution Patterns

Background: Description of the Data

As an interesting contrast, consider application of Student's t-Test for Independent Samples using a dataset where there is no semblance of normal distribution. The dataset for this part of the addendum is created using the `stats::runif()` function, the data do not follow a normal distribution pattern, and the data are focused on age breakouts and Diastolic Blood Pressure.

Background: Null Hypothesis

Null Hypothesis for Age Groups (e.g., subjects greater than or equal to 21 years and less than or equal to 49 years and their counterparts who are greater than or equal to 50 years and less than or equal to 99 years) v Diastolic Blood Pressure from the `DBPGTE21LTE99.df` dataset: There is no statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure between subjects from two distinct age groups—(1) subjects greater than or equal to 21 years and less than or equal to 49 years and (2) subjects who are greater than or equal to 50 years and less than or equal to 99 years. As a comment, due to the way

the dataset was created, it is expected that data should ostensibly fail to show normal distribution.¹⁶

Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

Use the `runif()` function to create a dataset where the measured object variable (e.g., Diastolic Blood Pressure) does not exhibit normal distribution. Similar to what was seen earlier, from among the many ways this aim could be achieved look below at the methods selected for this lesson, where a multi-staged approach is used.

Create a dataframe (`DBPGTE21LTE49.df`) of Diastolic Blood Pressure data for 500 subjects who range in age from 21 to 49.

R Input

```
DBPGTE21LTE49.df <- data.frame(
  Age <- replicate(500, "GTE21LTE49"),
  DBP <- round(runif(500, min=71, max=89))
)

colnames(DBPGTE21LTE49.df) <- c("Age", "DBP")
str(DBPGTE21LTE49.df)
names(DBPGTE21LTE49.df)
```

Create a dataframe called `DBPGTE21LTE49.df` (Diastolic Blood Pressure—Age 21–49 Years), consisting of two separate object variables: (1) replicate GTE21LTE49 500 times and place the output into the object `Age`, (2) apply the `runif()` function to create an object vector (`DBP`) in this dataframe of length = 500, minimum = 71, and maximum = 89, and (3) confirm the contents of the dataframe by applying the `str()` function and the `names()` function.

R Input

```
attach(DBPGTE21LTE49.df)      # Attach the dataset
head(DBPGTE21LTE49.df, 20)    # Look at the first 20 rows of data
summary(DBPGTE21LTE49.df)    # Summary descriptive statistics
```

Remove the individual object variables `Age` and `DBP` since they are no longer needed and then use the `ls()` function to confirm that they no longer show in the working directory:

¹⁶The terms GTE (e.g., Greater Than or Equal to) and LTE (e.g., Less Than or Equal to) are common means of expressing these conditions of comparison.

R Input

```
ls()  
rm(Age)  
rm(DBP)  
ls()
```

Create a dataframe (DBPGTE50LTE99.df) of Diastolic Blood Pressure data for 500 subjects who range in age from 50 to 99.

R Input

```
DBPGTE50LTE99.df <- data.frame(  
  Age <- replicate(500, "GTE50LTE99"),  
  DBP <- round(runif(500, min=72, max=96))  
)  
colnames(DBPGTE50LTE99.df) <- c("Age", "DBP")  
str(DBPGTE50LTE99.df)  
names(DBPGTE50LTE99.df)
```

Create a dataframe called DBPGTE21LTE49.df (Diastolic Blood Pressure—Age 50–99 Years), consisting of two separate object variables: (1) replicate GTE50LTE99 500 times and place the output into the object Age, (2) apply the runif() function to create an object vector (DBP) in this dataframe of length = 500, minimum = 72, and maximum = 96, and (3) confirm the contents of the dataframe by applying the str() function and the names() function.

R Input

```
attach(DBPGTE50LTE99.df)      # Attach the dataset  
head(DBPGTE50LTE99.df, 5)     # Look at the first 5 rows of data  
tail(DBPGTE50LTE99.df, 5)     # Look at the last 5 rows of data  
summary(DBPGTE50LTE99.df)     # Summary descriptive statistics
```

Remove the individual object variables Age and DBP since they are no longer needed and then use the ls() function to confirm that they no longer show in the working directory:

R Input

```
ls()  
rm(Age)  
rm(DBP)
```

```
ls()
```

Create a third dataframe in this sequence of steps. Merge the DBPGTE21LTE49.df and DBPGTE50LTE99.df dataframes. This new dataframe (DBPGTE21LTE99.df) will include the Diastolic Blood Pressure data for all 500 subjects who range in age from 21 to 49 years and also for all 500 subjects who range in age from 50 to 99 years.

R Input

```
DBPGTE21LTE99.df <- rbind(DBPGTE21LTE49.df, DBPGTE50LTE99.df)

attach(DBPGTE21LTE99.df)      # Attach the dataset
head(DBPGTE21LTE99.df, 5)    # Look at the first 5 rows of data
tail(DBPGTE21LTE99.df, 5)    # Look at the last 5 rows of data
summary(DBPGTE21LTE99.df)    # Summary descriptive statistics
```

As a QA check, review the length (e.g., N) and range (e.g., minimum and maximum) for all three dataframes, to gain a sense that DBPGTE21LTE99.df has values that are logical and in-range.

R Input

```
length(DBPGTE21LTE49.df$Age); range(DBPGTE21LTE49.df$DBP)
length(DBPGTE50LTE99.df$Age); range(DBPGTE50LTE99.df$DBP)
length(DBPGTE21LTE99.df$Age); range(DBPGTE21LTE99.df$DBP)
```

There is an initial indication that DBPGTE21LTE99.df is in good form. As a quality assurance measure, apply the R-based functions that the data are in the correct directory.

R Input

```
getwd()      # Identify the working directory
ls()         # List objects
```

Organize the Data and Display the Code Book

The dataframe on Diastolic Blood Pressure, similar to the dataframe on Systolic Blood Pressure, was created by combining two separate dataframes. The Code Book details the nature of the data.

R Input

```
#####
# Code Book for DBPGTE21LTE99.df          #
#####
#                                              #
# Age ..... Nominal #
#           GTE21LTE49 21 to 49 Years #
#           GTE50LTE99 50 to 99 Years #
#                                              #
# DBP ..... Numeric #
#           Range from about 70 to 100, + or - #
#####
#####
```

The `base::getwd()` function and the `base::ls()` function were previously used to confirm that each object variable was placed in the expected directory. Although it may seem redundant, one last check of the data in `DBPGTE21LTE99.df` is prudent to be sure that the data are as expected.

R Input

```
utils::str(DBPGTE21LTE99.df)      # Structure
utils::head(DBPGTE21LTE99.df, n=5) # First 5 cases
utils::tail(DBPGTE21LTE99.df, n=5) # Last 5 cases
base::summary(DBPGTE21LTE99.df)    # Summary statistics
```

R Output

Age	DBP
GTE21LTE49:500	Min. :71.0
GTE50LTE99:500	1st Qu.:77.0
	Median :82.0
	Mean :82.4
	3rd Qu.:87.0
	Max. :96.0

There are many more R-based functions available for further diagnostics about singular object variables and dataframes with multiple object variables, but the functions in this part of the addendum should be more than sufficient to confirm that the data are in good form.

Conduct a Visual Data Check Using Graphics (e.g., Figures)

Prepare simple graphics for quality assurance purposes, just to be sure that data follow along with expectations. The `ggplot2::ggplot()` function will serve

a useful purpose here. For a few figures `facet_grid()` is used to put age-related breakouts (21–49 years and 50–99 years) on a common scale (Fig. 3.18).

R Input

```
# Histogram breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(DBPGTE21LTE99.df,
  aes(x=DBP, fill=Age)) +
  geom_histogram(position="identity", color="white",
  bins = 15) +
  facet_grid(. ~ Age) +
  ggtitle("DBP by Age Breakouts (Histogram)") +
  labs(
    x = "\nDiastolic Blood Pressure", y = "Count\n") +
  theme_bw() # QA check for general trends

# Density Plot breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(DBPGTE21LTE99.df,
  aes(DBP, color=Age)) +
  geom_density(size=1.25) +
  facet_grid(. ~ Age) +
  ggtitle("DBP by Age Breakouts (Density Plot)") +
  labs(
    x = "\nDiastolic Blood Pressure", y = "Density\n") +
  theme_bw() # QA check for general trends

# Box Plot breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(DBPGTE21LTE99.df,
  aes(y=DBP, fill=Age)) +
  geom_boxplot() +
  facet_grid(. ~ Age) +
  ggtitle("DBP by Age Breakouts (Boxplot)") +
  labs(
    x = "Age", y = "Diastolic Blood Pressure\n") +
  theme_bw() # QA check for general trends

# Q-Q Plot and Q-Q Line breakouts using the ggplot2::ggplot()
# function
```

```

par(ask=TRUE)
ggplot2::ggplot(DBPGTE21LTE99.df,
  aes(sample=DBP)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ Age) +
  ggtitle("DBP by Age Breakouts (Q-Q Plot)") +
  labs(
    x = "\nTheoretical", y = "Diastolic Blood Pressure\n") +
  theme_bw() # QA check for general trends

# Violin Plot and superimposed Box Plot breakouts using the
# ggplot2::ggplot()function

par(ask=TRUE)
ggplot2::ggplot(DBPGTE21LTE99.df, aes(x=Age, y=DBP)) +
  geom_violin(aes(fill = Age), lwd=1.75) +
  geom_boxplot(width = 0.25, col=c("black"), lwd=1.25) +
  ggtitle("DBP by Age Breakouts (Violin Plot and Box Plot)") +
  labs(
    x = "\nAge", y = "Diastolic Blood Pressure\n") +
  theme_bw() # QA check for general trends
# Note how facet_grid() and/or facet_wrap() were not used and
# and how a box plot was placed inside each side-by-side
# violin plot.

```

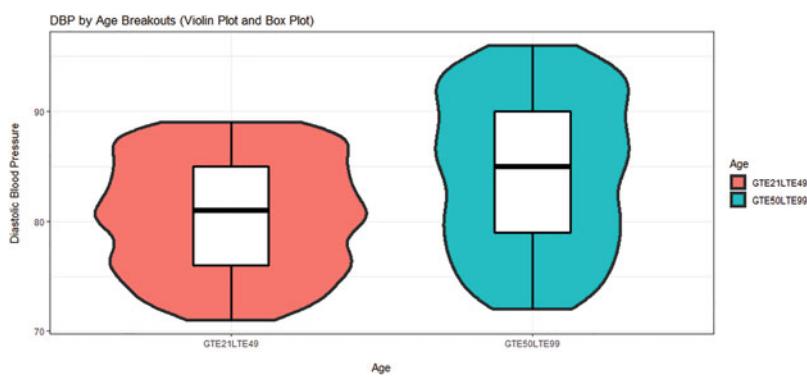


Figure 3.18: Distribution of diastolic blood pressure by age breakouts

R Input

```
# Dot Plot breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(DBPGTE21LTE99.df,
  aes(DBP, color=Age)) +
  geom_dotplot(binwidth=0.75, stackdir="center",
    dotsize = 0.75) +
  facet_grid(. ~ Age) +
  scale_x_continuous(breaks=c(70, 75, 80, 85, 90, 95,
    100)) +      # Force printout of these DBP measurements
  ggtitle("DBP by Age Breakouts (Dot Plot)") +
  labs(
    x = "\nDiastolic Blood Pressure", y = "") +
  theme_bw() +          # QA check for general trends
  theme(axis.title.y=element_blank(), # Avoid text on the
        axis.text.y=element_blank(), # Y axis
        axis.ticks.y=element_blank())
```

The gridExtra::grid.arrange() could have been used to place all six figures into q six in one figure. However, due to the use of two breakouts in each figure, the detail in the six in one figure would have been so small that the detail would be difficult to read. Use good judgment on when and how to present figures, either for formative analyses or final publication.

Descriptive Statistics for Initial Analysis of the Data

Although the above figures provide a sense of the data, descriptive statistics are also needed to have a more complete understanding of data distribution, measures of central tendency, and extreme measures—overall and by age breakouts. As demonstrated previously in this lesson and throughout this text, there are many functions that address these needs. For this dataset, the s20x::summaryStats(), arsenal::tableby(), and RcmdrMisc::numSummary() functions will likely satisfy immediate needs regarding output of descriptive statistics overall for Diastolic Blood Pressure and also by the two Age breakout groups.

R Input

```
with(DBPGTE21LTE99.df, s20x::summaryStats(DBP))
# Show descriptive statistics, overall.

DBPAgeBreakouts <-
```

```
arsenal::tableby(Age ~ DBP, data=DBPGTE21LTE99.df)
# Create the arsenal::tableby() model.

summary(DBPAgeBreakouts, text=TRUE)
# Show output from the arsenal::tableby() model.

RcmdrMisc::numSummary(DBPGTE21LTE99.df$DBP)
# Show descriptive statistics, overall.

RcmdrMisc::numSummary(DBPGTE21LTE99.df[,c("DBP")], groups=Age)
# Show descriptive statistics, by breakout groups.

asbio::Mode((subset(DBPGTE21LTE99.df$DBP,
DBPGTE21LTE99.df$Age == "GTE21LTE49")))
# Age 21 to 49 DBP

asbio::Mode((subset(DBPGTE21LTE99.df$DBP,
DBPGTE21LTE99.df$Age == "GTE50LTE99")))
# Age 50 to 99 DBP

asbio::Mode(DBPGTE21LTE99.df$DBP)
# 21 to 49 and 50 to 99 DBP
```

Quality Assurance, Data Distribution, and Tests for Normality

Quality assurance (QA) is addressed through diligent and constant attention to the data, from initial data entry to final interpretation of results. Be sure to observe how the many figures provide multiple views of the data. Then, observe how multiple functions are used to provide what may at first seem to be redundant descriptive statistics. Only through these many other actions can there be assurance that the data are in good form.

In concert with this high level of attention to the data, give attention also to tests that address concern over normal distribution. The Anderson–Darling test for normality is a common first choice to gauge distribution and deviation away from normal distribution, if any.

R Input

```
nortest::ad.test(DBPGTE21LTE99.df$DBP)
```

After application of the nortest::ad.test() function against the object variable DBPGTE21LTE99.df\$DBP (e.g., Diastolic Blood Pressure) it is observed that the calculated p-value (p-value = 0.00000000000000954) is less than the criterion

p-value of $p \leq 0.05$. This finding brings to question if there is normal distribution for Diastolic Blood Pressure.

There is still a question about normality for Diastolic Blood Pressure by the two age groups. Is it possible that subjects in the 21–49 years age group have normal distribution of Diastolic Blood Pressure but that this is not the case for subjects in the 50–99 years age group? Is it possible that subjects in the 21–49 years age group do not have normal distribution of Diastolic Blood Pressure but that subjects in the 50–99 years age group have normal distribution of Diastolic Blood Pressure?

To address this concern, the Anderson–Darling test will be used again, but it will be deployed by using the RcmdrMisc::normalityTest() function. In later lessons other functions will be used to achieve the same aim. Nearly every action in R can be addressed through multiple approaches.

R Input

```
RcmdrMisc::normalityTest(DBPGTE21LTE99.df$DBP,
  test=c("ad.test"),
  groups=DBPGTE21LTE99.df$Age)
```

R Output

```
-----
DBPGTE21LTE99.df$Age = GTE21LTE49

Anderson-Darling normality test

data: DBPGTE21LTE99.df$DBP
A = 5.345, p-value = 0.000000000000332

-----
DBPGTE21LTE99.df$Age = GTE50LTE99

Anderson-Darling normality test

data: DBPGTE21LTE99.df$DBP
A = 5.753, p-value = 0.000000000000351

-----
p-values adjusted by the Holm method:
unadjusted           adjusted
```

```
GTE21LTE49 0.0000000000003322 0.0000000000003322  
GTE50LTE99 0.000000000000351 0.0000000000000702
```

Recall that `DBPGTE21LTE99.df$DBP` was purposely created by using the `runif()` function so that the data for Diastolic Blood Pressure would not exhibit normal distribution and indeed that is the case. Overall and by the two age breakouts, normality p-values are less than 0.05, indicating failure to exhibit normality. This observation should be considered when selecting the best approach to determine if there is a statistically significant difference between the two age groups regarding Diastolic Blood Pressure.

Statistical Test(s)

The parametric Student's t-Test for Independent Samples will be used to determine if there is a statistically significant difference ($p \leq 0.05$) in the mean Diastolic Blood Pressure of subjects who are 21–49 years and their counterparts who are 50–99 years. However, it cannot be ignored that normal distribution is assumed for correct application of this test and it has been established that normal distribution is not evident for Diastolic Blood Pressure, overall and by age breakout groups. The nonparametric Mann–Whitney U Test will also be used to compare Diastolic Blood Pressure for the two age groups. Normal distribution is not assumed for correct application of the Mann–Whitney U Test.

Before applying either the Student's t-Test for Independent Samples or the Mann–Whitney U Test, it is necessary to consider the Null Hypothesis once again, to review the main focus of this set of analyses: There is no statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure between subjects from two distinct age groups—(1) subjects greater than or equal to 21 years and less than or equal to 49 years and (2) subjects who are greater than or equal to 50 years and less than or equal to 99 years.

As a recap of general trends for Diastolic Blood Pressure, it is useful to apply the `furniture::table1()` function. The `test=TRUE` provides interesting output, but application of the Student's t-Test for Independent Samples or the Mann–Whitney U Test serves as the final tests used to judge outcomes for the Null Hypothesis.

R Input

```
furniture::table1(DBPGTE21LTE99.df,  
  DBP,                      # Measured object variable  
  splitby=~Age,              # Grouping object variable  
  total=TRUE,                # Report total  
  rounding_perc=0,           # Report as whole numbers  
  digits=0,                  # Number of significant digits
```

```
test=TRUE) # Test of statistical significance
# Examine Diastolic Blood Pressure (DBP) by Age (21-49
# and 50-99). By default, the furniture::table1() function
# will provide the following statistics: N, Mean, and SD.
# By adding the test argument, the furniture::table1()
# function will also provide a p-value for the appropriate
# test of significance when the splitby argument has more
# than 1 level.
```

R Output

```
-----  
Age  
Total      GTE21LTE49  GTE50LTE99 P-Value  
n = 1000  n = 500      n = 500  
DBP                      <.001  
 82 (6)    80 (5)    84 (7)  
-----
```

As shown by using the `furniture::table1()` function, the mean Diastolic Blood Pressure for subjects 21–49 years is approximately 80 ($sd = 5$) and the mean Diastolic Blood Pressure for subjects 50–99 years is approximately 84 ($sd = 7$), with calculated $p \leq 0.001$. See if these statistics are in parity with results from the Student's t-Test for Independent Samples and the Mann–Whitney U Test.

Using a parametric approach, apply the Student's t-Test for Independent Samples.

R Input

```
t.test(DBP ~ Age, data=DBPGTE21LTE99.df)
# Measured object variable ..... DBP
# Grouping object variable ..... Age
```

Using a nonparametric approach to the analyses, apply the Mann–Whitney U Test.

R Input

```
onewaytests::mw.test(DBP ~ Age,
data=DBPGTE21LTE99.df, alpha=0.05, na.rm=TRUE,
verbose=TRUE)
```

Summary of Outcome(s)

Using Student's t-Test for Independent Samples and the confirming Mann–Whitney U Test, the calculated p-value is far less than 0.05. Therefore, there is a statistically significant difference in Diastolic Blood Pressure for subjects 21–49 years and subjects 50–99 years at $p \leq 0.05$, with the younger subjects having lower mean DBP values as compared to their older counterparts.

The final outcomes from this dataset were quite similar, both for the parametric Student's t-Test for Independent Samples and the confirming nonparametric Mann–Whitney U Test. Many inferential tests have an underlying assumption of normal distribution. Yet, as evidenced in this addendum, normal distribution is not always the case when working with data. Therefore, it is usually a good idea to deploy a confirming test that can accommodate data that fail to exhibit normality. It is not at all uncommon for the two approaches (parametric and nonparametric) to have a fair degree of parity in final outcomes, but it is still prudent to apply confirming tests.

3.12 Prepare to Exit, Save, and Later Retrieve This R Session

R Input

```
getwd()          # Identify the current working directory.  
ls()            # List all objects in the working  
                # directory.  
ls.str()         # List all objects, with finite detail.  
list.files()     # List files at the PC directory.  
  
save.image("R_Lesson_tTestIndependentSamples.rdata")  
  
getwd()          # Identify the current working directory.  
ls()            # List all objects in the working  
                # directory.  
ls.str()         # List all objects, with finite detail.  
list.files()     # List files at the PC directory.  
  
alarm()          # Alarm, notice of upcoming action.  
q()              # Quit this session.  
                # Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: File -> Load Workspace. Otherwise, use the load() function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use the .R script file (typically saved as a .txt ASCII-type file) and recreate the analyses and graphics, provided the data files remain available.

3.13 External Data and/or Data Resources Used in This Lesson

`MilkBreedButterfatProtein.csv` was the only external file directly imported into this lesson and it is available at the publisher's Web site associated with this text. Use this file to practice and replicate the outcomes used in this lesson.

All other data in this lesson were enumerated (e.g., user-created) directly in the R session, using a variety of R-based functions. The user-created data in this lesson are **not** included at the publisher's Web site associated with this text and they are instead easily created in R by following the syntax presented in this lesson.



Chapter 4

Student's t-Test for Matched Pairs

Abstract

The purpose of this lesson is to use R to examine differences to a singular measured variable between pairs, specifically by using Student's t-Test for Matched Pairs. Student's t-Test for Independent Samples is used to compare differences between two separate groups against a singular measured variable. In contrast, Student's t-Test for Matched Pairs is used to compare differences to a single measured variable when subjects are matched against a counterpart, often where subjects are their own counterparts. Although there are many possible applications, Student's t-Test for Matched Pairs is often used for pretest v posttest analyses, where a subject is measured for a specific variable, a treatment is applied, and the subject is measured again after the treatment. This lesson also provides an introduction to the use of unstacked data as compared to the use of stacked data. Finally, the issues of sample size (especially N of approximately 30) and sample representation are introduced in this lesson.

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_4) contains supplementary material, which is available to authorized users.

Keywords: Gosset (William Sealy), Independent samples, Long format data, Matched pairs, Pretest, Posttest, Repeated measures, Stacked data, Student's t-Test, t-Statistic, Treatment, Unstacked data, Wide format data

4.1 Background

This lesson has been designed to show how the R environment supports Student's t-Test for Matched Pairs, the empirical test used to determine if there are statistically significant differences to a measured variable between two samples where the two samples are organized in a paired (e.g., matched) fashion. Although there are many creative ways to structure matched pairs, Student's t-Test for Matched Pairs typically includes analyses organized as pretest measures compared to posttest measures, often after a treatment is applied after the pretest and before the posttest (e.g., Pretest—Treatment—Posttest). The key condition in most cases of a matched pairs analysis is that the same subjects are measured twice, with individual subjects serving as their own counterpart—thus they are matched. Note how this research design demands sufficient tracking information so that data are organized by specific subjects and not only overall group membership.

4.1.1 Description of the Data

This lesson on Student's t-Test for Matched Pairs has been designed to determine if there is a statistically significant difference ($p \leq 0.05$) in the weight (Kg) of individual biological specimens after the application of a mineral supplement in the feeding program. The individual charged with responsibility for data analysis knows very little about the specifics of the data, the nature of the biological specimen, the feeding program, and the added supplement.

For now, it is only sufficient to know that the biological specimens are all adults of both genders and that they are on a maintenance feeding program. Subjects receive sufficient feed, water, management, etc., to maintain their weight, but an overt attempt at weight gain is not a goal for the current feeding program. The feeding program and available rations, access to water, exercise, ventilation and housing, etc., remained constant throughout the experimental period.

Individual specimens were all weighed on the same day (e.g., pretest), soon after a mineral supplement was introduced into the feeding program (e.g., treatment). The mineral supplement is tasteless and odorless, has no noticeable texture, and it is assumed to have no impact on feed palatability. After a set, but unidentified, period of time, individual subjects are weighed again (e.g., posttest), all on the same day. Weights are in Kg and this measurement is viewed as a numeric value. Note how the specimens are not divided into breakout groups (e.g., there is no grouping by breed, color, gender, height, length, temperament,

etc.). Equally, the feeding program and application of the mineral supplement are also consistent, with no differences in treatment by breakout groups. The introduction of the mineral supplement, viewed as a treatment, is the only change.

This arrangement is a representation of a simple pretest—treatment—posttest experiment and by design, Student's t-Test for Matched Pairs will be used for statistical analysis:

- Subjects are organized into one common group where individual subjects are matched against their pretest measurement and their posttest measurement.
- Subjects are measured with a beginning pretest. In this case, the pretest was weight (Kg) immediately before the beginning of a treatment (e.g., feeding program).
- Subjects then experienced some type of treatment. In this case, the treatment was an added mineral supplement. The nature of the treatment and duration of the treatment is the same for all subjects.
- Subjects are measured with an ending posttest. In this case, the posttest was weight (Kg) at the end of the treatment (e.g., feeding program).

The data that will be used for this matched pairs experiment are the pretest and posttest weights for each individual subject. By design, individual subjects are their own matched pair, with weight measured at the beginning of the feeding program (KgPreSupplement) and weight measured again at the end of the treatment (KgPostSupplement).

Review prior discussion on the development, nature, and use of Student's t-Test. The pretest and posttest approach used in this lesson is a common approach for Student's t-Test for Matched Pairs.¹ In this lesson, individual subjects are their own match, using a simple pretest and posttest design.

This sample will be interesting in that it is a fairly large dataset of 150 subjects and it also reinforces the realities of working with missing data. There is a missing posttest datum for Subject 19. There are always reasons why data may be missing from a dataset: the subject may have been unavailable for measurement, the subject may have died, the field technician may have recorded a weight that is simply not valid (e.g., illogical or out of range), field notes for this measurement may have either been lost or damaged beyond use, the data entry specialist may have failed to enter the datum and field notes are not readily available for this subject, etc. Missing data are never desirable, but it is a condition that must be considered since it is common.

¹When a matched pairs research design is desired but it is not possible to have individual subjects serve as their own match, it is common to see siblings, cousins, similar counterparts, etc., serve as a match. However, that is not the approach used in this lesson.

Given the nature of the data for this lesson, it is judged that Student's t-Test for Matched Pairs is an appropriate approach in the attempt to determine if there is a statistically significant difference ($p \leq 0.05$) in weights (Kg) between pretest measures and posttest measures: KgPreSupplement and KgPostSupplement. Again, recall that there are no planned breakout analyses by gender, by varying mineral supplement strengths, by varying mineral supplement application periods, etc. Treatment is consistent.

4.1.2 Null Hypothesis

There is one Null Hypothesis associated with this part of the lesson, with $p \leq 0.05$ the criterion level of significance.

H_0 : There is no statistically significant difference ($p \leq 0.05$) in the pretest weight (Kg) of unidentified adult biological specimens and the posttest weight (Kg) of unidentified adult biological specimens at the end of a feeding experiment, where a tasteless and odorless mineral supplement was introduced into the feeding program.

4.1.3 Unstacked (e.g., Wide) Data and Stacked (e.g., Long) Data

Data for Student's t-Test typically involve identification of a grouping variable (e.g., binary variable; Male v Female, Location 1 v Location 2, supplemental light in the housing unit v natural light only in the housing unit, etc.) and a measured variable (e.g., speed, height, weight). There are largely two ways that data can be organized in electronic format for a Student's t-Test analysis and then used in R: unstacked (e.g., wide) data and stacked (e.g., long) data. An example separate from this lesson will help clarify these two methods for data organization, when data are put into electronic format.

Unstacked Data (e.g., Wide Format)

Unstacked data for ten separate subjects (2-year old Labrador Retrievers) for gender and weight (Lbs) follow:

Weight.LabDog.Female	Weight.LabDog.Male
51	55
53	61
50	58
49	60
52	64

With unstacked data, the two columns are typically next to each other. The standard way to use R to conduct a Student's t-Test with unstacked data is to use a comma to separate identification of the two data columns:

```
t.test(Column1, Column2) # Unstacked data uses , and not ~
t.test(Weight.LabDog.Female, Weight.LabDog.Male)
```

Of course, arguments can always be added to this general means of conducting a Student's t-Test, such as: (1) na.rm=TRUE for missing data and (2) either paired=FALSE or paired=TRUE to address whether the t-Test analysis is for independent samples (paired=FALSE) or matched pairs (paired=TRUE).

Stacked Data (e.g., Long Format)

Although the experienced researcher will be able to work with data organized in either format (unstacked or stacked, wide or long), it is perhaps more common in the biological sciences to see data put into stacked format than unstacked format.

Stacked data for ten separate subjects (2-year old Labrador Retrievers) for gender and weight (Lbs) follow:

Gender.LabDog	Weight.LabDog
Female	51
Female	53
Female	50
Female	49
Female	52
Male	55
Male	61
Male	58
Male	60
Male	64

The grouping variable, also called the binary variable, is Gender.LabDog (e.g., Female and Male). The measured datum is Weight.LabDog, measured in Lbs.

With stacked data the data are organized so that data for the grouping variable (Gender.LabDog, Female and Male) are placed in one column and the corresponding measured data (Weight.LabDog, measured in Lbs.) are found in an adjacent column.

The standard way to use R to conduct a Student's t-Test with stacked data is to use a tilde to separate the variable representing the measured data and the variable representing the two groups.

```
t.test(Measured.Data.Variable ~ Grouping.Variable)
t.test(Weight.LabDog ~ Gender.LabDog)
# Stacked data uses ~ and not ,
```

Of course, arguments can always be added to this general means of conducting a Student's t-Test, such as na.rm=TRUE for missing data.

4.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

For this lesson, the data have been organized as unstacked (e.g., wide format) data. The dataset has been prepared in .csv (e.g., comma-separated values) file format. The data are separated by commas. The data are neither separated by tabs nor spaces.

Assume that the data were originally transferred from paper field notes into a Gnumeric-based spreadsheet and saved in gnumeric file format. After the spreadsheet was reviewed by staff and put into final form, it was then saved in .csv file format, to facilitate data import into R and to also transfer the dataset into a common format that can easily be shared with colleagues, worldwide if needed. As is evident when the data are reviewed, there is a missing datum, due to difficulties encountered during field operations.

R Input

```
#####
# Housekeeping                                     Use for All Analyses #
#####
date()           # Current system time and date.
Sys.time()       # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls(all.names=TRUE) # List all objects in the working
                   # directory, including hidden files.
rm(list=ls())     # CAUTION: Remove all files in the
                   # working directory. If this
                   # action is not desired, use rm()
                   # one-by-one to remove the objects
                   # that are not needed.
ls.str()          # List all objects, with finite detail.
getwd()           # Identify the current working directory.
setwd("F:/R_BiostatisticsIntroduction")
                   # Set to a new working directory.
                   # Note the single forward slash and double
                   # quotes.
                   # This new directory should be the directory
                   # where the data file is located, otherwise
                   # the data file will not be found.
```

```
getwd()           # Confirm the working directory.  
list.files()     # List files at the PC directory.  
.libPaths()      # Library pathname.  
.Library         # Library pathname.  
sessionInfo()    # R version, locale, and packages.  
search()          # Attached packages and objects.  
searchpaths()     # Attached packages and objects.  
#####
```

The Housekeeping template shown above, with occasional changes, is used throughout this text. The purpose of this collection of R functions, all in one convenient beginning section, is to be sure that the working environment, selected directory, etc., are as desired.

Create an object called `PrePostWtUnstack.df`. The object `PrePostWtUnstack.df` will be a dataframe, as indicated by the enumerated `.df` extension to the object name. This object will represent the output of applying the `read.table()` function against the comma-separated values file called `WeightPrePostSupplementUnstacked.csv`. Note the arguments used with the `read.table()` function, showing that there is a header with descriptive variable names (`header = TRUE`) and that the separator between fields is a comma (`sep = ","`).

R Input

```
PrePostWtUnstack.df <- read.table (file =  
  "WeightPrePostSupplementUnstacked.csv",  
  header = TRUE,  
  sep = ",")               # Import the .csv file.  
  
getwd()           # Identify the working directory  
ls()              # List objects  
attach(PrePostWtUnstack.df) # Attach the data, for later use  
str(PrePostWtUnstack.df)   # Identify structure  
nrow(PrePostWtUnstack.df)  # List the number of rows  
ncol(PrePostWtUnstack.df)  # List the number of columns  
dim(PrePostWtUnstack.df)   # Dimensions of the dataframe  
names(PrePostWtUnstack.df) # Identify names  
colnames(PrePostWtUnstack.df) # Show column names  
rownames(PrePostWtUnstack.df) # Show row names  
head(PrePostWtUnstack.df)   # Show the head  
tail(PrePostWtUnstack.df)   # Show the tail  
PrePostWtUnstack.df       # Show the entire dataframe
```

```
summary(PrePostWtUnstack.df) # Summary statistics
```

R Output

Subject	KgPreSupplement	KgPostSupplement
Min. : 1.0	Min. : 5.57	Min. : 5.57
1st Qu.: 38.2	1st Qu.: 9.28	1st Qu.: 9.64
Median : 75.5	Median :10.60	Median :10.89
Mean : 75.5	Mean :10.79	Mean :11.19
3rd Qu.:112.8	3rd Qu.:12.24	3rd Qu.:12.56
Max. :150.0	Max. :16.47	Max. :16.30
NA's :1		

By completing these actions, an object called `PrePostWtUnstack.df` has been created. This R-based object is a dataframe and it consists of the data originally included in the file `WeightPrePostSupplementUnstacked.csv`, a comma-separated .csv file. To avoid possible conflicts, make sure that there are no prior R-based objects called `PrePostWtUnstack.df`. The prior use of `rm(list = ls())` accommodates this concern, removing all prior objects in the current R session.

It was only necessary to key the filename for the .csv file and not the full pathname since the R working directory is currently set to the directory and/or subdirectory where this .csv file is located (see the Housekeeping section at the beginning of this lesson).

When showing the entire dataset, notice the listing for Subject 19 and how NA (the R notation for missing data) has been inserted for the KgPostSupplement datum. Review the help page for the `read.table()` function to see how a blank cell in the .csv file, which is the case for Subject 19, is viewed as a missing value. With R, NA is inserted during execution of the `read.table()` function for what are otherwise missing data.

4.3 Organize the Data and Display the Code Book

For this lesson, the `class()` function, `str()` function, and `duplicated()` function will be used to be sure that data are organized as desired.

R Input

```
class(PrePostWtUnstack.df)
```

R Output

```
[1] "data.frame"
```

R Input

```
str(PrePostWtUnstack.df)
```

R Output

```
'data.frame': 150 obs. of 3 variables:  
 $ Subject      : int 1 2 3 4 5 6 7 8 9 10 ...  
 $ KgPreSupplement : num 12.4 10.64 8.97 10.01 10.94 ...  
 $ KgPostSupplement: num 13.3 10.2 9.6 10.9 10.8 ...
```

R Input

```
duplicated(PrePostWtUnstack.df)
```

R Output

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[10] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[19] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[28] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[46] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[55] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[64] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[82] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[91] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[100] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[118] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[127] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[136] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  
[145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

The class for each object seems to be correct and there are no duplicate rows of data in the dataframe. Saying this, a Code Book will help with future understanding of this dataset.

A Code Book is an essential aid for anyone involved in the day-to-day activities of the research and statistics process. The Code Book is typically brief and only serves as a useful reminder for what can be easily forgotten months (or even weeks) later. The Code Book is also a useful tool for when data and associated files are shared with colleagues. Assume little and document much.

The Code Book for this dataset represents how data are desired before analyses begin. Recoding may be needed to put data into desired formats.

R Input

```
#####
# Code Book for PrePostWtUnstack.df          #
#####
#                                         #
# Subject ..... Factor (e.g., nominal) #
#           A unique ID ranging from 1 to 150 #
#                                         #
# KgPreSupplement ..... Numeric (e.g., interval) #
#           Weight (Kg) of an unidentified biological #
#           specimen, ranging from 5.00 to 20.00 #
#                                         #
# KgPostSupplement ..... Numeric (e.g., interval) #
#           Weight (Kg) of an unidentified biological #
#           specimen, ranging from 5.00 to 20.00 #
#####
#####
```

Although possibly redundant, to be prudent apply the `str()` function against the dataframe to confirm the nature of each object variable as well as the dataframe.

R Input

```
str(PrePostWtUnstack.df)
```

Recall that the Code Book shows data in their desired formats, which often requires some degree of recoding which has not yet occurred. Once there is agreement that the data were brought into R in correct format, it is usually necessary to organize the data to some degree:

- The object variable `Subject` is currently viewed as an integer. Some R users may find it best to recode these identification-type numeric values into row names, but in this lesson `Subject` will instead be recoded into a factor object variable.
- Values for `KgPreSupplement` are viewed as numeric values.

- Values for KgPostSupplement are viewed as numeric values.

This transformation (again, typically called a recode action) is needed. There may be some unnecessary (perhaps redundant) actions with the demonstrated recode activities, but they are purposely done to provide assurance that each variable is in desired format:

R Input

```

PrePostWtUnstack.df$Subject           <- as.factor(
  PrePostWtUnstack.df$Subject)

PrePostWtUnstack.df$KgPreSupplement   <- as.numeric(
  PrePostWtUnstack.df$KgPreSupplement)

PrePostWtUnstack.df$KgPostSupplement  <- as.numeric(
  PrePostWtUnstack.df$KgPostSupplement)

getwd()                                # Identify the working directory
ls()                                    # List objects
attach(PrePostWtUnstack.df)            # Attach the data, for later use
str(PrePostWtUnstack.df)               # Identify structure
nrow(PrePostWtUnstack.df)              # List the number of rows
ncol(PrePostWtUnstack.df)              # List the number of columns
dim(PrePostWtUnstack.df)               # Dimensions of the dataframe
names(PrePostWtUnstack.df)             # Identify names
colnames(PrePostWtUnstack.df)          # Show column names
rownames(PrePostWtUnstack.df)          # Show row names
head(PrePostWtUnstack.df)              # Show the head
tail(PrePostWtUnstack.df)              # Show the tail
PrePostWtUnstack.df                   # Show the entire dataframe
summary(PrePostWtUnstack.df)           # Summary statistics

```

R Output

	Subject	KgPreSupplement	KgPostSupplement
1	: 1	Min. : 5.57	Min. : 5.57
2	: 1	1st Qu.: 9.28	1st Qu.: 9.64
3	: 1	Median :10.60	Median :10.89
4	: 1	Mean :10.79	Mean :11.19
5	: 1	3rd Qu.:12.24	3rd Qu.:12.56
6	: 1	Max. :16.47	Max. :16.30
(Other)	:144		NA's : 1

Note the formal nomenclature (`DataFrame$Object`) used in this recode action and the use of formal notation when working with object variables that are part of a dataframe. Note also how the \$ symbol is used to separate the name of the dataframe from the name of the object.

4.4 Conduct a Visual Data Check Using Graphics (e.g., Figures)

Now that the data are all in proper format, it would be common for those who are inexperienced with the research process to immediately apply the Student's t-Test for Matched Pairs algorithm against the data and to complete this matched pairs analysis. However, it is best to first prepare a few graphical displays of the data and to then reinforce comprehension of the data with descriptive statistics and measures of central tendency.

A few throwaway graphics will likely be sufficient at first, but fully-embellished graphical images may be needed later, for either publication or presentation. The full set of lessons for this text, both prior lessons and future lessons, provide many examples of how graphics enhance understanding of the data.

For now, use the `par(mfrow=c(2,4))` functions and arguments to put 8 figures into a 2 row by 4 column grid. Using a common set of R-based graphical tools, the next set of visual presentations will use the following functions: `hist()`, `plot()`, `lines(plot(density()))`, `rug(jitter())`, `boxplot()`, `stripchart()`, `dotchart()`, `qqnorm()`, `qqline()`, and `vioplot::vioplot()`. Although these many figures could have also been generated using `ggplot2::ggplot()` with different `geom_` selections, notice below how all of these figures are generated using functions immediately available from when R is first download, with the singular exception of the violin plot, which is based on use of the `vioplot` package.

R Input

```
install.packages("vioplot")
library(vioplot)                      # Load the vioplot package.
help(package=vioplot)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.
```

It is helpful to organize the figures into side-by-side comparisons, such as KgPreSupplement and KgPostSupplement. Be sure to notice the use of a common scale for each set of side-by-side figures, allowing meaningful comparisons. Use appropriate arguments, such as `xlim=c()` or `ylim=c()` to force common scales (Figs. 4.1 and 4.2).

R Input

```

par(ask=TRUE)
par(mfrow=c(2,4)) # 8 figures into a 2 row by 8 column grid
# More than 8 figures in one common plot makes it difficult to
# see results, read titles and scales, etc.
hist(PrePostWtUnstack.df$KgPreSupplement, col=c("red"),
  xlab="Weight (Kg)", ylab="Frequency",
  xlim=c(0,20), ylim=c(0,35),
  main="KgPreSupplement: Histogram")
hist(PrePostWtUnstack.df$KgPostSupplement, col=c("red"),
  xlab="Weight (Kg)", ylab="Frequency",
  xlim=c(0,20), ylim=c(0,35),
  main="KgPostSupplement: Histogram")
plot(PrePostWtUnstack.df$KgPreSupplement, col=c("red"),
  xlab="Subject ID", ylab="Weight (Kg)",
  xlim=c(0,150), ylim=c(0,20),
  main="KgPreSupplement: Plot")
plot(PrePostWtUnstack.df$KgPostSupplement, col=c("red"),
  xlab="Subject ID", ylab="Weight (Kg)",
  xlim=c(0,150), ylim=c(0,20),
  main="KgPostSupplement: Plot")
lines(plot(density(PrePostWtUnstack.df$KgPreSupplement,
  na.rm=TRUE), col="red", lwd=4, lty=2, ylim=c(0,0.25),
  xlab="Weight (Kg)", main="KgPreSupplement: Density Plot"))
rug(jitter(PrePostWtUnstack.df$KgPreSupplement))
lines(plot(density(PrePostWtUnstack.df$KgPostSupplement,
  na.rm=TRUE), col="red", lwd=4, lty=2, ylim=c(0,0.25),
  xlab="Weight (Kg)", main="KgPostSupplement: Density Plot"))
rug(jitter(PrePostWtUnstack.df$KgPostSupplement))
boxplot(PrePostWtUnstack.df$KgPreSupplement, col=c("red"),
  ylab="Weight (Kg)", ylim=c(0,20),
  main="KgPreSupplement: Box Plot")
boxplot(PrePostWtUnstack.df$KgPostSupplement, col=c("red"),
  ylab="Weight (Kg)", ylim=c(0,20),
  main="KgPostSupplement: Box Plot")
# Note how common scales for the X axis and/or Y axis have
# been used to allow for meaningful side-by-side comparisons.

```

R Input

```

par(ask=TRUE)
par(mfrow=c(2,4)) # 8 figures into a 2 row by 4 column grid

```

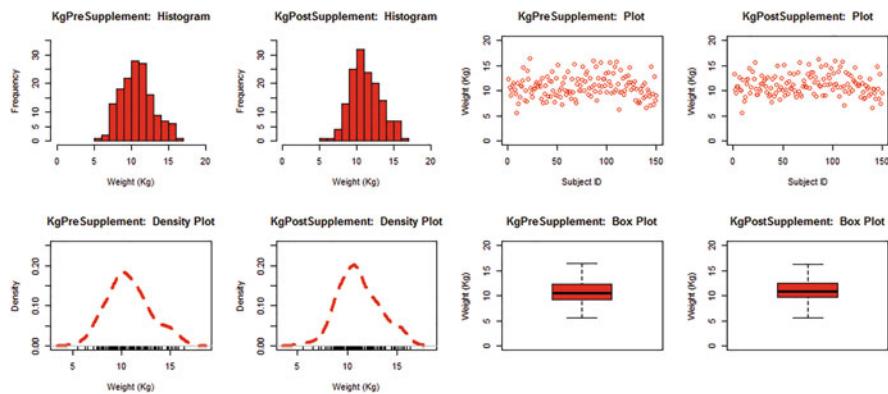


Figure 4.1: Comparison of pretest weights to posttest weights using unstacked data—1

```
# More than 8 figures in one common plot makes it difficult to
# see results, read titles and scales, etc.
stripchart(PrePostWtUnstack.df$KgPreSupplement, col=c("red"),
           xlab="Weight (Kg)", xlim=c(0,20),
           main="KgPreSupplement: Stripchart")
stripchart(PrePostWtUnstack.df$KgPostSupplement, col=c("red"),
           xlab="Weight (Kg)", xlim=c(0,20),
           main="KgPostSupplement: Stripchart")
dotchart(PrePostWtUnstack.df$KgPreSupplement, col=c("red"),
           xlab="Weight (Kg)", xlim=c(0,20),
           main="KgPreSupplement Dotchart")
dotchart(PrePostWtUnstack.df$KgPostSupplement, col=c("red"),
           xlab="Weight (Kg)", xlim=c(0,20),
           main="KgPostSupplement: Dotchart")
qqnorm(PrePostWtUnstack.df$KgPreSupplement, col=c("red"),
       main="KgPreSupplement Q-Q Plot\nand Q-Q Line")
qqline(PrePostWtUnstack.df$KgPreSupplement)
qqnorm(PrePostWtUnstack.df$KgPostSupplement, col=c("red"),
       main="KgPostSupplement: Q-Q Plot\nand Q-Q Line")
qqline(PrePostWtUnstack.df$KgPostSupplement)
vioplot::vioplot(PrePostWtUnstack.df$KgPreSupplement,
                 names=c("KgPreSupplement"),
                 drawRect=TRUE,                      # Add the box
                 col="red",                          # Color of violin
                 ylim=c(0,20),                      # Y axis scale
                 horizontal=FALSE,                  # Horizontal violin plot
                 lty=2)                             # Dashed line
title("KgPreSupplement: Violin Plot")
```

```
vioplot::vioplot(PrePostWtUnstack.df$KgPostSupplement,
  names=c("KgPostSupplement"),
  drawRect=TRUE,                                # Add the box
  col="red",                                     # Color of violin
  ylim=c(0,20),                                   # Y axis scale
  horizontal=FALSE,                             # Horizontal violin plot
  lty=2)                                         # Dashed line
  title("KgPostSupplement: Violin Plot")
```

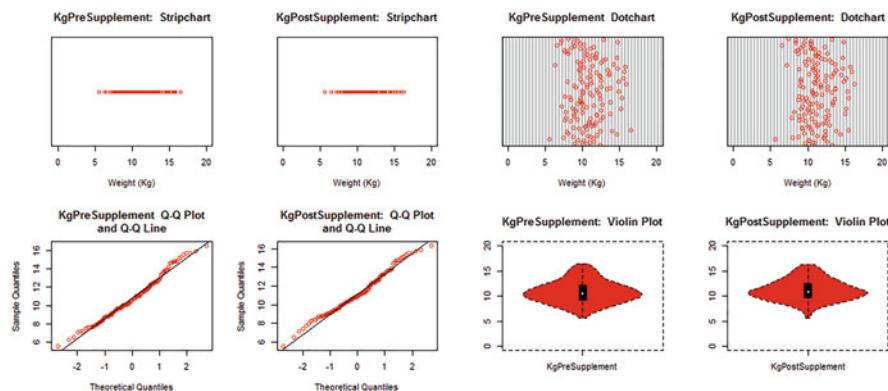


Figure 4.2: Comparison of pretest weights to posttest weights using unstacked data—2

After viewing these figures, which focused on different views of the two numeric object variables `KgPreSupplement` and `KgPostSupplement`, it seems that each object variable generally follows a normal distribution of values (e.g., the values approximate the bell-shaped curve). Although that observation is as yet unconfirmed, a few more slightly embellished graphics will help better understand the nature of the data, with more emphasis on normal distribution covered in a later part of this lesson (Fig. 4.3).

R Input

```
install.packages("descr")
library(descr)                                # Load the descr package.
help(package=descr)                            # Show the information page.
sessionInfo()                                 # Confirm all attached packages.

savelwd      <- par(lwd=4)                      # Heavy line
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
descr:::histkdnc(PrePostWtUnstack.df$KgPreSupplement,
  breaks=0, include.lowest=TRUE, right=TRUE, col = "wheat1",
```

```

border="black", xlim=c(0,20), ylim=c(0.0,0.25),
main ="KgPreSupplement with Density Plot Overlay (Red)
and Normal Curve (Blue)",
xlab = "Weight (Kg)", font.lab=2, font.axis=2)
descr:::histkdnc(PrePostWtUnstack.df$KgPostSupplement,
breaks=0, include.lowest=TRUE, right=TRUE, col = "wheat1",
border="black", xlim=c(0,20), ylim=c(0.0,0.25),
main ="KgPostSupplement with Density Plot Overlay (Red)
and Normal Curve (Blue)",
xlab = "Weight (Kg)", font.lab=2, font.axis=2)
par(savelwd) # Return to original value.
# The specialized function descr:::histkdnc() function
# generates an attractive and concise display, enhancing
# visualization of data distribution.

```

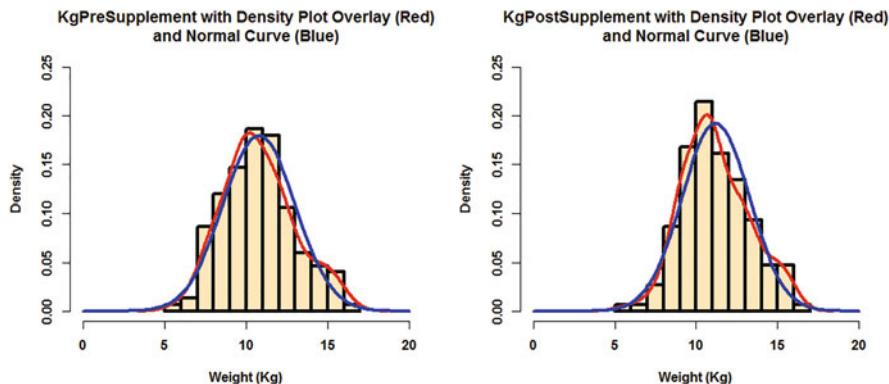


Figure 4.3: Distribution of pretest weights and posttest weights using unstacked data—1

Using the `descr:::histkdnc()` function and organizing a side-by-side comparison of density plots in comparison to the normal curve, there is some degree of assurance that the distribution of values for the two object variables `KgPreSupplement` and `KgPostSupplement` begin to approximate the normal curve. The exact nature of distribution values is addressed in a later part of this lesson. Even so, the early use of these graphics is quite helpful in investigating the nature of the data and how it can be used for later analyses.

4.5 Descriptive Statistics for Initial Analysis of the Data

It was identified previously that the dataframe `PrePostWtUnstack.df` has at least one missing datum. As a Quality Assurance practice, it is common to use the `is.na()` function and the `complete.cases()` function against the entire dataframe. Both functions return a TRUE or FALSE response, depending on

the function and the outcome of whether data are missing or data are not missing. To make the presentation to the screen compact, wrap the table() function around the is.na() function and the complete.cases() function.

R Input

```
table(is.na(PrePostWtUnstack.df)) # Missing data
```

R Output

FALSE	TRUE
449	1

R Input

```
table(complete.cases(PrePostWtUnstack.df)) # Complete cases
```

R Output

FALSE	TRUE
1	149

For this simple dataset of KgPreSupplement and KgPostSupplement for the otherwise unidentified biological specimens, the summary() function may be all that is necessary to gain a sense of the data. Note how the summary() function is applied against the entire dataframe, thus yielding information about all object variables including those that are not directly used, including ostensibly unnecessary information about Subject.

R Input

```
summary(PrePostWtUnstack.df)
```

R Output

Subject	KgPreSupplement	KgPostSupplement
1 : 1	Min. : 5.57	Min. : 5.57
2 : 1	1st Qu.: 9.28	1st Qu.: 9.64
3 : 1	Median :10.60	Median :10.89
4 : 1	Mean :10.79	Mean :11.19
5 : 1	3rd Qu.:12.24	3rd Qu.:12.56
6 : 1	Max. :16.47	Max. :16.30

(Other) : 144

NA's : 1

Although the `summary()` function is quite sufficient, a more complete set of descriptive statistics for individual object variables may be desired. To achieve this aim, review the prior lesson *Data Exploration, Descriptive Statistics, and Measures of Central Tendency* for a comprehensive review of the functions used for descriptive statistics, especially: `length()`, `asbio::Mode()`, `median()`, `mean()`, `sd()`, `quantile()`, `table()`, and finally `summary()`.

However, the use of these many one-by-one functions can be tedious and time-consuming. Consider, instead, functions from packages that provide a variety of descriptive statistics all as one attractive and compact output to the screen, allowing easy copy and paste to a word-processed document, if needed.

The `RcmdrMisc::numSummary()` function provides a very convenient and easy to use output of summary statistics. By default the `RcmdrMisc::numSummary()` function displays Mean, Standard Deviation, IQR (Interquartile Range, or Q3 minus Q1) Quartiles (0% (Minimum), 25%, 50% (Median), 75%, 100% (Maximum)), N, and NAs (the number of missing values, if any).

R Input

```
install.packages("RcmdrMisc", dependencies=TRUE)
library(RcmdrMisc)           # Load the RcmdrMisc package.
help(package=RcmdrMisc)      # Show the information page.
sessionInfo()                # Confirm all attached packages.

RcmdrMisc::numSummary(PrePostWtUnstack.df$KgPreSupplement,
                      statistics=c("mean", "sd", "quantiles"))
# Show descriptive statistics of KgPreSupplement.
```

R Output

mean	sd	0%	25%	50%	75%	100%	n
10.7863	2.21821	5.57	9.28	10.6	12.2375	16.47	150

R Input

```
RcmdrMisc::numSummary(PrePostWtUnstack.df$KgPostSupplement,
                      statistics=c("mean", "sd", "quantiles"))
# Show descriptive statistics of KgPostSupplement.
```

R Output

mean	sd	0%	25%	50%	75%	100%	n	NA
11.1885	2.07344	5.57	9.64	10.89	12.56	16.3	149	1

Output from the RcmdrMisc::numSummary() function also include details about missing data, when that is the case. As mentioned previously, it is never desirable to have missing data but it is a reality of the research process that cannot be ignored and must be accommodated as analyses are attempted.

The s20x::summaryStats() function is also quite good for generating detailed output of descriptive statistics. In contrast to the RcmdrMisc::numSummary() function, use of the s20x::summaryStats() function requires use of the na.rm=TRUE argument for when there are missing data. Otherwise, the s20x::summaryStats() will fail to generate output.²

R Input

```
install.packages("s20x", dependencies=TRUE)
library(s20x)                      # Load the s20x package.
help(package=s20x)                  # Show the information page.
sessionInfo()                      # Confirm all attached packages.

with(PrePostWtUnstack.df, s20x::summaryStats(KgPreSupplement))
# Show descriptive statistics, overall.
```

R Output

Minimum value:	5.57
Maximum value:	16.47
Mean value:	10.79
Median:	10.6
Upper quartile:	12.24
Lower quartile:	9.28
Variance:	4.92
Standard deviation:	2.22
Midspread (IQR):	2.96
Skewness:	0.34
Number of data values:	150

²The teaching dataset for this lesson was purposely structured to have at least one missing datum, to provide assurance that missing data would be addressed in this lesson.

R Input

```
with(PrePostWtUnstack.df, s20x::summaryStats(KgPostSupplement,
  na.rm=TRUE))
# Show descriptive statistics, overall.
```

R Output

Minimum value:	5.57
Maximum value:	16.3
Mean value:	11.19
Median:	10.89
Upper quartile:	12.56
Lower quartile:	9.64
Variance:	4.3
Standard deviation:	2.07
Midspread (IQR):	2.92
Skewness:	0.29
Number of data values:	150
Number of missing values:	1

Another function from an external package that is quite useful for presentation of descriptive statistics is the `pastecs::stat.desc()` function. The output is put into a very manageable side-by-side summary for `KgPreSupplement` and `KgPostSupplement`, allowing easy comparisons of a full set of descriptive statistics for the two object variables.^{3,4}

R Input

```
install.packages("pastecs", dependencies=TRUE)
library(pastecs)          # Load the pastecs package.
help(package=pastecs)      # Show the information page.
sessionInfo()              # Confirm all attached packages.

pastecs::stat.desc(PrePostWtUnstack.df[,2:3],
```

³If there are concerns about normal distribution and highly skewed data, it may be helpful to use the `pastecs::stat.pen()` function to obtain a full set of descriptive statistics that also include reference to Pennington's estimators of the mean.

⁴As an advance organizer to statistical measures of normal distribution, review output from the `pastecs::stat.desc()` function and focus on the `normtest.p` statistics. Compare these statistics to later statistical output on normal distribution.

```

basic=TRUE, # Show basic descriptive statistics.
desc=TRUE, # Show more descriptive statistics.
norm=TRUE, # Show normal distribution statistics.
p=0.95) # Probability for the mean's Confidence Interval.
# In the PrePostWtUnstack.df dataframe, Subject represents
# Column 1, KgPreSupplement represents Column 2, and
# KgPostSupplement represents Column 3. Note above how
# [,2:3] was used to obtain an attractive side-by-side
# comparison of output for the two object variables, allowing
# easy interpretation of similarities and differences in
# specific descriptive statistics (e.g., median, mean,
# std.dev, etc.) for each measured object variable.

```

R Output

	KgPreSupplement	KgPostSupplement
nbr.val	150.000000	149.000000
nbr.null	0.000000	0.000000
nbr.na	0.000000	1.000000
min	5.570000	5.570000
max	16.470000	16.300000
range	10.900000	10.730000
sum	1617.940000	1667.080000
median	10.600000	10.890000
mean	10.7862667	11.1884564
SE.mean	0.1811163	0.1698632
CI.mean.0.95	0.3578881	0.3356705
var	4.9204652	4.2991726
std.dev	2.2182122	2.0734446
coef.var	0.2056515	0.1853200
skewness	0.3416223	0.2880115
skew.2SE	0.8625153	0.7247801
kurtosis	-0.2700554	-0.2066746
kurt.2SE	-0.3430729	-0.2617061
normtest.W	0.9846847	0.9838892
normtest.p	0.0951337	0.0791077

Additional functions from other external R packages could be demonstrated, but the above functions should provide a broad representation of how descriptive statistics and measures of central tendency are determined when using R. Keep current with the R literature as there is a constantly evolving set of external packages, functions, and arguments in these packages to choose from for attractive presentation of descriptive statistics and measures of central tendency.

4.6 Quality Assurance, Data Distribution, and Tests for Normality

Quality assurance (QA) is a continuous and inclusive process. A great deal of attention has gone into the preparation of figures and descriptive statistics for the two numeric object variables of interest from the dataframe `PrePostWtUnstack.df`: `KgPreSupplement` and `KgPostSupplement`.

Although it may be redundant to once again look at descriptive statistics, use the `epiDisplay::summ()` function to gain a review of the descriptive statistics for each numeric object variable. Use the `graph=TRUE` argument to visually reinforce the nature of each numeric object variable, going beyond static descriptive statistics that are provided to the screen.

R Input

```
install.packages("epiDisplay", dependencies=TRUE)
library(epiDisplay) # Load the epiDisplay package.
help(package=epiDisplay) # Show the information page.
sessionInfo() # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
with(PrePostWtUnstack.df, epiDisplay::summ(KgPreSupplement,
    graph=TRUE, xlim=c(0,20)))
# Redundant confirmation of KgPresupplement.
with(PrePostWtUnstack.df, epiDisplay::summ(KgPostSupplement,
    graph=TRUE, xlim=c(0,20)))
# Redundant confirmation of KgPresupplement.
```

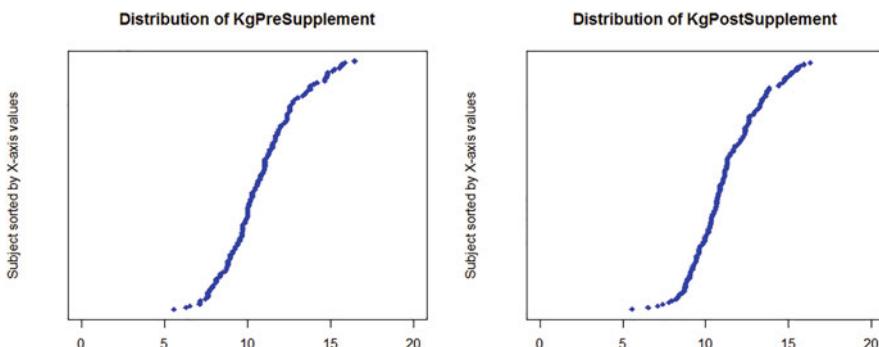


Figure 4.4: Distribution of pretest weights and posttest weights using unstacked data—2

These statistics are provided, again, to renew a sense of the data. A key interest when viewing the many figures and statistics on central tendency is the concern

over normal distribution for numeric object variables, or KgPreSupplement and KgPostSupplement in this lesson. Normal distribution is important in that many inferential tests are only used, appropriately, if the underlying data exhibit normal distribution. Specifically for this lesson, Student's t-Test for Matched Pairs is best used with numeric data that follow a pattern of normal distribution. Of course, that ideal is not always met and when that finding occurs a decision needs to be made on the degree of deviation away from normal distribution that can be accepted before another test, possibly a nonparametric test, is selected. The issue of test selection is discussed in more detail later in this lesson (Fig. 4.4).

There are many statistical tests that are specifically designed to offer judgment on adherence to normality, including:

- Anderson–Darling test for normality
- Cramer–von Mises test for normality
- Lilliefors (Kolmogorov–Smirnov) test for normality
- Pearson chi-square test for normality
- Shapiro–Francia test for normality

Each test provides a slightly different view toward normality. For this lesson, use the RVAideMemoire::byf.shapiro() function, which provides a fairly easy to understand summary of normality statistics for numeric object variables.

R Input

```
install.packages("RVAideMemoire", dependencies=TRUE)
library(RVAideMemoire)      # Load the RVAideMemoire package.
help(package=RVAideMemoire) # Show the information page.
sessionInfo()              # Confirm all attached packages.

RVAideMemoire::mshapiro.test(
  PrePostWtUnstack.df$KgPreSupplement)
# Normality test of KgPreSupplement
```

R Output

Multivariate Shapiro-Wilk normality test

```
data: ()
W = 0.9847, p-value = 0.0951
```

With a p-value of 0.0951 it can be said with a fair degree of confidence that the data from KgPreSupplement follow an acceptable pattern of normal distribution. Review the qqnorm() function and output from the pastecs::stat.desc() function (specifically the norm=TRUE argument) to confirm the declaration that there is an acceptable degree of normal distribution for KgPreSupplement.⁵

R Input

```
RVAideMemoire::mshapiro.test(
  PrePostWtUnstack.df$KgPostSupplement)
# Normality test of KgPostSupplement
```

R Output

```
Multivariate Shapiro-Wilk normality test

data: ()
W = 0.9839, p-value = 0.0791
```

With a p-value of 0.0791 it can be said with a fair degree of confidence that the data from KgPostSupplement follow an acceptable pattern of normal distribution. Review the qqnorm() function and output from the pastecs::stat.desc() function (specifically the norm=TRUE argument) to confirm the declaration that there is an acceptable degree of normal distribution for KgPostSupplement.

It is common to focus on a p-value of either $p \leq 0.05$ or $p \leq 0.01$ when considering normality tests, such as the Anderson–Darling Test for Normality or the Shapiro Test for Normality. Remember that due to the way normality tests are worded, in the affirmative:

- Lower calculated p-values provide evidence against the null hypothesis. That is to say, if the calculated p-value is less than or equal to the criterion p-value (e.g., significance level), the rules-based decision is to reject the null hypothesis and to declare that the data do not follow a pattern of normal distribution.
- Higher calculated p-values fail to provide evidence against the null hypothesis. That is to say, if the calculated p-value is greater than the criterion

⁵For nearly all statistical tests, the Null Hypothesis is worded in a negative fashion and is typically stated as *There is no statistically significant difference between A and B in terms of C*. Somewhat different, the Null Hypothesis for a normality test is instead worded in the affirmative and is typically stated as *The data follow a normal distribution*. Give attention to this different approach to the way the normality tests are worded when interpreting the p-values, significance levels, and outcomes.

p-value (e.g., significance level), the rules-based decision is to accept (e.g., fail to reject) the null hypothesis and to declare that the data seem to follow a pattern of normal distribution.

Although Anderson–Darling is possibly the most frequently used test for addressing normality, the Shapiro–Wilk test was used in this lesson and the results provided direction that normal distribution was seemingly evident for the two numeric object variables in question: KgPreSupplement and KgPostSupplement. In an abundance of caution, use the `ggplot2::ggplot()` function, with appropriate arguments, to again visually examine distribution of the two numeric object variables. For this task, produce a Q-Q plot and a density plot.

R Input

```
install.packages("ggplot2", dependencies=TRUE)
library(ggplot2)          # Load the ggplot2 package.
help(package=ggplot2)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("ggthemes", dependencies=TRUE)
library(ggthemes)          # Load the ggthemes package.
help(package=ggthemes)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("ggmosaic", dependencies=TRUE)
library(ggmosaic)          # Load the ggmosaic package.
help(package=ggmosaic)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("gridExtra", dependencies=TRUE)
library(gridExtra)          # Load the gridExtra package.
help(package=gridExtra)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("grid", dependencies=TRUE)
library(grid)               # Load the grid package.
help(package=grid)          # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)              # Load the scales package.
help(package=scales)        # Show the information page.
sessionInfo()             # Confirm all attached packages.
```

```

# KgPreSupplement, overall since there are no breakout groups
QQPlotKgPreSupplement <-
ggplot2::ggplot(PrePostWtUnstack.df,
  aes(sample=KgPreSupplement)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.25) +
  ggtitle(
    "KgPreSupplement QQ-Plot and QQ-Line, Overall") +
  labs(x = "\nTheoretical", y = "Weight (Kg)\n") +
  ylim(0,20) +
  theme_classic()

# KgPostSupplement, overall since there are no breakout groups
QQPlotKgPostSupplement <-
ggplot2::ggplot(PrePostWtUnstack.df,
  aes(sample=KgPostSupplement)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.25) +
  ggtitle(
    "KgPostSupplement QQ-Plot and QQ-Line, Overall") +
  labs(x = "\nTheoretical", y = "Weight (Kg)\n") +
  ylim(0,20) +
  theme_classic()

# KgPreSupplement, overall since there are no breakout groups
DensityCurveKgPreSupplement <-
ggplot2::ggplot(PrePostWtUnstack.df,
  aes(x=KgPreSupplement)) +
  geom_density(size=1.5, col=("red")) +
  ggtitle(
    "KgPreSupplement Density Curve, Overall") +
  scale_x_continuous(name="Weight (Kg)", limits=c(0,20),
    breaks=seq(0,20,1.0)) +
  scale_y_continuous(name="Density", limits=c(0.0,0.25),
    breaks=seq(0.0,0.25,0.05)) +
  theme_classic()

# KgPostSupplement, overall since there are no breakout groups
DensityCurveKgPostSupplement <-
ggplot2::ggplot(PrePostWtUnstack.df,
  aes(x=KgPostSupplement)) +
  geom_density(size=1.5, col=("red")) +
  ggtitle(

```

```
"KgPostSupplement Density Curve, Overall") +
  scale_x_continuous(name="Weight (Kg)", limits=c(0,20),
    breaks=seq(0,20,1.0)) +
  scale_y_continuous(name="Density", limits=c(0.0,0.25),
    breaks=seq(0.0,0.25,0.05)) +
  theme_classic()

par(ask=TRUE)
gridExtra::grid.arrange(
  QQPlotKgPreSupplement,
  QQPlotKgPostSupplement,
  DensityCurveKgPreSupplement,
  DensityCurveKgPostSupplement, ncol=2)
```

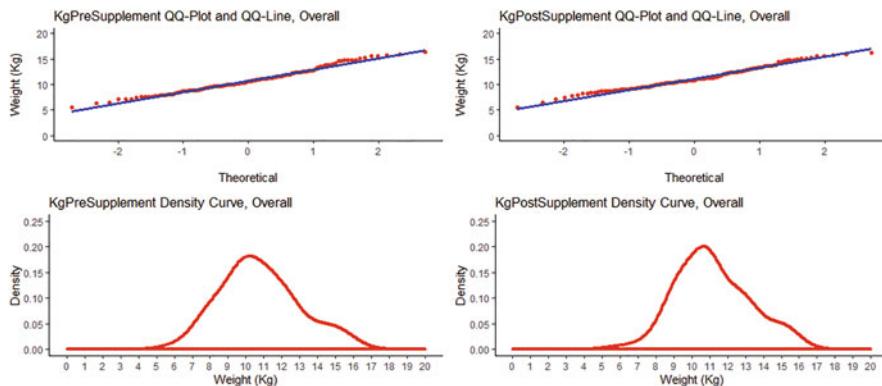


Figure 4.5: Distribution of pretest weights and posttest weights using unstacked data—3

For the Q-Q plot, observe how the data (represented as red dots) mostly follow along the straight line, which is an indicator of normal distribution. For the density plot, the curve is a reasonable facsimile of the normal (e.g., bell-shaped) curve, another indicator of normal distribution.

4.7 Statistical Test(s)

The dataset for this part of the lesson was prepared in unstacked format. A brief demonstration of the same data, but presented as a dataset in stacked format, follows later in this lesson. Review the help pages for the `stack()` function and the `unstack()` function to learn more about this issue (Fig. 4.5).

Although a great deal of information has been given to visual presentations of data distribution and descriptive statistics, the primary emphasis for this lesson revolves around Student's t-Test for Matched Pairs and the Null Hypothesis associated with this lesson:

Null Hypothesis (H₀) There is no statistically significant difference ($p \leq 0.05$) in the pretest weight (Kg) of unidentified adult biological specimens and the posttest weight (Kg) of unidentified adult biological specimens at the end of a feeding experiment, where a tasteless and odorless mineral supplement was introduced into the feeding program.

The t.test() function, with selection of appropriate arguments, should be used as a first attempt at conducting the Student's t-Test for Matched Pairs. The t.test() function is gained from the stats package, which is included among the packages first gained when R is downloaded.

R Input

```
# Default selections.
t.test(PrePostWtUnstack.df$KgPreSupplement, # Measured variable
       PrePostWtUnstack.df$KgPostSupplement,      # Measured variable
       paired=TRUE,                                # Matched pairs
       na.rm=TRUE)                                 # Missing data
# The data are in unstacked (e.g., wide) format. Notice how
# a comma was used to separate the one measured variable from
# the other measured variable.
#
# Notice how the paired=TRUE and na.rm=TRUE arguments were
# used.
```

R Output

Paired t-test

```
PrePostWtUnstack.df$KgPreSupplement and
PrePostWtUnstack.df$KgPostSupplement
t = -7.547, df = 148, p-value = 0.00000000000421
```

R Input

```
# Assumption that the two variances are equal.
t.test(PrePostWtUnstack.df$KgPreSupplement, # Measured variable
       PrePostWtUnstack.df$KgPostSupplement,      # Measured variable
       alternative="two.sided",                      # Two-sided t-Test
       paired=TRUE,                                # Matched pairs
       na.rm=TRUE,                                 # Missing data
       var.equal=TRUE)                            # Equal variance
```

R Output

Paired t-test

```
PrePostWtUnstack.df$KgPreSupplement and
PrePostWtUnstack.df$KgPostSupplement
t = -7.547, df = 148, p-value = 0.00000000000421
```

R Input

```
# Assumption that the two variances are not equal.
t.test(PrePostWtUnstack.df$KgPreSupplement, # Measured variable
       PrePostWtUnstack.df$KgPostSupplement,      # Measured variable
       alternative="two.sided",                  # Two-sided t-Test
       paired=TRUE,                            # Matched pairs
       na.rm=TRUE,                            # Missing data
       var.equal=FALSE)                      # Equal variance
                                              # is not TRUE
```

R Output

Paired t-test

```
PrePostWtUnstack.df$KgPreSupplement and
PrePostWtUnstack.df$KgPostSupplement
t = -7.547, df = 148, p-value = 0.00000000000421
```

Again, a comma and not a tilde was used to separate the two unstacked object variables in `PrePostWtUnstack.df`: `KgPreSupplement` and `KgPostSupplement`. There is no grouping variable with unstacked data as there is with stacked data since the two sets of unstacked data represent their own groups. By design, look at the error message that is generated by R when a `~` (e.g., tilde) instead of a `,` (e.g., comma) was used with unstacked data, a common mistake that is easily fixed by the use of correct syntax.

R Input

```
# Purposeful introduction of an error, by using ~ instead of ,
t.test(PrePostWtUnstack.df$KgPreSupplement ~ # Measured
       PrePostWtUnstack.df$KgPostSupplement,      # Measured
       paired=TRUE,                            # Matched pairs
       na.rm=TRUE)                           # Missing data
```

R Output

```
Error in t.test.formula(PrePostWtUnstack.df$KgPreSupplement ~
  PrePostWtUnstack.df$KgPostSupplement, :
  grouping factor must have exactly 2 levels
```

As useful as the `t.test()` function is, it is often a good idea to analyze the same dataset with a complementary function, as a redundant data check. It may also be helpful to question assumptions about data distribution, again as a redundant data check. For comparison of statistics generated by the `t.test()` function, consider the `PairedData::yuen.t.test()` function.

R Input

```
install.packages("PairedData", dependencies=TRUE)
library(PairedData)          # Load the PairedData package.
help(package=PairedData)     # Show the information page.
sessionInfo()                # Confirm all attached packages.

PairedData::yuen.t.test(PrePostWtUnstack.df$KgPreSupplement,
  PrePostWtUnstack.df$KgPostSupplement,
  tr = 0.1,                  # Trim 10 percent of the data.
  alternative = c("two.sided"), na.rm=TRUE, paired=TRUE)
# Ostensibly, extreme values are trimmed when using the tr
# argument, to mitigate the impact of extreme values on
# overall outcomes.
```

R Output

```
Paired Yuen test, trim=0.1

data: x and y
t = -6.911, df = 120, p-value = 0.000000000249
```

The Paired Yuen test *p*-value of 0.000000000249 is certainly not equivalent to the *p*-value that was obtained by using the `stats::t.test()` function, using the `paired=TRUE` argument. However, even with extreme values removed, by using the argument `tr = 0.1`, the calculated *p*=values are in range to each other and more importantly, the outcome in terms of the Null Hypothesis remains the same.

By trimming the largest and smallest datapoints, Yuen's trimmed mean test for paired samples reduces the effect of extreme values, which is especially useful for when the data do not exhibit a semblance of normal distribution. This may

not have been a concern for this specific dataset, but there are far too many times when data have extreme values and Yuen's trimmed mean test for paired samples is at least a reasonable consideration for statistical analyses—serving as another Quality Assurance measure even if considered redundant by some.

4.8 Summary of Outcomes

The graphics and statistics in this lesson provided a great deal of useful information in support of informed decision-making about the data associated with this study on the introduction of a mineral supplement into a feeding program. Of immediate importance, however, focus on outcomes of the Student's t-Test for Matched Pairs, used to address the Null Hypothesis.

Assuming that the two variances are equal, the following statistics of importance have been gained from two sets of analyses (descriptive statistics and Student's t-Test) for this problem:

	n	mean	sd
KgPreSupplement	150	10.79	2.22
KgPostSupplement	149	11.19	2.07

$$p\text{-value} = 0.0000000000421$$

Attention to the calculated p-value is perhaps the easiest way to view differences for this problem:

- The calculated p-value is 0.0000000000421.
- The criterion p-value from the previously stated Null Hypothesis is 0.05.
- The calculated p-value (0.0000000000421) is less than the criterion p-value (0.05).

Therefore, the Null Hypothesis is rejected and it can be claimed that there is a statistically significant difference ($p \leq .05$) between KgPreSupplement and KgPostSupplement.⁶ The mean weight for subjects was greater after the supplement was provided (11.19 Kg, KgPostSupplement) than the mean weight of subjects before the supplement was provided (10.79 Kg, KgPreSupplement). This difference is not due to chance but instead represents a true difference between the two measured object variables, at $p \leq 0.05$.

There is no knowledge if gender, breed, etc., have any possible influence of outcomes. Data were not provided for these factors. Instead, as this pretest and posttest matched pairs lesson was structured, it is only known that matched

⁶Instead of stating *The Null Hypothesis is rejected.*, it is not uncommon to see the expression *The Null Hypothesis is not accepted.* in the literature. Discussion on differences in the fine points of these two phrases is beyond the purpose of this lesson.

subjects, overall at $p \leq 0.05$, had a significant difference (e.g., gain) in weight after administration of the feeding supplement compared to weights before administration of the supplement.

4.9 Addendum 1: R-Based Tools for Unstacked (e.g., Wide) Data

It is very common to see data in unstacked (e.g., wide) format, especially data that are shown to the public. However, it is also common for researchers to work with, often by preference, data that are instead in stacked (e.g., long) format. Look below at one way (of many ways) R can be used to accommodate this concern, transforming unstacked data into stacked format.

For demonstration purposes, create a sample dataset and put it into unstacked (e.g.) wide format.

R Input

```
Addendum1Wide.df <- read.table(text = "
Group      A      B      C      D      E      F
Group1  0.67  0.56  0.54  0.03 -0.97 -1.09
Group2  0.52  0.43  0.51  0.05 -0.86 -0.96
Group3  0.20  0.30  0.19  0.08  0.02  0.09
Group4 -1.30 -1.13 -1.78 -0.05 -0.23 -0.23
Group5 -1.60 -1.30 -1.34 -0.23 -0.45 -0.45", header = TRUE)

getwd()                      # Identify the working directory
ls()                          # List objects
attach(Addendum1Wide.df)      # Attach the data, for later use
Addendum1Wide.df              # Show the entire dataframe
summary(Addendum1Wide.df)    # Summary statistics
str(Addendum1Wide.df)         # Identify structure
```

R Output

```
Selected output is not shown, to save space.
'data.frame':   5 obs. of  7 variables:
 $ Group: Factor w/ 5 levels "Group1","Group2",...: 1 2 3 4 5
 $ A     : num  0.67 0.52 0.2 -1.3 -1.6
 $ B     : num  0.56 0.43 0.3 -1.13 -1.3
 $ C     : num  0.54 0.51 0.19 -1.78 -1.34
 $ D     : num  0.03 0.05 0.08 -0.05 -0.23
 $ E     : num  -0.97 -0.86 0.02 -0.23 -0.45
 $ F     : num  -1.09 -0.96 0.09 -0.23 -0.45
```

Use the `reshape2::melt()` function to put the data that are in unstacked (e.g., wide) format into stacked (e.g., long) format.⁷ As the name of the `reshape2` package suggests, the data will be reshaped, from unstacked format to stacked format in this lesson, or perhaps it is easier to say that the data will be changed from wide format (`Addendum1Wide.df`) to long format (`Addendum1Long.df`).

R Input

```
install.packages("reshape2", dependencies=TRUE)
library(reshape2)                      # Load the reshape2 package.
help(package=reshape2)                 # Show the information page.
sessionInfo()                         # Confirm all attached packages.

Addendum1Long.df <- reshape2::melt(Addendum1Wide.df,
  id.vars = "Group")
# Create the dataframe AddendumLong1.df by reshaping
# the dataframe Addendum1Wide.df. They are not changed but
# are instead only changed from unstacked (e.g., wide)
# format to stacked (e.g., long) format.

getwd()                                # Identify the working directory
ls()                                    # List objects
attach(Addendum1Long.df)                # Attach the data, for later use
Addendum1Long.df                        # Show the entire dataframe
summary(Addendum1Long.df)              # Summary statistics
str(Addendum1Long.df)                  # Identify structure
```

R Output

```
Selected output is not shown, to save space.
'data.frame': 30 obs. of 3 variables:
 $ Group    : Factor w/ 5 levels "Group1","Group2",...: 1 ...
 $ variable: Factor w/ 6 levels "A","B","C","D",...: 1 1 ...
 $ value    : num  0.67 0.52 0.2 -1.3 -1.6 0.56 0.43 0.3 ...
```

Prepare a simple graphic of the data in `Addendum1Long.df` to confirm that the data are organized as desired and can be used with R. Recall that this dataset is merely for demonstration purposes, showing the ease by which data can be

⁷Although the `reshape2` package has many useful functions, it cannot be ignored that the tidyverse environment supports many tools that are also quite popular with the R community, with these tools used in an attempt to put data into desired format. This early-on lesson uses the `reshape2` package. The more encompassing tidyverse package is addressed in later lessons.

reshaped.

R Input

```
ggplot2::ggplot(Addendum1Long.df,
  aes(x=Group, y=value, color=variable)) +
  geom_point() +
  geom_line() +
  coord_flip() +
  facet_wrap(~ variable) +
  ggtitle(
  "Long-Format Datapoints by Breakout Groups A to F") +
  theme_classic()
# Data are put into a grid, where each
# value shows.
# Look for, but ignore, Warning messages
# since there is only one observation for
# each group.
```

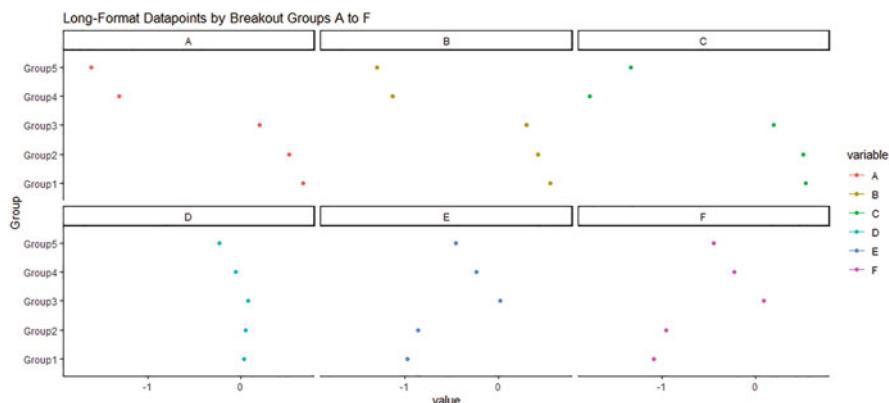


Figure 4.6: Distribution of datapoints by groups using stacked data

The `reshape2::melt()` function was used in this addendum to transform data from wide format to long format. The `reshape2::melt()` function is fairly easy to use and it generates desired results. However, tools found in the tidyverse are increasingly the preferred approach for transforming data from one format to another format, such as wide to long and these tools are demonstrated in a later lesson (Fig. 4.6).

4.10 Addendum 2: Stacked Data and Student's t-Test for Matched Pairs

Apply the reshape2::melt() function against the unstacked (e.g., wide) dataframe PrePostWtUnstack.df and put the data into stacked (e.g., long) format. Then, apply the Student's t-Test for Matched Pairs against the data, but now using the appropriate symbols to accommodate data in stacked (e.g., long) format.

R Input

```
head(PrePostWtUnstack.df, n=3)
# Review the data, prior to transformation
# of the data from wide to long.
```

R Output

Subject	KgPreSupplement	KgPostSupplement
1	12.40	13.28
2	10.64	10.18
3	8.97	9.60

R Input

```
tail(PrePostWtUnstack.df, n=3)
# Review the data, prior to transformation
# of the data from wide to long.
```

R Output

Subject	KgPreSupplement	KgPostSupplement
148	12.86	13.32
149	8.13	7.42
150	8.97	9.60

R Input

```
str(PrePostWtUnstack.df)
# Review the data, prior to transformation
# of the data from wide to long.
```

R Output

```
'data.frame': 150 obs. of 3 variables:
 $ Subject       : Factor w/ 150 levels "1","2","3","4",...
 $ KgPreSupplement : num 12.4 10.64 8.97 10.01 10.94 ...
 $ KgPostSupplement: num 13.3 10.2 9.6 10.9 10.8 ...
```

R Input

```
PrePostWtLONG.df <- reshape2::melt(PrePostWtUnstack.df,
  id.vars = "Subject")
# Transform the dataframe PrePostWtUnstack.df from
# unstacked (e.g., wide) format to stacked (e.g.,
# long) format.

getwd()                      # Identify the working directory
ls()                          # List objects
attach(PrePostWtLONG.df)     # Attach the data, for later use

head(PrePostWtLONG.df, n=3)
# Review the data, after transformation
# of the data from wide to long.
```

R Output

	Subject	variable	value
1	1	KgPreSupplement	12.40
2	2	KgPreSupplement	10.64
3	3	KgPreSupplement	8.97

R Input

```
tail(PrePostWtLONG.df, n=3)
# Review the data, after transformation
# of the data from wide to long.
```

R Output

	Subject	variable	value
298	148	KgPostSupplement	13.32
299	149	KgPostSupplement	7.42
300	150	KgPostSupplement	9.60

R Input

```
str(PrePostWtLONG.df)
# Review the data, after transformation
# of the data from wide to long.
```

R Output

```
'data.frame': 300 obs. of 3 variables:
 $ Subject : Factor w/ 150 levels "1","2","3","4",...: 1 2 3 ...
 $ variable: Factor w/ 2 levels "KgPreSupplement",...: 1 1 1 ...
 $ value   : num 12.4 10.64 8.97 10.01 10.94 ...
```

R Input

```
PrePostWtLONG.df          # Show the entire dataframe
summary(PrePostWtLONG.df) # Summary statistics
```

R Output

	Subject	variable	value
1	: 2	KgPreSupplement :150	Min. : 5.57
2	: 2	KgPostSupplement:150	1st Qu.: 9.55
3	: 2		Median :10.77
4	: 2		Mean :10.99
5	: 2		3rd Qu.:12.40
6	: 2		Max. :16.47
(Other)	:288		NA's :1

Based on review of the data in unstacked format as well as now, where the data are in long format, it is known that there is a missing datum in this dataframe:

R Input

```
complete.cases(PrePostWtLONG.df)
```

By careful review and use of the `complete.cases()` function, it was observed that Subject 19 was missing a measurement for KgPostSupplement. It is necessary to remove data for Subject 19—both the KgPreSupplement datum (which exists) and not only the KgPostSupplement datum (which is missing). There must be a match when using any type of matched pairs analysis and there can be no match if a needed datum is missing. It is not enough to only remove the row for Subject

19 KgPostSupplement but instead the row for Subject 19 KgPreSupplement must also be removed.

Although the syntax below may not be the most elegant, look at one way from among many potential ways missing data can be accommodated.⁸ This is especially a concern for matched pairs data since a missing measured datum means that there is no match.

R Input

```
SubjectListToRemove <- 19
# Subject 19 is the subject with missing data.

PrePostWtLONGAdjusted.df <-
  PrePostWtLONG.df [ ! PrePostWtLONG.df$Subject %in%
  SubjectListToRemove, ]
```

Identify the Subject (Subject 19 in this example) that must be removed in a separate file (e.g., SubjectListToRemove). Then, use standard R syntax such as ! (which means not) to remove the data in all cases (e.g., KgPreSupplement and not only KgPostSupplement).

R Input

```
getwd()                      # Working directory
ls()                          # List objects
attach(PrePostWtLONGAdjusted.df) # Attach the data
str(PrePostWtLONGAdjusted.df)   # Identify structure
PrePostWtLONGAdjusted.df       # Show the entire dataframe
summary(PrePostWtLONGAdjusted.df) # Summary statistics
# Note how there are 149 pairs,
# not 150 pairs.
```

R Output

Subject	variable	value
1 : 2	KgPreSupplement :149	Min. : 5.57

⁸Tools from the tidyverse environment deserve consideration as these tools can accommodate deletion of data from a dataset. The syntax in this lesson for data deletion is purposely used to show the heuristics of how data are deleted, prior to the simpler use of functions that somewhat mask the mechanics of this action. As an example, the `tidy::drop_na()` function is often used to remove (e.g., `drop`) rows containing missing values, but it is not as descriptive in terms of understanding how data are removed as the syntax purposely used in this lesson.

2	:	2	KgPostSupplement:149	1st Qu.: 9.55
3	:	2		Median :10.79
4	:	2		Mean :11.00
5	:	2		3rd Qu.:12.40
6	:	2		Max. :16.47
(Other):286				

Now, with Subject 19 removed use the adjusted dataframe to conduct the analysis.

R Input

```
t.test(value ~ variable,      # Measured Value ~ Grouped Variable
       data=PrePostWtLONGAdjusted.df,    # Dataframe
       alternative="two.sided",   # Two-sided t-Test
       paired=TRUE,                  # Matched pairs
       var.equal=FALSE)             # Equal variance
```

R Output

```
Paired t-test

data: value by variable
t = -7.547, df = 148, p-value = 0.00000000000421
```

The result of applying Student's t-Test for Matched Pairs against data in stacked (e.g., long) format is equivalent to what was seen earlier when Student's t-Test for Matched Pairs was applied against data in unstacked (e.g., wide) format, with the calculated p-value at 0.00000000000421. And, for this addendum a different approach was used to accommodate missing data. As is nearly always the case with R, there is no one and only one way to structure a solution to the problem at hand.

4.11 Addendum 3: The Impact of N on Student's t-Test

The results of the Student's t-Test for Matched Pairs provided evidence that there was a statistically significant difference ($p \leq 0.5$) in KgPreSupplement weights and KgPostSupplement weights for the entire dataset of 150 subjects, where individual biological specimens were matched against their weight change (if any, pretest to posttest) after a supplement of some type was introduced into the feeding program:

```
#           n  mean    sd
# KgPreSupplement 150 10.79  2.22
# KgPostSupplement 149 11.19  2.07
#
# p-value = 0.000000000004206
```

Think of the cost (money for salaries and supporting infrastructure as well as time on task) for managing a research study with 150 biological subjects. If non-human, these specimens need special attention for housing, feed, water, lighting, ventilation, exercise, daily care and management, etc. The cost of care for biological specimens is by no means insignificant. When engaged in research, is it possible to obtain the same general outcomes with a smaller, and therefore less expensive, sample of 30 subjects? To explore this question, obtain the base statistics from the original sample, then use the sample() function against the dataframe to obtain a sample of only 30 subjects.

R Input

```
summary(PrePostWtUnstack.df) # N = 150
```

R Output

	Subject	KgPreSupplement	KgPostSupplement
1	: 1	Min. : 5.57	Min. : 5.57
2	: 1	1st Qu.: 9.28	1st Qu.: 9.64
3	: 1	Median :10.60	Median :10.89
4	: 1	Mean :10.79	Mean :11.19
5	: 1	3rd Qu.:12.24	3rd Qu.:12.56
6	: 1	Max. :16.47	Max. :16.30
(Other)	:144	NA's :1	

R Input

```
base::set.seed(8) # Set the seed.
# See prior lessons why the set.seed() function
# is used to provide consistent outcomes when
# constructing user-created datasets.

# Apply the sample() function
Sample30.PrePostWtUnstack.df <- PrePostWtUnstack.df[
  sample(1:dim(PrePostWtUnstack.df)[1],
  size=30, replace=FALSE),
  # As shown in the descriptive name for this object,
```

```
# Sample30.PrePostWtUnstack.df consists of only 30
# subjects, taken from PrePostWtUnstack.df, the
# original dataframe.

summary(Sample30.PrePostWtUnstack.df)      # N = 30
```

R Output

Subject	KgPreSupplement	KgPostSupplement
1	Min. : 7.63	Min. : 8.21
13	1st Qu.: 9.03	1st Qu.:10.18
19	Median :11.25	Median :11.35
28	Mean :11.02	Mean :11.71
30	3rd Qu.:12.40	3rd Qu.:13.28
31	Max. :15.71	Max. :15.71
(Other):24	NA's :1	

R Input

```
# ---> N = 150 < ---
# Assumption that the two variances are equal.
t.test(PrePostWtUnstack.df$KgPreSupplement,
       PrePostWtUnstack.df$KgPostSupplement,
       alternative="two.sided",
       paired=TRUE, na.rm=TRUE, var.equal=TRUE)
```

R Output

Paired t-test

PrePostWtUnstack.df\$KgPreSupplement and
 PrePostWtUnstack.df\$KgPostSupplement
 $t = -7.547$, $df = 148$, $p\text{-value} = 0.00000000000421$

R Input

```
# ---> N = 30 < ---
# Assumption that the two variances are equal.
t.test(Sample30.PrePostWtUnstack.df$KgPreSupplement,
       Sample30.PrePostWtUnstack.df$KgPostSupplement,
       alternative="two.sided",
       paired=TRUE, na.rm=TRUE, var.equal=TRUE)
```

R Output

Paired t-test

```
Sample30.PrePostWtUnstack.df$KgPreSupplement and
Sample30.PrePostWtUnstack.df$KgPostSupplement
t = -5.756, df = 28, p-value = 0.00000354
```

Observe the p-value for when $N = 150$ and then observe the p-value for when $N = 30$. Use the sample() function multiple times with multiple Ns ($N = 125$, $N = 100$, $N = 50$, $N = 25$, to see if results are generally consistent.

Typically, the general outcome is upheld so that in this example, for both $N = 150$ and $N = 30$, there is a statistically significant difference ($p \leq 0.05$) in Pre-Supplement Weight (Kg) and the matched Post-Supplement Weight (Kg). And of course, a sample of $N = 30$ would be far less expensive and subsequently easier to manage than a sample of $N = 150$. Research and statistical analyses are as much about budgets and the prudent management of resources as they are about descriptive statistics, graphical output, and generated p values.

Although there are those who focus on gaining a large sample, it is perhaps more important to focus on the issue of sample representation. Is the above iteration of 30 subjects representative of the collection of all 150 subjects? That issue is beyond the purpose of this lesson, but sample representation should always be considered when conducting statistical analyses, regardless of the computing platform.

4.12 Addendum 4: Parametric v Nonparametric

It is assumed that the data for this lesson followed a normal distribution pattern. Based on this assumption, apply the Student's t-Test for Matched Pairs again to confirm outcomes. In an effort to introduce new material in this lesson that may be helpful in the future:

- Use another approach to accommodate missing data (other than tools in the tidyverse), a special concern for a matched pairs research study.
- Use the jmv::ttestPS() function to see the same general outcome, but viewing a different screen interface at output.

It is known that Subject 19 has a missing datum for KgPostSupplement. To accommodate this problem, a simple *kludge* is used, where Pair 19 is deleted from the PrePostWtStack.df dataset.⁹ Because of this action there will now be

⁹Review the meaning of *kludge*, from a computing perspective, if this is a new term.

an equal number of comparisons (e.g., pairs) between KgPreSupplement and KgPostSupplement.

R Input

```
PrePostWtUnstack.df  
  # Show the original dataset (PrePostWtStack.df),  
  # with all rows.  
  
nrow(PrePostWtUnstack.df) # List the number of rows.
```

R Output

```
[1] 150
```

R Input

```
RemoveS19PrePostWtUnstack.df <- PrePostWtUnstack.df[c(-19),]  
  # Create a new dataset (RemoveS19PrePostWtUnstack.df),  
  # with row 19 eliminated from this new dataset.  
  
RemoveS19PrePostWtUnstack.df  
  # Show the new dataset (RemoveS19PrePostWtUnstack.df),  
  # which should not have data for Subject 19.  
  
nrow(RemoveS19PrePostWtUnstack.df) # List the number of rows.
```

R Output

```
[1] 149
```

Use `RemoveS19PrePostWtUnstack.df`, the newly created dataset, to conduct the Student's t-Test for Matched Pairs using the `jmv::ttestPS()` function, but now with no concern about unequal lengths (e.g., missing data for one of the pairs). As an advance comment about defaults for the `jmv::ttestPS()` function, observe how the calculated p-value is presented as `p <= .001`, not `p <= 0.05`.

R Input

```
install.packages("jmv", dependencies=TRUE)  
library(jmv)                      # Load the jmv package.  
help(package=jmv)                  # Show the information page.
```

```
sessionInfo() # Confirm all attached packages.

jmv::ttestPS(RemoveS19PrePostWtUnstack.df,
  pairs=list(
    list(i1='KgPreSupplement', i2='KgPostSupplement')),
  students=TRUE, # Student's t-Test
  hypothesis='different', # Hypothesis for difference
  qq=TRUE, # QQ Plot
  desc=TRUE) # Descriptive statistics
# A QQ plot of residuals is generated.
```

R Output

Paired Samples T-Test

			p
KgPreSupplement	KgPostSupplement	Student's t	< .001

Descriptives

	N	Mean	Median	SD	SE
KgPreSupplement	149	10.8	10.6	2.21	0.181
KgPostSupplement	149	11.2	10.9	2.07	0.170

With nonparametric data, which is not the case for PrePostWtUnstack.df, it is prudent to use the wilcox.test() function to conduct the matched pairs analysis. As a useful QA exercise, however, apply the wilcox.test() function against the data to see the degree of consistency in outcomes, parametric v nonparametric.

R Input

```
wilcox.test(RemoveS19PrePostWtUnstack.df$KgPreSupplement,
  RemoveS19PrePostWtUnstack.df$KgPostSupplement,
  alternative=c("two.sided"), paired=TRUE, exact=TRUE,
  correct=TRUE, conf.int=TRUE, conf.level=0.95)
# Notice how there was no grouping factor. Instead,
# KgPreSupplement and KgPostSupplement were the two
# measured variables for this analysis.
```

R Output

```
Wilcoxon signed rank test with continuity correction

RemoveS19PrePostWtUnstack.df$KgPreSupplement and
RemoveS19PrePostWtUnstack.df$KgPostSupplement
V = 1644, p-value = 0.0000000000976
```

Using a nonparametric approach the calculated p-value remains less than 0.001. Whether the data are viewed from a parametric perspective or a nonparametric perspective, the outcome remains consistent. There was statistically significant weight gain at the end of the feeding study after application of the mineral supplement. The difference in pretest and posttest results was statistically significant ($p \leq 0.05$).

4.13 Addendum 5: Additional Practice Datasets for Data with Normal Distribution Patterns and Data That Do Not Exhibit Normal Distribution Patterns

Purpose of this Addendum

The purpose of this addendum is to continue with demonstrations on how the R environment supports Student's t-Test for Matched Pairs, the inferential test used to determine if there is a statistically significant differences for when data from two groups are presented in pairs. Perhaps the most common application of Student's t-Test for Matched Pairs is when it is used for pretest–posttest testing, but with clever thought there are many possible scenarios where Student's t-Test for Matched Pairs is used appropriately.¹⁰

Note In the front matter to this lesson, the syntax is presented and is then immediately followed in most cases by either a copy of screen output or an accompanying figure. That approach is not used in this addendum. View this addendum as practice homework-type bonus materials. Syntax is presented and descriptive text goes along with the syntax. However, screen output and figures are generally excluded and when they show it is only to offer mid-point guidance that analyses follow along in a correct manner. Either key the syntax in this addendum or copy and paste it into an editor—whatever is feasible and most convenient. And then, to use the common expression, Practice—Practice—Practice!

¹⁰Syntax is provided throughout this addendum, but of course this syntax is only a suggestion. Experiment and take other approaches to how the data can be analyzed and outcomes presented by using other functions and other arguments. Use this addendum as a confidence-building resource on how R is used with increasingly complex analyses.

Data That Exhibit Normal Distribution Patterns

R Input

```
ParametricPre <- round(rnorm(15, mean=120, sd=5))
# Use the rnorm() function to create ParametricPre, an
# object variable with N = 15, mean = 120, and sd = 5).
# Wrap the round() function around the output so that
# the datapoints are rounded as whole numbers.

ParametricPost <- round(rnorm(15, mean=130, sd=6))
# Use the rnorm() function to create ParametricPost, an
# object variable with N = 15, mean = 130, and sd = 6).
# Wrap the round() function around the output so that
# the datapoints are rounded as whole numbers.
```

The rnorm() function is used to create a set of numbers that are expected to show normal distribution. The round() function with no arguments is used to generate whole numbers.

R Input

```
ParametricExample.df <- data.frame(cbind(ParametricPre,
ParametricPost))
# Use the cbind() function to join ParametricPre and
# ParametricPost into ParametricExample.df. Use the
# data.frame() function to be sure that the new object is
# indeed a dataframe.

attach(ParametricExample.df)
str(ParametricExample.df)
ParametricExample.df

pastecs::stat.desc(ParametricExample.df,
basic=TRUE, # Show basic descriptive statistics.
desc=TRUE, # Show more descriptive statistics.
norm=TRUE, # Show normal distribution statistics.
p=0.95) # Probability for the mean's Confidence Interval.
# Along with a focus on basic descriptive statistics, give
# attention to the output associated with normtest.p.
```

R Output

	ParametricPre	ParametricPost
nbr.val	15.0000000	15.0000000
nbr.null	0.0000000	0.0000000
nbr.na	0.0000000	0.0000000
min	110.0000000	121.0000000
max	130.0000000	139.0000000
range	20.0000000	18.0000000
sum	1815.0000000	1967.0000000
median	120.0000000	132.0000000
mean	121.0000000	131.1333333
SE.mean	1.5766148	1.3232955
CI.mean.0.95	3.3815025	2.8381867
var	37.2857143	26.2666667
std.dev	6.1062029	5.1251016
coef.var	0.0504645	0.0390831
skewness	0.0000000	-0.2739225
skew.2SE	0.0000000	-0.2360915
kurtosis	-1.3130782	-1.0311488
kurt.2SE	-0.5857265	-0.4599659
normtest.W	0.9470544	0.9730576
normtest.p	0.4793110	0.9004681

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
qqnorm(ParametricExample.df$ParametricPre, col=c("red"),
       main="ParametricPre Q-Q Plot and Q-Q Line",
       , ,
       ylim=c(100,150)); qqline(ParametricExample.df$ParametricPre)
qqnorm(ParametricExample.df$ParametricPost, col=c("red"),
       main="ParametricPost Q-Q Plot and Q-Q Line",
       , ,
       ylim=c(100,150)); qqline(ParametricExample.df$ParametricPost)
# The QQ-Plots visually reinforce the normtest.p values and
# provide further assurance that there is some degree of
# semblance of normal distribution. Note how with a sample
# of only 15 subjects, one or two extreme values have a far
# greater impact on the totality of interpretation than what
# is the case with larger samples.

```

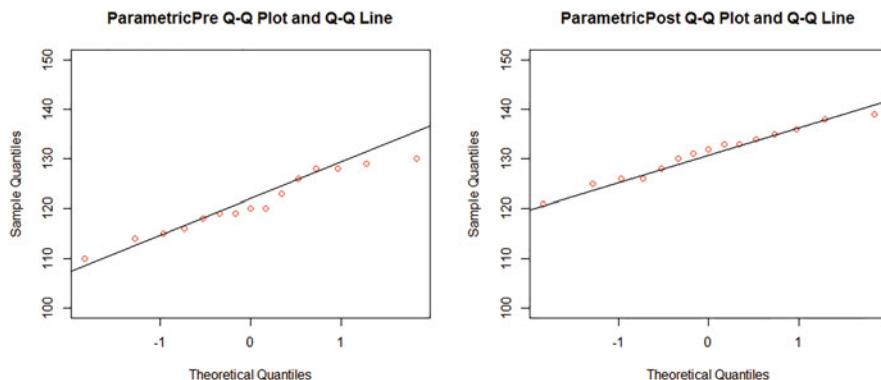


Figure 4.7: Comparison of parametric pre and parametric post datapoints

R Input

```
t.test(ParametricExample.df$ParametricPre,
       ParametricExample.df$ParametricPost, na.rm=TRUE,
       alternative="two.sided", paired=TRUE, var.equal=TRUE)
# With all of the preparations completed, conduct a Student's
# t-Test for Matched Pairs against the data.
```

Is there sufficient evidence to state that there is, or is not, a statistically significant difference ($p \leq 0.05$) between ParametricPre means and ParametricPost means, as evidenced by results of Student's t-Test for Matched Pairs and the associated `t.test()` function? Write the outcome in a descriptive manner so that someone with only a basic understanding of biostatistics would understand the trends for each variable and the overall summary. Use this example to make practice datasets and continue along with the process of practicing with Student's t-Test for Matched Pairs.

Data That Do Not Exhibit Normal Distribution Patterns

R Input

```
NonParametricPre <- round(runif(15, 100, 135))
# Use the runif() function to create NonParametricPre, an
# object variable with N = 15 and values that range from
# 100 to 135. Wrap the round() function around the output
# so that the datapoints are rounded as whole numbers.
```

```
NonParametricPost <- round(runif(15, 115, 155))
# Use the runif() function to create NonParametricPost, an
# object variable with N = 15 and values that range from
```

```
# 115 to 155. Wrap the round() function around the output
# so that the datapoints are rounded as whole numbers.
```

The runif() function is used to create a set of numbers that are not expected to show normal distribution. The round() function with no arguments is used to generate whole numbers (Fig. 4.7).

R Input

```
NonParametricExample.df <- data.frame(cbind(NonParametricPre,
NonParametricPost))
# Use the cbind() function to join NonParametricPre and
# NonParametricPost into NonParametricExample.df. Use the
# data.frame() function to be sure that the new object is
# indeed a dataframe.

attach(NonParametricExample.df)
str(NonParametricExample.df)
NonParametricExample.df

pastecs::stat.desc(NonParametricExample.df,
basic=TRUE, # Show basic descriptive statistics.
desc=TRUE, # Show more descriptive statistics.
norm=TRUE, # Show normal distribution statistics.
p=0.95) # Probability for the mean's Confidence Interval.

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
qqnorm(NonParametricExample.df$NonParametricPre, col=c("red"),
main="NonParametricPre Q-Q Plot and Q-Q Line",
xlim=c(-4, 4), ylim=c(100,160))
qqline(NonParametricExample.df$NonParametricPre)
qqnorm(NonParametricExample.df$NonParametricPost, col=c("red"),
main="NonParametricPost Q-Q Plot and Q-Q Line",
xlim=c(-4, 4), ylim=c(100,160))
qqline(NonParametricExample.df$NonParametricPost)
# The QQ-Plots visually reinforce the normtest.p values and
# provide further assurance that there are concerns about
# normal distribution. View QQ-Plots and how some would say
# that datapoints are serpentine along the QQ line.
```

With all of the preparations completed, including the QQ plots, normal distribution is questioned and the data for NonParametricExample.df do not support

analyses dependent on parametric assumptions. With this concern, it may be best to use the `wilcox.test()` function to address any statistical comparison of the two variables and in turn make judgment on statistical significance.

R Input

```
wilcox.test(NonParametricExample.df$NonParametricPre,
            NonParametricExample.df$NonParametricPost,
            alternative=c("two.sided"), paired=TRUE, exact=TRUE,
            correct=TRUE, conf.int=TRUE, conf.level=0.95)
# Notice how there was no grouping factor. Instead,
# NonParametricPre and NonParametricPost were the two
# measured variables for this analysis.
```

Is there sufficient evidence to state that there is, or is not, a statistically significant difference ($p \leq 0.05$) between `NonParametricPre` and `NonParametricPost` values, as evidenced by results of the Wilcoxon Matched Pairs Signed Ranks Test and the associated `wilcox.test()` function? Write the outcome in a descriptive manner so that someone with only a basic understanding of biostatistics would understand the trends for each variable and the overall summary. Use this example to make practice datasets and continue along with the process of practicing with the Wilcoxon Matched Pairs Signed Ranks Test, the nonparametric proxy test for when it may not be appropriate to use Student's t-Test for Matched Pairs.

4.14 Prepare to Exit, Save, and Later Retrieve This R Session

R Input

```
getwd()          # Identify the current working directory.
ls()            # List all objects in the working
                # directory.
ls.str()        # List all objects, with finite detail.
list.files()    # List files at the PC directory.

save.image("R_Lesson_tTestMatchedPairs.rdata")

getwd()          # Identify the current working directory.
ls()            # List all objects in the working
                # directory.
ls.str()        # List all objects, with finite detail.
```

```
list.files()      # List files at the PC directory.  
  
alarm()          # Alarm, notice of upcoming action.  
q()              # Quit this session.  
                 # Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: File -> Load Workspace. Otherwise, use the load() function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use the .R script file (typically saved as a .txt ASCII-type file) and recreate the analyses and graphics, provided the data files remain available.

4.15 External Data and/or Data Resources Used in This Lesson

The file `WeightPrePostSupplementUnstacked.csv`, the only external file directly imported into this lesson, is available at the publisher's Web site associated with this text. Use this file to practice and replicate the outcomes used in this lesson.

All other data in this lesson were enumerated (e.g., user-created) directly in the R session, using a variety of R-based functions. The user-created data in this lesson are **not** included at the publisher's Web site associated with this text and they are instead easily created in R by following the syntax presented in this lesson.



Chapter 5

Oneway Analysis of Variance (ANOVA)

Abstract

The purpose of this lesson is to demonstrate how the R environment supports Oneway Analysis of Variance (Oneway ANOVA), the empirical test used to determine if there are statistically significant differences between the means of three or more samples of continuous data (e.g., is there a statistically significant ($p \leq 0.05$) difference between the different race-ethnicity groups regarding Systolic Blood Pressure? Is there a statistically significant ($p \leq 0.05$) difference between Guernsey cows, Jersey cows, and Holstein cows regarding the percentage fat content of their milk? Is there a statistically significant ($p \leq 0.05$) difference between Orchard grass (*Dactylis glomerata*), Tall Fescue (*Festuca arundinacea*), and Timothy grass (*Phleum pratense*) regarding crude protein content?) Oneway ANOVA is a common inferential test that is typically used when three or more groups are examined against one continuous measured variable and mastery of this test is essential for those who regularly conduct research in the biological sciences.

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_5) contains supplementary material, which is available to authorized users.

Keywords: ANOVA, Analysis of variance, Balanced design, Duncan's multiple range test, Fisher (Ronald), Least significant difference (LSD), Mean comparison technique, Oneway ANOVA, Scheffé's mean comparison test, Student-Newman-Keuls (SNK), Tukey's honestly significant difference (Tukey HSD), Unbalanced design

5.1 Background

5.1.1 Description of the Data

There is one dataset used in this lesson, but as the lesson continues the data are organized and used in two different ways:

- The data for this lesson are first organized in wide (e.g., unstacked) format, where each column represents the data for one variable. Data are often organized in wide format as row-by-column tables, especially if the data are viewed by the public or researchers with limited experience, because it is easy for those without adequate training to have some sense of the data and the research project since the rows represent subjects and the columns represent variables.
- The data for this lesson are later organized in long (e.g., stacked) format, to follow along with expected norms for data organization and in turn better accommodate R-based functions that call for data in long format. Not only R, but nearly all other statistical analysis software are best (or at least most frequently) accommodated when data are in long format.

The data, whether in wide format or long format, are the same data. The only difference in the data, ostensibly, is their organizational structure—wide v long (e.g., unstacked v stacked):

- The dataset in wide format was self-generated, using R-based tools. The dataset is created using the `stats::rnorm()` function and the data follow a generally normal distribution pattern.
- As this lesson continues, the `tidy::gather()` function is applied against the dataset in wide format and as a result of the correct use of this function the data were put into long (e.g., stacked) format.

It cannot be overly-emphasized that the data are the same, whether in wide format or long format. It is certainly possible to accommodate data in wide format, but the norm behavior of data scientists is to work with data in long format if at all possible and that is why the structure for the data was transformed in this lesson.

In an effort to provide consistency and outcomes that allow reproducible results, the `base::set.seed()` function is used at the start of this lesson. Self-generated datasets, such as the dataset used in this lesson, if prepared correctly using the `base::set.seed()` function, will then allow for equivalent replication of the data if generated again and/or if generated by other researchers. This approach for the way practice datasets are self-generated is common when different methods of data exploration are tried, where data with known values are first used to conceptualize methods on how to achieve desired outcomes for later on when real data are used and outcomes are not immediately certain.

5.1.2 Null Hypothesis

There is one Null Hypothesis associated with this part of the lesson, with $p \leq 0.05$ the criterion level of significance. The data used in this demonstration of Oneway ANOVA are focused on the differences in Systolic Blood Pressure between three groups of individuals, with group membership based on lifestyle and subsequent exercise activity:¹

- **Sedentary:** Those individuals who engage in little to no purposeful exercise.
- **Moderate:** Those individuals who engage in only a moderate amount of exercise.
- **Active:** Those individuals who engage in active exercise programs.

Null Hypothesis (H_0): There is no statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure between subjects based on Lifestyle (e.g., typical exercise patterns such as Sedentary, Moderate, Active). Due to the way the dataset was created, it is expected that data should ostensibly show normal distribution.

5.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

Instead of importing a .csv file, the data in this lesson are self-generated, using R-based tools. After the `base::set.seed()` function is used, R-based functions are used to self-generate the dataset and put the data into proper order.

¹Oneway Analysis of Variance was developed in large part by Ronald Fisher in support of agricultural research, approximately 100 years ago. Oneway ANOVA is now a frequently used test in the physical sciences and social sciences.

R Input

```
#####
# Housekeeping                                     Use for All Analyses #
#####
date()          # Current system time and date.
Sys.time()       # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls(all.names=TRUE) # List all objects in the working
                   # directory, including hidden files.
rm(list=ls())      # CAUTION: Remove all files in the
                   # working directory. If this
                   # action is not desired, use rm()
                   # one-by-one to remove the objects
                   # that are not needed.
ls.str()         # List all objects, with finite detail.
getwd()           # Identify the current working directory.
setwd("F:/R_BiostatisticsIntroduction")          # Set to a new working directory.
                                                 # Note the single forward slash and double
                                                 # quotes.
                                                 # This new directory should be the directory
                                                 # where the data file is located, otherwise
                                                 # the data file will not be found.
getwd()           # Confirm the working directory.
list.files()        # List files at the PC directory.
.libPaths()        # Library pathname.
.Library           # Library pathname.
sessionInfo()      # R version, locale, and packages.
search()            # Attached packages and objects.
searchpaths()       # Attached packages and objects.
#####
```

The purpose of the collection of R functions in the beginning Housekeeping section is to be sure that the working environment is immediately organized as desired. This syntax, with only minor if any change, is used throughout this text.

R Input

```
# Set the seed
base::set.seed(8)
# The base::set.seed() function is commonly used
# immediately before any attempt to generate random
```

```
# numbers. This process supports consistency and  
# replication when self-creating data.
```

Distribution that Exhibits Normality

Create a dataframe that will eventually consist of data in wide format related to Systolic Blood Pressure for subjects based on three Lifestyle breakouts (e.g., typical exercise patterns such as Sedentary, Moderate, Active). Using R, there are many ways to create this eventual dataframe.

R Input

```
ExerciseWide.df <- data.frame(  
  Sedentary <- round(rnorm(50, mean=136, sd=08)),  
  Moderate  <- round(rnorm(50, mean=134, sd=06)),  
  Active    <- round(rnorm(50, mean=122, sd=04))  
)  
# Create a dataframe called ExerciseWide.df consisting of  
# three separate object variables, with the data in three  
# separate columns. The data consist of Systolic Blood  
# Pressure (Systolic Blood Pressure) measurements for:  
# 50 subjects with a sedentary lifestyle.  
# 50 subjects with a lifestyle that has only a moderate  
# amount of exercise.  
# 50 subjects with an active exercise lifestyle.  
# Note how the round() function was wrapped around the  
# rnorm() function to display the data as whole numbers.  
# Later, observe how each object variable is unique, with a  
# different mean and standard deviation.  
  
colnames(ExerciseWide.df) <-  
  c("Sedentary", "Moderate", "Active")  
  
str(ExerciseWide.df)
```

R Output

```
'data.frame': 50 obs. of 3 variables:  
 $ Sedentary: num 135 143 132 132 142 135 135 127 112 ...  
 $ Moderate : num 140 132 138 133 140 137 129 130 125 139 ...  
 $ Active   : num 123 114 115 115 122 120 124 121 123 124 ...
```

R Input

```
names(ExerciseWide.df)
```

R Output

```
[1] "Sedentary" "Moderate" "Active"
```

Use different R-based functions to put the data into proper format and then obtain a first view of the data. However, these actions against the data in wide format will be minimal since the data are best subjected to statistical analysis and graphical presentation when put into long format.

R Input

```
attach(ExerciseWide.df)      # Attach the dataset
head(ExerciseWide.df, 5)     # Look at the first 5 rows of data
tail(ExerciseWide.df, 5)     # Look at the last 5 rows of data
summary(ExerciseWide.df)    # Summary descriptive statistics
```

R Output

[Selected output is not shown, to save space.]

Sedentary	Moderate	Active
Min. :112	Min. :116	Min. :112
1st Qu.:131	1st Qu.:129	1st Qu.:117
Median :135	Median :134	Median :122
Mean :136	Mean :133	Mean :122
3rd Qu.:141	3rd Qu.:138	3rd Qu.:124
Max. :152	Max. :148	Max. :132

5.3 Organize the Data and Display the Code Book

The dataframe of interest for the first part of this lesson was created by organizing data into wide format. The Code Book details the nature of the data.

R Input

```
#####
# Code Book for ExerciseWide.df          #
#####
```

```
#  
# Sedentary ..... Numeric #  
#   Systolic Blood Pressure values may range #  
#       from approximately 100 to 160 #  
#  
# Moderate ..... Numeric #  
#   Systolic Blood Pressure values may range #  
#       from approximately 100 to 160 #  
#  
# Active ..... Numeric #  
#   Systolic Blood Pressure values may range #  
#       from approximately 100 to 160 #  
#  
#####
```

Although it may seem redundant, one last check of the data in `ExerciseWide.df` is prudent to be sure that the data are as expected.

R Input

```
str(ExerciseWide.df)      # Identify structure  
head(ExerciseWide.df, n=5) # Show the head, first 5 cases
```

R Output

	Sedentary	Moderate	Active
1	135	140	123
2	143	132	114
3	132	138	115
4	132	133	115
5	142	140	122

R Input

```
tail(ExerciseWide.df, n=5)      # Show the tail, last 5 cases
```

R Output

	Sedentary	Moderate	Active
46	120	139	119
47	131	136	124
48	138	134	125
49	138	130	118
50	131	134	120

R Input

```
summary(ExerciseWide.df)      # Summary statistics
mean(ExerciseWide.df$Sedentary); sd(ExerciseWide.df$Sedentary)
# The ; character in this context allows the deployment of
# two functions on the same line.
```

R Output

```
[1] 135.52
[1] 8.14722
```

R Input

```
mean(ExerciseWide.df$Moderate); sd(ExerciseWide.df$Moderate)
```

R Output

```
[1] 133.2
[1] 6.88091
```

R Input

```
mean(ExerciseWide.df$Active); sd(ExerciseWide.df$Active)
```

R Output

```
[1] 121.56  
[1] 4.80289
```

For the `ExerciseWide.df$` dataframe, using the `mean()` and `sd()` functions as a first attempt at Quality Assurance, it is seen early on that the values for Sedentary, Moderate, and Active seem to follow along with general expectations, where physical exercise tends to have an impact on Systolic Blood Pressure—but for now this is only an unfounded general observation absent any empirical testing for statistically significant differences. Rounded to whole numbers, observe the mean and sd for each of the three Systolic Blood Pressure variables:

```
ExerciseWide.df$Sedentary .... Mean = 136 and SD = 8  
ExerciseWide.df$Moderate .... Mean = 133 and SD = 7  
ExerciseWide.df$Active ..... Mean = 122 and SD = 5
```

Going beyond the `mean()`, `sd()`, and `summary()` functions, use `epiDisplay::summ()` to gain a sense of the data. Later on in this lesson additional efforts will be used to learn more about the data, distribution patterns, and statistically significant differences (if any) in means (Fig. 5.1).

R Input

```
install.packages("epiDisplay")  
library(epiDisplay)           # Load the epiDisplay package.  
help(package=epiDisplay)      # Show the information page.  
sessionInfo()                # Confirm all attached packages.  
  
par(ask=TRUE)                # Pause  
par(mfrow=c(1,3))            # 3 figures - 1 row by 3 column grid  
epiDisplay::summ(ExerciseWide.df$Sedentary, xlim=c(100,160))  
epiDisplay::summ(ExerciseWide.df$Moderate, xlim=c(100,160))  
epiDisplay::summ(ExerciseWide.df$Active, xlim=c(100,160))  
# Along with the comparative side-by-side figure, review  
# the descriptive statistics printed to the screen.
```

The data have been self-created and placed in the `ExerciseWide.df` dataframe. The `epiDisplay::summ()` function was used to provide an initial graphic and to provide a first view that data are as desired. A few R-based functions are used to be sure that the data are in the correct location, as a possibly redundant but still useful quality assurance action.

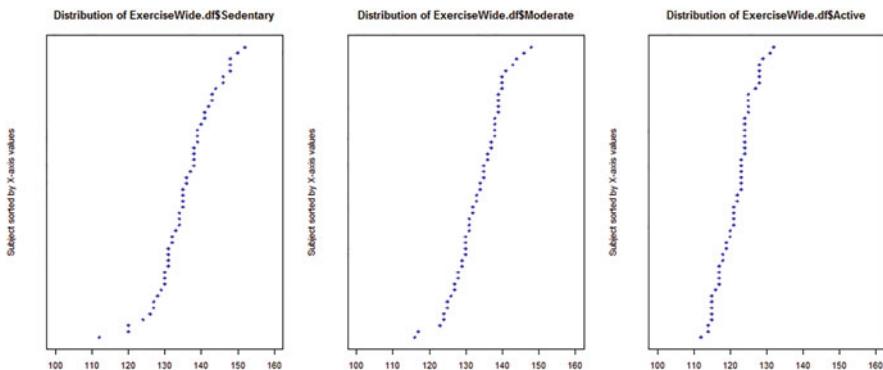


Figure 5.1: Distribution of systolic blood pressure by lifestyle breakouts—1

R Input

```
getwd()      # Identify the working directory
```

R Output

```
[1] "F:/R_BiostatisticsIntroduction"
```

R Input

```
ls()      # List objects
```

R Output

```
[1] "Active"   "ExerciseWide.df"   "Moderate"   "Sedentary"
```

Given that `ExerciseWide.df` consists of 50 subjects for each Lifestyle breakout (e.g., typical exercise patterns such as Sedentary, Moderate, Active) and is far too large to show on the screen at one time, the `head()` function and `tail()` function were used to examine the data. It is not practical to show the entire dataset which is an issue that only magnifies as datasets increase in size. There are many more R-based functions available for further diagnostics about singular object variables and dataframes with multiple object variables, but these functions should be more than sufficient to confirm that the data are in good form.

5.4 Conduct a Visual Data Check Using Graphics (e.g., Figures)

For immediate use as a data check, produce throw-away graphics and avoid too many embellishments. This approach, using graphics as a quality assurance measure, provides a convenient way to review the data and have confidence that the data are acceptable for later analyses.

For each numeric object variable, it is a good practice to examine the data using a histogram, a density plot, a boxplot, and a Q-Q plot. If desired or if there are any concerns about the data, it is also useful to use a dot plot and a violin plot to graphically examine numeric data. Each type of graphic provides a different perspective of the data, regardless of whether the figures are ever shared with others.

If possible, meaningful comparisons between and among numerical data are gained using side-by-side figures. When preparing comparative figures, such as graphical presentation of members from one group to members from another group, be sure to use a common scale. Note below how the `xlim()` argument and the `ylim()` arguments were used to achieve this aim. Saying that, review outcomes from the `summary()` function to observe the minimum and maximum value for each variable and then use this information, perhaps after a few tries, to decide on the best scale for each axis. Some trial-and-error may be needed to finally select the best scale for common viewing of all comparative figures. Note how very few embellishments were used in these initial figures. The focus here is only on the desire to review the data in wide format, the original format for these data (Figs. 5.2, 5.3, 5.4, 5.5, and 5.6).

R Input

```
# Histogram

par(ask=TRUE)      # Pause
par(mfrow=c(1,3))  # 3 figures - 1 row by 3 column grid
hist(ExerciseWide.df$Sedentary,
     main="Systolic Blood Pressure - Sedentary",
     col="red",        # Add color
     breaks=15,        # Increase granularity of histogram
     font.lab=2,        # Bold labels
     xlab="Systolic Blood Pressure",      # X label
     xlim=c(100,160), # X axis scale
     ylim=c(0,8))     # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
```

```
hist(ExerciseWide.df$Moderate,
  main="Systolic Blood Pressure - Moderate",
  col="red",          # Add color
  breaks=15,          # Increase granularity of histogram
  font.lab=2,          # Bold labels
  xlab="Systolic Blood Pressure",      # X label
  xlim=c(100,160),    # X axis scale
  ylim=c(0,8))        # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold

hist(ExerciseWide.df$Active,
  main="Systolic Blood Pressure - Active",
  col="red",          # Add color
  breaks=15,          # Increase granularity of histogram
  font.lab=2,          # Bold labels
  xlab="Systolic Blood Pressure",      # X label
  xlim=c(100,160),    # X axis scale
  ylim=c(0,8))        # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
```

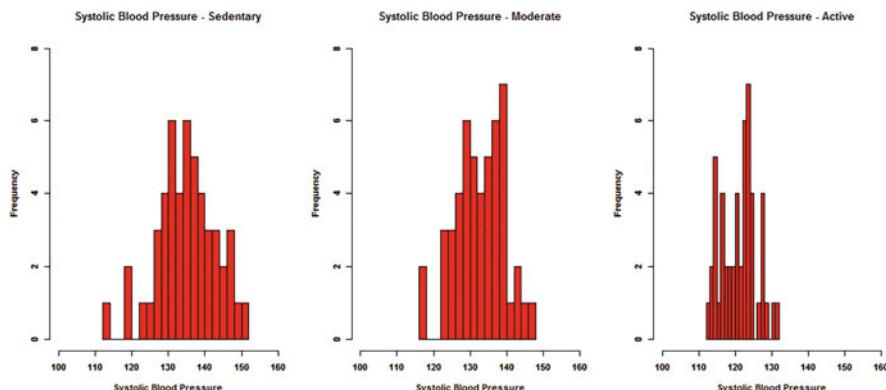


Figure 5.2: Distribution of systolic blood pressure by lifestyle breakouts—2

R Input

```
# Density Plot

par(ask=TRUE)      # Pause
par(mfrow=c(1,3)) # 3 figures - 1 row by 3 column grid
plot(density(ExerciseWide.df$Sedentary),
  main="Systolic Blood Pressure - Sedentary",
```

```

col="red",                      # Add color
lwd=3,                          # Thick line
font.lab=2,                     # Bold labels
xlab="Density",                 # X axis label
xlim=c(100,160),               # X axis scale
ylim=c(0,0.08))                # Y axis scale
axis(side=1, font=2)            # X axis bold
axis(side=2, font=2)            # Y axis bold
plot(density(ExerciseWide.df$Moderate),
main="Systolic Blood Pressure - Moderate",
col="red",                      # Add color
lwd=3,                          # Thick line
font.lab=2,                     # Bold labels
xlab="Density",                 # X axis label
xlim=c(100,160),               # X axis scale
ylim=c(0,0.08))                # Y axis scale
axis(side=1, font=2)            # X axis bold
axis(side=2, font=2)            # Y axis bold
plot(density(ExerciseWide.df$Active),
main="Systolic Blood Pressure - Active",
col="red",                      # Add color
lwd=3,                          # Thick line
font.lab=2,                     # Bold labels
xlab="Density",                 # X axis label
xlim=c(100,160),               # X axis scale
ylim=c(0,0.08))                # Y axis scale
axis(side=1, font=2)            # X axis bold
axis(side=2, font=2)            # Y axis bold

```

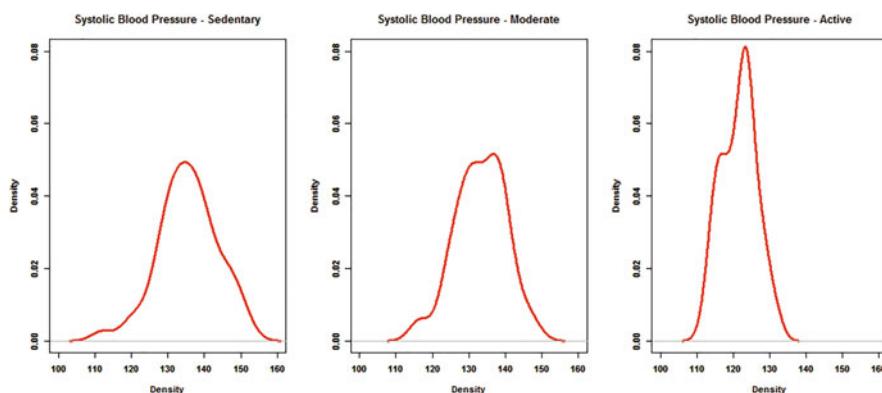


Figure 5.3: Distribution of systolic blood pressure by lifestyle breakouts—3

R Input

```
# Boxplot

par(ask=TRUE)      # Pause
par(mfrow=c(1,3))  # 3 figures - 1 row by 3 column grid
boxplot(ExerciseWide.df$Sedentary,
        main="Systolic Blood Pressure - Sedentary",
        xlab="Boxplot",           # X axis label
        ylab="Systolic Blood Pressure",      # Y axis label
        cex.axis=1.15,            # Axis size
        cex.lab=1.15,             # Label size
        col="red",                # Box color
        lwd=1,                   # Line thickness
        font.lab=2,               # Bold labels
        font=2,                   # Bold font
        ylim=c(100,160))         # Y axis scale

boxplot(ExerciseWide.df$Moderate,
        main="Systolic Blood Pressure - Moderate",
        xlab="Boxplot",           # X axis label
        ylab="Systolic Blood Pressure",      # Y axis label
        cex.axis=1.15,            # Axis size
        cex.lab=1.15,             # Label size
        col="red",                # Box color
        lwd=1,                   # Line thickness
        font.lab=2,               # Bold labels
        font=2,                   # Bold font
        ylim=c(100,160))         # Y axis scale

boxplot(ExerciseWide.df$Active,
        main="Systolic Blood Pressure - Active",
        xlab="Boxplot",           # X axis label
        ylab="Systolic Blood Pressure",      # Y axis label
        cex.axis=1.15,            # Axis size
        cex.lab=1.15,             # Label size
        col="red",                # Box color
        lwd=1,                   # Line thickness
        font.lab=2,               # Bold labels
        font=2,                   # Bold font
        ylim=c(100,160))         # Y axis scale
```

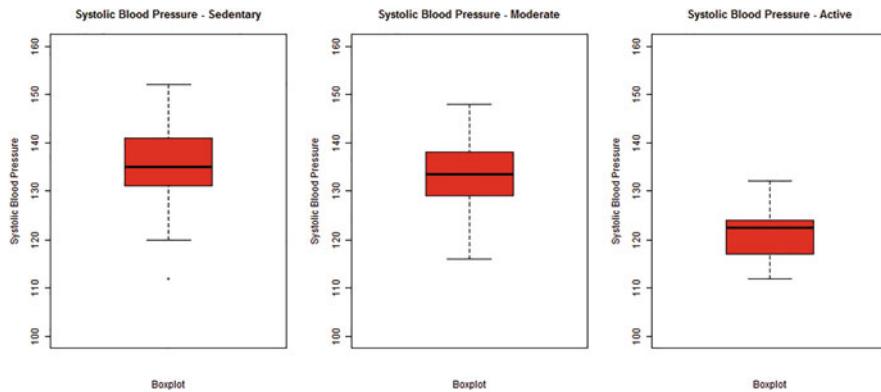


Figure 5.4: Distribution of systolic blood pressure by lifestyle breakouts—4

R Input

```
# Q-Q Plot

par(ask=TRUE)      # Pause
par(mfrow=c(1,3)) # 3 figures - 1 row by 3 column grid
qqnorm(ExerciseWide.df$Sedentary,
       main="Systolic Blood Pressure - Sedentary",
       col="blue", xlim=c(-4,4), ylim=c(100,160), font.axis=2,
       font.lab=2)
qqline(ExerciseWide.df$Sedentary, col="red", lwd=4, lty=2)
qqnorm(ExerciseWide.df$Moderate,
       main="Systolic Blood Pressure - Moderate",
       col="blue", xlim=c(-4,4), ylim=c(100,160), font.axis=2,
       font.lab=2)
qqline(ExerciseWide.df$Moderate, col="red", lwd=4, lty=2)
qqnorm(ExerciseWide.df$Active,
       main="Systolic Blood Pressure - Active",
       col="blue", xlim=c(-4,4), ylim=c(100,160), font.axis=2,
       font.lab=2)
qqline(ExerciseWide.df$Active, col="red", lwd=4, lty=2)
```

R Input

```
# Dotchart

par(ask=TRUE)      # Pause
par(mfrow=c(1,3)) # 3 figures - 1 row by 3 column grid
dotchart(ExerciseWide.df$Sedentary,
```

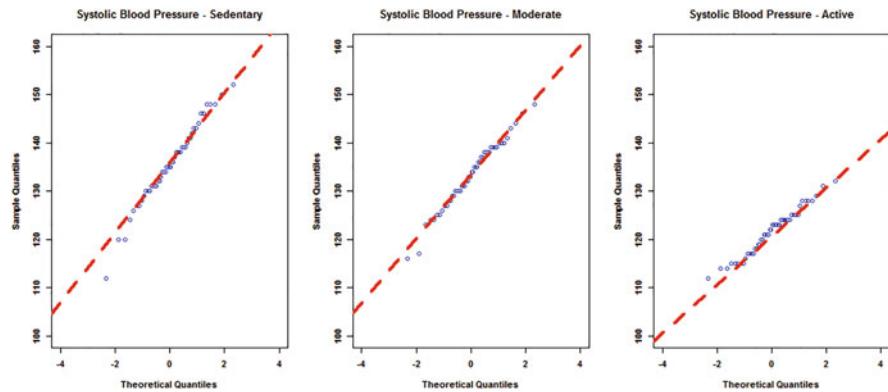


Figure 5.5: Distribution of systolic blood pressure by lifestyle breakouts—5

```

main="Systolic Blood Pressure - Sedentary",
xlab="Systolic Blood Pressure",           # X axis label
cex.axis=1.15,                          # Axis size
cex.lab=1.15,                           # Label size
pch=16,                                 # Dot symbol
pt.cex=1.15,                           # Dot size
col="red",                               # Dot color
font.lab=2,                             # Bold labels
font=2,                                 # Bold font
xlim=c(100,160))                      # X axis scale
dotchart(ExerciseWide.df$Moderate,
main="Systolic Blood Pressure - Moderate",
xlab="Systolic Blood Pressure",           # X axis label
cex.axis=1.15,                          # Axis size
cex.lab=1.15,                           # Label size
pch=16,                                 # Dot symbol
pt.cex=1.15,                           # Dot size
col="red",                               # Dot color
font.lab=2,                             # Bold labels
font=2,                                 # Bold font
xlim=c(100,160))                      # X axis scale
dotchart(ExerciseWide.df$Active,
main="Systolic Blood Pressure - Active",
xlab="Systolic Blood Pressure",           # X axis label
cex.axis=1.15,                          # Axis size
cex.lab=1.15,                           # Label size
pch=16,                                 # Dot symbol
pt.cex=1.15,                           # Dot size
col="red",                               # Dot color

```

```
font.lab=2,          # Bold labels
font=2,              # Bold font
xlim=c(100,160))    # X axis scale
```

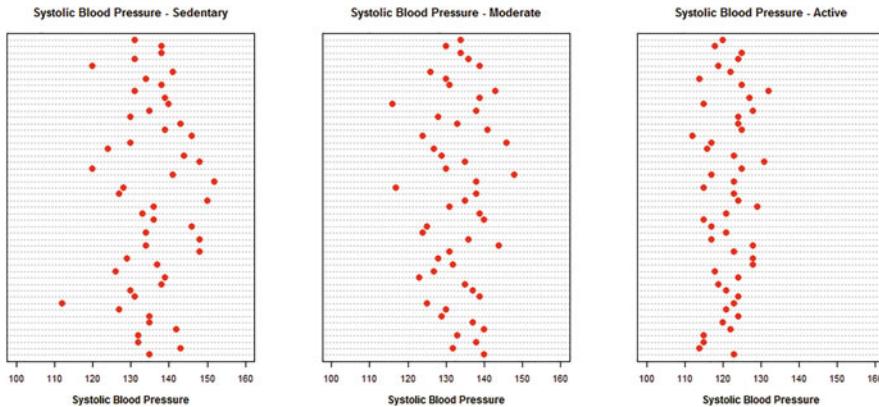


Figure 5.6: Distribution of systolic blood pressure by lifestyle breakouts—6

R Input

```
# Violin Plot

install.packages("vioplot")
library(vioplot)           # Load the vioplot package.
help(package=vioplot)       # Show the information page.
sessionInfo()               # Confirm all attached packages.

par(ask=TRUE)               # Pause
par(mfrow=c(1,3))           # 3 figures - 1 row by 3 column grid
vioplot::vioplot(ExerciseWide.df$Sedentary,
  names=c("Systolic Blood Pressure - Sedentary"),
  drawRect=TRUE,              # Add the box
  col="cyan",                 # Color of violin
  ylim=c(100,160),            # Y axis scale
  horizontal=FALSE,           # Horizontal violin plot
  lty=2)                      # Dashed line
  title("Systolic Blood Pressure - Sedentary")
vioplot::vioplot(ExerciseWide.df$Moderate,
  names=c("Systolic Blood Pressure - Moderate"),
  drawRect=TRUE,              # Add the box
  col="cyan",                 # Color of violin
  ylim=c(100,160),            # Y axis scale
  horizontal=FALSE,           # Horizontal violin plot
```

```

lty=2) # Dashed line
title("Systolic Blood Pressure - Moderate")
vioplot::vioplot(ExerciseWide.df$Active,
  names=c("Systolic Blood Pressure - Active"),
  drawRect=TRUE, # Add the box
  col="cyan", # Color of violin
  ylim=c(100,160), # Y axis scale
  horizontal=FALSE, # Horizontal violin plot
  lty=2) # Dashed line
title("Systolic Blood Pressure - Active")

```

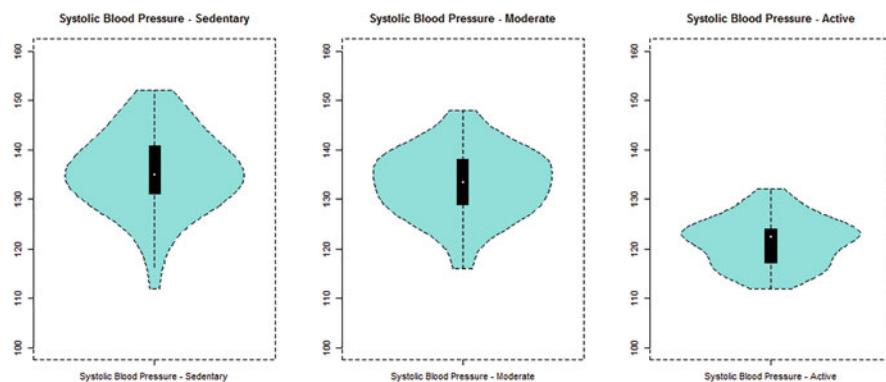


Figure 5.7: Distribution of systolic blood pressure by lifestyle breakouts—7

These figures, although fairly simple in presentation, should provide a good sense of the data and general trends. Far more embellished figures are presented later in this lesson, offering finer detail in support of statistical analyses (Fig. 5.7).

5.5 Descriptive Statistics for Initial Analysis of the Data

Going beyond the few descriptive statistics that have been shown, such as use of the `summary()`, `mean()`, `sd()`, and `epiDisplay::summ()` functions, the first task is to obtain a fairly good understanding of the data by using the most common functions for descriptive statistics and measures of central tendency. Prior lessons, including materials in the addenda, have provided extensive documentation on the many R-based functions used to generate descriptive statistics and measures of central tendency.

In the interest of brevity, only a few additional R functions will be used to offer a view of descriptive statistics, at this point in the lesson with the data in wide format. Of course, the full set of R-based functions demonstrated through these lessons is always available if needed.

A function that generates an attractive display of breakout descriptive statistics is the RcmdrMisc::numSummary() function. Look at the many ways this function can be used to produce output that can easily copied and put into a word-processed document, for data in wide format and later in this lesson when data are in long format.

R Input

```
install.packages("RcmdrMisc", dependencies=TRUE)
library(RcmdrMisc)           # Load the RcmdrMisc package.
help(package=RcmdrMisc)      # Show the information page.
sessionInfo()                # Confirm all attached packages.

RcmdrMisc::numSummary(ExerciseWide.df$Sedentary)
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n
135.52	8.14722	9.75	112	131	135	140.75	152	50

R Input

```
RcmdrMisc::numSummary(ExerciseWide.df$Moderate)
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n
133.2	6.88091	9	116	129	133.5	138	148	50

R Input

```
RcmdrMisc::numSummary(ExerciseWide.df$Active)
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n
121.56	4.80289	6.75	112	117.25	122.5	124	132	50

Do not overlook the descriptive statistics that can be generated in association with a boxplot, the boxplot.stats() function, and the fivenum() function. These functions generate statistics that improve upon an understanding of not only

measures of central tendency but also distribution of the entire set of data for each object variable.

R Input

```
boxplot.stats(ExerciseWide.df$Sedentary) # Summary statistics
```

R Output

```
stats  
[1] 120 131 135 141 152  
n  
[1] 50  
conf  
[1] 132.766 137.234  
out  
[1] 112
```

R Input

```
boxplot.stats(ExerciseWide.df$Moderate) # Summary statistics
```

R Output

```
stats  
[1] 116.0 129.0 133.5 138.0 148.0  
n  
[1] 50  
conf  
[1] 131.489 135.511  
$out  
numeric(0)
```

R Input

```
boxplot.stats(ExerciseWide.df$Active) # Summary statistics
```

R Output

```
stats  
[1] 112.0 117.0 122.5 124.0 132.0  
n
```

```
[1] 50  
conf  
[1] 120.936 124.064  
out  
numeric(0)
```

In order, the boxplot statistics are: Lower-Whisker, Lower-Hinge, Median, Upper-Hinge, and Upper-Whisker. The boxplot.stats() also identifies N and Outliers (if any). The boxplot.stats() function is included in the grDevices package, which is available when R is first downloaded.

R Input

```
fivenum(ExerciseWide.df$Sedentary) # Summary statistics
```

R Output

```
[1] 112 131 135 141 152
```

R Input

```
fivenum(ExerciseWide.df$Moderate) # Summary statistics
```

R Output

```
[1] 116.0 129.0 133.5 138.0 148.0
```

R Input

```
fivenum(ExerciseWide.df$Active) # Summary statistics
```

R Output

```
[1] 112.0 117.0 122.5 124.0 132.0
```

The fivenum() function is included in the stats package and generates Tukey's Five-Number Summary: Minimum, Lower-Hinge, Median, Upper-Hinge, and Maximum. It adds another dimension to the use of descriptive statistics and how these statistics are used to fully understand the nature of data.

5.6 Quality Assurance, Data Distribution, and Tests for Normality

Quality Assurance (QA) should be viewed as a continuous process. Quality assurance needs to be inclusive throughout all research activities, from pre-planning to ending actions. To that end, a great deal of attention has gone into the production of figures that offer a graphical sense of the data. Descriptive statistics have been explored from many perspectives. Summative descriptive statistics were provided for Systolic Blood Pressure, for all three lifestyle groups of interest in this lesson.

There is a reasonable degree of assurance that the data represent expected Systolic Blood Pressure values for the three previously identified lifestyles. With this initial assurance of data integrity, it is now necessary to focus on data distribution and investigations into normality. From among the many tests that focus on normality, the Shapiro test will be demonstrated in this lesson, using the `stats::shapiro.test()` function. Recall that the `stats` package is one of the many packages obtained when R is first downloaded and that it is not an external package. Accordingly, the syntax `shapiro.test()` will be used in this lesson, but it would have been just as accurate to use `stats::shapiro.test()` if there were a desire to be more formal.²

It is common to focus on either a p-value of either $p \leq 0.05$ or $p \leq 0.01$ when considering normality tests, such as the Shapiro Test for Normality. Due to the way the Shapiro Test for Normality is worded, in the affirmative:

- Lower calculated p-values provide evidence against the null hypothesis. That is to say, if the calculated p-value is less than or equal to the criterion p-value (e.g., significance level), the rules-based decision is to reject the null hypothesis and to declare that the data do not follow a pattern of normal distribution.
- Higher calculated p-values fail to provide evidence against the null hypothesis. That is to say, if the calculated p-value is greater than the criterion p-value (e.g., significance level), the rules-based decision is to accept (e.g., fail to reject) the null hypothesis and to declare that the data seem to follow a pattern of normal distribution.

A key interest when viewing the many figures and statistics on central tendency is the concern over normal distribution for numeric object variables, or Systolic

²For nearly all statistical tests, the Null Hypothesis is worded in a negative fashion and is typically stated as *There is no statistically significant difference between A and B in terms of C*. Somewhat different, the Null Hypothesis for the Shapiro Test of Normality is instead worded in the affirmative and is typically stated as *The data follow a normal distribution*. Give attention to this different approach to the way the Shapiro Test for Normality is worded when interpreting the p-values, significance levels, and outcomes from a Shapiro Test for Normality.

Blood Pressure—Sedentary, Systolic Blood Pressure—Moderate, and Systolic Blood Pressure—Active in this lesson. Normal distribution is important in that many inferential tests are only used, appropriately, if the underlying data exhibit normal distribution. Of course, compromised decisions are often made and it is not uncommon to see the application of statistical tests with data that do not follow a normal distribution pattern even if the test calls for the use of normally-distributed data. The question then is how much deviation away from normal distribution can be accepted before a nonparametric approach to statistical analysis is necessary. This is always a judgment call, but the Shapiro Test for Normality will at least provide a sense of distribution and from that outcome, whether it is accepted that the data exhibit a reasonable pattern of normal distribution or if the distribution pattern of the data should be questioned.

R Input

```
shapiro.test(ExerciseWide.df$Sedentary)
```

R Output

```
Shapiro-Wilk normality test  
data: ExerciseWide.df$Sedentary  
p-value = 0.723
```

Applied against the numeric object variable `ExerciseWide.df$Sedentary` and using the Shapiro test, the calculated p-value equals 0.723 which is greater than the criterion p-value of 0.05. This finding provides assurance that there is normal distribution for Systolic Blood Pressure—Sedentary.

R Input

```
shapiro.test(ExerciseWide.df$Moderate)
```

R Output

```
Shapiro-Wilk normality test  
data: ExerciseWide.df$Moderate  
p-value = 0.787
```

Applied against the numeric object variable `ExerciseWide.df$Moderate` and using the Shapiro test, the calculated p-value equals 0.787 which is greater than the criterion p-value of 0.05. This finding confirms that there is normal distribution for Systolic Blood Pressure—Moderate.

R Input

```
shapiro.test(ExerciseWide.df$Active)
```

R Output

```
Shapiro-Wilk normality test  
data: ExerciseWide.df$Active  
p-value = 0.265
```

Applied against the numeric object variable `ExerciseWide.df$Active` and using the Shapiro test, the calculated p-value equals 0.265 which is greater than the criterion p-value of 0.05. This finding confirms that there is normal distribution for Systolic Blood Pressure—Active.

As a quality assurance measure it seems prudent to redundantly check results for the Shapiro Test for Normality against all three object variables, but by using a different function—deploying the `RcmdrMisc::normalityTest()` function in this case.

R Input

```
RcmdrMisc::normalityTest(ExerciseWide.df$Sedentary,  
test=c("shapiro.test"))
```

R Output

```
p-value = 0.723
```

R Input

```
RcmdrMisc::normalityTest(ExerciseWide.df$Moderate,  
test=c("shapiro.test"))
```

R Output

```
p-value = 0.787
```

R Input

```
RcmdrMisc::normalityTest(ExerciseWide.df$Active,  
test=c("shapiro.test"))
```

R Output

```
p-value = 0.265
```

Other than the possibility of slight differences due to rounding, the results obtained from the RcmdrMisc::normalityTest() function are consistent with what was obtained from use of the stats::shapiro.test() function.

The values for use with the rnorm() function were purposely selected to make this teaching dataset interesting, reinforcing the need for Quality Assurance throughout the research process. Although the rnorm() function was used to create three separate object variables (Systolic Blood Pressure—Sedentary, Systolic Blood Pressure—Moderate, and Systolic Blood Pressure—Active), attention to Quality Assurance needs to be considered throughout the entire process.

As part of the Quality Assurance process, replicate the prior QQ Plots and Density Curves, but use the ggplot2::ggplot() function to provide another approach in final presentation. Ostensibly, the QQ Plots should have the data follow along the QQ Line and the Density Curves should have some degree of parity with a normal curve.

R Input

```
install.packages("ggplot2", dependencies=TRUE)  
library(ggplot2) # Load the ggplot2 package.  
help(package=ggplot2) # Show the information page.  
sessionInfo() # Confirm all attached packages.  
  
install.packages("ggthemes", dependencies=TRUE)  
library(ggthemes) # Load the ggthemes package.  
help(package=ggthemes) # Show the information page.  
sessionInfo() # Confirm all attached packages.  
  
install.packages("ggmosaic", dependencies=TRUE)  
library(ggmosaic) # Load the ggmosaic package.  
help(package=ggmosaic) # Show the information page.  
sessionInfo() # Confirm all attached packages.  
  
install.packages("gridExtra", dependencies=TRUE)
```

```

library(gridExtra)           # Load the gridExtra package.
help(package=gridExtra)     # Show the information page.
sessionInfo()               # Confirm all attached packages.

install.packages("grid", dependencies=TRUE)
library(grid)                # Load the grid package.
help(package=grid)          # Show the information page.
sessionInfo()               # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)              # Load the scales package.
help(package=scales)        # Show the information page.
sessionInfo()               # Confirm all attached packages.

```

Use the most current version of R to obtain full functionality of selected packages and functions. However, with the `ggplot2` package, it is essential to have the most current version. Otherwise, it is possible that figures will not be produced and that error messages will instead be generated (Fig. 5.8).

R Input

```

QQSedentary <-
ggplot2::ggplot(ExerciseWide.df,
  aes(sample=Sedentary)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.05) +
  ggtitle(
  "Sedentary Lifestyle QQ-Plot and QQ-Line, Overall SBP") +
  labs(x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
  ylim(100,160) +
  theme_classic()

QQModerate <-
ggplot2::ggplot(ExerciseWide.df,
  aes(sample=Moderate)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.05) +
  ggtitle(
  "Moderate Lifestyle QQ-Plot and QQ-Line, Overall SBP") +
  labs(x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
  ylim(100,160) +
  theme_classic()

```

```
QQActive <-
ggplot2::ggplot(ExerciseWide.df,
aes(sample=Active)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.05) +
ggttitle(
"Active Lifestyle QQ-Plot and QQ-Line, Overall SBP") +
labs(x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
ylim(100,160) +
theme_classic()

DensityCurveSedentary <-
ggplot2::ggplot(ExerciseWide.df,
aes(x=Sedentary)) +
geom_density(size=1.5, col=("red")) +
ggttitle(
"Sedentary Lifestyle Density Curve, Overall SBP") +
scale_x_continuous(name="Systolic Blood Pressure",
limits=c(100,160), breaks=seq(100,160,10)) +
scale_y_continuous(name="Density", limits=c(0.0,0.09),
breaks=seq(0.0,0.09,0.025)) +
theme_classic()

DensityCurveModerate <-
ggplot2::ggplot(ExerciseWide.df,
aes(x=Moderate)) +
geom_density(size=1.5, col=("red")) +
ggttitle(
"Moderate Lifestyle Density Curve, Overall SBP") +
scale_x_continuous(name="Systolic Blood Pressure",
limits=c(100,160), breaks=seq(100,160,10)) +
scale_y_continuous(name="Density", limits=c(0.0,0.09),
breaks=seq(0.0,0.09,0.025)) +
theme_classic()

DensityCurveActive <-
ggplot2::ggplot(ExerciseWide.df,
aes(x=Active)) +
geom_density(size=1.5, col=("red")) +
ggttitle(
"Active Lifestyle Density Curve, Overall SBP") +
scale_x_continuous(name="Systolic Blood Pressure",
limits=c(100,160), breaks=seq(100,160,10)) +
```

```

scale_y_continuous(name="Density", limits=c(0.0,0.09),
breaks=seq(0.0,0.09,0.025)) +
theme_classic()

par(ask=TRUE)
gridExtra::grid.arrange(
  QQSedentary,
  QQModerate,
  QQActive,
  DensityCurveSedentary,
  DensityCurveModerate,
  DensityCurveActive, ncol=2)

```

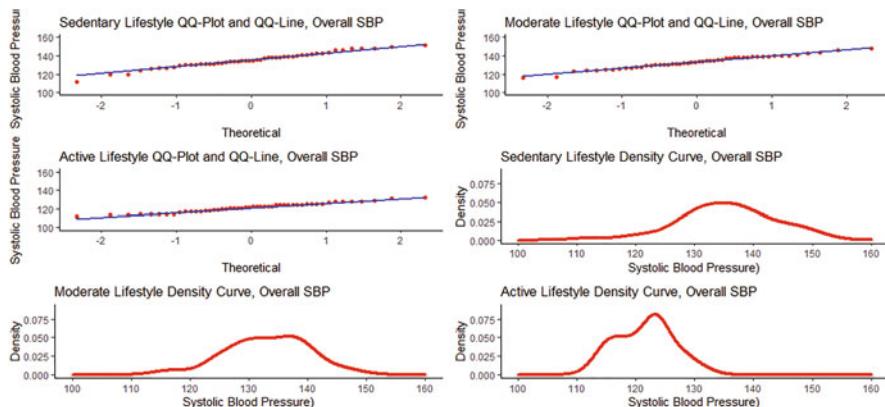


Figure 5.8: Distribution of systolic blood pressure by lifestyle breakouts—8

As expected for the QQ plot, observe how the data (represented as red dots) mostly follow along the straight line, which is an indicator of normal distribution. For the density curve, the curve is a reasonable facsimile of the normal (e.g., bell-shaped) curve, another indicator of normal distribution. But of course, the previously selected statistical test for normal distribution (e.g., Anderson–Darling Test, Shapiro Test, etc.) is still required to make authoritative judgment on normality.

5.7 Statistical Test(s)

This part of the lesson will demonstrate Exploratory Oneway Analysis of Variance (Oneway ANOVA) and from this general approach to Oneway ANOVA two detailed methods will be demonstrated: Oneway ANOVA Method 1, the use of `lm()` and `anova()` functions; and Oneway ANOVA Method 2, the use of `aov()` and `TukeyHSD()` functions. Before these different Oneway ANOVA demonstrations are shown, it is best at first to look at the way data are cur-

rently organized as wide or unstacked data and to then put the data into the more commonly used long or stacked data format.

Stacked Data

The analyses and graphics up to this point of the lesson have all been based on the use of data in wide format, using the dataset ExerciseWide.df. With data in wide format, the data for each variable are in a separate column.

However, when engaged in the research process, it is far more common for data to be prepared in stacked format. Many R-based functions that call for some type of differentiation by breakout groups require data in stacked format—only. With data in stacked format, the grouping variable data that differentiates one breakout group (e.g., Gender: Female v Male; Dairy Cattle: Ayrshire v Holstein v Jersey v Brown Swiss, etc.) from another are in one column and the data that represent a measured value are in a separate column.

Separate from the main dataset associated with this lesson, step back and consider the following sample datasets, shown only for demonstration purposes, to see differences in the way the same data are organized, comparing wide (e.g., unstacked) format data to long (e.g., stacked) format data. To remain consistent with this lesson, the data will focus on Blood Pressure, but in this brief example that is totally separate from the analyses in this lesson, Diastolic Blood Pressure is viewed for Males by Region:

```
# Diastolic Blood Pressure (DBP) in Wide (e.g., Unstacked)
# Format
```

	DBPMaleNorth	DBPMaleEast	DBPMaleSouth	DBPMaleWest
060	073	083	085	
069	068	074	076	
071	081	077	088	

```
# Diastolic Blood Pressure (DBP) in Long (e.g., Stacked)
# Format
```

Region	MaleSBP
North	060
North	069
North	071
East	073
East	068
East	081
South	083
South	074
South	077
West	085

West	076
West	088

For a small dataset it would be possible to manually copy and paste or even rekey the data that are in wide format and then put the data into long format, but either action would be unwise:

- There would be a considerable degree of time and effort on manually altering a dataset from wide format to long format and if the dataset is of any great size it would be nearly impossible to think that this action could be completed in a reasonable period of time.
- Data that are moved about by copy and paste and/or rekeying open up the possibility of misplaced and/or miskeyed data, subjecting future analyses to error.

Instead, there are a few different R-based functions that can far more easily and accurately accomplish the task of organizing data from wide format to long format. For this lesson, the `tidy::gather()` function will be used to put the data into desired format, from wide to long. That is to say, the data in `ExerciseWide.df` will be reorganized and put into a new dataset called `ExerciseLong.df`. The data in `ExerciseLong.df` will be the same data as in `ExerciseWide.df`, but the long-format data will better accommodate R functions for Oneway ANOVA.³ Use the `tidy::gather()` function to put data that are in wide format into stacked format. Recall, however, that there are other R-based tools that could achieve the same aim:⁴

- In wide format, the data for Systolic Blood Pressure are in three separate columns, one column for Systolic Blood Pressure—Sedentary, a second column for Systolic Blood Pressure—Moderate, and a third column for Systolic Blood Pressure—Active.
- In stacked format, all of the data for Systolic Blood Pressure will be in one column and there will be a new column created, with the new column representing a grouping object variable. The grouping object variable will make it possible to distinguish between the three groups of Systolic Blood Pressure data, Systolic Blood Pressure—Sedentary, Systolic Blood Pressure—Moderate, and Systolic Blood Pressure—Active.

³As a general concluding comment, it is very common to see data in stacked format. Yet, this is not always the case, thus the need for tools to transform data from one format to another format, without ever changing the values of the data.

⁴The reshape package and accompanying functions were an early R-based tool used to restructure data. Later, the reshape2 package and accompanying functions demonstrated in an earlier lesson were (and still are) used to put data into desired format. However, it cannot be ignored that the tidy package is now viewed by many as the successor to the reshape2 package and will be used to put wide data into long format. Although the `tidy::gather()` function is used in this lesson, the `tidy::pivot_longer()` function, another function in the tidy package, is also frequently used to restructure data.

R Input

```
install.packages("tidyverse", dependencies=TRUE)
library(tidyverse)                      # Load the tidyverse package.
help(package=tidyverse)                  # Show the information page.
sessionInfo()                           # Confirm all attached packages.

ExerciseLong.df <- ExerciseWide.df %>%
  tidyverse::gather(Lifestyle, SBP, Sedentary:Active)
# Lifestyle ..... key      new grouping variable
# SBP ..... value      existing measured values
# Sedentary:Active .. wide columns to gather into long format
# Note use of the %>% operator
```

In wide format, the data consisted of three columns, with 50 datapoints in each column. In long format, the data will consist of two columns, with one column representing the grouping variable (e.g., Lifestyle) and the other column representing the measured variable (e.g., SBP) values. As a sidebar comment, it is also common to see the `tidyverse::pivot_longer()` function used to restructure data. That function will not be used in this lesson, but if it were observe how the syntax would be structured, with preceding comment characters to be sure it is not used in this lesson:

```
# ExerciseLong2.df <- ExerciseWide.df %>%
#   tidyverse::pivot_longer(c(Sedentary, Moderate, Active),
#   names_to = "Lifestyle", values_to = "SBP")
```

Use standard R-based functions to confirm that the data were reformatted correctly, from wide to long:

R Input

```
attach(ExerciseLong.df)      # Attach the dataset
str(ExerciseLong.df)        # Structure
```

R Output

```
'data.frame': 150 obs. of 2 variables:
 $ Lifestyle: chr "Sedentary" "Sedentary" "Sedentary" ...
 $ SBP       : num 135 143 132 132 142 135 135 127 112 131 ...
```

R Input

```
length(ExerciseLong.df)      # Length (e.g., N)
head(ExerciseLong.df, 5)     # Look at the first 5 rows of data
tail(ExerciseLong.df, 5)     # Look at the last 5 rows of data
summary(ExerciseLong.df)    # Summary descriptive statistics
```

R Output

Lifestyle	SBP
Length:150	Min. :112
Class :character	1st Qu.:124
Mode :character	Median :130
	Mean :130
	3rd Qu.:137
	Max. :152

It may be helpful to add some type of subject number for each row (e.g., subject) in the ExerciseLong.df dataframe.

R Input

```
ExerciseLong.df$Subject <- factor(seq(from=1, to=150, by=1))

str(ExerciseLong.df)      # Structure
attach(ExerciseLong.df)   # Attach the dataset
length(ExerciseLong.df)   # Length (.g., N)
head(ExerciseLong.df, 5)  # Look at the first 5 rows of data
tail(ExerciseLong.df, 5)  # Look at the last 5 rows of data
summary(ExerciseLong.df) # Summary descriptive statistics
```

R Output

Lifestyle	SBP	Subject
Length:150	Min. :112	1 : 1
Class :character	1st Qu.:124	2 : 1
Mode :character	Median :130	3 : 1
	Mean :130	4 : 1
	3rd Qu.:137	5 : 1
	Max. :152	6 : 1
		(Other):144

R Input

```
names(ExerciseLong.df)      # Column names
```

R Output

```
[1] "Lifestyle" "SBP"      "Subject"
```

Use the `table()` function with `ExerciseLong.df$Lifestyle` to know more about the way data are organized:

R Input

```
table(ExerciseLong.df$Lifestyle)
# Summary information as a table
```

R Output

Active	Moderate	Sedentary
50	50	50

With everything in good order, use the `colnames()` function to put the dataframe into final form.

R Input

```
colnames(ExerciseLong.df) <-
  c("Lifestyle", "SBP", "SubjectID")

attach(ExerciseLong.df)      # Attach the dataset
str(ExerciseLong.df)        # Structure
length(ExerciseLong.df)     # Length (.g., N)
head(ExerciseLong.df, 5)    # Look at the first 5 rows of data
tail(ExerciseLong.df, 5)    # Look at the last 5 rows of data
summary(ExerciseLong.df)   # Summary descriptive statistics
names(ExerciseLong.df)     # Column names
```

R Output

```
[1] "Lifestyle" "SBP"      "SubjectID"
```

The desired names for the columns were forced by using the colnames() function. The column name SubjectID is more descriptive than Subject.

Calculate descriptive statistics at Lifestyle breakout levels for Systolic Blood Pressure, to confirm that the values for the data were not changed and only their format was changed, from wide to long. Then, as a Quality Assurance measure, compare Systolic Blood Pressure by Lifestyle breakout values where the data are in long format back to where the data are in wide format.

Breakout Descriptive Statistics of Systolic Blood Pressure: Long Format

R Input

```
RcmdrMisc::numSummary(
  ExerciseLong.df[,c("SBP")], groups=Lifestyle,
  statistics=c("mean", "sd", "quantiles"))
```

R Output

	mean	sd	0%	25%	50%	75%	100%	data:n
Active	121.56	4.80289	112	117.25	122.5	124.00	132	50
Moderate	133.20	6.88091	116	129.00	133.5	138.00	148	50
Sedentary	135.52	8.14722	112	131.00	135.0	140.75	152	50

Breakout Descriptive Statistics of Systolic Blood Pressure: Wide Format

R Input

```
RcmdrMisc::numSummary(ExerciseWide.df$Active,
  statistics=c("mean", "sd", "quantiles"))
```

R Output

mean	sd	0%	25%	50%	75%	100%	n
121.56	4.80289	112	117.25	122.5	124	132	50

R Input

```
RcmdrMisc::numSummary(ExerciseWide.df$Moderate,
  statistics=c("mean", "sd", "quantiles"))
```

R Output

```
mean      sd  0% 25%  50% 75% 100% n
133.2 6.88091 116 129 133.5 138 148 50
```

R Input

```
RcmdrMisc::numSummary(ExerciseWide.df$Sedentary,
  statistics=c("mean", "sd", "quantiles"))
```

R Output

```
mean      sd  0% 25%  50% 75% 100% n
135.52 8.14722 112 131 135 140.75 152 50
```

The RcmdrMisc::numSummary() function provided assurance that there is agreement on the breakout descriptive statistics (e.g., Systolic Blood Pressure—Sedentary, Systolic Blood Pressure—Moderate, and Systolic Blood Pressure—Active) for the data in both wide format (`ExerciseWide.df`) and stacked format (`ExerciseLong.df`). After a few more Quality Assurance actions, now with graphical tools as well as statistical measures, the actual Oneway Analysis of Variance test can be conducted to determine if the observed differences in Systolic Blood Pressure by Lifestyle are statistically significant ($p \leq 0.05$) or if the differences are due only to chance (Figs. 5.9 and 5.10).

R Input

```
par(ask=TRUE)
epiDisplay::tab1(ExerciseLong.df$SBP, graph= TRUE)
# Be sure to review the screen output as well as
# the figure.
```

R Input

```
par(ask=TRUE)
epiDisplay::summ(ExerciseLong.df$SBP,
  by=ExerciseLong.df$Lifestyle, graph= TRUE)
# Be sure to review the screen output as well as
# the figure.
```

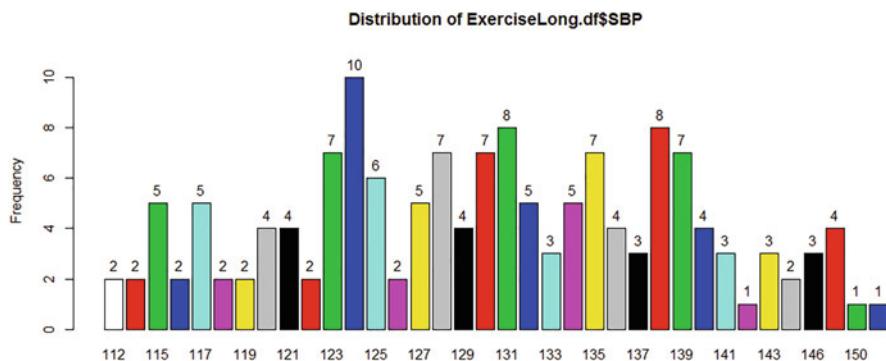


Figure 5.9: Distribution of systolic blood pressure

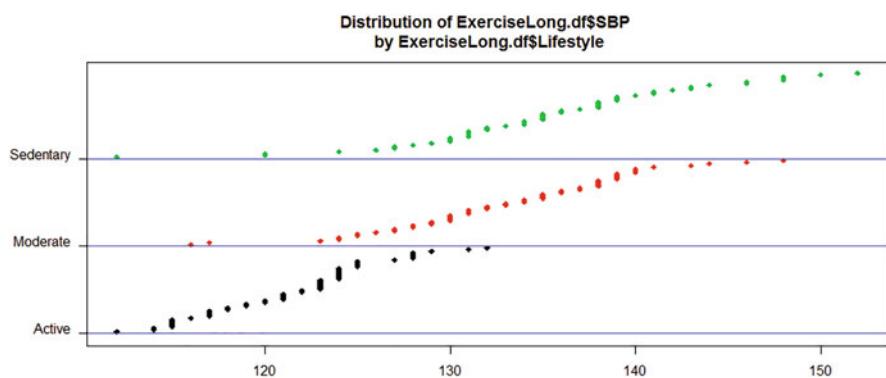


Figure 5.10: Distribution of systolic blood pressure by lifestyle breakouts—9

R Input

```
par(ask=TRUE)
epiDisplay::tabpct(ExerciseLong.df$SBP,
  ExerciseLong.df$Lifestyle,
  main=
  "Systolic Blood Pressure by Lifestyle: Active (Red),
  Moderate (Blue), and Sedentary (Black)",
  xlab="Systolic Blood Pressure",
  ylab="Lifestyle",
  col=c("red", "blue", "black"),
  percent=c("col"))
```

The `ggplot2::ggplot()` function is used to show Systolic Blood Pressure values for each of the three breakout groups side-by-side. This common presentation, with common scales facilitating comparisons, neither dictates nor confirms statistical outcomes, but it does provide a convenient way of comparing values for each breakout group (e.g., Systolic Blood Pressure—Sedentary, Systolic Blood

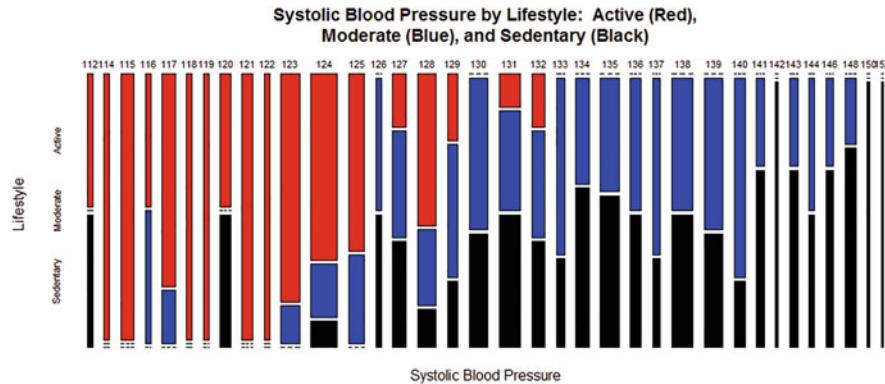


Figure 5.11: Distribution of systolic blood pressure by lifestyle breakouts—10

Pressure—Moderate, and Systolic Blood Pressure—Active) (Fig. 5.11).

Far more detail on themes is presented in later lessons. For now, use the enumerated theme (e.g., theme_MacYates) to see if it is helpful in improving presentation. Notice how the emphasis of this enumerated theme is on large, bold, and vibrant display of outcomes—all to make it easier to see the figure and accompanying details. Recall that theme_MacYates() is a function with specific attributes (Figs. 5.12, 5.13, 5.14, 5.15, and 5.16):

R Input

```
#####
theme_MacYates <- function(base_size=12, base_family="sans"){
  theme(
    plot.title=element_text(face="bold", size=14, hjust=0),
    plot.caption=element_text(face="bold", size=08, hjust=0),
    axis.title.x=element_text(face="bold", size=14,
      hjust=0.5),
    axis.text.x=element_text(face="bold", size=12, angle=00,
      color="black"),
    axis.title.y=element_text(face="bold", size=12, vjust=1,
      angle=90),
    axis.text.y=element_text(face="bold", size=10, angle=00,
      color="black", hjust=1),
    legend.title=element_text(face="bold", size=12),
    legend.text=element_text(face="bold", size=12),
    axis.ticks.x=element_line(size=1.0),
    axis.ticks.y=element_line(size=1.0),
    axis.ticks.length=unit(0.20,"cm"),
    axis.line=element_line(color="black", size=1.5,
      linetype = "solid"),
```

```

    panel.background=element_rect(fill="whitesmoke")
  )
}

# hjust - horizontal justification; 0 = left edge to 1 = right
# edge, with 0.5 the default
# vjust - vertical justification; 0 = bottom edge to 1 = top
# edge, with 0.5 the default
# angle - rotation; generally 0 to 90 degrees, with 0 the
# default
#####
#####
```

R Input

```

# Histogram breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(ExerciseLong.df,
  aes(x=SBP, fill=Lifestyle)) +
  geom_histogram(position="identity", color="white",
  bins = 15) +
  facet_grid(. ~ Lifestyle) +
  ggtitle("Systolic Blood Pressure (Histogram) by Lifestyle:
Active, Moderate, and Sedentary") +
  labs(
    x = "\nSystolic Blood Pressure", y = "Count\n") +
  # Quality Assurance check to see general trends
  theme_MacYates()
```

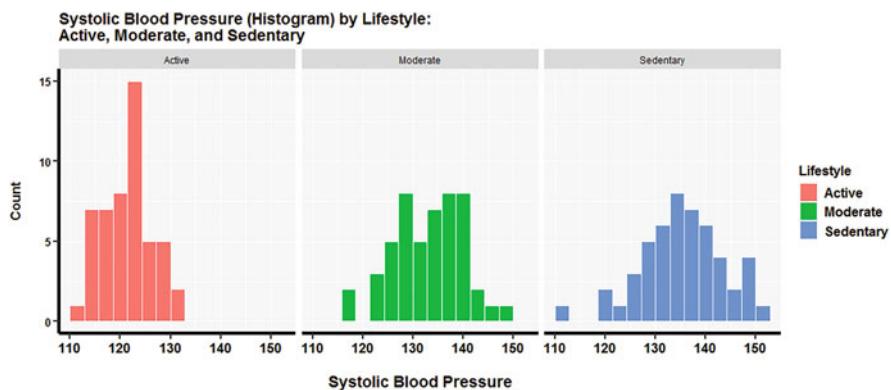


Figure 5.12: Distribution of systolic blood pressure by lifestyle breakouts—11

R Input

```
# Density Plot breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(ExerciseLong.df,
  aes(SBP, color=Lifestyle)) +
  geom_density(size=1.25) +
  ggtitle("Systolic Blood Pressure (Density Plot) by Lifestyle:
  Active, Moderate, and Sedentary") +
  labs(
    x = "\nSystolic Blood Pressure", y = "Density\n") +
  # Quality Assurance check to see general trends
  theme_MacYates()
```

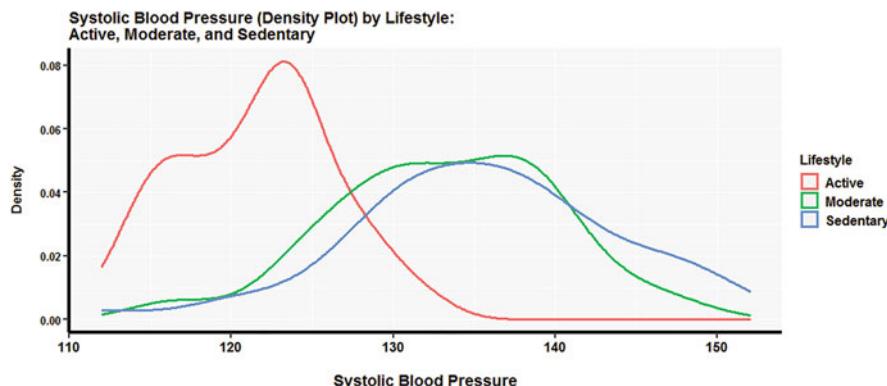


Figure 5.13: Distribution of systolic blood pressure by lifestyle breakouts—12

R Input

```
# Box Plot breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(ExerciseLong.df,
  aes(x=Lifestyle, y=SBP)) +
  geom_boxplot(aes(fill=Lifestyle)) +
  ggtitle("Systolic Blood Pressure (Boxplot) by Lifestyle:
  Active, Moderate, and Sedentary") +
  labs(
    x = "\nLifestyle", y = "Systolic Blood Pressure\n") +
  # Quality Assurance check to see general trends
```

```
theme_MacYates()
```

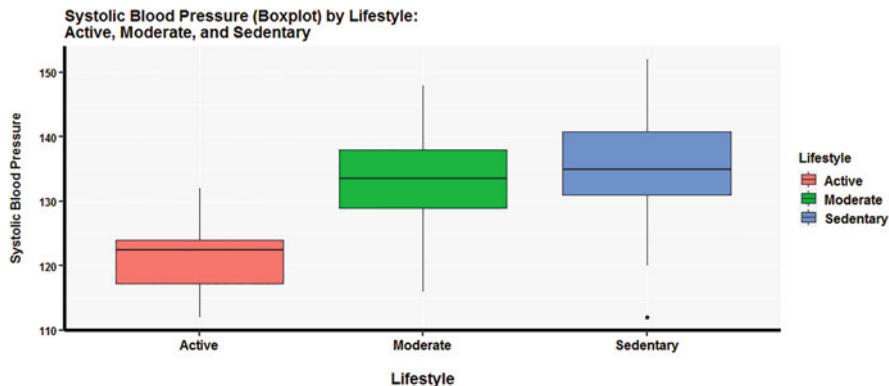


Figure 5.14: Distribution of systolic blood pressure by lifestyle breakouts—13

R Input

```
# Q-Q Plot and Q-Q Line breakouts using the ggplot2::ggplot()
# function

par(ask=TRUE)
ggplot2::ggplot(ExerciseLong.df,
  aes(sample=SBP)) +
  stat_qq(color="red") +
  stat_qq_line(color="black", size=1.75) +
  facet_grid(. ~ Lifestyle) +
  ggttitle(
    "Systolic Blood Pressure (QQ-Plot and QQ-Line) by Lifestyle:
    Active, Moderate, and Sedentary") +
  labs(
    x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
  # Quality Assurance check to see general trends
  theme_MacYates()
```

R Input

```
# Violin Plot breakouts using the ggplot2::ggplot() function

par(ask=TRUE)
ggplot2::ggplot(ExerciseLong.df,
  aes(x = Lifestyle, y = SBP)) +
  geom_violin(aes(fill=Lifestyle)) +
```

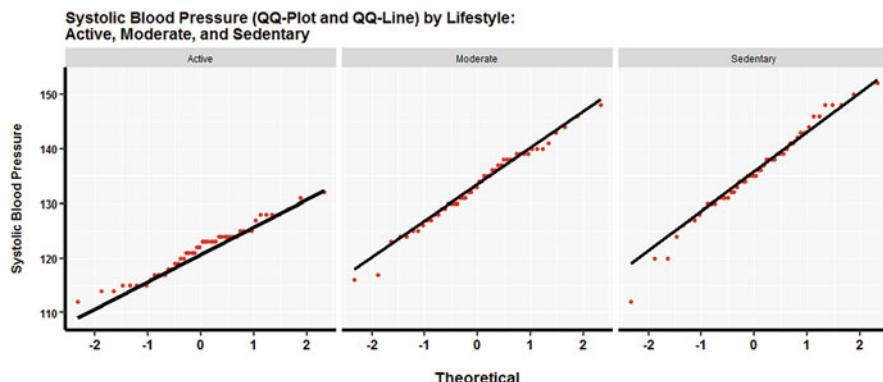


Figure 5.15: Distribution of systolic blood pressure by lifestyle breakouts—14

```
ggtitle("Systolic Blood Pressure (Violin Plot) by Lifestyle: Active, Moderate, and Sedentary") +
  labs(
    x = "\nLifestyle", y = "Systolic Blood Pressure\n") +
  # Quality Assurance check to see general trends
  theme_MacYates()
```

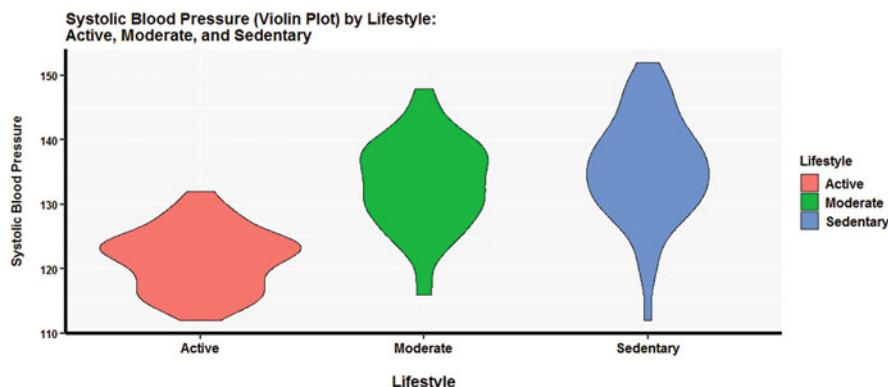


Figure 5.16: Distribution of systolic blood pressure by lifestyle breakouts—15

R Input

```
# Dot Plot breakouts using the ggplot2::ggplot() function
par(ask=TRUE)
ggplot2::ggplot(ExerciseLong.df,
  aes(x = Lifestyle, y = SBP)) +
  geom_dotplot(binaxis = "y", stackdir = "center",
```

```

aes(fill=Lifestyle), binwidth=2) +
ggtitle(
  "Systolic Blood Pressure (Dot Plot) by Lifestyle:
  Active, Moderate, and Sedentary") +
labs(
  x = "\nLifestyle", y = "Systolic Blood Pressure\n") +
# Quality Assurance check to see general trends
theme_MacYates()

```

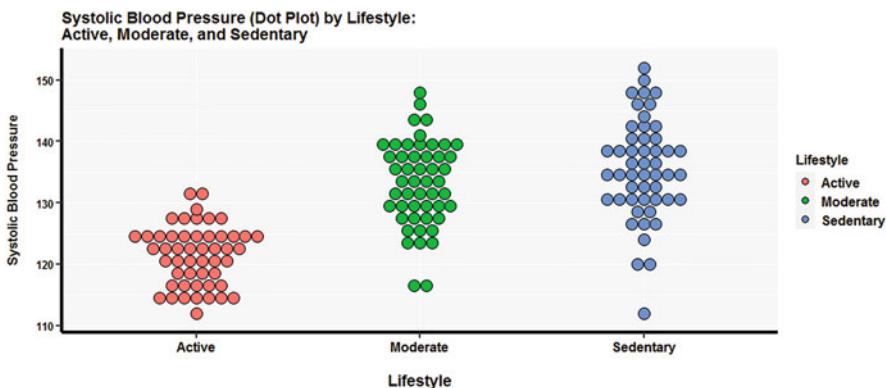


Figure 5.17: Distribution of systolic blood pressure by lifestyle breakouts—16

Different R-based Quality Assurance tools were used to determine that the data, in wide (e.g., unstacked) format, were acceptable. Now, with the data in long (e.g., stacked) format, once again the data seem to be in good format. Given assurance that the data are acceptable, it is possible to demonstrate the many different functions associated with Oneway ANOVA. Be sure to notice how (Fig. 5.17):

- Some functions only identify if there is, or is not, a statistically significant difference between and among the breakout groups (Systolic Blood Pressure—Sedentary, Systolic Blood Pressure—Moderate, and Systolic Blood Pressure—Active).
- Other functions are more detailed and identify not only if there is, or is not, a statistically significant difference between the breakout groups, but there is also identification of which breakout groups are in parity in terms of mean Systolic Blood Pressure and which breakout groups are unique in terms of mean Systolic Blood Pressure.

5.7.1 Exploratory Oneway ANOVA

Use the `oneway.test()` function to examine means between and among the breakout groups. The `oneway.test()` function is associated with the `stats` package, which is made available when R is first downloaded.

R Input

```
oneway.test(SBP ~ Lifestyle, data=ExerciseLong.df,  
var.equal=TRUE)  
# Assume variance between samples is equal
```

R Output

```
One-way analysis of means  
p-value <0.0000000000000002
```

R Input

```
oneway.test(SBP ~ Lifestyle, data=ExerciseLong.df,  
var.equal=FALSE)  
# Assume variance between samples is not equal
```

R Output

```
One-way analysis of means (not assuming equal variances)  
p-value <0.0000000000000002
```

Using the `oneway.test()` function, the calculated p-value (0.0000000000000002) is certainly less than the $p \leq 0.05$ level of significance, going back to the criterion level of significance associated with the Null Hypothesis. Accordingly, application of the `oneway.test()` function provides assurance that there is a statistically significant ($p \leq 0.05$) difference in Systolic Blood Pressure means by Lifestyle (e.g., Sedentary, Moderate, and Active), but the exact nature of the difference is not known. It is only known that there is a difference.

5.7.2 Oneway ANOVA Method 1: `lm()` and `anova()` functions

The `oneway.test()` function, used for exploratory purposes, is a good beginning at inquiries into a Oneway ANOVA, but there are limitations on the use of this function since it is not known which breakout groups differ from each other and which breakout groups are in parity with each other in terms of mean Systolic Blood Pressure differences at $p \leq 0.05$. Similar R-based functions are demonstrated below, which provide more information (specifically, a standard ANOVA table) than the `oneway.test()` function, but still have limitations in terms of support for a full understanding of outcomes:

R Input

```
anova(lm(SBP ~ Lifestyle -1, data=ExerciseLong.df,
na.action=na.exclude))
# Note how this analysis accommodates missing data.
# The -1 argument provides a fit model without an intercept.
# Groups are compared only to each other.
```

R Output

Analysis of Variance Table

Response: SBP

	Df	Sum Sq	Mean Sq	F value	Pr(>F)						
Lifestyle	3	2544237	848079	18599	<0.0000000000000002 ***						
Residuals	147	6703	46								

Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'.'	0.1	' '	1

This method, based on use of the `anova()` function wrapped around the `lm()` function, provides evidence that there is a statistically significant difference at $p \leq .05$ for SBP values by Lifestyle breakout groups. Give special attention to the `Signif.codes` part of the ANOVA table. Note, however, that groupwise comparisons are absent. It is currently unclear which Lifestyle groups differ in statistical significance ($p \leq 0.05$) by SBP (e.g., Systolic Blood Pressure) values.

5.7.3 Oneway ANOVA Method 2: `aov()` and `TukeyHSD()` Functions

Use the `aov()` function to start more detailed comparisons of statistically significant differences in means between and among the Lifestyle breakout groups. The `aov()` function is associated with the `stats` package, which is made available when R is first downloaded.

R Input

```
summary(aov(SBP ~ Lifestyle, data=ExerciseLong.df))
# Wrap the summary() function around the aov()
# function to obtain useful output
```

R Output

	Df	Sum Sq	Mean Sq	F value	Pr(>F)						
Lifestyle	2	5596	2798	61.4	<0.0000000000000002 ***						
Residuals	147	6703	46								
<hr/>											
Signif. codes:	0	'***'	0.001	'**'	0.01	'*'	0.05	'.'	0.1	'	1

The `aov()` function yields a calculated p-value of 0.0000000000000002, which is certainly less than the $p \leq 0.05$ level of significance, going back to the criterion level of significance associated with the Null Hypothesis. Again, the `aov()` function provides assurance that there is a statistically significant ($p \leq 0.05$) difference in Systolic Blood Pressure means by Lifestyle (e.g., Sedentary, Moderate, and Active), but the exact nature of differences still remains unknown.

Unlike a test where only two sample means are compared, such as Student's t-Test for Independent Samples, this Oneway Analysis of Variance example deals with three breakout groups, which begs the questions:

- Is the Systolic Blood Pressure mean for Sedentary unique and different from the Systolic Blood Pressure mean for all other Lifestyle breakout groups, Moderate and Active.?
- Is the Systolic Blood Pressure mean for Moderate unique and different from the Systolic Blood Pressure mean for all other Lifestyle breakout groups, Sedentary and Active?
- Is the Systolic Blood Pressure mean for Active unique and different from the Systolic Blood Pressure mean for all other Lifestyle breakout groups, Sedentary and Moderate?
- Are the Systolic Blood Pressure means for Sedentary and Moderate in common and different from the Systolic Blood Pressure mean for Active?
- Are the Systolic Blood Pressure means for Sedentary and Active in common and different from the Systolic Blood Pressure mean for Moderate?
- Are the Systolic Blood Pressure means for Moderate and Active in common and different from the Systolic Blood Pressure mean for Sedentary?
- Are the Systolic Blood Pressure means for Sedentary, Moderate, and Active different, with no commonality?

Neither the `oneway.test()` function, the `anova(lm())` functions, nor the `summary(aov())` functions on their own provide the type of information needed to answer these questions.

To easily determine where sample means are in parity and where sample means differ, individually and by overlapping breakout groups, apply Tukey's HSD (Honestly Significant Difference) test. With R, the `TukeyHSD()` function is used to achieve the aim of examining multiple comparisons. However, there are many possible mean comparison tests—with Tukey's HSD test a frequent first choice, but not the only choice. Give attention to a few of the many mean comparison tests, which are commonly used for the purpose of comparing differences between means in a Oneway ANOVA (Table 5.1):

Table 5.1: Oneway ANOVA mean comparison tests

Selected tests

Duncan	Duncan's multiple range test
LSD	Least-significant difference
Scheffé	Scheffé's test
SNK	Student-Newman-Keuls
Tukey	Tukey's honestly significant difference (HSD)

Statistical tests that can account for increased complexity are needed if meaningful decisions involving statistics in the large are to be effected. Again, be sure to consider the increased complexity of decision-making and comparisons supported by a simple Oneway ANOVA.⁵

Similar to the `oneway.test()` function and the `aov()` function, the `TukeyHSD()` function is associated with the `stats` package. Note how the `TukeyHSD()` function is wrapped around the `aov()` function.

R Input

```
TukeyHSD((aov(SBP ~ Lifestyle, data=ExerciseLong.df)),
conf.level=0.95)
```

R Output

```
Tukey multiple comparisons of means
95% family-wise confidence level
```

⁵ As an advance organizer, even more complex comparisons and analyses are supported in Twoway ANOVA, but that is not the purpose of this lesson.

```
Fit: aov(formula = SBP ~ Lifestyle, data = ExerciseLong.df)

Lifestyle
      diff      lwr      upr   p adj
Moderate-Active 11.64 8.442394 14.83761 0.0000000
Sedentary-Active 13.96 10.762394 17.15761 0.0000000
Sedentary-Moderate 2.32 -0.877606 5.51761 0.201941
```

By focusing on the p adj (e.g., calculated p-value) column, the output associated with the TukeyHSD() function is fairly easy to interpret:

- Calculated p-value (e.g., p adj) is less than 0.05 for Moderate-Active breakout group comparisons, indicating a statistically significant difference at $p \leq 0.05$.
- Calculated p-value (e.g., p adj) is less than 0.05 for Sedentary-Active breakout group comparisons, indicating a statistically significant difference at $p \leq 0.05$.
- Calculated p-value (e.g., p adj) is not less than 0.05 for Sedentary-Moderate breakout group comparisons, indicating that there is no statistically significant difference at $p \leq 0.05$.

If it helps to understand where there are common groupings and where there are differences, generate a simple boxplot to once again view the distribution of Systolic Blood Pressure values for the three Lifestyle breakout groups:

R Input

```
par(ask=TRUE)
ggplot2::ggplot(data=ExerciseLong.df,
  aes(x=Lifestyle, y=SBP, fill=Lifestyle)) +
  geom_boxplot() +
  stat_summary(fun.y=mean, color="black", geom="point",
    shape=18, size=2, show.legend=FALSE) +
  stat_summary(fun.y=mean, color="black", geom="text",
    size=6, show.legend=FALSE, vjust=1.50,
    aes(label=round(..y.., digits=1))) +
  labs(title=
  "Mean Systolic Blood Pressure by Lifestyle:
  Active, Moderate, and Sedentary",
  caption="Mean shows as a small black dot inside the box
  and median shows as a solid horizontal line.\n",
  x="\nLifestyle",
```

```

y="Systolic Blood Pressure\n") +
theme(plot.caption=element_text(face="bold", size=08,
hjust=0.5, color="black")) +
theme(legend.position="none") +
theme_MacYates()

```

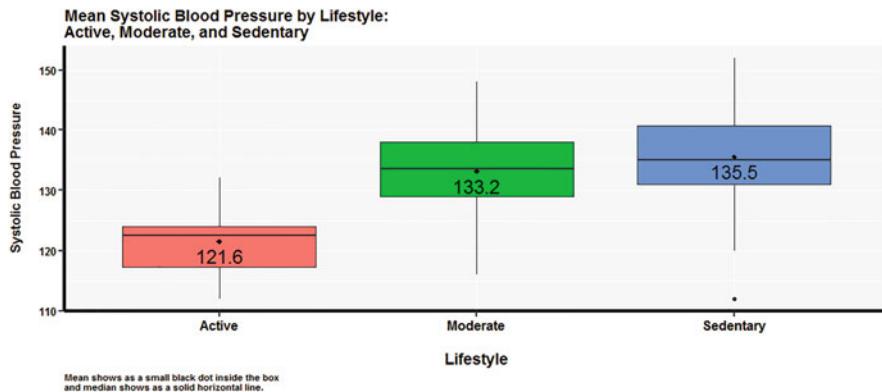


Figure 5.18: Distribution of systolic blood pressure by lifestyle breakouts—17

A determination of statistical significance should not be declared based on graphical output only. Yet, the simple boxplot generated here certainly reinforces common and unique breakout group memberships, now that calculated p-values are known (Fig. 5.18).

An inferential test is needed for precise interpretation of the data and any declaration of statistically significant differences, based on how the Null Hypothesis is worded. For this lesson on Systolic Blood Pressure values by Lifestyle, the TukeyHSD() function provided detailed outcomes on where there are statistically significant differences ($p \leq .05$) for Systolic Blood Pressure means by Lifestyle breakout groups and equally, were there is no statistically significant differences ($p \leq .05$) for Systolic Blood Pressure means by Lifestyle breakout groups.

5.8 Summary of Outcomes

In this lesson the graphics and generated statistics provided a great deal of information in support of decision-making. Of immediate importance, however, focus on the Null Hypothesis and the breakout analyses deriving from use of the TukeyHSD() function:

Null Hypothesis (H_0): There is no statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure between subjects based on Lifestyle (e.g., typical exercise patterns such as Sedentary, Moderate, Active).

For this lesson, there are overall statistically significant differences ($p \leq 0.05$)

in mean Systolic Blood Pressure by Lifestyle breakout groups. Oneway ANOVA served as the primary approach to determine if a statistically significant difference existed and the TukeyHSD() function then served as the tool by which finite comparisons of Systolic Blood Pressure by Lifestyle breakout group comparisons were made.

Review the edited summary comparison to give another view of which Lifestyle breakout groups are significantly different from each other ($p \leq 0.05$) and which are not.

Lifestyle	Mean SBP	Lifestyle	Mean SBP	p adj	p <= 0.05
Sedentary	135.52	Moderate	133.20	0.201941	No
Sedentary	135.52	Active	121.56	0.000000	Yes
Moderate	133.20	Active	121.56	0.000000	Yes

Overall, there is no statistically significant difference ($p \leq 0.05$) in mean Systolic Blood Pressure for those subjects who lead a Sedentary Lifestyle (Mean SBP = 135.52) and those subjects with a Moderate Lifestyle (Mean SBP = 133.20). The same outcome cannot be said when an Active Lifestyle is brought into comparison. There is a statistically significant difference ($p \leq 0.05$) in mean Systolic Blood Pressure for those subjects who lead a Sedentary Lifestyle (Mean SBP = 135.52) and those who have an Active Lifestyle (Mean SBP = 121.56). Also, there is a statistically significant difference ($p \leq 0.05$) in mean Systolic Blood Pressure for those subjects who lead a Moderate Lifestyle (Mean SBP = 133.20) and those who have an Active Lifestyle (Mean SBP = 121.56).

A graphical representation of the TukeyHSD values may help better reinforce the group comparisons. To achieve this aim, wrap the plot() function around application of the TukeyHSD() function. A plot related to the TukeyHSD() function is generally helpful for when only a few subgroups are compared to each other. See below if in this lesson, where there are three Lifestyle breakout groups, if the plot is of any practical value for presentation to others. If not, it still might be helpful for internal decision-making (Fig. 5.19).

R Input

```
font      <- par(font=2) # Font bold
savecex.lab <- par(cex=1.0) # Label size
savelwd     <- par(lwd=4)   # Line width
par(ask=TRUE)
plot(TukeyHSD((aov(SBP ~ Lifestyle,
                    data=ExerciseLong.df)),
               conf.level=0.95), las=3, col="red")
par(savefont); par(savecex.lab); par(savelwd)
# Look for breakout groups that overlap and
```

```
# those breakout groups that do not overlap.
```

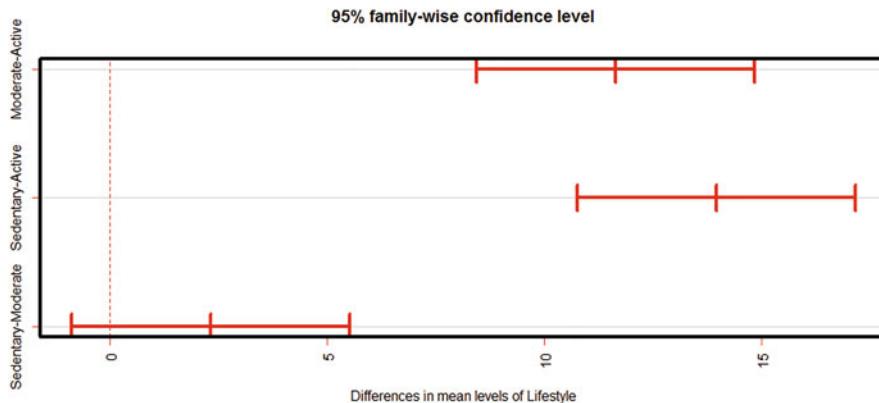


Figure 5.19: Systolic blood pressure comparative family-wise confidence levels

A simple `ggplot2::qplot()` function is used to create a simple graphic, reinforcing distribution of Systolic Blood Pressure values by Lifestyle breakout groups (Fig. 5.20).

R Input

```
par(ask=TRUE)
ggplot2::qplot(Lifestyle, SBP, data=ExerciseLong.df,
  main="Systolic Blood Pressure by Lifestyle",
  xlab="\nLifestyle", ylab="Systolic Blood Pressure\n",
  geom="auto", col="darkred", size=1.5) +
  scale_y_continuous(limits=c(100,160),
    breaks=seq(100,160,10)) +
  theme(legend.position="none") +
  theme_MacYates()
# This graphic uses the ggplot2::qplot() function
# and not the ggplot2::ggplot function.
```

The `ggplot2::qplot()` function is interesting, but the `stripchart()` function is also useful and it may yield a more complete view of how data are distributed, allowing detailed comparisons of Systolic Blood Pressure by Lifestyle breakout groups (Fig. 5.21).

R Input

```
par(ask=TRUE)
stripchart(ExerciseLong.df$SBP ~ ExerciseLong.df$Lifestyle,
```

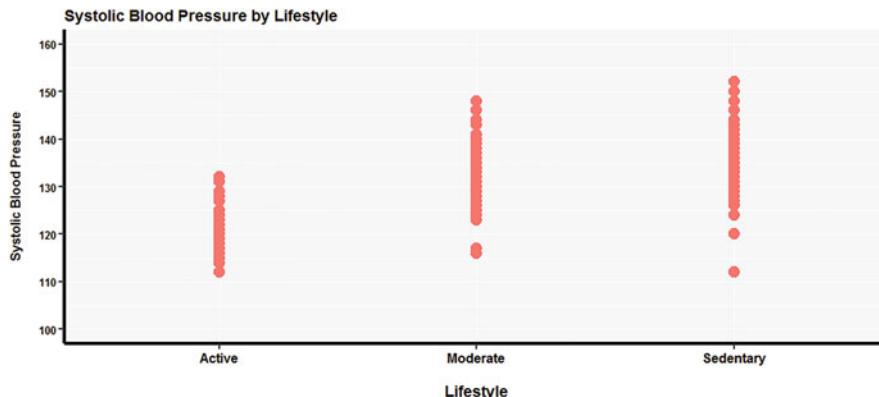


Figure 5.20: Distribution of systolic blood pressure by lifestyle breakouts—18

```
method="jitter", jitter=.1, vertical=TRUE,
main=
"Jitter Stripchart of Systolic Blood Pressure by Lifestyle",
xlab="\nLifestyle", ylab="Systolic Blood Pressure\n",
cex.lab=1.25, cex.axis=1.25, ylim=c(100,160), pch=19,
col="darkred")
```

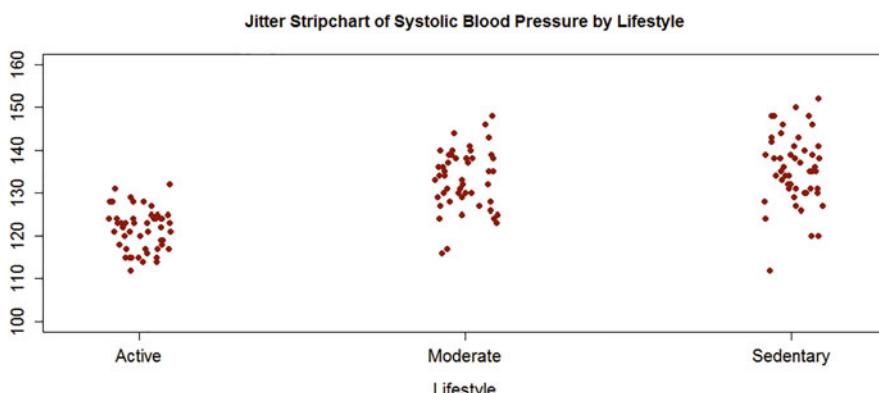


Figure 5.21: Distribution of systolic blood pressure by lifestyle breakouts—19

Finally, go back to the graphical images at the level of group comparisons (Mean and Confidence Interval) to see how these images parallel the Oneway ANOVA findings. Syntax for complementary graphical images of Systolic Blood Pressure by Lifestyle follows, based on use of the `sciplot::lineplot()` function, the `s20x::boxqq()` function, and the `s20x::onewayPlot()` function (Figs. 5.22 and 5.23).

R Input

```

install.packages("sciplot")
library(sciplot)                      # Load the sciplot package.
help(package=sciplot)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

par.ask=TRUE)
sciplot::lineplot.CI(ExerciseLong.df$Lifestyle,
  ExerciseLong.df$SBP,
  main="Mean and CI Systolic Blood Pressure by Lifestyle",
  xlab="\nLifestyle", ylab="\nSystolic Blood Pressure",
  font=2, font.lab=2, legend=FALSE, type="b",
  x.cont=FALSE, lwd=3, col="red", err.width=0.75,
  err.col="darkblue", err.lty="solid", ylim=c(120,140))
# The Y axis scale for Mean and CI of Systolic Blood
# Pressure was set to focus on Systolic Blood Pressure
# means for each Lifestyle breakout group.

```

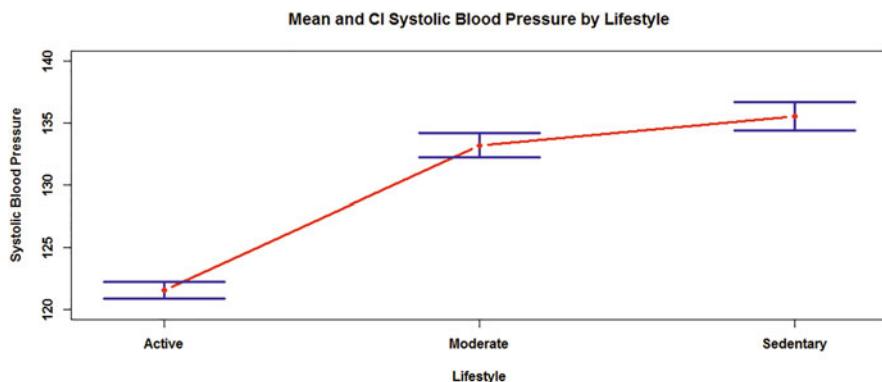


Figure 5.22: Distribution of systolic blood pressure by lifestyle breakouts—20

R Input

```

install.packages("s20x")
library(s20x)                          # Load the s20x package.
help(package=s20x)                     # Show the information page.
sessionInfo()                         # Confirm all attached packages.

par.ask=TRUE)             # Pause
s20x::boxqq(SBP ~ Lifestyle, data=ExerciseLong.df)

```

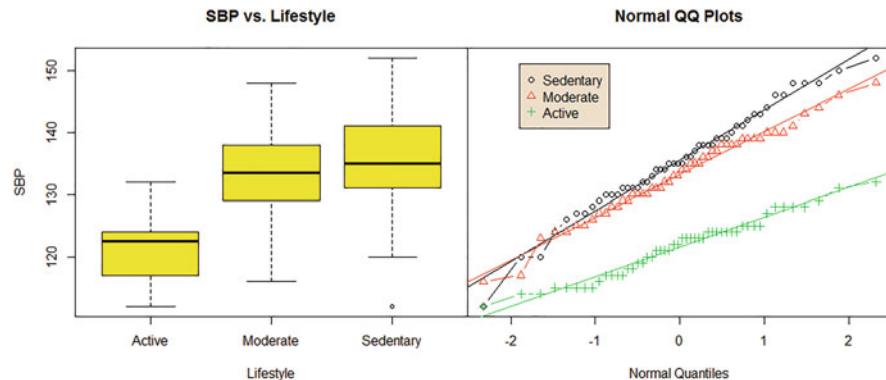


Figure 5.23: Distribution of systolic blood pressure by lifestyle breakouts—21

R Input

```
par(ask=TRUE)      # Pause
s20x::onewayPlot(SBP ~ Lifestyle, data=ExerciseLong.df)
```

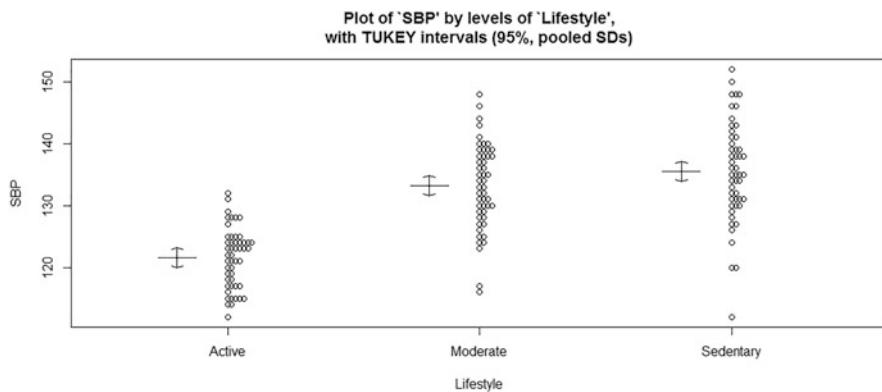


Figure 5.24: Distribution of systolic blood pressure by lifestyle breakouts—22

Although the figures in this Summary of Outcomes section are fairly basic and may not be of publishable quality, they certainly reinforce in graphical format the statistics and eventual conclusions gained from Oneway ANOVA and the TukeyHSD() function. There is a clear difference between Active Lifestyle and the two other breakout groups, Sedentary Lifestyle and Moderate Lifestyle, in terms of mean Systolic Blood Pressure (Fig. 5.24).

A long list of additional functions could easily continue (and are addressed to a small degree in Addendum 1) for this lesson on Oneway ANOVA, both functions that focus on graphical presentation and functions that focus on different type of statistics. The important thing to consider in terms of outcomes for this lesson are:

- Based on user-created data specific to this lesson, it was determined that there was a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure means between and among the three separate Lifestyle breakout groups based on exercise: Sedentary, Moderate, and Active.
- Knowing that there was a statistically significant difference ($p \leq 0.5$), it was further determined that:
 - Subjects in the Active Lifestyle breakout group had a mean Systolic Blood Pressure of 121.56 and the Systolic Blood Pressure of Active Lifestyle subjects was statistically different ($p \leq 0.05$) than the Systolic Blood Pressure of members from either the Sedentary Lifestyle (Mean SBP = 135.52) breakout group or the Moderate Lifestyle (Mean SBP = 133.20) breakout group.
 - There was no statistically significant ($p \leq 0.05$) difference in the Systolic Blood Pressure of Sedentary Lifestyle (Mean SBP = 135.52) breakout group members or Moderate Lifestyle (Mean SBP = 133.20) breakout group members. That is to say, the mean Systolic Blood Pressure for Sedentary subjects and Moderate was in parity.

Never forget that research demands replication and representation. This lesson was a one-time look at user-created theoretical values for Systolic Blood Pressure. Yes, the measured values were selected to parallel expected norms, but no broad judgment should ever be made from one and only one study. In this lesson, subjects with an Active lifestyle had a mean Systolic Blood Pressure that was less than subjects who are not as active. Yet, replication and representation must be observed before actions are justified.

5.9 Addendum 1: Other Packages for Display of Oneway ANOVA

Along with the functions available in the stats package and the other packages obtained when R is first downloaded, the agricolae::HSD.test() function is also commonly used to examine Oneway ANOVA outcomes. Notice how the output is slightly different compared to what has been seen previously, but the outcomes are the same in terms: (1) means of the three breakout groups, (2) which breakout group means are in common, and (3) which breakout group means are different.

R Input

```
install.packages("agricolae", dependencies=TRUE)
library(agricolae)           # Load the agricolae package.
help(package=agricolae)       # Show the information page.
```

```
sessionInfo() # Confirm all attached packages.

agricolae::tapply.stat(ExerciseLong.df$SBP,
  ExerciseLong.df$Lifestyle,
  function(x) mean(x,na.rm=TRUE))
# Accommodate missing values, when a concern.
```

R Output

	ExerciseLong.df\$Lifestyle	ExerciseLong.df\$SBP
1	Active	121.56
2	Moderate	133.20
3	Sedentary	135.52

R Input

```
agricolae::HSD.test(
  aov(SBP ~ Lifestyle, data=ExerciseLong.df), # Model
  trt="Lifestyle", # Treatment
  group=TRUE, console=TRUE, alpha=0.05, # Arguments
  main="Systolic Blood Pressure by Lifestyle - Exercise Level")
# Wrap the agricolae::HSD.test() function around the
# Oneway ANOVA model obtained by using the aov()
# function. Select desired arguments, such as group,
# console, and alpha (e.g., p-value).
```

R Output

Study: Systolic Blood Pressure by Lifestyle - Exercise Level

HSD Test for SBP

Mean Square Error: 45.5973

Lifestyle, means

	SBP	std	r	Min	Max
Active	121.56	4.80289	50	112	132
Moderate	133.20	6.88091	50	116	148
Sedentary	135.52	8.14722	50	112	152

Alpha: 0.05 ; DF Error: 147

Critical Value of Studentized Range: 3.34842

Minimum Significant Difference: 3.19761

Treatments with the same letter are not significantly different.

SBP groups		
Sedentary	135.52	a
Moderate	133.20	a
Active	121.56	b

The agricolae::HSD.test() function provides a very convenient summary of the information needed to make an informed interpretation of Oneway ANOVA outcomes:

- Descriptive statistics for each breakout group are provided, including the mean, standard deviation, number, minimum, and maximum.
- Lowercase letters are used to identify common groups and by extension groups that are not in common. In this example, the means for Systolic Blood Pressure—Sedentary and Systolic Blood Pressure—Moderate are common to each other and are identified as group **a**. In contrast, the mean for Systolic Blood Pressure—Active is listed by itself as group **b**. The outcomes are the same as what was seen before and only the output to the screen is different, which is common for many otherwise redundant R functions.

Along with the statistics associated with the agricolae::HSD.test() function when wrapped around the aov() function, look at the graphical output gained by wrapping the agricolae::diffograph() function around the HSD.test() and aov() functions. Give attention to the way the statistically significant difference ($p \leq 0.05$) is presented:

- The intersection of mean Systolic Blood Pressure for Active Lifestyle and Moderate Lifestyle shows a red line, indicating a significant difference between the two lifestyle groups ($p \leq 0.05$).
- The intersection of mean Systolic Blood Pressure for Active Lifestyle and Sedentary Lifestyle shows a red line, indicating a significant difference between the two lifestyle groups ($p \leq 0.05$).
- The intersection of mean Systolic Blood Pressure for Moderate Lifestyle and Sedentary Lifestyle shows a blue line, indicating no significant difference between the two lifestyle groups ($p \leq 0.05$).

R Input

```
agricolae::diffograph((HSD.test(aov(SBP ~ Lifestyle,
  data=ExerciseLong.df), "Lifestyle", alpha=0.05,
  group=FALSE)), cex.main=1.25, cex.lab=1.25, cex.axis=1.25,
  lwd=4, xlab="\nSystolic Blood Pressure",
  ylab="\nSystolic Blood Pressure")
```

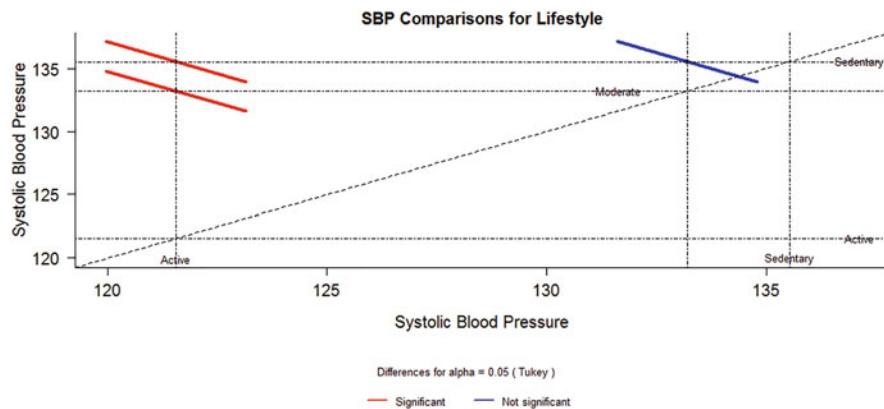


Figure 5.25: Differences of systolic blood pressure means by lifestyle breakouts (Tukey—alpha = 0.05)

The graphical output that can be achieved with R should be viewed as a complementary advantage to statistical output only. Graphics help reinforce outcomes for researchers and are essential for many (Fig. 5.25).

5.10 Addendum 2: Parametric v Nonparametric

5.10.1 Parametric Approach to Oneway ANOVA

There has been more than ample evidence displayed in this lesson that the data originally presented in `ExerciseWide.df` and later reorganized into `ExerciseLong.df` display an acceptable degree of normal distribution. Given this finding, functions typically associated with the use of a parametric approach to statistical analyses were used and it was determined that statistically significant differences ($p \leq 0.05$) were evident. See the prior Summary of Outcomes section for more detail.

5.10.2 Nonparametric Alternative to Oneway ANOVA

A common expression in research is *What-If*. Although there is every assurance that the data in `ExerciseLong.df` are fit for a parametric approach of Oneway ANOVA, it is still prudent to examine the data from a nonparametric

perspective, based on a *What-If* concern about normal distribution findings. *What-If* the data were nonparametric? This approach will provide even greater assurance in the final interpretation of outcomes.

A first attempt at Oneway ANOVA for nonparametric data is to apply the Kruskal–Wallis Oneway Analysis of Variance test, which with R is typically based on the `kruskal.test()` function:

R Input

```
kruskal.test(SBP ~ as.factor(Lifestyle),
  data=ExerciseLong.df)
# The as.factor() function was wrapped
# around the Lifestyle object variable
```

R Output

```
Kruskal-Wallis rank sum test
p-value <0.0000000000000002
```

The `kruskal.test()` function provided the same outcome as was provided using the `oneway.test()` function and the `aov()` function. The calculated p-value ($p \leq 0.000000000000022$) is certainly less than the criterion p-value of 0.05. Again, however, this test only provides evidence that there is a statistically significant difference in Systolic Blood Pressure means between and/or among the `Lifestyle` breakout groups. The precise `Lifestyle` breakout groups of commonality and difference are not identified using the `kruskal.test()` function.

To gain more insight into actual commonalities and differences by breakout groups, use Dunn's Test of Multiple Comparisons Using Rank Sums for when data are viewed from a nonparametric perspective. Dunn's Test is to some degree similar to Tukey's HSD (Honestly Significant Difference) test, but for use when data are considered nonparametric.

R Input

```
install.packages("dunn.test", dependencies=TRUE)
library(dunn.test)           # Load the dunn.test package.
help(package=dunn.test)      # Show the information page.
sessionInfo()                # Confirm all attached packages.

dunn.test::dunn.test(ExerciseLong.df$SBP,
  ExerciseLong.df$Lifestyle, method="bonferroni",
  kw=TRUE, label=TRUE, table=TRUE, list=TRUE)
```

```
# Produce a table of Kruskal-Wallis pairwise
# comparisons (e.g., kw=TRUE) and p-values for
# each comparison.
```

R Output

```
Kruskal-Wallis rank sum test

data: x and group
Kruskal-Wallis chi-squared = 72.8839, df = 2, p-value = 0

Comparison of x by group
(Bonferroni)

Col Mean-
Row Mean | Active   Moderate
-----+-----
Moderate | -6.791733
          | 0.0000*
          |
Sedentary | -7.875508 -1.083775
          | 0.0000*    0.4177

List of pairwise comparisons: Z statistic (adjusted p-value)
-----
Active - Moderate : -6.791733 (0.0000)*
Active - Sedentary : -7.875508 (0.0000)*
Moderate - Sedentary : -1.083775 (0.4177)

alpha = 0.05
Reject Ho if p <= alpha/2
```

Give attention to the adjusted p-values and the common use of an asterisk to indicate statistically significant difference ($p \leq 0.05$). Similar to what was seen using a parametric view toward the data, but now using Dunn's Test (a non-parametric test), it is again demonstrated that there is a statistically significant ($p \leq 0.05$) difference in Systolic Blood Pressure between: (1) subjects with an Active lifestyle compared to subjects with a Moderate lifestyle (calculated $p = 0.0000$) and (2) subjects with an Active lifestyle compared to subjects with a Sedentary lifestyle (calculated $p = 0.0000$). There was no statistically significant (calculated $p \leq 0.4177$, which is greater than the criterion $p \leq 0.05$)

difference in Systolic Blood Pressure between subjects with a Moderate lifestyle and subjects with a Sedentary lifestyle.⁶

Perhaps it was not necessary to revisit the data using a nonparametric approach toward Oneway ANOVA. Yet, this degree of redundancy gained by using the `dunn.test::dunn.test()` function and accompanying `kw=TRUE` argument is useful in that it confirms prior outcomes.

5.11 Addendum 3: Additional Practice Datasets

Practice Oneway ANOVA using the `BiologicalSpecimen.csv` dataset, which has been placed on the publisher's Web site associated with this text. Start the practice with an emphasis on Oneway ANOVA for measured object variable `M1` by factor-type object variable `F2b`, which has been detailed to a limited degree below. Then, continue along with Oneway ANOVA for factor-type object variable `F2b` against the measured object variables `M2`, `M3a`, `M3b`, and `M3c`. As part of this practice activity, start with a parametric approach to the data but after inquiry into normal distribution patterns of the data, it may also be prudent to approach the analyses from a nonparametric approach.⁷

A few guided directions and syntax for a Oneway ANOVA of `M1` by `F2b` follows, to prompt actions, but the best way to put this practice exercise into place is to follow along exactly with what has been presented up to this point in the lesson, but now for the data in what will be called `BioSpmen.df`. Download the file `BiologicalSpecimen.csv` (again, available at the publisher's Web site associated with this text) into the appropriate directory and continue along with practice activities against the dataset.⁸

Note In the front matter to this lesson, the syntax is presented and is then immediately followed in most cases by either a copy of screen output or an accompanying figure. That approach is not used in this addendum. View this addendum as practice homework-type bonus materials. Syntax is presented and descriptive text goes along with the syntax. However, screen output and figures are generally excluded and when they show it is only to offer mid-point guidance that analyses follow along in a correct manner. Either key the syntax in this addendum or copy and paste it into an editor—whatever is feasible and most convenient. And then, to use the common expression, Practice—Practice—Practice!

⁶With R and a most other statistically-focused software, it is not uncommon to see output show $p = 0.0000$. Know that this printout is theoretically incorrect and that a calculated p -value may be incredibly small, but there are those who question if it can ever reach 0.0000.

⁷The naming scheme used for this dataframe is to use **F** to indicate a factor-type object variable and **M** to indicate a measured object variable.

⁸Syntax is provided throughout this addendum, but of course this syntax is only a suggestion. Experiment and take other approaches to how the data can be analyzed and outcomes presented by using other functions and other arguments. Use this addendum as a confidence-building resource on how R is used with increasingly complex analyses.

R Input

```
BioSpmen.df <- read.table (file =
  "BiologicalSpecimen.csv",
  header = TRUE,
  sep = ",")                                # Import the .csv file.

getwd()                                     # Identify the working directory
ls()                                         # List objects
attach(BioSpmen.df)                         # Attach the data, for later use
str(BioSpmen.df)                            # Identify structure
nrow(BioSpmen.df)                           # List the number of rows
ncol(BioSpmen.df)                           # List the number of columns
dim(BioSpmen.df)                            # Dimensions of the data frame
names(BioSpmen.df)                           # Identify names
colnames(BioSpmen.df)                        # Show column names
rownames(BioSpmen.df)                        # Show row names
head(BioSpmen.df)                            # Show the head
tail(BioSpmen.df)                            # Show the tail
BioSpmen.df                                 # Show the entire data frame
summary(BioSpmen.df)                         # Summary statistics

#####
# Code Book for BioSpmen.df                 #
######
#                                             #
# Variable Labels                            #
# =====
# SubjetID      Subject Identification Number #
# M1           Measurement 1                  #
# M2           Measurement 2                  #
# M3a          Measurement 3a                 #
# M3b          Measurement 3b                 #
# M3c          Measurement 3c                 #
# F1           Factor 1                      #
# F2a          Factor 2a                     #
# F2b          Factor 2b                     #
#####
#                                             #
# Variable Values                           #
# =====
# ID            ..... Factor (e.g., nominal) #
```

```

# A unique ID ranging from 1 to 4,660 #
#
# M1 ..... Numeric (e.g., interval) #
# An unidentified biological variable #
# that ranges from 0.00 to 600.0 #
#
# M2 ..... Numeric (e.g., interval) #
# An unidentified biological variable #
# that ranges from 0.00 to 10.00 #
#
# M3a
# Numeric (e.g., interval) #
# An unidentified biological variable #
# that ranges from 0.00 to 4.00 #
#
# M3b
# Numeric (e.g., interval) #
# An unidentified biological variable #
# that ranges from 0.00 to 4.00 #
#
# M3c
# Numeric (e.g., interval) #
# An unidentified biological variable #
# that ranges from 0.00 to 4.00 #
#
# F1 ..... Factor (e.g., nominal) #
# Group F1-1 #
# Group F1-2 #
#
# F2a.....Factor (e.g., nominal) #
# Group F2a-1 #
# Group F2a-2 #
#
# F2b.....Factor (e.g., nominal) #
# Group F2b-1 Group F2b-2 Group F2b-3 #
# Group F2b-4 Group F2b-5 #
#####
#####
```

5.11.1 Data with Normal Distribution Patterns

Guided Direction: Produce a set of descriptive statistics for the measured object variable M1 by the factor-type object variable F2b.

R Input

```
RcmdrMisc::numSummary(
  BioSpmen.df[,c("M1")], groups=as.factor(F2b),
  statistics=c("mean", "sd"))
# It is necessary to wrap the as.factor() function
# around the object variable F2b to be sure that it
# is treated as a factor-type object variable.
```

R Output

	mean	sd	data:n	data:NA
1	469.812	16.9806	2064	40
2	460.590	19.7288	650	20
3	461.475	17.7980	691	16
4	469.927	17.4978	600	9
5	466.994	18.1982	515	24

At this early stage of analysis it is not possible to even suggest which F2b breakout groups are in common and which F2b breakout groups are different in terms of the measured object variable M1. However, do these descriptive statistics offer a general sense of the data and possible trends? Complement these descriptive statistics by generating appropriate figures.

Guided Direction: Conduct a normality test (e.g., Anderson–Darling, Shapiro, etc.) against the measured datum M1, overall and by F2b breakout groups. Does the M1 measured object variable approximate normal distribution, overall and by F2b breakout groups? Complement the generated statistics by preparing QQ plots, density curves, etc.

R Input

```
RVAideMemoire::mshapiro.test(BioSpmen.df$M1)

RVAideMemoire::byf.shapiro(M1 ~ as.factor(F2b),
  data=BioSpmen.df)
```

R Output

```
Shapiro-Wilk normality tests
data: M1 by as.factor(F2b)
```

W	p-value
---	---------

```

1 0.9909 0.0000000004534 ***
2 0.9932      0.004699 **
3 0.9950      0.023723 *
4 0.9933      0.009068 **
5 0.9930      0.016647 *

---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Do QQ plots and density curve graphics complement the breakout group p-values? Will concerns about normality (if any) impact the selection of either a parametric approach to Oneway ANOVA or a nonparametric approach to Oneway ANOVA?

Guided Direction: Structure the syntax so that the agricolae::HSD.test() function is wrapped around the aov() function, with appropriate selections for a Oneway ANOVA of M1 by F2b. Do not forget that F2b is a number and therefore needs to be treated as a factor-type object variable, using the as.factor() function to achieve this aim.

R Input

```

agricolae::HSD.test(
  aov(M1 ~ as.factor(F2b), data=BioSpmen.df), # Model
  trt="as.factor(F2b)",                         # Treatment
  group=TRUE, console=TRUE, alpha=0.05,          # Arguments
  main="M1 by F2b")
# Wrap the agricolae::HSD.test() function around the
# Oneway ANOVA model obtained by using the aov()
# function. Select desired arguments, such as group,
# console, and alpha (e.g., p-value).

```

Do the results of the agricolae::HSD.test() function, when used correctly, generate a list of groups similar to below?

R Output

M1 groups	
4 469.927	a
1 469.812	a
5 466.994	b
3 461.475	c
2 460.590	c

How is the screen printout interpreted in terms of M1 means and F2b breakout groups that are in common (e.g., where there is no statistically significant difference at $p \leq 0.05$)? How is the screen printout interpreted in terms of M1 means and F2b breakout groups where there is a statistically significant difference ($p \leq 0.05$)?

5.11.2 Data That Do Not Exhibit Normal Distribution Patterns

Guided Direction: Is there any concern about the distribution pattern of the measured object variable M1 (as well as the other measured object variables, M2, M3a, M3b, and M3c, if the entire dataset were used for practice) and if so, is it necessary to approach the Oneway ANOVA from a nonparametric perspective?

R Input

```
install.packages("pastecs", dependencies=TRUE)
library(pastecs)                      # Load the pastecs package.
help(package=pastecs)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

pastecs::stat.desc(BioSpmen.df[,2:6],
  basic=TRUE, # Show basic descriptive statistics.
  desc=TRUE,  # Show more descriptive statistics.
  norm=TRUE,  # Show normal distribution statistics.
  p=0.95)    # Probability for the mean's Confidence Interval.
# In the BioSpmen.df dataframe, the measured object variables
# are in columns 2 to 6. Use [,2:6] to obtain an attractive
# side-by-side comparison of output for the object variables.
# Give special attention to the statistics for assessment of
# normal distribution, normtest.p -- the associated
# probability for the Shapiro-Wilk test of normality.
```

Just to be prudent even if normal distribution were not a concern, apply the dunn.test::dunn.test() function appropriately and generate a screen printout of pairwise comparisons, where breakout groups for F2b are compared to each other in view of M1.

R Input

```
dunn.test::dunn.test(BioSpmen.df$M1,
  as.factor(BioSpmen.df$F2b), method="bonferroni",
  kw=TRUE, label=TRUE, table=TRUE, list=TRUE)
# Produce a table of Kruskal-Wallis pairwise
# comparisons and p-values for each comparison.
```

R Output

```
List of pairwise comparisons: Z statistic (adjusted p-value)
-----
1 - 2 : 11.20599 (0.0000)*
1 - 3 : 10.20238 (0.0000)*
2 - 3 : -1.017781 (1.0000)
1 - 4 : -0.323570 (1.0000)
2 - 4 : -9.167774 (0.0000)*
3 - 4 : -8.304573 (0.0000)*
1 - 5 : 3.355152 (0.0040)*
2 - 5 : -5.742196 (0.0000)*
3 - 5 : -4.863713 (0.0000)*
4 - 5 : 3.001026 (0.0135)*
```

Look at the F2b breakout group by breakout group comparisons for M1 generated using the dunn.test::dunn.test() function. Compare this output to the F2b breakout group by breakout group comparisons for M1 generated using the agricolae::HSD.test() function. Are there differences in how groups are viewed, in common and by difference, for this dataset and selected R functions? Are Oneway ANOVA results from a parametric perspective the same as Oneway ANOVA results from a nonparametric perspective?

5.12 Prepare to Exit, Save, and Later Retrieve This R Session

R Input

```
getwd()           # Identify the current working directory.
ls()              # List all objects in the working
                  # directory.
ls.str()          # List all objects, with finite detail.
list.files()       # List files at the PC directory.

save.image("R_Lesson_OnewayANOVA.rdata")

getwd()           # Identify the current working directory.
ls()              # List all objects in the working
                  # directory.
ls.str()          # List all objects, with finite detail.
list.files()       # List files at the PC directory.
```

```
alarm()           # Alarm, notice of upcoming action.  
q()              # Quit this session.  
# Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: File -> Load Workspace. Otherwise, use the load() function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use the .R script file (typically saved as a .txt ASCII-type file) and recreate the analyses and graphics, provided the data files remain available.

5.13 External Data and/or Data Resources Used in This Lesson

The publisher's Web site associated with this text includes the file **BiologicalSpecimen.csv**, which was the only external file directly imported into this lesson. Use this file to practice and replicate the outcomes used in this lesson.

All other data in this lesson were enumerated (e.g., user-created) directly in the R session, using a variety of R-based functions. The user-created data in this lesson are **not** included at the publisher's Web site associated with this text and they are instead easily created in R by following the syntax presented in this lesson.



Chapter 6

Two-way Analysis of Variance (ANOVA)

Abstract

The purpose of this lesson on Two-way Analysis of Variance (ANOVA) is to provide guidance on how R can be used to see if two or more group means (e.g., Mean of High-Density Lipoprotein (HDL) Cholesterol for Gender breakouts (e.g., Female and Male) **and** Race-Ethnicity breakouts (e.g., Asian, Black, Hispanic, Indigenous, Unidentified, and White); Mean Degree of Marbling for Breed breakouts (e.g., Hereford, Limousin, and Simmental) **and** Feeding Program Breakouts (e.g., Pasture with Supplemental Grain and Pasture Only); Mean of Soybean Yield for Tillage Program (e.g., No-Till and Conventional) **and** Seasonal Rainfall (e.g., Dry, Normal, Wet), etc.) differ due to chance, or if observed differences are indeed the result of true difference and/or interactions between phenomena. Specifically, the Two-way ANOVA statistical test has been structured to examine differences and possible interactions when multiple variables have two or more separate categories. The degree of complexity supported by Two-way ANOVA begins to model real-

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_6) contains supplementary material, which is available to authorized users.

world concerns for the many ways variables, and possible interaction between and among variables, impact outcomes.

Keywords: ANOVA, Analysis of variance, Balanced design, Between-subjects factors, Dependent variable, Factorial ANOVA, Factorial design, Fisher (Ronald), Independent variable, Interaction, Mean comparison technique, Oneway ANOVA, Predictor variable, Twoway ANOVA, Unbalanced design, Within-subjects factors

6.1 Background

This lesson provides guidance on how the R environment supports Twoway Analysis of Variance (Twoway ANOVA), a more complex approach to research and biostatistics than is the case when using Oneway Analysis of Variance (Oneway ANOVA):

- The Oneway ANOVA test was used to determine if a statistically significant difference existed between a factor-type object variable (e.g., independent variable) organized into three or more groupings (e.g., independent variable; Dairy breeds: Guernsey, Holstein, Jersey, etc.) and a measured (e.g., dependent) variable (e.g., pounds of milk per lactation).
- The Twoway ANOVA expands on this research model, now bringing in the complexity of accounting for two or more factor-type object variables and their relationship (if any) with the measured variable. Following along with the prior example on milk production per lactation, consider a research design appropriate for Twoway ANOVA:
 - Factor-type object variable 1—Dairy breeds, with three breakouts (e.g., Guernsey, Holstein, and Jersey)
 - Factor-type object variable 2—Mineral Supplement, with two breakouts (e.g., Received a mineral supplement, Did not receive a mineral supplement)
 - Measured object variable—Pounds of milk per lactation

By using Twoway ANOVA it is possible to determine not only if there are statistically significant differences ($p \leq 0.05$) in group means but it is also possible to determine if there are statistically significant interactions ($p \leq 0.05$) when variables have two or more categories. When Twoway ANOVA is used, it is possible to determine:

- Do variables act independently of each other, causing difference?
- Are joint effects (e.g., interaction) a reason for difference?

The advantage of Twoway ANOVA is that it represents a more real-world research design on how measured variables such as pounds of milk production per lactation, measured teat-end callosity, live birth weight, etc., are impacted by far more than only one factor-type object variable. Following along with the prior example on milk production by dairy breed and by management practice (e.g., application of a mineral supplement), the Twoway ANOVA test would be a likely choice to determine if there is a statistically significant ($p \leq 0.05$) difference in pounds of milk production per lactation: (1) by dairy breed (e.g., Guernsey, Jersey, and Holstein), (2) by the management practice of supplying mineral supplements (e.g., mineral supplement was supplied, mineral supplement was not supplied), and (3) the possibility of interaction between dairy breed and management practice. Twoway ANOVA is one of the most commonly used inferential tests when multiple factors are examined against measured variables and mastery of this test is essential for those who regularly conduct research in the biological sciences. There are many different ways Twoway ANOVA supports the growing complexity of research designs for the biological sciences.

6.1.1 Description of the Data

There is one dataset used in this lesson and the data are organized in long (e.g., stacked) format. The data, in long format, generally follow the expected format used among researchers in biostatistics. Prior lessons in this text used data that were self-generated, often by using the `rnorm()` and `runif()` functions. The data for this lesson are entered into this session by wrapping the `read.table()` function around the `textConnection()` function. That is to say, the data are entered directly into the R session associated with this lesson:

- There are 80 subjects in the dataset. If there had been many more subjects it would have been best to organize the data in a .csv (Comma-Separated Values) text file and import the saved .csv file into the R session. However, the `textConnection()` function was purposely used to demonstrate how this tool has value, especially for simple datasets that are not too large.
- The data in this lesson model actual data typically encountered in biostatistics. Yet, the eventual dataframe should be viewed as a teaching dataset designed to result in desired outcomes.

6.1.2 Null Hypothesis

There is one Null Hypothesis associated with this part of the lesson, with $p \leq 0.05$ the criterion level of significance.

Null Hypothesis (H_0): There is no statistically significant difference ($p \leq 0.05$) and there is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure between and among subjects based on Gender (Female and Male), Drug (Drug A, Drug B, Drug C, Drug D, and Placebo), and Race-Ethnicity (Black, Hispanic, Other, and White).

Although it may be readily evident in the above Null Hypothesis, it is still prudent to recapitulate that the data used in this demonstration of Twoway ANOVA are focused on group differences and possible interactions in Systolic Blood Pressure (SBP) by:

- Gender: Female and Male
- Drug: Drug A, Drug B, Drug C, Drug D, and Placebo
- Race-Ethnic: Black, Hispanic, Other, and White

6.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

Instead of importing a .csv file, the data in this lesson are self-generated, using R-based tools. Although prior lessons used theoretical datasets self-generated by using the rnorm() function and the runif() function, for this lesson the data are entered directly into this R session. To create the dataframe, the read.table() function is wrapped around the textConnection() function. The data start with a header row to name the object variables and data are organized in a fixed-column format.

R Input

```
#####
# Housekeeping                                         Use for All Analyses #
#####
date()          # Current system time and date.
Sys.time()       # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls(all.names=TRUE) # List all objects in the working
                   # directory, including hidden files.
rm(list=ls())     # CAUTION: Remove all files in the
                   # working directory. If this
                   # action is not desired, use rm()
                   # one-by-one to remove the objects
                   # that are not needed.
ls.str()         # List all objects, with finite detail.
getwd()          # Identify the current working directory.
setwd("F:/R_BiostatisticsIntroduction")
                   # Set to a new working directory.
                   # Note the single forward slash and double
```

```
# quotes.  
# This new directory should be the directory  
# where the data file is located, otherwise  
# the data file will not be found.  
getwd()          # Confirm the working directory.  
list.files()     # List files at the PC directory.  
.libPaths()      # Library pathname.  
.Library         # Library pathname.  
sessionInfo()    # R version, locale, and packages.  
search()          # Attached packages and objects.  
searchpaths()     # Attached packages and objects.  
#####
```

The purpose of this Housekeeping template is to be sure that the working environment, selected directory, etc., are as desired, with these key functions all in one convenient beginning section. In advance, it can be stated that because the data are entered directly into the R session, the base::set.seed() function, as seen in prior lessons, is not used in this part of the lesson.

R Input

```
SBPGenderDrugRaceEthnicLong.df <- read.table(textConnection("
```

PatientID	Gender	Drug	RaceEthnic	SBP
ID001	Female	1	Black	122
ID002	Female	1	Hispanic	124
ID003	Female	1	Other	118
ID004	Female	1	White	120
ID005	Female	1	Black	120
ID006	Female	1	Hispanic	122
ID007	Female	1	Other	118
ID008	Female	1	White	116
ID009	Female	2	Black	136
ID010	Female	2	Hispanic	138
ID011	Female	2	Other	128
ID012	Female	2	White	128
ID013	Female	2	Black	138
ID014	Female	2	Hispanic	138
ID015	Female	2	Other	124
ID016	Female	2	White	134
ID017	Female	3	Black	138
ID018	Female	3	Hispanic	118
ID019	Female	3	Other	124
ID020	Female	3	White	124
ID021	Female	3	Black	144

```
"))
```

ID022	Female	3	Hispanic	144
ID023	Female	3	Other	118
ID024	Female	3	White	122
ID025	Female	4	Black	142
ID026	Female	4	Hispanic	126
ID027	Female	4	Other	118
ID028	Female	4	White	134
ID029	Female	4	Black	124
ID030	Female	4	Hispanic	138
ID031	Female	4	Other	128
ID032	Female	4	White	128
ID033	Female	5	Black	128
ID034	Female	5	Hispanic	120
ID035	Female	5	Other	124
ID036	Female	5	White	124
ID037	Female	5	Black	120
ID038	Female	5	Hispanic	118
ID039	Female	5	Other	120
ID040	Female	5	White	122
ID041	Male	1	Black	137
ID042	Male	1	Hispanic	134
ID043	Male	1	Other	129
ID044	Male	1	White	126
ID045	Male	1	Black	126
ID046	Male	1	Hispanic	128
ID047	Male	1	Other	118
ID048	Male	1	White	120
ID049	Male	2	Black	141
ID050	Male	2	Hispanic	129
ID051	Male	2	Other	115
ID052	Male	2	White	124
ID053	Male	2	Black	144
ID054	Male	2	Hispanic	139
ID055	Male	2	Other	121
ID056	Male	2	White	121
ID057	Male	3	Black	153
ID058	Male	3	Hispanic	132
ID059	Male	3	Other	124
ID060	Male	3	White	131
ID061	Male	3	Black	137
ID062	Male	3	Hispanic	134
ID063	Male	3	Other	129
ID064	Male	3	White	130

ID065	Male	4	Black	136
ID066	Male	4	Hispanic	138
ID067	Male	4	Other	118
ID068	Male	4	White	124
ID069	Male	4	Black	131
ID070	Male	4	Hispanic	129
ID071	Male	4	Other	115
ID072	Male	4	White	126
ID073	Male	5	Black	144
ID074	Male	5	Hispanic	139
ID075	Male	5	Other	121
ID076	Male	5	White	121
ID077	Male	5	Black	153
ID078	Male	5	Hispanic	122
ID079	Male	5	Other	124
ID080	Male	5	White	130"), header=TRUE)

In this dataframe, text and not numeric codes are used for data relating to Gender (Female and Male) and RaceEthnic (Black, Hispanic, Other, and White). As a purposeful contrast to the way breakout groups can be identified, numeric codes have been used for data relating to Drug (1, 2, 3, 4, 5). Whole numbers are used to enter data for SBP.¹

R Input

```
colnames(SBPGenderDrugRaceEthnicLong.df) <-
  c("Patient", "Gender", "Drug", "RaceEthnic", "SBP")
# Although somewhat redundant, clearly identify the name
# for each column. Note how PatientID was changed to
# Patient, largely to reinforce how column names can be
# changed easily, if desired.
```

Use different R-based functions to put the data into proper format and then obtain a first view of the data.

¹Blood pressure readings, representing a measure of mmHg or millimeters of mercury, were once only gained by trained personnel using a manual sphygmomanometer and accompanying equipment, where due to concerns with accuracy the readings were set to even numbers only. Now however, with the use of digital devices that deploy automatic inflation of the cuff and multiple cuff sizes, blood pressure readings are now recorded with both even numbers and odd numbers, allowing for more accurate measurements.

R Input

```
attach(SBPGenderDrugRaceEthnicLong.df)      # Attach the dataframe
names(SBPGenderDrugRaceEthnicLong.df)        # Column names
head(SBPGenderDrugRaceEthnicLong.df, 5)       # First 5 rows
tail(SBPGenderDrugRaceEthnicLong.df, 5)        # Last 5 row
str(SBPGenderDrugRaceEthnicLong.df)           # Dataframe structure
```

R Output

[Selected output is not shown, to save space.]

```
'data.frame': 80 obs. of 5 variables:
 $ Patient : Factor w/ 80 levels "ID001","ID002",...: 1 2 3 4 ...
 $ Gender   : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 ...
 $ Drug     : int 1 1 1 1 1 1 1 1 2 2 ...
 $ RaceEthnic: Factor w/ 4 levels "Black","Hispanic",...: 1 2 3 ...
 $ SBP      : int 122 124 118 120 120 122 118 116 136 138 ...
```

R Input

```
summary(SBPGenderDrugRaceEthnicLong.df) # Summary statistics
```

R Output

Patient	Gender	Drug	RaceEthnic	SBP
ID001 : 1	Female:40	Min. :1	Black :20	Min. :115
ID002 : 1	Male :40	1st Qu.:2	Hispanic:20	1st Qu.:121
ID003 : 1		Median :3	Other :20	Median :126
ID004 : 1		Mean :3	White :20	Mean :128
ID005 : 1		3rd Qu.:4		3rd Qu.:134
ID006 : 1		Max. :5		Max. :153
(Other):74				

Look closely at the output of the str() function and see why the following data for Drug and SBP need to be put into another format and then observe how this is accomplished, using functions such as factor() and as.numeric(). Further, for the object variable Drug, apply a label so that numeric codes such as 1, 2, 3, 4, and 5 are instead presented as Drug A, Drug B, Drug C, Drug D, and Placebo:

- Transform the object variable Drug from integer to factor.
- Transform the object variable SBP from integer to numeric.

R Input

```
SBPGenderDrugRaceEthnicLong.df$Drug <-
  factor(SBPGenderDrugRaceEthnicLong.df$Drug,
  labels=c("Drug A", "Drug B", "Drug C", "Drug D",
  "Placebo"))

levels(SBPGenderDrugRaceEthnicLong.df$Drug)
```

R Output

```
[1] "Drug A"   "Drug B"   "Drug C"   "Drug D"   "Placebo"
```

R Input

```
summary(SBPGenderDrugRaceEthnicLong.df$Drug)
# Transform the object variable Drug from an
# integer-type object variable to a factor-type
# object variable and provide meaningful text to
# describe the nature of each numeric code.
```

R Output

Drug A	Drug B	Drug C	Drug D	Placebo
16	16	16	16	16

R Input

```
SBPGenderDrugRaceEthnicLong.df$SBP <-
  as.numeric(SBPGenderDrugRaceEthnicLong.df$SBP)
summary(SBPGenderDrugRaceEthnicLong.df$SBP)
# Transform the object variable SBP from an
# integer-type object variable to a numeric-type
# object variable.
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
115	121	126	128	134	153

R Input

```
attach(SBPGenderDrugRaceEthnicLong.df)      # Attach the dataframe
str(SBPGenderDrugRaceEthnicLong.df)          # Dataframe structure
names(SBPGenderDrugRaceEthnicLong.df)         # Column names
head(SBPGenderDrugRaceEthnicLong.df, 5)       # First 5 rows
tail(SBPGenderDrugRaceEthnicLong.df, 5)        # Last 5 rown
summary(SBPGenderDrugRaceEthnicLong.df)        # Summary statistics
```

R Output

Patient	Gender	Drug	RaceEthnic	SBP
ID001 : 1	Female:40	Drug A :16	Black :20	Min. :115
ID002 : 1	Male :40	Drug B :16	Hispanic:20	1st Qu.:121
ID003 : 1		Drug C :16	Other :20	Median :126
ID004 : 1		Drug D :16	White :20	Mean :128
ID005 : 1		Placebo:16		3rd Qu.:134
ID006 : 1				Max. :153
(Other):74				

Give special attention to the summary() function output for the object variable Drug. It is now quite different since Drug was transformed from a set of numeric codes to factor-type breakouts, using text to identify the different drugs used in this study.

6.3 Organize the Data and Display the Code Book

The dataframe of interest for the first part of this lesson was created entering data from 80 patients into the R session. The Code Book details the nature of the data.²

R Input

```
#####
# Code Book for SBPGenderDrugRaceEthnicLong.df #
#####
#
# Patient ..... Factor #
#             Patient IDs for 80 subjects, from #
#                           ID001 to ID080 #
```

²Observe how the 80 Systolic Blood Pressure measurements associated with this dataframe came from 80 unique subjects. Each subject was measured one time and only one time. Further, the factor-type object variables such as Gender, Drug, and RaceEthnic are balanced in terms of an equal N for each breakout group.

```
#                                     #
# Gender                         Factor #
#                                     Female and Male #
#                                     #
# Drug                          Factor #
#           Integer-based codes (1, 2, 3, 4, 5) #
#           transformed to descriptive text-based #
#           factor-type codes (Drug A, Drug B, #
#           Drug C, Drug D, and Placebo) #
#                                     #
# RaceEthnic                     Factor #
#           Black, Hispanic, Other, White #
#                                     #
# SBP                           Numeric #
#           SBP values may range from approximately #
#           110 to 160 mmHg and were transformed #
#           from integer to numeric #
#                                     #
#####
```

The data seem to be in good form, but before any inferential test is attempted, it is prudent to use R-based tools to create figures and develop descriptive statistics. Only then is it a good practice to engage in an inferential analysis such as a Twoway ANOVA.

6.4 Conduct a Visual Data Check Using Graphics (e.g., Figures)

A comparative advantage of R is that it can be used to prepare simple black, white, and shades of gray throwaway figures that provide a sense of the data, but would never be shared with others other than to view general trends. Yet, R can also be used to prepare figures that provide extreme detail using graphical functions from external packages such as ggplot2.

As a start to the type of figures that can be produced, use the `hist()` function and the `density()` function to visually represent the distribution of numeric values of Systolic Blood Pressure. Give attention to the line-by-line arguments and documentation for each argument (Fig. 6.1).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures - 1 row by 2 column grid
plot(density(SBPGenderDrugRaceEthnicLong.df$SBP),
      main="Systolic Blood Pressure - Overall",
      col="red",           # Add color
      lwd=3,               # Thick line
      font.lab=2,          # Bold labels
      xlab="SBP",           # X axis label
      xlim=c(100,160),     # X axis scale
      ylim=c(0,0.05))      # Y axis scale
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
hist(SBPGenderDrugRaceEthnicLong.df$SBP,
      main="Systolic Blood Pressure - Overall",
      xlab="SBP", breaks=20, col="red",
      xlim=c(100,160), ylim=c(0,12), font.lab=2,
      tck=-0.05)
axis(side=1, font=2) # X axis bold
axis(side=2, font=2) # Y axis bold
```

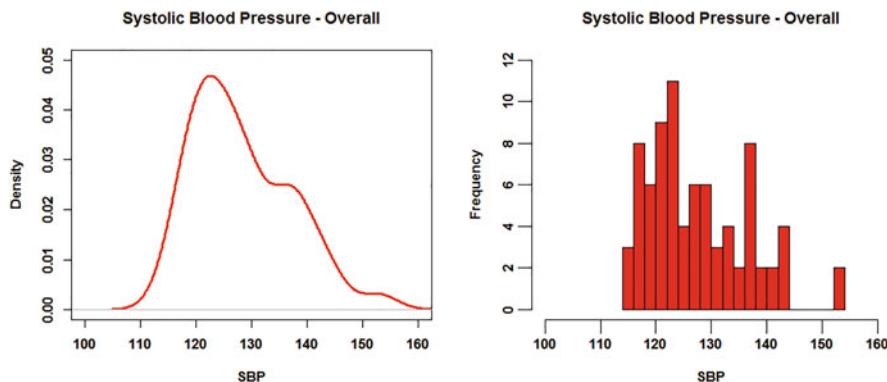


Figure 6.1: Distribution of systolic blood pressure overall

This simple set of figures, using the `density()` function and the `hist()` function, is quite valuable in that it brings to attention the distribution pattern of Systolic Blood Pressure values. This issue will be addressed in more detail later. For now, look at the way other R-based functions, functions from external packages such as `ggplot2`, can be used to examine the full set of factor-type object variables in relation to Systolic Blood Pressure associated with this lesson and how the figures can be presented as *Beautiful Graphics*, a term common to use of the `ggplot` package.

R Input

```
install.packages("ggplot2", dependencies=TRUE)
library(ggplot2)          # Load the ggplot2 package.
help(package=ggplot2)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("ggthemes", dependencies=TRUE)
library(ggthemes)          # Load the ggthemes package.
help(package=ggthemes)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("ggmosaic", dependencies=TRUE)
library(ggmosaic)          # Load the ggmosaic package.
help(package=ggmosaic)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("gridExtra", dependencies=TRUE)
library(gridExtra)          # Load the gridExtra package.
help(package=gridExtra)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("grid", dependencies=TRUE)
library(grid)               # Load the grid package.
help(package=grid)          # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)              # Load the scales package.
help(package=scales)        # Show the information page.
sessionInfo()             # Confirm all attached packages.
```

Detail on the many themes available with R and the ggplot2 package is presented throughout this set of lessons. For now, use the enumerated theme (e.g., theme_MacYates) shown immediately below. The emphasis of this enumerated theme is on large, bold, and vibrant display of outcomes—all to make it easier to see the figure and accompanying details. Recall that theme_MacYates() is a function with specific attributes, as defined immediately below (Fig. 6.2):

R Input

```
#####
theme_MacYates <- function(base_size=12, base_family="sans"){
```

```

theme(
  plot.title=element_text(face="bold", size=14, hjust=0),
  plot.caption=element_text(face="bold", size=8, hjust=0),
  axis.title.x=element_text(face="bold", size=14,
    hjust=0.5),
  axis.text.x=element_text(face="bold", size=12, angle=00,
    color="black"),
  axis.title.y=element_text(face="bold", size=12, vjust=1,
    angle=90),
  axis.text.y=element_text(face="bold", size=10, angle=00,
    color="black", hjust=1),
  legend.title=element_text(face="bold", size=12),
  legend.text=element_text(face="bold", size=12),
  axis.ticks.x=element_line(size=1.0),
  axis.ticks.y=element_line(size=1.0),
  axis.ticks.length=unit(0.20,"cm"),
  axis.line=element_line(color="black", size=1.5,
    linetype = "solid"),
  panel.background=element_rect(fill="whitesmoke")
)
}

# hjust - horizontal justification; 0 = left edge to 1 = right
# edge, with 0.5 the default
# vjust - vertical justification; 0 = bottom edge to 1 = top
# edge, with 0.5 the default
# angle - rotation; generally 0 to 90 degrees, with 0 the
# default
#####
#####
```

R Input

```

# Systolic Blood Pressure (Violin Plot) measurements, overall

SBPViolinOverall <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
  aes(x=SBP, y=SBP)) +
  # SBP has been used for both the x and y aesthetic
  # (e.g., aes). Note below how the coordinates are
  # flipped, thus the reason for what may otherwise seem
  # to be a mistake for X axis and Y axis placement.
  geom_violin(fill="red", color="black",
  lwd=2) +                               # Violin Plot geom
  ggttitle(
  "Systolic Blood Pressure, Overall\n") +
  scale_x_discrete(name="") +
```

```
scale_y_continuous(name="Systolic Blood Pressure\n",
  limits=c(110,155), breaks=seq(110,155,20)) +
theme_MacYates() +
coord_flip()      # Rotate (e.g., flip) the presentation

# Systolic Blood Pressure (Violin Plot) measurements by
# singular group breakouts: Gender, Drug, Race-Ethnicity

SBPViolinGender <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
  aes(x=Gender, y=SBP, group=Gender)) +
  geom_violin(aes(fill=Gender)) +           # Violin Plot geom
  ggtitle(
  "Systolic Blood Pressure by Gender\n") +
  scale_x_discrete(name="") +
  scale_y_continuous(name="Systolic Blood Pressure\n",
    limits=c(110,155), breaks=seq(110,155,20)) +
  # The coordinates are flipped, thus the reason for what
  # seems to be a mistake for the X axis and Y axis.
  theme_MacYates() +
  theme(legend.position="none") +
  coord_flip()      # Rotate (e.g., flip) the presentation

SBPViolinDrug <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
  aes(x=Drug, y=SBP, group=Drug)) +
  geom_violin(aes(fill=Drug)) +           # Violin Plot geom
  ggtitle(
  "Systolic Blood Pressure by Drug\n") +
  scale_x_discrete(name="") +
  scale_y_continuous(name="Systolic Blood Pressure\n",
    limits=c(110,155), breaks=seq(110,155,20)) +
  theme_MacYates() +
  theme(legend.position="none") +
  coord_flip()      # Rotate (e.g., flip) the presentation

SBPViolinRaceEthnic <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
  aes(x=RaceEthnic, y=SBP, group=RaceEthnic)) +
  geom_violin(aes(fill=RaceEthnic)) +       # Violin Plot geom
  ggtitle(
  "Systolic Blood Pressure by Race-Ethnicity\n") +
  scale_x_discrete(name="") +
```

```

scale_y_continuous(name="Systolic Blood Pressure\n",
  limits=c(110,155), breaks=seq(110,155,20)) +
theme_MacYates() +
theme(legend.position="none") +
coord_flip()      # Rotate (e.g., flip) the presentation

par(ask=TRUE)
gridExtra::grid.arrange(
  SBPViolinOverall,
  SBPViolinGender,
  SBPViolinDrug,
  SBPViolinRaceEthnic, ncol=2)

```

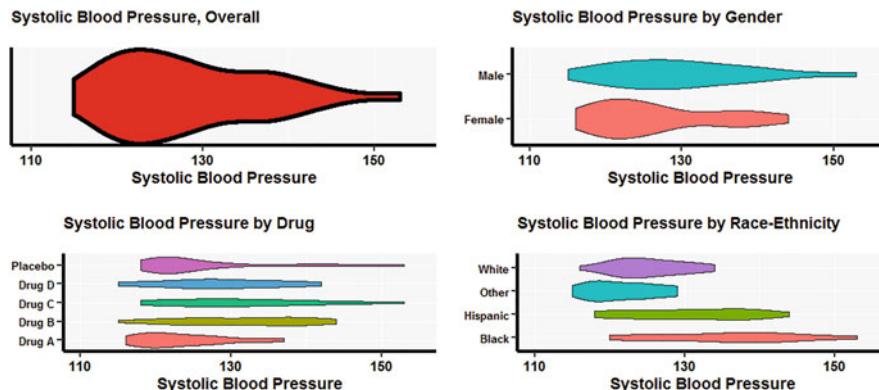


Figure 6.2: Distribution of systolic blood pressure overall and by breakouts—1

R Input

```

# Systolic Blood Pressure (Violin Plot) measurements by
# two-group breakouts: Gender and by Drug, Gender and by
# Race-Ethnicity, and Drug and by Race-Ethnicity

SBPViolinGenderDrug <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
  aes(x=Gender, y=SBP, group=Gender)) +
  geom_violin(aes(fill=Gender)) +           # Violin Plot geom
  facet_grid(. ~ Drug) +
  ggtitle(
  "Systolic Blood Pressure by Gender and by Drug\n") +
  scale_x_discrete(name "") +
  scale_y_continuous(name="Systolic Blood Pressure\n",

```

```
limits=c(110,155), breaks=seq(110,155,20)) +
theme_MacYates() +
theme(axis.text.x=element_text(face="bold", size=10,
angle=00, color="black")) +
# Change X axis text to size=10 to improve presentation
theme(legend.position="none") +
coord_flip()      # Rotate (e.g., flip) the presentation

SBPViolinGenderRaceEthnic <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
aes(x=Gender, y=SBP, group=Gender)) +
geom_violin(aes(fill=Gender)) +           # Violin Plot geom
facet_grid(. ~ RaceEthnic) +
ggttitle(
"Systolic Blood Pressure by Gender and by
Race-Ethnicity\n") +
scale_x_discrete(name "") +
scale_y_continuous(name="Systolic Blood Pressure\n",
limits=c(110,155), breaks=seq(110,155,20)) +
theme_MacYates() +
theme(axis.text.x=element_text(face="bold", size=10,
angle=00, color="black")) +
# Change X axis text to size=10 to improve presentation
theme(legend.position="none") +
coord_flip()      # Rotate (e.g., flip) the presentation

SBPViolinDrugRaceEthnic <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
aes(x=Drug, y=SBP, group=Drug)) +
geom_violin(aes(fill=Drug)) +           # Violin Plot geom
facet_grid(. ~ RaceEthnic) +
ggttitle(
"Systolic Blood Pressure by Drug and by
Race-Ethnicity\n") +
scale_x_discrete(name "") +
scale_y_continuous(name="Systolic Blood Pressure\n",
limits=c(110,155), breaks=seq(110,155,20)) +
theme_MacYates() +
theme(axis.text.x=element_text(face="bold", size=10,
angle=00, color="black")) +
# Change X axis text to size=10 to improve presentation
theme(legend.position="none") +
coord_flip()      # Rotate (e.g., flip) the presentation

par(ask=TRUE)
gridExtra::grid.arrange(
```

```
SBPViolinOverall,
SBPViolinGenderDrug,
SBPViolinGenderRaceEthnic,
SBPViolinDrugRaceEthnic, ncol=2)
```

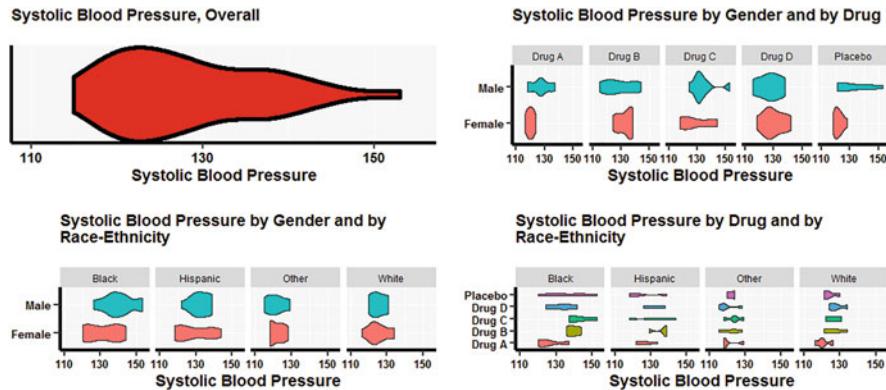


Figure 6.3: Distribution of systolic blood pressure overall and by breakouts—2

R Input

```
# Systolic Blood Pressure (Dot Plot) measurements at the
# overall level of Gender and by Race-Ethnicity and by Drug and
# All

par(ask=TRUE)
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
  aes(x=Gender, y=SBP, group=Gender, fill="darkred",
  color="black")) +
  geom_point(shape=21, position="jitter") +           # Point geom
  facet_grid(RaceEthnic ~ Drug, margins=TRUE) +
  # facet_grid() using margins=TRUE provides Total (all)
  labs(title =
    "\nSystolic Blood Pressure by Gender and by Drug and by
    Race-Ethnicity and All\n",
    x="\nGender\n", y="\nSystolic Blood Pressure\n") +
  scale_y_continuous(name="Systolic Blood Pressure\n",
    limits=c(110,155), breaks=seq(110,155,20)) +
  theme_MacYates() +
  theme(axis.text.y=element_text(face="bold", size=10,
    angle=00, color="black")) +
  # Change Y axis text to size=10 to improve presentation
  theme(legend.position="none")
```

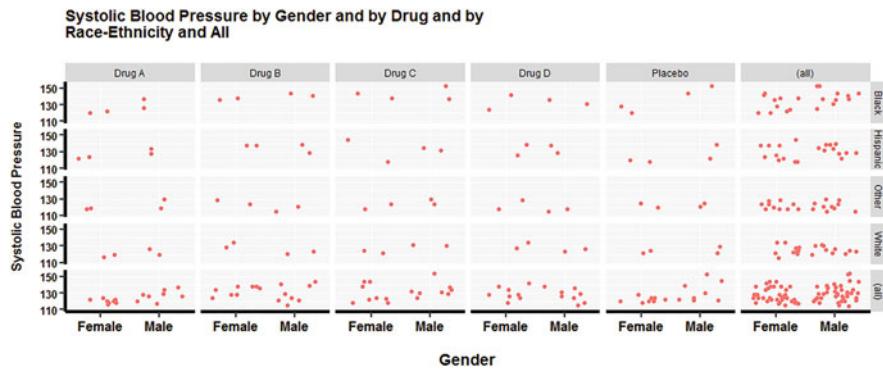


Figure 6.4: Distribution of systolic blood pressure overall and by breakouts—3

Although the `ggplot2::ggplot()` function, with its many geoms and supporting capabilities, is quite popular in the desire for *Beautiful Graphics*, the `epiDisplay` package should also be considered when examining data from a visual perspective. A particular advantage of the `epiDisplay` package is that some functions provide not only graphical output but they also provide a complementary set of descriptive statistics printed to the screen in text format, suitable for copy and paste into a word-processed document if desired (Figs. 6.3 and 6.4).

R Input

```
install.packages("epiDisplay")
library(epiDisplay)           # Load the epiDisplay package.
help(package=epiDisplay)      # Show the information page.
sessionInfo()                 # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(2,2))  # 4 figures - 2 row by 2 column grid
with(SBPGenderDrugRaceEthnicLong.df,
     epiDisplay::summ(SBP))
with(SBPGenderDrugRaceEthnicLong.df,
     epiDisplay::summ(SBP, by=Gender))
with(SBPGenderDrugRaceEthnicLong.df,
     epiDisplay::summ(SBP, by=Drug))
with(SBPGenderDrugRaceEthnicLong.df,
     epiDisplay::summ(SBP, by=RaceEthnic))
```

R Output

[Selected output is not shown, to save space.]

For RaceEthnic = Black

obs.	mean	median	s.d.	min.	max.
20	135.7	137	9.948	120	153

For RaceEthnic = Hispanic

obs.	mean	median	s.d.	min.	max.
20	130.5	130.5	8.01	118	144

For RaceEthnic = Other

obs.	mean	median	s.d.	min.	max.
20	121.7	121	4.52	115	129

For RaceEthnic = White

obs.	mean	median	s.d.	min.	max.
20	125.25	124	4.876	116	134

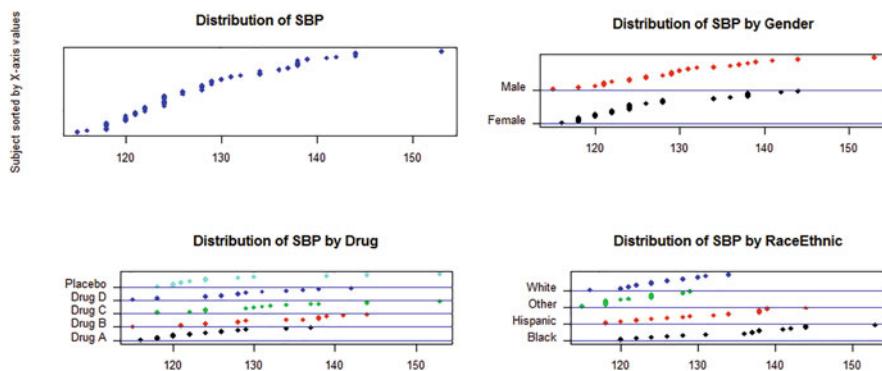


Figure 6.5: Distribution of systolic blood pressure overall and by breakouts—4

The epiDisplay::summ() function produces figures that are, at first, perhaps not as appealing as those produced using the ggplot2::ggplot() function, but notice the descriptive statistics printed to the screen. The addition of these descriptive statistics provides a useful complement to the generated figure and adds value to the use of functions from the epiDisplay package (Fig. 6.5).

Gain an additional sense of the data and the way data are organized in the data frame by listing selected data to the screen. The head() function and tail() function could be used, but another way can be used:

R Input

```
SBPGenderDrugRaceEthnicLong.df[1:8,] # first 8 observations
```

R Output

	Patient	Gender	Drug	RaceEthnic	SBP
1	ID001	Female	Drug A	Black	122
2	ID002	Female	Drug A	Hispanic	124
3	ID003	Female	Drug A	Other	118
4	ID004	Female	Drug A	White	120
5	ID005	Female	Drug A	Black	120
6	ID006	Female	Drug A	Hispanic	122
7	ID007	Female	Drug A	Other	118
8	ID008	Female	Drug A	White	116

Notice how Patient ID001 and Patient ID005 are two different individuals, but they otherwise share the same factor-type classifications in terms of Gender (both are Female), Drug (both received Drug A), and RaceEthnic (both are Black). Yet, Patient ID001 had a Systolic Blood Pressure of 122 and Patient ID005 had a Systolic Blood Pressure of 120. This type of research design is repeated for all subjects in that there are two unique subjects in each cell, with a cell representing a similar organizational structure of Gender, Drug, and RaceEthnic.

Because Twoway ANOVA involves data organized in some type of factorial design, it is helpful to graphically represent the idea of group membership in terms of the factor-type object variables, which for this study are Gender, Drug, and RaceEthnic. Use the `vcd::structable()` function to better understand the factorial design for this study on Systolic Blood Pressure by Gender, by Drug, and by RaceEthnic:

R Input

```
install.packages("vcd")
library(vcd)          # Load the vcd package.
help(package=vcd)    # Show the information page.
sessionInfo()        # Confirm all attached packages.
# Select the most local mirror site using Set CRAN mirror.

vcd::structable(RaceEthnic+Gender ~ Drug,
                SBPGenderDrugRaceEthnicLong.df)
```

R Output

Drug	Black	Hispanic	Other	White			
	Female	Male	Female	Male	Female	Male	Female
Drug A	2	2	2	2	2	2	2
Drug B	2	2	2	2	2	2	2
Drug C	2	2	2	2	2	2	2
Drug D	2	2	2	2	2	2	2
Placebo	2	2	2	2	2	2	2

Although there are multiple ways to organize output using the vcd::structable() function, all configurations for this study result in the same general outcome: there are two unique subjects in each cell and there are 40 cells, confirming that cell membership results in a total of 80 subjects.

To offer a graphical perspective on the factorial design associated with this dataset, use the ggplot2::ggplot() function and accompanying geom_mosaic geom. The geom_mosaic geom requires use of the ggmosaics package (Fig. 6.6).

R Input

```
ggplot2::ggplot(data=SBPGenderDrugRaceEthnicLong.df) +
  geom_mosaic(aes(x=product(RaceEthnic, Drug, Gender),
    fill=RaceEthnic), na.rm=TRUE, show.legend=FALSE) +
  labs(title =
  "Representation of Cells for a Factorial Design of
  Systolic Blood Pressure by Race-Ethnic and
  Gender and by Drug\n",
  caption="There are 2 patients for each cell and there are no
  missing data.\n",
  x="\nRace-Ethnic and Gender", y="\nDrug\n") +
  theme_MacYates() +
  theme(axis.text.x=element_text(size=10, angle=15,
    color="black", hjust=0.75))
  # Change X axis text to size=10 and put the angle at 15 to
  # improve presentation.
```

Many more graphics could be produced, but functions found in the external epiDisplay and ggplot2 packages are always a good first choice for the production of figures that provide a sense of the data. Of course, do not forget the value of the many graphically oriented functions available from when R is first downloaded, such as hist(), plot(density()), boxplot(), qqnorm() and qqline(), and dotchart() for measured object variables and barplot(table)) for factor-type object variables.

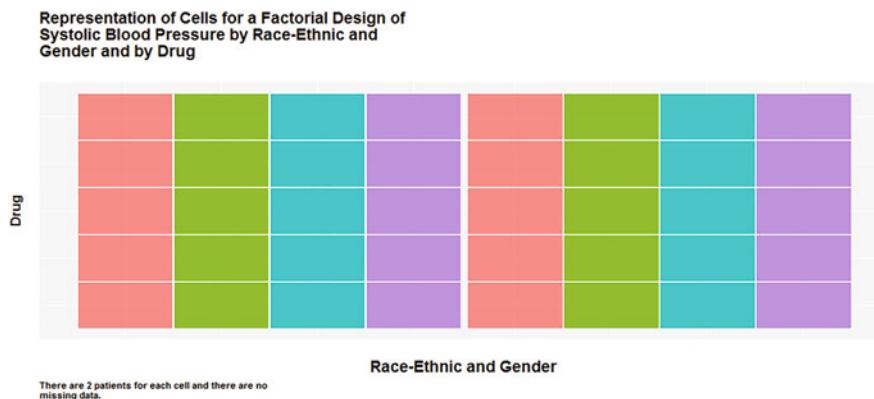


Figure 6.6: Factorial representation of systolic blood pressure overall and by breakouts

6.5 Descriptive Statistics for Initial Analysis of the Data

The many figures previously shown as well as the nearly countless number of figures that could have been produced provide a sense of the data, both for the factor-type object variables (e.g., Gender, Drug, RaceEthnic) as well as the numeric-type object variable (e.g., SBP). Given that Systolic Blood Pressure measurements may be impacted by the factor-type object variables, it is necessary to do far more than merely obtain descriptive statistics of Systolic Blood Pressure. Instead, it is necessary to obtain descriptive statistics of Systolic Blood Pressure by each of the different factor-type object variable breakouts.

First, use simple R-based functions such as the `summary()` function to reinforce a sense of Systolic Blood Pressure. However, there are many more R-based functions available for further diagnostics about measured object variables such as SBP and breakout descriptive statistics of the measured object variable by one or more factor-type object variables. The `RcmdrMisc::numSummary()` function is a good starting point for this type of inquiry.

R Input

```
install.packages("RcmdrMisc", dependencies=TRUE)
library(RcmdrMisc)           # Load the RcmdrMisc package.
help(package=RcmdrMisc)      # Show the information page.
sessionInfo()                # Confirm all attached packages.

RcmdrMisc::numSummary(SBPGenderDrugRaceEthnicLong.df$SBP)
```

R Output

```
mean      sd   IQR  0% 25% 50%  75% 100% n
128.287 8.85065 13.5 115 121 126 134.5 153 80
```

R Input

```
RcmdrMisc::numSummary(SBPGenderDrugRaceEthnicLong.df[
, c("SBP")], groups=Gender, statistics=c("mean", "sd"))
```

R Output

	mean	sd	data:n
Female	126.750	8.17360	40
Male	129.825	9.32844	40

R Input

```
RcmdrMisc::numSummary(SBPGenderDrugRaceEthnicLong.df[
, c("SBP")], groups=Drug, statistics=c("mean", "sd"))
```

R Output

	mean	sd	data:n
Drug A	123.625	6.02080	16
Drug B	131.125	8.52350	16
Drug C	131.375	9.95239	16
Drug D	128.438	7.79717	16
Placebo	126.875	9.98582	16

R Input

```
RcmdrMisc::numSummary(SBPGenderDrugRaceEthnicLong.df[
, c("SBP")], groups=RaceEthnic, statistics=c("mean", "sd"))
```

R Output

	mean	sd	data:n
Black	135.70	9.94776	20
Hispanic	130.50	8.00986	20
Other	121.70	4.52013	20

White	125.25	4.87610	20
-------	--------	---------	----

In the next set of figures, notice how a simple boxplot has been generated to further reinforce SBP descriptive statistics by breakouts of each factor-type object variable as well as overall. These throwaway boxplots are simple in presentation but they still have great value, reinforcing the many descriptive statistics associated with SBP by breakouts of the three factor-type object variables, Gender, Drug, and RaceEthnic (Fig. 6.7).

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures - 2 row by 2 column grid
boxplot(SBPGenderDrugRaceEthnicLong.df$SBP,
        main="SBP -- All Patients", xlab="", ylab="")
boxplot(SBPGenderDrugRaceEthnicLong.df$SBP ~
        SBPGenderDrugRaceEthnicLong.df$Gender,
        main="SBP by Gender", xlab="", ylab="")
boxplot(SBPGenderDrugRaceEthnicLong.df$SBP ~
        SBPGenderDrugRaceEthnicLong.df$Drug,
        main="SBP by Drug", xlab="", ylab="")
boxplot(SBPGenderDrugRaceEthnicLong.df$RaceEthnic,
        main="SBP by RaceEthnic", xlab="", ylab="")
```

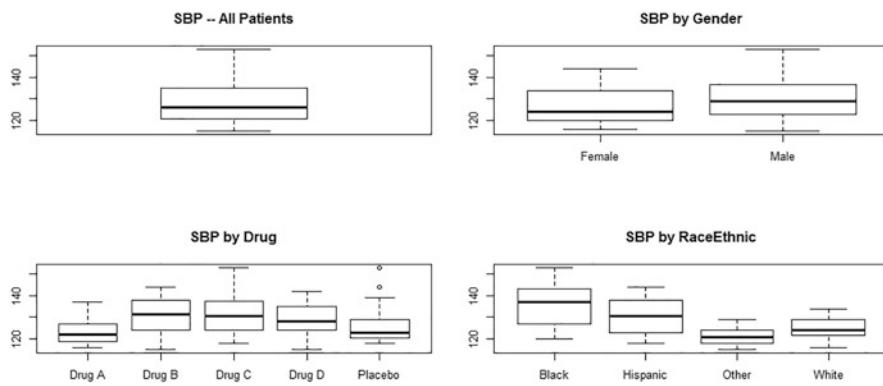


Figure 6.7: Distribution of systolic blood pressure overall and by breakouts—5

These singular attempts at presenting figures and preparing descriptive statistics for the measured object variable (e.g., SBP) by breakouts of the three different factor-type object variables (e.g., Gender, Drug, RaceEthnic) are certainly useful and make an attractive summary that can be easily copied from the R session and pasted into a word-processed document. However, consider how the

epiDisplay::statStack() function also provides descriptive statistics (specifically, mean and sd), but in a more concise presentation and for all declared factor-type object variables. Observe how iqr (inter-quartile range) arguments of the epiDisplay::statStack() function can differentiate between Mean (iqr="NULL") and Median (iqr="auto"), which may be helpful later on if data distribution patterns call for a nonparametric approach to statistical analysis and not only a parametric approach.³

R Input

```
epiDisplay::statStack(SBP, by=c(Gender, Drug, RaceEthnic),
  dataFrame=SBPGenderDrugRaceEthnicLong.df, decimal=2,
  iqr="NULL")      # Generate Mean and Standard Deviation
```

R Output

	Total	SBP	mean (SD)
Gender			
Female	40	126.75	(8.17)
Male	40	129.82	(9.33)
Drug			
Drug A	16	123.62	(6.02)
Drug B	16	131.12	(8.52)
Drug C	16	131.38	(9.95)
Drug D	16	128.44	(7.8)
Placebo	16	126.88	(9.99)
RaceEthnic			
Black	20	135.7	(9.95)
Hispanic	20	130.5	(8.01)
Other	20	121.7	(4.52)
White	20	125.25	(4.88)

For now, ignore the screen printout section detailing Test and p-value, which was purposely deleted in the screen printout for this lesson. This information is important, but it focuses on a singular approach at statistical analysis (e.g., Student's t-Test for Independent Samples and Oneway ANOVA), whereas this lesson is focused on a more encompassing factorial design using Twoway ANOVA.

³As a very broad statement, recall that parametric inferential tests tend to focus on the mean and nonparametric inferential tests tend to focus on the median.

R Input

```
epiDisplay::statStack(SBP, by=c(Gender, Drug, RaceEthnic),
  dataFrame=SBPGenderDrugRaceEthnicLong.df, decimal=2,
  iqr="auto")      # Generate Median and Inter-Quartile Range
```

R Output

	Total	SBP	median(IQR)
Gender			
Female	40	124	(120,134)
Male	40	129	(123.5,136.25)
Drug			
Drug A	16	123.62	(6.02)
Drug B	16	131.12	(8.52)
Drug C	16	131.38	(9.95)
Drug D	16	128.44	(7.8)
Placebo	16	126.88	(9.99)
RaceEthnic			
Black	20	137	(127.5,142.5)
Hispanic	20	130.5	(123.5,138)
Other	20	121	(118,124)
White	20	124	(121.75,128.5)

There are many R-based functions that could be used on this dataset to generate descriptive statistics and measures of central tendency, as demonstrated throughout this set of lessons. However, the RcmdrMisc::numSummary() function and the epiDisplay::statStack() function are especially desirable in that they provide detailed descriptive statistics for the measured object variable by breakouts of declared factor-type object variables. These two functions are also helpful in that they put the many statistics in an attractive format that can be easily copied from the R session and pasted into a word-processed document, either in original format or formatted by placing some type of L^AT_EXwrapper around the text (e.g., The Hmisc::latex() function is often used to transform a R-generated table-like object into a format suitable for a L^AT_EXdocument.)

6.6 Quality Assurance, Data Distribution, and Tests for Normality

Quality assurance (QA) is by no means a simple activity where one and only one attempt is made to examine the data for quality issues, distribution patterns,

and examination of normality. From pre-planning to ending summary, quality assurance needs to be inclusive throughout the entire research process. To meet the immediate needs for quality assurance, a great deal of attention has gone into the production of figures that offer a graphical sense of the data. Descriptive statistics have been explored from many perspectives. Summative descriptive statistics were provided for Systolic Blood Pressure, overall and for breakouts of all three factor-type object variables (e.g., Gender, Drug, RaceEthnic).

For this lesson the Shapiro Test for Normality and the stats::shapiro.test() function will be used to address inquiries into normal distribution patterns—overall. The RVAideMemoire::byf.shapiro() function will be used to address normal distribution of Systolic Blood Pressure by the three factor-type object variables (e.g., Gender, Drug, RaceEthnic).⁴

It is common to focus on either a p-value of either $p \leq 0.05$ or $p \leq 0.01$ when considering normality tests, such as the Shapiro Test for Normality. Remember that due to the way the Shapiro Test for Normality is worded, in the affirmative:

- Lower calculated p-values provide evidence against the null hypothesis. That is to say, if the calculated p-value is less than or equal to the criterion p-value (e.g., significance level), the rules-based decision is to reject the null hypothesis and to declare that the data do not follow a pattern of normal distribution.
- Higher calculated p-values fail to provide evidence against the null hypothesis. That is to say, if the calculated p-value is greater than the criterion p-value (e.g., significance level), the rules-based decision is to accept (e.g., fail to reject) the null hypothesis and to declare that the data seem to follow a pattern of normal distribution.

R Input

```
shapiro.test(SBPGenderDrugRaceEthnicLong.df$SBP)
```

R Output

```
Shapiro-Wilk normality test
p-value = 0.000741
```

⁴For nearly all statistical tests, the Null Hypothesis is worded in a negative fashion and is typically stated as *There is no statistically significant difference between A and B in terms of C*. Somewhat different, the Null Hypothesis for a Shapiro Test of Normality is instead worded in the affirmative and is typically stated as *The data follow a normal distribution*. Give attention to this different approach to the way the Shapiro Test for Normality is worded when interpreting the p-values, significance levels, and outcomes from a Shapiro Test for Normality.

The calculated p-value equals 0.000741, which is less than the criterion p-value of 0.05, when the Shapiro test is applied against the numeric object variable SBPGenderDrugRaceEthnicLong.df\$SBP. This finding forces the concern that normal distribution was not evident for Systolic Blood Pressure—overall.

Although Systolic Blood Pressure may not exhibit normal distribution at the overall level, the issue of normal distribution must also be addressed at the different breakout levels for each factor-type object variables (e.g., Gender, Drug, RaceEthnic). Perhaps Systolic Blood Pressure follows normal distribution for some of these breakouts, but that possibility currently remains unknown until tested. The RVAideMemoire::byf.shapiro() function is an excellent choice to examine these issues since this function provides a convenient summary of normality p-values by individual breakouts of each factor-type object variable.

R Input

```
install.packages("RVAideMemoire", dependencies=TRUE)
library(RVAideMemoire)          # Load the RVAideMemoire package.
help(package=RVAideMemoire)    # Show the information page.
sessionInfo()                  # Confirm all attached packages.

# Normality test of SBP by Gender

RVAideMemoire::byf.shapiro(SBP ~ Gender,
                           data=SBPGenderDrugRaceEthnicLong.df)
```

R Output

Shapiro-Wilk normality tests		
	W	p-value
Female	0.8879	0.0008664 ***
Male	0.9564	0.1257107

Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1	

Based on the RVAideMemoire::byf.shapiro() function, normal distribution is not met for Female patients (calculated p of 0.0008664 compared to criterion p <= 0.05). At the p <= 0.05 level of significance it can be said that normal distribution is met for Male patients, with the calculated p-value at 0.1257107.

R Input

```
# Normality test of SBP by Drug

RVAideMemoire::byf.shapiro(SBP ~ Drug,
  data=SBPGenderDrugRaceEthnicLong.df)
```

R Output

```
Shapiro-Wilk normality tests
      W   p-value
Drug A 0.9149 0.1395181
Drug B 0.9455 0.4221139
Drug C 0.9513 0.5104679
Drug D 0.9689 0.8208433
Placebo 0.7635 0.0009271 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Based on the RVAideMemoire::byf.shapiro() function, normal distribution is not met for patients who used the Placebo (calculated p of 0.0009271 compared to criterion p ≤ 0.05). At the p ≤ 0.05 level of significance it can be said that normal distribution is met for patients who used Drug A, Drug B, Drug C, and Drug D, with p-values for those four Drug breakouts exceeding p ≤ 0.05 .

R Input

```
# Normality test of SBP by RaceEthnic

RVAideMemoire::byf.shapiro(SBP ~ RaceEthnic,
  data=SBPGenderDrugRaceEthnicLong.df)
```

R Output

```
Shapiro-Wilk normality tests
      W   p-value
Black    0.9461 0.31196
Hispanic 0.9329 0.17541
Other    0.9095 0.06243 .
White    0.9639 0.62454
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Based on the RVAideMemoire::byf.shapiro() function, normal distribution is met at the $p \leq 0.05$ level of significance for all four RaceEthnic breakout groups. For all four RaceEthnic breakout groups, the calculated p-value exceeded $p \leq 0.05$. Do not be confused by the notation about significance at the $p \leq 0.10$ level of significance for the Other breakout group. This condition ($p \leq 0.10$, observed for RaceEthnic equals Other) is greater than the criterion level of significance associated with the Null Hypothesis ($p \leq 0.05$) in this lesson.⁵

A graphic of QQ plots may help better understand these findings. Notice how four separate figures generated by using the ggplot2::ggplot() function are placed into one common figure (Fig. 6.8).

R Input

```
QQSBPbyOverall <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
aes(sample=SBP)) +
  stat_qq(color="red") +
  stat_qq_line(color="black", size=1.75) +
  ggtitle(""
Systolic Blood Pressure (QQ-Plot and QQ-Line) --
Overall") +
  labs(
    x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
  theme_MacYates()

QQSBPbyGender <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
aes(sample=SBP)) +
  stat_qq(color="red") +
  stat_qq_line(color="black", size=1.75) +
  facet_grid(. ~ Gender) +
  ggtitle(""
Systolic Blood Pressure (QQ-Plot and QQ-Line)
by Gender") +
  labs(
    x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
  theme_MacYates()

QQSBPbyDrug <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
```

⁵Give attention to the Signif. codes and specifically ‘.’ 0.1, with a . character showing the significance level for Other.

```

aes(sample=SBP)) +
  stat_qq(color="red") +
  stat_qq_line(color="black", size=1.75) +
  facet_grid(. ~ Drug) +
  ggtitle("Systolic Blood Pressure (QQ-Plot and QQ-Line)
by Drug") +
  labs(
    x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
  theme_MacYates()

QQSBPbyRaceEthnic <-
ggplot2::ggplot(SBPGenderDrugRaceEthnicLong.df,
  aes(sample=SBP)) +
  stat_qq(color="red") +
  stat_qq_line(color="black", size=1.75) +
  facet_grid(. ~ RaceEthnic) +
  ggtitle("Systolic Blood Pressure (QQ-Plot and QQ-Line)
by RaceEthnic") +
  labs(
    x = "\nTheoretical", y = "Systolic Blood Pressure\n") +
  theme_MacYates()

par(ask=TRUE)
gridExtra::grid.arrange(
  QQSBPbyOverall,
  QQSBPbyGender,
  QQSBPbyDrug,
  QQSBPbyRaceEthnic, ncol=2)
# Conduct a Quality Assurance check of normality to confirm
# output from the RVAideMemoire::byf.shapiro() function.

```

Quality Assurance has been attempted throughout this lesson, by generating graphical images of the relationship between and among data and also be providing descriptive statistics and measures of central tendency between and among data. Normality has been addressed by examining distribution patterns of Systolic Blood Pressure, overall and by individual breakouts of the factor-type object variables (e.g., Gender, Drug, RaceEthnic).

Overall, it was observed that Systolic Blood Pressure measures do not follow a pattern of normal distribution, using $p \leq 0.05$ for the criterion value of acceptance or rejection of the associated Null Hypothesis on normality. However,

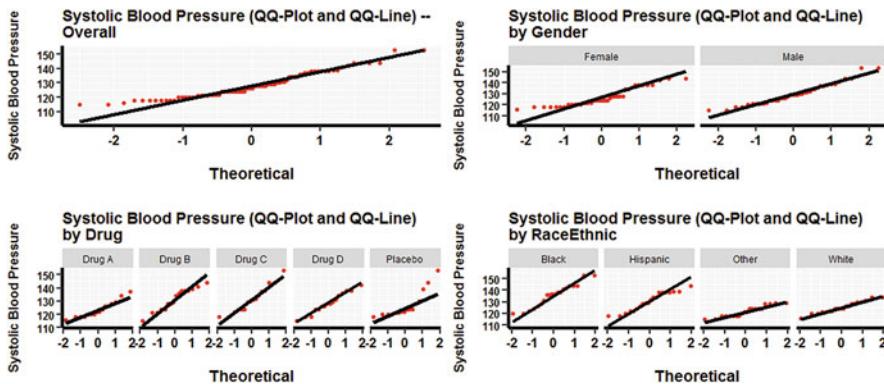


Figure 6.8: Distribution of systolic blood pressure overall and by breakouts—6

at the level of individual breakouts for the factor-type object variables normal distribution ($p \leq 0.05$) was usually evident:

Gender ($p \leq 0.05$)

Female Normal distribution was not evident
Male Normal distribution was evident

Drug ($p \leq 0.05$)

Drug A Normal distribution was evident
Drug B Normal distribution was evident
Drug C Normal distribution was evident
Drug D Normal distribution was evident
Placebo ... Normal distribution was not evident

RaceEthnic ($p \leq 0.05$)

Black Normal distribution was evident
Hispanic .. Normal distribution was evident
Other Normal distribution was evident
White Normal distribution was evident

Go back to the start of this lesson and the statements on how the data in this lesson model actual data typically encountered in biostatistics, yet the data for this lesson represent a dataframe that should be viewed as a teaching dataset designed to result in desired outcomes. The way data were constructed and normal distribution for Systolic Blood Pressure is addressed in this lesson was purposely structured to present an issue that demands an informed and thoughtful decision.

Two-way ANOVA is an inferential test that is ideally dependent on normal distribution. With the preponderance of factor-type object variable breakouts showing normal distribution for Systolic Blood Pressure, it is judged acceptable to use Two-way ANOVA for this study, to determine if there are group (e.g.,

Gender, Drug, RaceEthnic) differences in Mean SBP and to also address the possibility that there may be interactions between and among the group object variables (e.g., Gender, Drug, RaceEthnic) and breakouts of these variables in terms of SBP. However, in an abundance of caution, the data will also be examined from a nonparametric perspective at the end of this lesson, largely to confirm if outcomes from application of the nonparametric approach (e.g., Friedman Twoway Analysis of Variance by Ranks) are in parity with the parametric approach (e.g., Twoway ANOVA) outcome.

As a reminder, do not confuse any finding of normality by breakout groups with statistical significance by breakout groups. It is not at all unlikely that a factor-type object variable with five breakout groups will have normal distribution for all breakouts but that there will (or will not) be statistically significant differences between those same breakout groups in terms of the measured variable in question. Tests of statistical significance are needed to make such an inference.

6.7 Statistical Test(s)

As a brief reminder, the focus of this lesson is an examination of possible differences and/or interactions in mean Systolic Blood Pressure (SBP) by three separate factor-type object variables: Gender, Drug, and RaceEthnic. By using the Twoway ANOVA inferential test, it will be possible to determine not only if there are statistically significant ($p \leq 0.05$) differences in mean SBP by breakouts of Gender, Drug, RaceEthnic, but it is also possible to determine if there are any statistically significant ($p \leq 0.05$) interactions between and among Gender, Drug, and RaceEthnic in terms of SBP. This application of Twoway ANOVA takes on a more real-world approach to the complex way that different biological constructs impact other phenomena.

Although it may be redundant, once again look at the descriptive statistics and measures of central tendency for Systolic Blood Pressure by Gender, Drug, and RaceEthnic. When observing differences in means by different breakouts, avoid the temptation to assume that a large difference immediately suggests statistical difference. Statistically significant differences in this context are only confirmed by using the appropriate inferential test(s) against a declared (e.g., *a priori*) criterion level of significance, often $p \leq 0.05$. Equally important, potential interactions between and among the object variables are not at all addressed when viewing descriptive statistics and measures of central tendency at the singular level of comparison.

R Input

```
install.packages("s20x", dependencies=TRUE)
library(s20x) # Load the s20x package.
```

```
help(package=s20x)          # Show the information page.  
sessionInfo()               # Confirm all attached packages.  
  
with(SBPGenderDrugRaceEthnicLong.df,  
  s20x::summaryStats(SBP))  
# Show descriptive statistics, overall.
```

R Output

```
Minimum value:           115  
Maximum value:          153  
Mean value:             128.29  
Median:                 126  
Upper quartile:         134.5  
Lower quartile:         121  
Variance:               78.33  
Standard deviation:     8.85  
Midspread (IQR):        13.5  
Skewness:                0.7  
Number of data values:   80
```

R Input

```
with(SBPGenderDrugRaceEthnicLong.df,  
  s20x::summaryStats(SBP ~ Gender))  
# Show descriptive statistics, by breakout groups.
```

R Output

	Sample Size	Mean	Median	Std Dev	Midspread
Female	40	126.750	124	8.17360	14.00
Male	40	129.825	129	9.32844	12.75

R Input

```
with(SBPGenderDrugRaceEthnicLong.df,  
  s20x::summaryStats(SBP ~ Drug))  
# Show descriptive statistics, by breakout groups.
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
Drug A	16	123.625	122.0	6.02080	7.00	
Drug B	16	131.125	131.5	8.52350	14.00	
Drug C	16	131.375	130.5	9.95239	13.25	
Drug D	16	128.438	128.0	7.79717	10.50	
Placebo	16	126.875	123.0	9.98582	7.75	

R Input

```
with(SBPGenderDrugRaceEthnicLong.df,
  s20x::summaryStats(SBP ~ RaceEthnic))
# Show descriptive statistics, by breakout groups.
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
Black	20	135.70	137.0	9.94776	15.00	
Hispanic	20	130.50	130.5	8.00986	14.50	
Other	20	121.70	121.0	4.52013	6.00	
White	20	125.25	124.0	4.87610	6.75	

From among the many different ways a Twoway ANOVA can be implemented, it may be easiest to merely use the `aov()` function, which is included in the stats package. For convenience, wrap the `summary()` function around the `aov()` function to put all syntax into one simple approach.

R Input

```
summary(aov(SBP ~ Gender * Drug * RaceEthnic,
  data=SBPGenderDrugRaceEthnicLong.df))
# Twoway ANOVA of SBP by Gender, Drug, and
# RaceEthnic
```

R Output

[Selected output is not shown, to save space.]

	Df	Mean Sq	F value	Pr(>F)
Gender	1	189	5.30	0.0267 *
Drug	4	165	4.63	0.0036 **
RaceEthnic	3	750	20.99	0.000000025 ***

Gender:Drug	4	144	4.03	0.0077 **
Gender:RaceEthnic	3	97	2.71	0.0580 .
Drug:RaceEthnic	12	43	1.19	0.3215
Gender:Drug:RaceEthnic	12	24	0.66	0.7760
Residuals	40	36		
<hr/>				
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1				

Immediately, note the following conclusions from this concise summary of the Twoway ANOVA of SBP by Gender, Drug, and RaceEthnic, with all inferences based on $p \leq 0.05$:

- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Gender.
- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Drug.
- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by RaceEthnic.
- There is a statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender:Drug—Gender and Drug.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender:RaceEthnic—Gender and RaceEthnic.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Drug:RaceEthnic—Drug and RaceEthnic.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender:Drug:RaceEthnic—Gender, Drug, and RaceEthnic.

Use of the `summary(aov())` functions provides sufficient information to make a first attempt at interpreting Twoway ANOVA results. As a quality assurance attempt that is more than prudent, use the `car:::Anova()` function to generate, ideally, the same Twoway ANOVA statistics. Although presentation of the output may be slightly different due to the way different functions are organized and the use of rounding, output gained from use of the `summary(aov())` functions should be in close parity, if not identical, to what is generated by using the `car:::Anova()` function.

R Input

```
install.packages("car")
library(car)           # Load the car package.
help(package=car)      # Show the information page.
sessionInfo()          # Confirm all attached packages.

car:::Anova(aov(SBP ~ Gender * Drug * RaceEthnic,
                data=SBPGenderDrugRaceEthnicLong.df))
```

R Output

Anova Table (Type II tests)

Response: SBP

	Sum Sq	Df	F value	Pr(>F)
Gender	189.1	1	5.295	0.02667 *
Drug	661.4	4	4.630	0.00363 **
RaceEthnic	2249.2	3	20.994	0.0000000249 ***
Gender:Drug	575.4	4	4.028	0.00774 **
Gender:RaceEthnic	289.9	3	2.706	0.05803 .
Drug:RaceEthnic	511.0	12	1.192	0.32152
Gender:Drug:RaceEthnic	283.7	12	0.662	0.77600
Residuals	1428.5	40		
<hr/>				

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

There may (or may not) be a slight difference in rounding and the number of places to the right of the decimal, but the Twoway ANOVA table generated by using the car:::Anova() function is nearly equivalent to the Twoway ANOVA table generated by using the summary(aov()) functions. Both approaches provide the same general set of outcomes:

- There is a statistically significant difference ($p \leq 0.05$) in mean Systolic Blood Pressure by Gender, Drug, and RaceEthnic breakout groups. However, this finding does not provide finite information on which breakout groups are in common and which breakout groups are different regarding Systolic Blood Pressure.
- There is a statistically significant interaction ($p \leq 0.05$) in mean Systolic Blood Pressure between Gender and Drug. However, it is unclear just what this interaction means in terms of actionable use of this information.

It was determined that for each of the three factor-type object variables (e.g., Gender, Drug, RaceEthnic) there was a statistically significant ($p \leq 0.05$)

difference between the associated breakouts and SBP. To examine these differences closely, consider how the agricolae package has many functions that may be of value for those who need even more detailed information about examination of these singular comparisons when conducting a Twoway ANOVA.

For both convenience and better understanding of later outcomes, create an enumerated object that represents a Twoway ANOVA approach to the data:

R Input

```
TwowayGDRE <-
  aov(SBP ~ Gender * Drug * RaceEthnic,
  data=SBPGenderDrugRaceEthnicLong.df)
# Twoway ANOVA for G(ender), D(rug),
# and R(ace)E(thnic) -- TwowayGDRE
```

Wrap the summary() function around the enumerated object TwowayGDRE to confirm that all is correct:

R Input

```
summary(TwowayGDRE) # Generate a Twoway ANOVA table
```

R Output

[Selected output is not shown, to save space.]

	Df	Mean Sq	F value	Pr(>F)
Gender	1	189	5.30	0.0267 *
Drug	4	165	4.63	0.0036 **
RaceEthnic	3	750	20.99	0.000000025 ***
Gender:Drug	4	144	4.03	0.0077 **
Gender:RaceEthnic	3	97	2.71	0.0580 .
Drug:RaceEthnic	12	43	1.19	0.3215
Gender:Drug:RaceEthnic	12	24	0.66	0.7760
Residuals	40	36		
<hr/>				

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Then, apply the TukeyHSD() function to the TwowayGDRE enumerated object to obtain calculated p-values for all of the pairings—all pairings! It would take page after page to include the output from these more than 1000 pairings so in the interest of closure this output is not included, but it should still be attempted to see the large volume of information included.

R Input

```
TukeyHSD(TwowayGDRE) # Caution: Reams of output
```

Once these actions are completed, look at the three factor-type object variables in this lesson and consider how the breakouts for each compare in terms of mean SBP. As an interesting approach to this aim, use the `agricolae::HSD.test()` function to more closely examine where there are differences for mean Systolic Blood Pressure by Gender, Drug, and RaceEthnic breakout groups. The results of this function are slightly edited to save space from an otherwise far more verbose set of statistics.

R Input

```
install.packages("agricolae")
library(agricolae)           # Load the agricolae package.
help(package=agricolae)       # Show the information page.
sessionInfo()                 # Confirm all attached packages.

agricolae::HSD.test(TwowayGDRE, "Gender", group=TRUE,
                     console=TRUE)
# Use Tukey's HSD (Honestly Significant Difference) Mean
# Comparison Test to determine which breakouts are in common
# groups, and which breakout groups are not.
```

R Output

```
Study: TwowayGDRE ~ "Gender"
HSD Test for SBP

      SBP      std   r Min Max
Female 126.750 8.17360 40 116 144
Male   129.825 9.32844 40 115 153

      SBP groups
Male   129.825     a
Female 126.750     b
```

R Input

```
agricolae::HSD.test(TwowayGDRE, "Drug", group=TRUE,
                     console=TRUE)
# Use Tukey's HSD (Honestly Significant Difference) Mean
```

```
# Comparison Test to determine which breakouts are in common
# groups, and which breakout groups are not.
```

R Output

Study: TwowayGDRE ~ "Drug"

HSD Test for SBP

	SBP	std	r	Min	Max
Drug A	123.625	6.02080	16	116	137
Drug B	131.125	8.52350	16	115	144
Drug C	131.375	9.95239	16	118	153
Drug D	128.438	7.79717	16	115	142
Placebo	126.875	9.98582	16	118	153

	SBP groups
Drug C	131.375 a
Drug B	131.125 a
Drug D	128.438 ab
Placebo	126.875 ab
Drug A	123.625 b

R Input

```
agricolae::HSD.test(TwowayGDRE, "RaceEthnic", group=TRUE,
                     console=TRUE)
# Use Tukey's HSD (Honestly Significant Difference) Mean
# Comparison Test to determine which breakouts are in common
# groups, and which breakout groups are not.
```

R Output

Study: TwowayGDRE ~ "RaceEthnic"

HSD Test for SBP

	SBP	std	r	Min	Max
Black	135.70	9.94776	20	120	153
Hispanic	130.50	8.00986	20	118	144
Other	121.70	4.52013	20	115	129
White	125.25	4.87610	20	116	134

	SBP groups
--	------------

Black	135.70	a
Hispanic	130.50	b
White	125.25	c
Other	121.70	c

There is no denying that it takes time to study, learn, and correctly interpret the output from these statistics, knowing how to discern where means are in common, where means are different, and where there are degrees of overlap in means. Fortunately, it is possible to wrap the `plot()` function around the `agricolae::HSD.test()` function to more easily make sense of the data by factor-type object variable breakouts (Fig. 6.9).

R Input

```
savelwd      <- par(lwd=4)           # Heavy line
par(ask=TRUE)      # Pause
par(mfrow=c(1,3)) # 3 figures - 1 row by 3 column grid
plot(agricolae::HSD.test(TwowayGDRE,"Gender",
  alpha=0.05, group=TRUE),
  main="Common Subgoups of SBP by Gender Breakouts")
plot(agricolae::HSD.test(TwowayGDRE,"Drug",
  alpha=0.05, group=TRUE),
  main="Common Subgoups of SBP by Drug Breakouts")
plot(agricolae::HSD.test(TwowayGDRE,
  "RaceEthnic", alpha=0.05, group=TRUE),
  main="Common Subgoups of SBP by RaceEthnic Breakouts")
par(savelwd)
# Plot Tukey's HSD (Honestly Significant Difference) to
# determine which breakouts are in common groups, and which
# are not.
# Notice the X axis and how breakout groups are ordered, not
# by alpha ordering but instead by mean.
```

The graphical display of differences by factor-type object variables makes it fairly easy to understand where there are common breakout group means for Systolic Blood Pressure and where there are statistically significant ($p \leq 0.05$) differences in mean Systolic Blood Pressure by breakout groups. Going along with this approach, a graphical image is often the best way to prepare for an understanding of interaction, which was statistically significant ($p \leq 0.05$) for Gender and Drug (e.g., Gender:Drug calculated $p \leq 0.00774$) (Fig. 6.10).

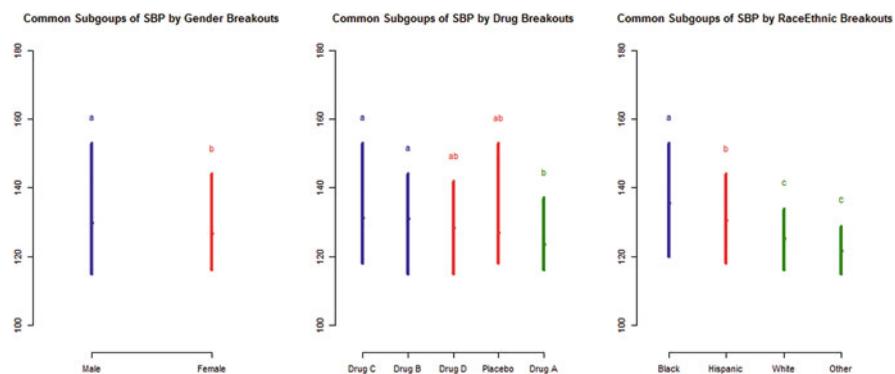


Figure 6.9: Distribution of systolic blood pressure by breakouts

R Input

```
par(ask=TRUE)
ggplot2::ggplot() +
  aes(x=Gender, color=Drug, group=Drug, y=SBP) +
  stat_summary(fun=mean, geom="point") +
  stat_summary(fun=mean, geom="line", size=1.25) +
  ggtitle(
    "Systolic Blood Pressure Means by Gender and Drug") +
  scale_x_discrete(name="Gender") +
  scale_y_continuous(name="Systolic Blood Pressure\n") +
  # Prepare a plot of Systolic Blood Pressure means by Gender
  # and by Drug. These two variables were the only pairing
  # where there was a statistically significant (p <= 0.05)
  # interaction.
  theme_MacYates()
```

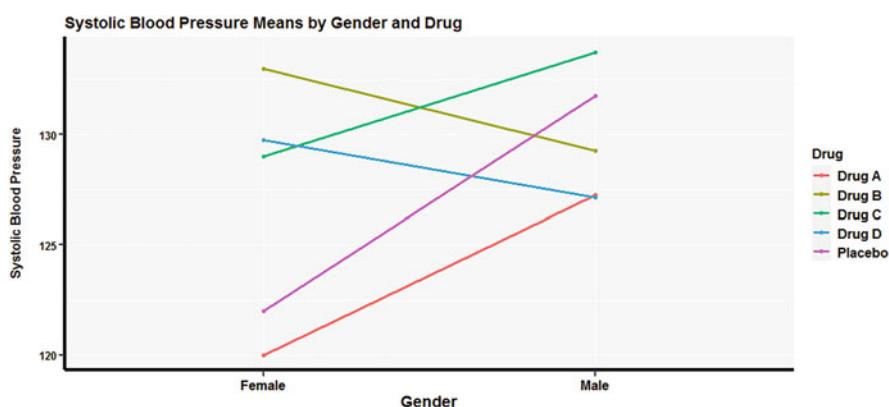


Figure 6.10: Interaction of gender and drug for systolic blood pressure—1

The differences at the singular level of comparison are usually easy to understand, such as Males had significantly ($p \leq 0.05$) higher levels of mean Systolic Blood Pressure than their Female counterparts.⁶ Interactions, however, can be more problematic to understand. For this lesson, it was observed that there is a statistically significant ($p \leq 0.05$) interaction between Gender and Drug regarding Systolic Blood Pressure. As is often the case, additional figures on interaction may improve understanding of this finding (Figs. 6.11 and 6.12).

R Input

```
savelwd      <- par(lwd=4)           # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab <- par(cex.lab=1.25)    # Label
savecex.axis <- par(cex.axis=1.25)   # Axis
par(ask=TRUE)
interaction.plot(SBPGenderDrugRaceEthnicLong.df$Drug,
                  SBPGenderDrugRaceEthnicLong.df$Gender,    # Note the order
                  SBPGenderDrugRaceEthnicLong.df$SBP,      # of the variables.
                  main="Interaction: Gender, Drug, and SBP",
                  fun=mean,                      # Use mean instead of median.
                  legend=TRUE, trace.label="Gender", fixed=TRUE,
                  col=c("red", "blue"), lwd=4, lty=c("solid", "dashed"),
                  xlab="Drug",
                  ylab="Systolic Blood Pressure", font.lab=2,
                  ylim=c(120,135), xtick=TRUE)
par(savelwd); par(savefont); par(savecex.lab)
par(savecex.axis)                      # Return to original setting.
```

R Input

```
savelwd      <- par(lwd=4)           # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab <- par(cex.lab=1.25)    # Label
savecex.axis <- par(cex.axis=1.25)   # Axis
par(ask=TRUE)
interaction.plot(SBPGenderDrugRaceEthnicLong.df$Gender,
                  SBPGenderDrugRaceEthnicLong.df$Drug,    # Note the order
                  SBPGenderDrugRaceEthnicLong.df$SBP,      # of the variables.
```

⁶Give attention to the means of Systolic Blood Pressure for the two genders by different drugs. This interaction plot of Systolic Blood Pressure by Gender and by Drug is an excellent example of why this type of identifying demographic information is essential in the biological sciences. It cannot be assumed that treatments are impartial to biological differences, including differences such as gender.

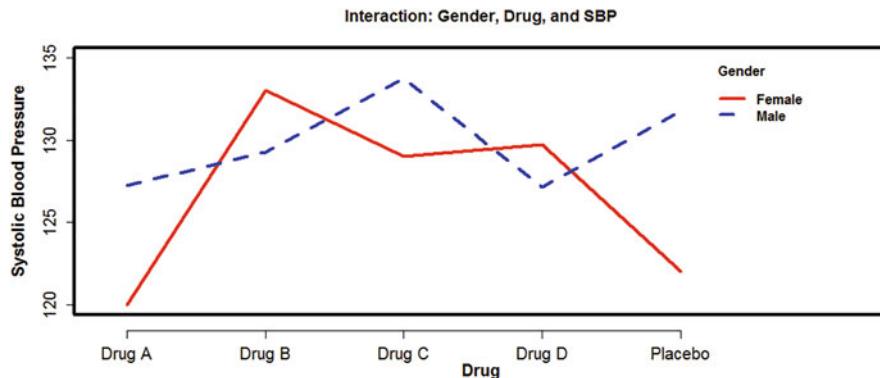


Figure 6.11: Interaction of gender and drug for systolic blood pressure—2

```
main="Interaction: Drug, Gender, and SBP",
fun=mean,                      # Use mean instead of median.
legend=TRUE, trace.label="Drug", fixed=TRUE,
col=c("red", "blue", "black", "green", "orange"),
lwd=4, lty=c("solid", "dashed", "dotted", "dotdash",
"longdash"),
xlab="Gender",
ylab="Systolic Blood Pressure", font.lab=2,
ylim=c(120,135), xtick=TRUE)
par(savelwd); par(savefont); par(savecex.lab)
par(savecex.axis)               # Return to original setting.
```

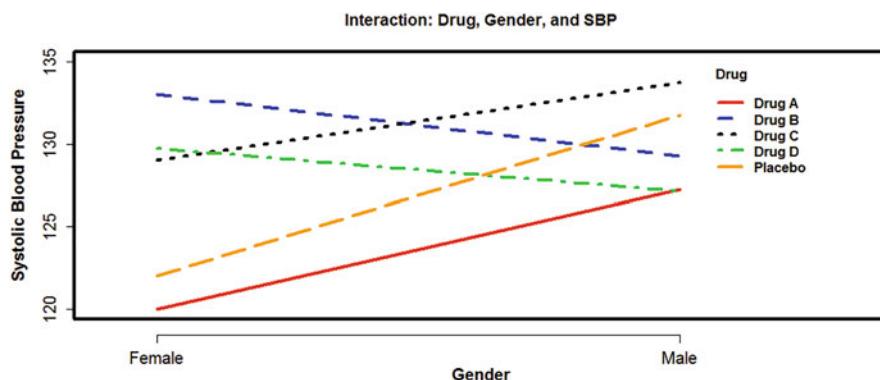


Figure 6.12: Interaction of gender and drug for systolic blood pressure—3

6.8 Summary of Outcomes

This lesson provided an opportunity to apply the inferential test Twoway Analysis of Variance (Twoway ANOVA) against data relating to Systolic Blood Pressure (SBP) and three factor-type object variables: Gender, Drug, RaceEthnic.

Recall that for this lesson Twoway ANOVA is not only used to look at statistically significant ($p \leq 0.05$) differences in mean SBP by breakout groups for the three factor-type object variables Gender, Drug, and RaceEthnic, but Twoway ANOVA is also used to determine if there are statistically significant ($p \leq 0.05$) interactions between and among the factor-type object variables (Gender, Drug, and RaceEthnic) and their relationship with the measured object variable, SBP in this lesson.

To further understand the many outcomes from this study, review the previously generated descriptive statistics to best understand the outcomes of the Twoway ANOVA:

R Input

```
epiDisplay::statStack(SBP, by=c(Gender, Drug, RaceEthnic),
  dataFrame=SBPGenderDrugRaceEthnicLong.df, decimal=2,
  iqr=NULL)      # Generate Mean and Standard Deviation
```

R Output

	Total	SBP	mean (SD)
Gender			
Female	40	126.75	(8.17)
Male	40	129.82	(9.33)
Drug			
Drug A	16	123.62	(6.02)
Drug B	16	131.12	(8.52)
Drug C	16	131.38	(9.95)
Drug D	16	128.44	(7.8)
Placebo	16	126.88	(9.99)
RaceEthnic			
Black	20	135.7	(9.95)
Hispanic	20	130.5	(8.01)
Other	20	121.7	(4.52)
White	20	125.25	(4.88)

Although the descriptive statistics provide a hint of outcomes, once again consider how Twoway ANOVA, affected by application of the agricolae::HSD.test() function, provided definitive results on common and unique memberships between and among the three breakout groups regarding Systolic Blood Pressure:

- SBP by Gender: There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Gender:

#	SBP groups
# Male	129.825 a
# Female	126.750 b

- SBP by Drug: There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Drug:

#	SBP groups
# Drug C	131.375 a
# Drug B	131.125 a
# Drug D	128.438 ab
# Placebo	126.875 ab
# Drug A	123.625 b

- SBP by RaceEthnic: There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by RaceEthnic:

#	SBP groups
# Black	135.70 a
# Hispanic	130.50 b
# White	125.25 c
# Other	121.70 c

- SBP by Gender and Drug: There is a statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender and Drug.
- SBP by Gender and RaceEthnic: There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender and RaceEthnic.
- SBP by Drug and RaceEthnic: There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Drug and RaceEthnic.
- SBP by Gender, Drug, and RaceEthnic: There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender, Drug, and RaceEthnic.

Because there was some concern about normality and data distribution patterns for some breakouts (e.g., Gender—Female and Drug—Placebo), a nonparametric approach will be used in Addendum 2 for Twoway ANOVA. This upcoming approach should provide closure to inquiries into statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Gender, Drug, and RaceEthnic. The goal, of course, is to determine if there is parity in outcomes between a parametric or nonparametric approach toward data analysis.

At the broadest level of discussion, it can be said that there are statistically significant differences ($p \leq 0.05$) in Systolic Blood Pressure by Gender, by

Drug, and by RaceEthnic. It can also be stated that there is a statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender and Drug. All other differences and all other interactions, if any, are the result of chance and they are not the result of statistical significance ($p \leq 0.05$).

Going farther than this broad summary of Twoway ANOVA, there have also been analyses of singular differences by breakout groups of the three factor-type object variables. These outcomes of mean SBP differences are identified in detail and a graphic was also provided that visually reinforced these outcomes, using standard ANOVA notation to distinguish between common breakout groups, unique breakout groups, and breakout groups where there is some degree of overlap.

6.9 Addendum 1: Other Packages for Display of Twoway ANOVA

Functions from the external car package and the external agricolae package were used to provide additional support for a Twoway ANOVA inquiry into the data associated with this lesson. The WRS2 package also has potential value for Twoway ANOVA and its use should be considered, especially in that the WRS2::t3way() function can easily accommodate trimmed means for an analysis of variance of one measured object variable (e.g., SBP) and three separate factor-type object variables (e.g., Gender, Drug, RaceEthnic).⁷

R Input

```
install.packages("WRS2")
library(WRS2)                      # Load the WRS2 package.
help(package=WRS2)                  # Show the information page.
sessionInfo()                      # Confirm all attached packages.

WRS2::t3way(SBP ~ Gender * Drug * RaceEthnic,
            data=SBPGenderDrugRaceEthnicLong.df,
            tr=0.1)                         # Trim level of the mean
```

⁷There are times when extreme values in a dataset have an impact on outcomes. Functions that take into consideration these extreme values, by using a trimmed means procedure, represent another attempt at Quality Assurance. There is no mandate that analyses must be based on adjustments to the dataset by trimming means or some other process. Yet, it is valuable to consider the impact of extreme values and to then make an informed judgment on whether extreme values should be used or in some manner excluded or adjusted for impact. The literature and discussions with colleagues should guide any discussion to use adjusted data.

R Output

	value	p.value
Gender	5.29541	0.041
Drug	29.56882	0.008
RaceEthnic	67.81761	0.001
Gender:Drug	24.65813	0.016
Gender:RaceEthnic	8.49844	0.120
Drug:RaceEthnic	16.45837	0.702
Gender:Drug:RaceEthnic	8.77819	0.942

Although the accommodation for trimmed means, using the WRS2::t3way() function, produced somewhat different p-values, the net result remains the same. At $p \leq 0.05$, it is confirmed that:

- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Gender.
- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Drug.
- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by RaceEthnic.
- There is a statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender and Drug.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender and RaceEthnic.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Drug and RaceEthnic.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender, Drug, and RaceEthnic.

Quality Assurance is a continuous process. Use of the WRS2::t3way() function provided another tool in the attempt to apply appropriate statistical analyses against the data, now taking into account the possible impact of extreme values. The net result, however, was that the prior inferences gained from use of the summary(aov()), car::Anova() function, and agricolae::HSD.test() functions are confirmed, even with trimmed extreme values considered.

6.10 Addendum 2: Parametric v Nonparametric

Overall, there was a concern that Systolic Blood Pressure did not exhibit normal distribution. Saying that, it would have been convenient to immediately

take a nonparametric approach to Twoway ANOVA. However, due to breakout analyses of normality, it was found that Systolic Blood Pressure did indeed exhibit normal distribution in most cases, when viewed from the perspective of breakout groups for the three factor-type object variables:

Gender ($p \leq 0.05$)

Female Normal distribution was not evident
 Male Normal distribution was evident

Drug ($p \leq 0.05$)

Drug A Normal distribution was evident
 Drug B Normal distribution was evident
 Drug C Normal distribution was evident
 Drug D Normal distribution was evident
 Placebo ... Normal distribution was not evident

RaceEthnic ($p \leq 0.05$)

Black Normal distribution was evident
 Hispanic .. Normal distribution was evident
 Other Normal distribution was evident
 White Normal distribution was evident

Accordingly, the parametric approach to Twoway ANOVA demonstrated throughout this lesson is in good form and does not need to be repeated. In an abundance of caution, however, approach the Twoway ANOVA from a nonparametric view. Compare outcomes from analyses from the prior Twoway ANOVA to the Twoway ANOVA table generated by using the Rfit::raov() function, a function that is used when data are viewed from a nonparametric perspective. Using $p \leq 0.05$ as the criterion metric for statistical significance, compare the similarities and differences from when data are viewed from a parametric perspective and for when data are viewed from a nonparametric perspective.

R Input

```
install.packages("Rfit")
library(Rfit)          # Load the Rfit package.
help(package=Rfit)      # Show the information page.
sessionInfo()          # Confirm all attached packages.

Rfit::raov(SBP ~ Gender + Drug + RaceEthnic,
            data=SBPGenderDrugRaceEthnicLong.df)
# This is a nonparametric approach to Twoway ANOVA.
```

R Output

Robust ANOVA Table

	DF	RD	Mean RD	F	p-value
Gender	1	10.9808	10.98077	3.14378	0.08383
Drug	4	67.3662	16.84156	4.82172	0.00287
RaceEthnic	3	202.0003	67.33345	19.27748	0.00000
Gender:Drug	4	52.8994	13.22485	3.78626	0.01056
Gender:RaceEthnic	3	23.7336	7.91119	2.26496	0.09567
Drug:RaceEthnic	12	43.1387	3.59489	1.02921	0.44219
Gender:Drug:RaceEthnic	12	23.9006	1.99172	0.57023	0.85260

From this different perspective toward the data, offered again as another Quality Assurance action due to possible concerns about deviation away from normal distribution, the Rfit::raov() function provided evidence that:

- There is **no** statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Gender.
- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by Drug.
- There is a statistically significant difference ($p \leq 0.05$) in Systolic Blood Pressure by RaceEthnic.
- There is a statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender and Drug.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender and RaceEthnic.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Drug and RaceEthnic.
- There is no statistically significant interaction ($p \leq 0.05$) in Systolic Blood Pressure by Gender, Drug, and RaceEthnic.

Compare outcomes from a parametric view of the data using the summary(aov()) functions to a nonparametric view of the data using the Rfit::raov() function. The only discrepancy in whether there is statistically significant ($p \leq 0.05$) difference between the two approaches toward the data is the general outcome for SBP and Gender:

R Input

```
RcmdrMisc::numSummary(SBPGenderDrugRaceEthnicLong.df [
, c("SBP")], groups=Gender, statistics=c("mean", "quantiles"))
```

R Output

	mean	0%	25%	50%	75%	100%	data:n
Female	126.750	116	120.0	124	134.00	144	40
Male	129.825	115	123.5	129	136.25	153	40

Parametric Gender calculated $p \leq 0.02667$ resulting in the conclusion that there **is** a statistically significant difference in SBP means between the two genders, Female (Mean = 126.750) and Male (Mean = 129.825), using $p \leq 0.05$ as the criterion p-value.

Nonparametric Gender calculated $p \leq 0.08383$ resulting in the conclusion that there **is no** statistically significance difference in SBP medians between the two genders, Female (Median = 124) and Male (Median = 129), using $p \leq 0.05$ as the criterion p-value.

There is no simple resolution to this disparity toward Gender and the issue of statistical significance ($p \leq 0.05$) at the singular level of comparison for Systolic Blood Pressure, whether the data are viewed from a parametric perspective or a nonparametric perspective. Review the literature, communicate with colleagues, etc., to eventually make a decision on how to approach the two Gender breakout groups in terms of Systolic Blood Pressure. Document all work, methods, and conclusions and make data available to others if allowed. Transparency is the best approach toward reporting.

6.11 Addendum 3: Additional Practice Datasets

The purpose of this addendum is to continue with demonstrations on how the R environment supports Twoway ANOVA, the inferential test used to determine if there is a statistically significant difference for when data are typically viewed from a factorial perspective. Perhaps the most common application of Twoway ANOVA involves analyses for when multiple factor-type variables, each with multiple breakouts (e.g., levels) are examined against a measured object variable. This type of approach not only allows for singular analyses that examine differences by breakout groups, but it also allows inquiries into the possibility of interaction between and among the measured object variable(s).⁸

Note In the front matter to this lesson, the syntax is presented and is then immediately followed in most cases by either a copy of screen output or an accompanying figure. That approach is not used in this addendum. View this

⁸Syntax is provided throughout this addendum, but of course this syntax is only a suggestion. Experiment and take other approaches to how the data can be analyzed and outcomes presented by using other functions and other arguments. Use this addendum as a confidence-building resource on how R is used with increasingly complex analyses.

addendum as practice homework-type bonus materials. Syntax is presented and descriptive text goes along with the syntax. However, screen output and figures are generally excluded and when they show it is only to offer mid-point guidance that analyses follow along in a correct manner. Either key the syntax in this addendum or copy and paste it into an editor—whatever is feasible and most convenient. And then, to use the common expression, Practice—Practice—Practice!

6.11.1 Data with Normal Distribution Patterns

The data from `NewJerseyBlueberryYield.csv` are available at the publisher's Web site associated with this text. The data are interesting in that overall, the measured object variable (`NewJerseyBlueberryYield.df$LbsPerAcre`) exhibits normal distribution for some breakouts of factor-type object variables, but not all.

To address this dataset and the finding of inconsistency in normality, practice with the data in `NewJerseyBlueberryYield.df` and follow along with the sequence of activities detailed in this lesson:

- Briefly describe the data and prepare a plausible Null Hypothesis on blueberry yield by County, by Harvest Method, and by Year.
- Download the dataset, a .csv file, from the publisher's Web page associated with this text. Then, bring that dataset into a R session. Or, as a very unique exercise for those with special interest, recreate a similar dataset using the `DoE.base::fac.design()` function.
- Organize the data (see the prior comment on column names, if needed) and prepare a Code Book.
- Generate appropriate graphics to gain a good sense of the data, measured data (e.g., `LbsPerAcre`) and factor-type data (e.g., County, Harvest Method, Year).
- Generate appropriate descriptive statistics and measures of central tendency to gain a good sense of the data, measured data (e.g., `LbsPerAcre`) and factor-type data (e.g., County, Harvest Method, Year).
- Practice Quality Assurance throughout the process, beginning to end. Conduct normality tests on `LbsPerAcre` overall and `LbsPerAcre` by the factor-type object data (e.g., County, Harvest Method, Year).
- Using the previously declared Null Hypothesis, subject the data to a Twoway ANOVA.

- Using results from the Twoway ANOVA, develop tenable outcomes relating to any finding of statistically significant differences ($p \leq 0.05$) between and among the variables and statistically significant interactions ($p \leq 0.05$) between and among the variables.

As a few hints on how to continue with this practice activity, model the syntax presented below:

R Input

```
# Import the data

NewJerseyBlueberryYield.df <-
  read.table(file="NewJerseyBlueberryYield.csv",
  header=TRUE, dec=". ", sep=",")

getwd()
ls()
attach(NewJerseyBlueberryYield.df)
str(NewJerseyBlueberryYield.df)
nrow(NewJerseyBlueberryYield.df)
ncol(NewJerseyBlueberryYield.df)
dim(NewJerseyBlueberryYield.df)
names(NewJerseyBlueberryYield.df)
colnames(NewJerseyBlueberryYield.df)
head(NewJerseyBlueberryYield.df)
tail(NewJerseyBlueberryYield.df)
# NewJerseyBlueberryYield.df
summary(NewJerseyBlueberryYield.df)

NewJerseyBlueberryYield.df$Year <-
  as.factor(NewJerseyBlueberryYield.df$Year)
# Year is numeric but should be treated as a factor.

# Examine normality

shapiro.test(NewJerseyBlueberryYield.df$LbsPerAcre)

par(ask=TRUE); qqnorm(NewJerseyBlueberryYield.df$LbsPerAcre,
  main="Q-Q Plot Blueberry Yield (LbsPerAcre), Overall",
  col="blue", xlim=c(-4,4), ylim=c(0,8000), font.axis=2,
  font.lab=2)

RVAideMemoire::byf.shapiro(LbsPerAcre ~ County,
```

```

data=NewJerseyBlueberryYield.df)

RVAideMemoire::byf.shapiro(LbsPerAcre ~ Harvest,
  data=NewJerseyBlueberryYield.df)

RVAideMemoire::byf.shapiro(LbsPerAcre ~ Year,
  data=NewJerseyBlueberryYield.df)

```

Guided Direction: Make a first set of conclusions about normality using criterion $p \leq 0.05$. Which breakout groups show normal distribution at criterion $p \leq 0.05$? Then, make a second set of conclusions about normality using criterion $p \leq 0.01$. Which breakout groups show normal distribution at criterion $p \leq 0.01$? How will these findings, if any, impact the decision to use a parametric approach to Twoway ANOVA as opposed to a nonparametric approach to Twoway ANOVA?

R Input

```

# Build a Twoway ANOVA table

TwowayBlueberryCHY <-
  aov(LbsPerAcre ~ County * Harvest * Year,
  data=NewJerseyBlueberryYield.df)

summary(TwowayBlueberryCHY) # Generate a Twoway ANOVA table

```

R Output

[Selected output is not shown, to save space.]

	Df	F value	Pr(>F)
County	3	726.12 <0.0000000000000002	***
Harvest	1	6.83	0.0109 *
Year	2	0.16	0.8485
County:Harvest	3	5.61	0.0016 **
County:Year	6	2.07	0.0679 .
Harvest:Year	2	1.39	0.2563
County:Harvest:Year	6	0.40	0.8767
Residuals	72		
<hr/>			
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1			

R Input

```
# Use Tukey's HSD to examine the means of singular breakouts,
# to see which breakout groups share a common grouped mean and
# which breakout groups do not share a common grouped mean.

agricolae::HSD.test(TwowayBlueberryCHY, "County", group=TRUE,
console=TRUE)
```

R Output

[Selected output is not shown, to save space.]

	LbsPerAcre	groups
Atlantic	6580.12	a
Burlington	6181.54	b
Gloucester	5782.54	c
Camden	5168.71	d

R Input

```
ggplot2::ggplot(NewJerseyBlueberryYield.df,
aes(x=County, y=LbsPerAcre, group=County)) +
geom_violin(aes(fill=County)) + # Violin Plot geom
ggttitle(
"Blueberry Yield (Pounds per Acre) by New Jersey County\n") +
scale_x_discrete(name="") +
scale_y_continuous(name="\nBlueberries -- Pounds per Acre",
limits=c(0,7000), breaks=seq(0,7000,500)) +
theme_MacYates() +
theme(legend.position="none") +
coord_flip() # Rotate (e.g., flip) the presentation

agricolae::HSD.test(TwowayBlueberryCHY, "Harvest", group=TRUE,
console=TRUE)
```

R Output

[Selected output is not shown, to save space.]

	LbsPerAcre	groups
Hand	5957.42	a
Machine	5899.04	b

R Input

```
ggplot2::ggplot(NewJerseyBlueberryYield.df,
  aes(x=Harvest, y=LbsPerAcre, group=Harvest)) +
  geom_violin(aes(fill=Harvest)) +           # Violin Plot geom
  ggtitle(
    "Blueberry Yield (Pounds per Acre) by Harvest Method\n") +
  scale_x_discrete(name "") +
  scale_y_continuous(name="\nBlueberries -- Pounds per Acre",
    limits=c(0,7000), breaks=seq(0,7000,500)) +
  theme_MacYates() +
  theme(legend.position="none") +
  coord_flip()      # Rotate (e.g., flip) the presentation

agricolae::HSD.test(TwowayBlueberryCHY,"Year", group=TRUE,
  console=TRUE)
```

R Output

	LbsPerAcre	groups
2018	5936.28	a
2017	5927.81	a
2019	5920.59	a

R Input

```
ggplot2::ggplot(NewJerseyBlueberryYield.df,
  aes(x=Year, y=LbsPerAcre, group=Year)) +
  geom_violin(aes(fill=Year)) +           # Violin Plot geom
  ggtitle(
    "Blueberry Yield (Pounds per Acre) by Year\n") +
  scale_x_discrete(name "") +
  scale_y_continuous(name="\nBlueberries -- Pounds per Acre",
    limits=c(0,7000), breaks=seq(0,7000,500)) +
  theme_MacYates() +
  theme(legend.position="none") +
  coord_flip()      # Rotate (e.g., flip) the presentation
```

Was there a concern about normality, overall and by breakout groups? In case this concern demands attention, as a quality assurance measure use the Rfit::raov() function against the data in NewJerseyBlueberryYield.df, to consider Twoway ANOVA outcomes from a nonparametric perspective.

R Input

```
Rfit::raov(LbsPerAcre ~ County + Harvest + Year,
            data=NewJerseyBlueberryYield.df)
```

R Output

Robust ANOVA Table

	DF	RD	Mean RD	F	p-value
County	3	41732.4169	13910.8056	263.25328	0.00000
Harvest	1	296.5811	296.5811	5.61261	0.02052
Year	2	37.2532	18.6266	0.35250	0.70414
County:Harvest	3	1055.1569	351.7190	6.65606	0.00050
County:Year	6	745.2817	124.2136	2.35067	0.03954
Harvest:Year	2	157.5009	78.7505	1.49030	0.23217
County:Harvest:Year	6	248.1854	41.3642	0.78279	0.58619

Guided Direction: Was normal distribution a concern at criterion $p \leq 0.05$? Was normal distribution a concern at criterion $p \leq 0.01$? Look at the Twoway ANOVA table generated by using the Rfit::raov() function and compare the output to the Twoway ANOVA table generated by using the stats::aov() function. Are the p-values for statistically significant differences and statistically significant interactions *generally* the same? Explain the differences in outcomes between the two approaches to the Twoway ANOVA, parametric v nonparametric, if any.

6.11.2 Data That Do Not Exhibit Normal Distribution Patterns

The data from KentuckyCornYield.csv are available at the publisher's Web site associated with this text. The data are interesting in that overall, the measured object variable of interest (`KentuckyCornYield.df$BuPerAcre`) exhibits normal distribution for some breakouts of factor-type object variables, but not all.

To address this dataset and the finding of inconsistency in normality, practice with the data in `KentuckyCornYield.df` and follow along with the sequence of activities detailed in this lesson:

- Briefly describe the data and prepare a plausible Null Hypothesis on corn yield by County, by Management Practice, by Year, and by Soil.

- Download the dataset, a .csv file, from the publisher's Web page associated with this text. Then, bring that dataset into a R session. Or, as a very unique exercise for those with special interest, recreate a similar dataset using the DoE.base::fac.design() function.
- Organize the data (see the prior comment on column names, if needed) and prepare a Code Book.
- Generate appropriate graphics to gain a good sense of the data, measured data (e.g., BuPerAcre) and factor-type data (e.g., County, Management, Year, Soil).
- Generate appropriate descriptive statistics and measures of central tendency to gain a good sense of the data, measured data (e.g., BuPerAcre) and factor-type data (e.g., County, Management, Year, Soil).
- Practice Quality Assurance throughout the process, beginning to end. Conduct normality tests on BuPerAcre overall and BuPerAcre by the factor-type object data (e.g., County, Management, Year, Soil).
- Using the previously declared Null Hypothesis, subject the data to a Twoway ANOVA.
- Using results from the Twoway ANOVA, develop tenable outcomes relating to any finding of statistically significant differences ($p \leq 0.05$) between and among the variables and statistically significant interactions ($p \leq 0.05$) between and among the variables.

As a few hints on how to continue with this practice activity, model the syntax presented below:

R Input

```
# Import the data

KentuckyCornYield.df <-
  read.table(file="KentuckyCornYield.csv",
  header=TRUE, dec=". ", sep=",")

getwd()
ls()
attach(KentuckyCornYield.df)
str(KentuckyCornYield.df)
nrow(KentuckyCornYield.df)
ncol(KentuckyCornYield.df)
dim(KentuckyCornYield.df)
names(KentuckyCornYield.df)
```

```

colnames(KentuckyCornYield.df)
head(KentuckyCornYield.df)
tail(KentuckyCornYield.df)
# KentuckyCornYield.df
summary(KentuckyCornYield.df)

KentuckyCornYield.df$Year <-
  as.factor(KentuckyCornYield.df$Year)
# Year is expressed a a number but should be treated
# as a factor.

# Examine normality

shapiro.test(KentuckyCornYield.df$BuPerAcre)

```

R Output

```

Shapiro-Wilk normality test
p-value = 0.000000000323

```

R Input

```

par(ask=TRUE); qnorm(KentuckyCornYield.df$BuPerAcre,
  main="Q-Q Plot Corn Yield (BuPerAcre), Overall",
  col="blue", xlim=c(-4,4), ylim=c(0,225), font.axis=2,
  font.lab=2)

RVAideMemoire:::byf.shapiro(BuPerAcre ~ County,
  data=KentuckyCornYield.df)

```

R Output

```

Shapiro-Wilk normality tests
data: BuPerAcre by County
      W p-value
Allen 0.9882 0.07225 .
Barren 0.9891 0.10133
Logan 0.9891 0.10133
Simpson 0.9891 0.10133
Todd   0.9869 0.04462 *
Warren 0.9947 0.64843
---
```

```
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(BuPerAcre ~ Management,
  data=KentuckyCornYield.df)

RVAideMemoire::byf.shapiro(BuPerAcre ~ Year,
  data=KentuckyCornYield.df)

RVAideMemoire::byf.shapiro(BuPerAcre ~ Soil,
  data=KentuckyCornYield.df)
```

R Output

```
Shapiro-Wilk normality tests
data: BuPerAcre by Soil
      W     p-value
Clay 0.9837 0.00008668 ***
Sand 0.9827 0.00004900 ***
Silt 0.9833 0.00007137 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Guided Direction: Make a first set of conclusions about normality using criterion $p \leq 0.05$. Which breakout groups show normal distribution at criterion $p \leq 0.05$? Then, make a second set of conclusions about normality using criterion $p \leq 0.01$. Which breakout groups show normal distribution at criterion $p \leq 0.01$? How will these differences, if any, impact the decision to use a parametric approach to Twoway ANOVA as opposed to a nonparametric approach to Twoway ANOVA? (Figs. 6.13 and 6.14)

R Input

```
# Build a Twoway ANOVA table

TwowayCornCMYS <-
  aov(BuPerAcre ~ County * Management * Year * Soil,
  data=KentuckyCornYield.df)

summary(TwowayCornCMYS) # Generate a Twoway ANOVA table
```

```
# Use Tukey's HSD to examine the means of unitary breakouts,
# to see which breakout groups share a common grouped mean and
# which breakout groups do not share a common grouped mean.

agricolae::HSD.test(TwowayCornCMYS, "County", group=TRUE,
                     console=TRUE)

ggplot2::ggplot(KentuckyCornYield.df,
                 aes(x=County, y=BuPerAcre, group=County)) +
  geom_violin(aes(fill=County)) +           # Violin Plot geom
  ggtitle(
    "Corn Yield (Bushels per Acre) by County\n") +
  scale_x_discrete(name="") +
  scale_y_continuous(name="\nCorn -- Bushels per Acre",
                     limits=c(0,220), breaks=seq(0,220,20)) +
  theme_MacYates() +
  theme(legend.position="none") +
  coord_flip()           # Rotate (e.g., flip) the presentation
```

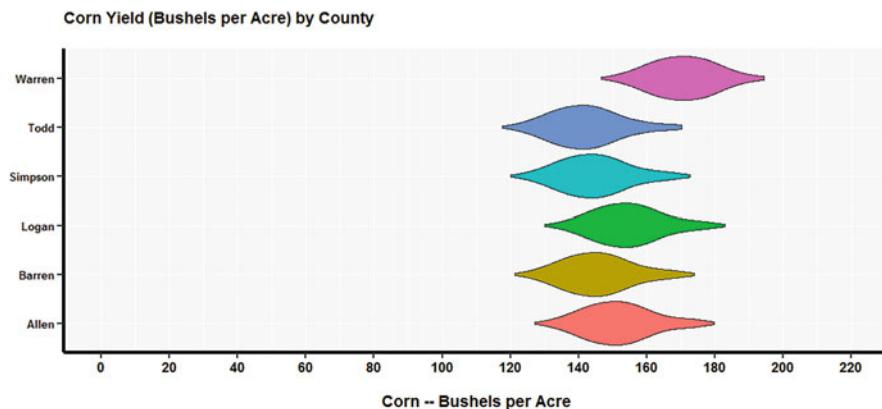


Figure 6.13: Corn yield by county breakouts

R Input

```
agricolae::HSD.test(TwowayCornCMYS, "Management", group=TRUE,
                     console=TRUE)
```

R Output

[Selected output is not shown, to save space.]

BuPerAcre groups

Conventional	157.751	a
No-Till	152.832	b
Organic	144.401	c

R Input

```
ggplot2::ggplot(KentuckyCornYield.df,
  aes(x=Management, y=BuPerAcre, group=Management)) +
  geom_violin(aes(fill=Management)) +      # Violin Plot geom
  ggtitle(
    "Corn Yield (Bushels per Acre) by Management Practice\n") +
  scale_x_discrete(name="") +
  scale_y_continuous(name="\nCorn -- Bushels per Acre",
    limits=c(0,220), breaks=seq(0,220,20)) +
  theme_MacYates() +
  theme(legend.position="none") +
  coord_flip()          # Rotate (e.g., flip) the presentation
```

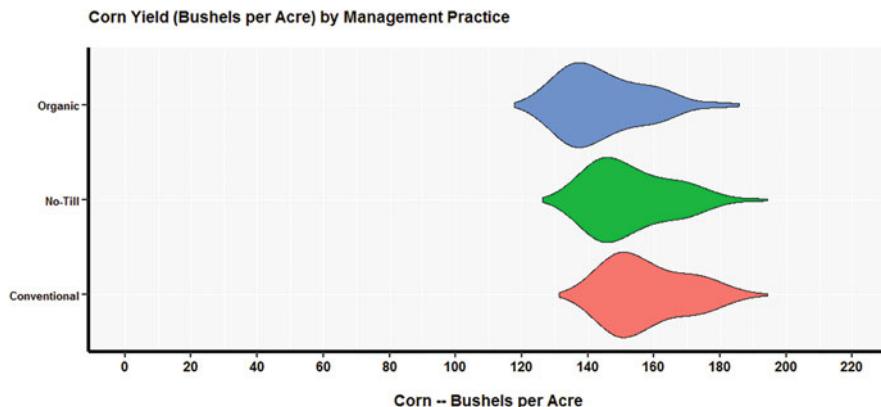


Figure 6.14: Corn yield by management breakouts

R Input

```
agricolae::HSD.test(TwowayCornCMYS, "Year", group=TRUE,
  console=TRUE)

ggplot2::ggplot(KentuckyCornYield.df,
  aes(x=Year, y=BuPerAcre, group=Year)) +
  geom_violin(aes(fill=Year)) +      # Violin Plot geom
  ggtitle(
    "Corn Yield (Bushels per Acre) by Year\n") +
  scale_x_discrete(name="") +
```

```

scale_y_continuous(name="\nCorn -- Bushels per Acre",
  limits=c(0,220), breaks=seq(0,220,20)) +
theme_MacYates() +
theme(legend.position="none") +
coord_flip()           # Rotate (e.g., flip) the presentation

agricolae::HSD.test(TwowayCornCMYS, "Soil", group=TRUE,
  console=TRUE)

ggplot2::ggplot(KentuckyCornYield.df,
  aes(x=Soil, y=BuPerAcre, group=Soil)) +
  geom_violin(aes(fill=Soil)) +           # Violin Plot geom
  ggtitle(
    "Corn Yield (Bushels per Acre) by Soil\n") +
  scale_x_discrete(name="") +
  scale_y_continuous(name="\nCorn -- Bushels per Acre",
    limits=c(0,220), breaks=seq(0,220,20)) +
  theme_MacYates() +
  theme(legend.position="none") +
  coord_flip()           # Rotate (e.g., flip) the presentation

```

Was there a concern about normality, overall and by breakout groups? In case this concern demands attention, as a quality assurance measure use the Rfit::raov() function against the data in KentuckyCornYield.df, to consider Twoway ANOVA outcomes from a nonparametric perspective.

R Input

```

Rfit::raov(BuPerAcre ~ County + Management + Year + Soil,
  data=KentuckyCornYield.df)
# Be patient. This analysis may take a few minutes.

```

Guided Direction: Was normal distribution a concern at criterion $p \leq 0.05$? Was normal distribution a concern at criterion $p \leq 0.01$? Look at the Twoway ANOVA table generated by using the Rfit::raov() function and compare the output to the Twoway ANOVA table generated by using the stats::aov() function. Are the p-values for statistically significant differences and statistically significant interactions *generally* the same? Explain the differences in outcomes between the two approaches to the Twoway ANOVA, parametric v nonparametric.

Guided Direction: Consider the N for NewJerseyBlueberryYield.csv (N = 96) and KentuckyCornYield.csv (N = 1296). Then, as time permits, review the literature on assumptions of normality and the Central Limit Theorem.

How does $N \geq 30$ fit into these constructs. It is far beyond the purpose of this lesson, but time spent reviewing the impact of N on normality would help with a good understanding of the research process on why N needs to be sufficiently large to obtain a representative sample, but N needs to be equally small enough to allow for efficient time-on-task, management of resources, and adherence to budgets. And then to add more complexity to this issue, consider the issue of representation, regardless of N .

6.12 Prepare to Exit, Save, and Later Retrieve This R Session

R Input

```
getwd()          # Identify the current working directory.  
ls()            # List all objects in the working  
                # directory.  
ls.str()         # List all objects, with finite detail.  
list.files()     # List files at the PC directory.  
  
save.image("R_Lesson_TwowayANOVA.rdata")  
  
getwd()          # Identify the current working directory.  
ls()            # List all objects in the working  
                # directory.  
ls.str()         # List all objects, with finite detail.  
list.files()     # List files at the PC directory.  
  
alarm()          # Alarm, notice of upcoming action.  
q()              # Quit this session.  
                # Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: File -> Load Workspace. Otherwise, use the load() function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use the .R script file (typically saved as a .txt ASCII-type file) and recreate the analyses and graphics, provided the data files remain available.

6.13 External Data and/or Data Resources Used in this Lesson

`KentuckyCornYield.csv` and `NewJerseyBlueberryYield.csv` were the only external files directly imported into this lesson and both files are found at the publisher's Web site associated with this text. Use these files to practice and replicate the outcomes used in this lesson.

All other data in this lesson were enumerated (e.g., user-created) directly in the R session, using a variety of R-based functions. The user-created data in this lesson are **not** included at the publisher's Web site associated with this text and they are instead easily created in R by following the syntax presented in this lesson.



Chapter 7

Correlation, Association, Regression, Likelihood, and Prediction

Abstract

The purpose of this lesson on correlation, association, regression, likelihood, and prediction is to provide guidance on how R can be used to first determine the association between two variables and to then use this degree of association, if any, to predict future outcomes. *Past behavior is the best predictor of future behavior.* This concept applies in the biological sciences, physical sciences, social sciences, and also in economics. Often, by knowing past relationships between and among variables it is then possible to build a prediction equation to make a reasonable estimate of future values for selected variables. This lesson will focus on Pearson's Product Moment Coefficient of Correlation (Pearson's r, perhaps the most common test for determining if there is an association between phenomena that display normal distribution), Spearman's Rank Correlation Coefficient (Spearman's rho, perhaps the most common test for determining

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_7) contains supplementary material, which is available to authorized users.

if there is an association between phenomena that do not display normal distribution), different types of regression analyses for prediction, as well as concepts such as probability, likelihood, odds, and odds ratio.

Keywords: Association, Binary logistic regression, Coefficient of correlation, Correlation, Galton (Francis), Likelihood, Linear regression, Minimal adequate model (MAM), Odds, Odds ratio, Ordinal regression, Pearson (Karl), Pearson's r, Probability, Regression, Scatter plot, Scatter plot matrix, Spearman (Charles), Spearman's rho, SPLOM (scatterplot matrix), Stepwise regression

7.1 Background

This lesson continues a series of demonstrations on how the R environment supports statistical analyses, with a focus on tests of correlation (e.g., association), such as Pearson's r and Spearman's rho. Building on initial analyses of correlation, this lesson then continues with simple linear regression, multiple linear regression, binary logistic regression, odds ratio, etc.—all in an effort to make reasonable predictions of future outcomes based on past events:

- Pearson's r is a frequently used estimate of the degree of association between two numeric object variables, assuming that each object variable represents an interval measurement. A typical example would be the feed-after-weaning weight gain of hogs up to 200 days after birth, where a Pearson's r estimate of association would help determine if there were a statistically significant increase in weight gain for varying daily amounts of conventional soybean meal (43–44% crude protein) that contains spent hulls fed to young hogs. Ideally, this information would help determine the most desirable high-protein feed rations needed to achieve desired marketing weight for hogs, taking into account the association between weight gain, amount of feed, cost of feed, management costs, live weight market price of hogs, and all other factors related to return on investment (ROI).
- Spearman's rho is a frequently used estimate of the degree of association between two numeric object variables, assuming that either (and possibly both) object variable represents an ordinal measurement. A typical example would be the association between the number of cigarettes smoked per day and Systolic Blood Pressure. It is the rare individual who smokes precisely the same number of cigarettes each day and equally smokes these cigarettes in the same manner (e.g., light puffs for each cigarette, deep inhalation of each cigarette, etc.). Because of this variance in smoking behavior as well as differences in cigarettes (e.g., differences between cigarette brands, differences between filtered v non-filtered cigarettes, differences between regular cigarettes v flavored cigarettes), it

is generally imprecise to assume that an individual who reports that they smoke 15 cigarettes per day is in parity with anyone else who makes the same claim.¹ Accordingly, there is an order to the number of cigarettes smoked per day, but the ordering has too much variance to be precise and it is best viewed as an ordinal datum, not an interval datum. The non-parametric Spearman's rho is likely the better test of association for any analysis of smoking and blood pressure, compared to the possible misuse of Pearson's r as a test of association which is dependent on data that are interval.

Past behavior is the best predictor of future behavior is a common expression when dealing with estimates of association. Saying that, simple linear regression, multiple linear regression, binary logistic regression, and odds ratio also have roles in this lesson. In a typical scenario, the value for X can be predicted with some degree of certainty **if** the value for Y and Z are known **and if** there is a known association between X, Y, and Z.

Perhaps the most important thing to recall when calculating estimates of association is the premise that *Correlation does not imply causation*. That is to say, there may be an association between X and Y, but not for one minute should it be assumed without strong evidence that X causes Y or that Y causes X. It is useful to know that the past behavior of X can be used to predict Y, but it should not be immediately assumed that X caused Y. Again, *Past behavior is the best predictor of future behavior* but equally recall that *Correlation does not imply causation*.

7.1.1 Description of the Data

There is one dataset used in this part of the lesson, with the data found in a .csv (e.g., Comma-Separated Values) file format. The data are organized in a typical row-by-column format, where each row represents the data for a specific subject and each column represents a specific object variable. Column headers are generally descriptive and are presented as text in the first row, the header row.

In some prior lessons the data were self-generated, often by using the rnorm() function. In an attempt to now more closely parallel the realities of how R is used in biostatistics, the data for this lesson are imported into the R session from an external source (e.g., the F:\ drive) by using the read.table() function.

The data are inspired by a wellness program, conceivably offered by a large company. There are 3945 subjects in the dataset and similar to prior lessons, the dataset should be viewed as a teaching dataset, where hypothetical data were organized to achieve desired results. There are 10 object variables in the dataset

¹ Assume that there may be some degree of undercount for when human subjects are asked to report on behaviors for tobacco, alcohol, and non-prescription drugs.

associated with this lesson, where one column represents an identification code for each unique subject and the remaining nine columns represent unique health-related object variables such as Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), etc., and eventually the calculated object variable Body Mass Index (BMI).

If the dataset were from an actual wellness study there would be far more than 10 object variables. Wellness-type data typically include health-related, personal, and social concerns such as: HDL-Cholesterol (mg/dL), LDL-Cholesterol (mg/dL), Triglyceride (mg/dL), Estrone (E1, pg/mL) and Estradiol (E2, pg/mL) for women, Prostate-Specific Antigen (PSA, ng/mL) for men, Pulse, Waist, Weight, Height, Individual Annual Salary, Overall Family Income, Marital Status, Number of Family Members at Household, Behaviors (e.g., Alcoholic Drinks, Prescription Drugs as well as Over the Counter Drugs and Herbal Supplements, Substance Abuse and Recreational Drugs, Tobacco Use, etc.), Family Medical History (e.g., Grandparents, Parents, Siblings, Aunts, Uncles, Cousins, etc.), and Known Medical Conditions Including Date of Onset. These issues are all important, but for teaching purposes their inclusion in this lesson would be overwhelming.

Although the data are inspired by a company-sponsored wellness program, for this lesson assume that data collection and presentation practices included the following, all in an effort to assure patient privacy:

- An outsourced third-party diagnostic service provider would be used for all services.
- All activities would be conducted off-campus, away from company property and personnel, with transportation to the off-campus location provided free-of-charge and at convenient times during regular work hours.
- The company would never have access to the resulting data in original format.
- Outcomes would be aggregated at a level of presentation such that individuals could never be discerned in any printout of results provided to the company and to the public.
- Subjects would have access to their personal data, in a secured manner, on-demand and without question. Guidance would be provided to explain the meaning of all personal data.
- There are missing data for some object variables, but not all.
- Again, for the teaching dataset used in this lesson, data have been organized to achieve desired outcomes. Accordingly, the dataset used in this lesson is inspired by wellness data but the dataset was constructed for demonstration and teaching purposes only.

7.1.2 Null Hypothesis (Ho)

There is one Null Hypothesis associated with this part of the lesson, with $p \leq 0.05$ the criterion level of significance.

Null Hypothesis (Ho): There is no statistically significant association ($p \leq 0.05$) between and among the numeric object variables found in the dataset: Age (Years), Total Cholesterol (mg/dL), Systolic Blood Pressure (mmHg), Diastolic Blood Pressure (mmHg), and Body Mass Index (BMI).

Although it may be readily evident in the above Null Hypothesis, it is still prudent to recapitulate that the data used in this demonstration of association, using a teaching dataset, are focused on measurements gained by biometric screening of a fairly large group of human subjects participating in a wellness program. Assume that the data in this lesson are representative of the population, although little is known about the population beyond what is self-evident from the dataset `EmpWellAge21to79.csv`.

7.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

The dataset consists of nearly 4000 subjects. The data have been put into .csv (Comma-Separated Values) file format and the file has been placed at an external location, the F:\ drive in this case. Appropriate R-based functions are used to set the desired working directory and the data are then imported into this directory by using the `read.table()` function. The data start with a header row to name the object variables and a unique identification code is used for each subject. Details for each object variable are found in the Code Book.

R Input

```
#####
# Housekeeping                                     Use for All Analyses #
#####
date()          # Current system time and date.
Sys.time()       # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls(all.names=TRUE) # List all objects in the working
                   # directory, including hidden files.
rm(list=ls())     # CAUTION: Remove all files in the
                   # working directory. If this
                   # action is not desired, use rm()
```

```

#           one-by-one to remove the objects
#           that are not needed.
ls.str()      # List all objects, with finite detail.
getwd()        # Identify the current working directory.
setwd("F:/R_BiostatisticsIntroduction")
               # Set to a new working directory.
               # Note the single forward slash and double
               # quotes.
               # This new directory should be the directory
               # where the data file is located, otherwise
               # the data file will not be found.
getwd()        # Confirm the working directory.
list.files()   # List files at the PC directory.
.libPaths()    # Library pathname.
.Library       # Library pathname.
sessionInfo() # R version, locale, and packages.
search()       # Attached packages and objects.
searchpaths()  # Attached packages and objects.
#####
#####
```

With only occasional variation, the Housekeeping template shown above is used throughout this text. The purpose of this collection of R functions, all in one convenient beginning section, is to be sure that the working environment, selected directory, etc., are as desired.²

Create an object called `EmployeeBiometric.df` which will be a dataframe, as indicated by the enumerated `.df` extension to the object name. This object will represent the output of applying the `read.table()` function against the comma-separated values file called `EmpWellAge21to79.csv`. Note the descriptive nature and corresponding long name of the `.csv` file and equally note the arguments used with the `read.table()` function:

- Descriptive file names are an excellent means of providing internal documentation about the nature of the data. As detailed in the filename `EmpWellAge21to79.csv`, it is evident that the data relate to Employee

²Throughout this text, the `attach()` function has been presented as a *Good Programming Practice*. It should be stated, now that this lesson begins to address some very advanced topics in the use of R for biostatistics, that there are those individuals who do not share this opinion and instead view the `attach()` function differently. There are sufficient Internet-based resources on this topic, pro and con, for those who wish to know more about this topic, which goes beyond the purpose of this lesson. Instead, it is suggested that the `attach()` function should be used, along with formal use of a `Dataframe$Object` convention for naming variables when preparing syntax.

Wellness and that the data are restricted to those employees who are greater than or equal to age 21 and less than or equal to age 79. Complete details are provided in the Code Book, but this descriptive file name, alone, serves as an advance organizer about the data.

- The first row in the file `EmpWellAge21to79.csv` represents a header, not data, with descriptive variable names (`header=TRUE`). With attention to good documentation practices, the names for each variable should be rich and fully descriptive even if verbose. Shorter variable names for objects can be used later, if needed.
- Decimals are indicated by using a period (`dec=".."`) and not a comma or some other symbol. Although it is certainly common to expect the `.` character when placing data into decimal format, it is still best to declare the character used for decimals. A period character is not the one and only character used for decimal notation and unverified assumptions can cause havoc with later analyses.
- Inherent to comma-separated values (`.csv`) file structure, the separator between fields is a comma (`sep=","`) and is neither a tab nor white space. R can accommodate tabs and white space as separators between data, but that would not be the case for a `.csv` file.

R Input

```
EmployeeBiometric.df <-  
  read.table(file="EmpWellAge21to79.csv",  
            header=TRUE, dec="..", sep=",")  
  # Use the read.table() function to import the .csv file  
  # EmpWellAge21to79.csv and place it into the object  
  # EmployeeBiometric.df, which: (1) has a header row,  
  # (2) uses a period for decimals, and (3) uses a comma  
  # to separate one field from another.  
  
getwd()                      # Identify the working directory  
ls()                          # List objects  
attach(EmployeeBiometric.df)  # Attach the data, for later use  
str(EmployeeBiometric.df)    # Identify structure  
nrow(EmployeeBiometric.df)   # List the number of rows  
ncol(EmployeeBiometric.df)   # List the number of columns  
dim(EmployeeBiometric.df)    # Dimensions of the data frame  
names(EmployeeBiometric.df)  # Identify names  
colnames(EmployeeBiometric.df) # Show column names  
head(EmployeeBiometric.df)   # Show the head  
tail(EmployeeBiometric.df)   # Show the tail  
# EmployeeBiometric.df       # Show the entire dataframe  
summary(EmployeeBiometric.df) # Summary statistics
```

R Output

[Selected output is not shown, to save space.]

EmployeeID	Gender	AgeYears	RaceEthnicity
ID10005:	1	Min. :1.00	Min. :21.0
ID10010:	1	1st Qu.:1.00	1st Qu.:34.0
ID10016:	1	Median :1.00	Median :47.0
ID10028:	1	Mean :1.47	Mean :47.7
ID10041:	1	3rd Qu.:2.00	3rd Qu.:61.0
ID10051:	1	Max. :2.00	Max. :79.0
(Other):	3939		
TotalCholesterolmgdL	SBPmmHg	DBPmmHg	
Min. : 78	Min. : 74	Min. : 34.0	
1st Qu.:151	1st Qu.:108	1st Qu.: 60.0	
Median :175	Median :118	Median : 70.0	
Mean :180	Mean :122	Mean : 69.3	
3rd Qu.:205	3rd Qu.:132	3rd Qu.: 78.0	
Max. :380	Max. :222	Max. :122.0	
NA's :894	NA's :209	NA's :217	
BMI Metric	BMI Status	Obesity	
Min. :13.4	Min. :1.00	Min. :1.00	
1st Qu.:22.6	1st Qu.:2.00	1st Qu.:1.00	
Median :26.4	Median :3.00	Median :1.00	
Mean :27.7	Mean :2.86	Mean :1.31	
3rd Qu.:31.3	3rd Qu.:4.00	3rd Qu.:2.00	
Max. :70.1	Max. :4.00	Max. :2.00	
NA's :20	NA's :20	NA's :20	

When viewing this process for bringing data from an external drive into the active R session, observe how it is not necessary to use the .df extension for a dataframe, but this is certainly a good programming practice and use of appropriate documentation. By adding the .df extension to the object name, there is no confusion that the object is indeed a dataframe.

Review the syntax and note how a # comment character was placed in front of EmployeeBiometric.df. With nearly 4000 cases (e.g., rows) the dataset is simply far too large for any meaningful review on the screen, line-by-line. The # character is used to temporarily *comment-out* the syntax and keep it from being put into effect. Commenting-out syntax is a useful practice, when needed, serving as a reminder of the possibility of later placing the syntax into effect by removing the comment character.

7.3 Organize the Data and Display the Code Book

The Code Book for `EmployeeBiometric.df`, shown below, provides detailed information on the object variables selected for this lesson. This dataset represents only the smallest part of the many types of data associated with employee wellness programs used by many companies, but it still serves as a useful introduction to working with a large and complex dataset specific to biostatistics.

R Input

```
#####
# Code Book for EmployeeBiometric.df          #
#####
```

EmployeeID - Employee Identification Number #
Factor (e.g., nominal) #

Each employee has an assigned number, separate from #
their government-issued Health Insurance Card #
Number, Social Security Number, etc., that is used #
for identification by the outsourced company #
conducting biometric screening. The EmployeeID #
code is created so that there is no way to identify #
employee name, gender, race-ethnicity, age, etc., #
by referring to this identifier. It is a unique #
code and only the outsourced company conducting #
biometric screening has the key needed to identify #
specific employees participating in the wellness #
program.
#-----#

Gender - Gender #
Factor (e.g., nominal) #
1 Female #
2 Male #

Because there is no hierarchy or order to nominal #
objects such as Gender, note how the two nominal #
values (e.g., Female and Male) are presented in #
alphabetical order. Alphabetical ordering is #
generally a good practice when there is otherwise #
no specific order to breakout values.
#-----#
#

```
# AgeYears - Age (Years) at Time of Screening      #
# Numeric (e.g., interval)                         #
#
# By design, the data for this study are restricted #
# to employees who range in age from 21 Years to 79 #
# Years. The few workers who were less than 21      #
# Years, if any, were allowed to participate in the  #
# wellness program and obtain their data, but their #
# data have been excluded from this study to avoid  #
# any possible concerns about special permissions and #
# data issues associated with Institutional Review   #
# Board (IRB). The few workers who were older than    #
# 79 Years, if any, were also allowed to participate   #
# in the wellness program and obtain their data, but   #
# their data have been excluded from this study to    #
# avoid concerns about identification of the few       #
# individuals in this age group.                      #
#-----#
#
# RaceEthnicity - Race-Ethnicity                  #
# Factor (e.g., nominal)                          #
# 1 Asian                                         #
# 2 Black                                         #
# 3 Hispanic                                       #
# 4 Other                                         #
# 5 White                                         #
#
# Race and ethnicity are self-identified and based on #
# accepted protocols. They are different constructs. #
# The five breakout groups for RaceEthnicity in this  #
# dataset are a common way of identifying both Race    #
# Ethnicity into one common variable, but there are    #
# many other ways by which these constructs are        #
# organized.                                       #
#-----#
#
# TotalCholesterolmgdL - Total Cholesterol(mg/dL)  #
# Numeric (e.g., interval)                         #
# Possible Range 65 onward                        #
#-----#
#
# SBPmmHg - Systolic Blood Pressure mmHg          #
# Numeric (e.g., interval)                         #
```

```
# Possible Range 65 onward          #
#-----#
#
# DBPmmHg - Diastolic Blood Pressure mmHg      #
# Numeric (e.g., interval)                      #
# Possible Range 20 onward                     #
#-----#
#
# BMI - Body Mass Index (BMI Metric = kg/m^2)    #
# Numeric (e.g., interval)                      #
# Possible Range 12.00 onward                   #
#
# Body measurements (weight and height, along with   #
# waist) were obtained concurrent to when blood     #
# pressure data were obtained. Accommodations were   #
# made, when required, for those with special needs. #
#
# Personnel from the diagnostic center used the      #
# metric system to record weight (kg) and height      #
# (cm). However, given that that the metric system    #
# is unfamiliar to many subjects, the data were later  #
# converted to the Imperial measurement system (e.g.,  #
# pounds and inches). However, this dataset only       #
# includes metric measures and the metric algorithm   #
# was used to calculate Body Mass Index (BMI).         #
#-----#
#
# BMIStatus - BMI Collapsed Into Four Groups        #
# Factor (e.g., ordinal)                          #
#
# 1  BMI <= 18.599           Underweight      #
# 2  BMI >= 18.600 and <= 24.999 Normal Weight #
# 3  BMI >= 25.000 and <= 29.999 Overweight   #
# 4  BMI >= 30.000           Obese          #
#-----#
#
# Obesity - BMI Collapsed Into Two Groups          #
# Factor (e.g., ordinal)                          #
#
# 1  BMI <= 29.999 Not Obese      #
# 2  BMI >= 30.000 Obese          #
#####
#####
```

The dataframe `EmployeeBiometric.df` is fairly simple to understand and should be manageable for beginning students and researchers with limited experience even though there are nearly 4000 subjects and 10 object variables in the dataset. There may be a few challenges due to missing data, but that is a common occurrence in biostatistics that must be addressed in real-world situations, which this lesson parallels.

For this dataset the `class()` function, `str()` function, and `table(duplicated())` functions will be sufficient first steps to provide some degree of assurance that data are organized as desired.

R Input

```
class(EmployeeBiometric.df) # Object-type
```

R Output

```
[1] "data.frame"
```

R Input

```
str(EmployeeBiometric.df) # Object-structure
```

R Output

```
'data.frame': 3945 obs. of 10 variables:
 $ EmployeeID      : Factor w/ 3945 levels "ID10005",
 $ Gender          : int  1 1 1 2 1 1 1 2 1 1 ...
 $ AgeYears        : int  34 67 41 57 35 40 40 34 40
 $ RaceEthnicity   : int  5 5 5 1 5 3 2 5 2 2 ...
 $ TotalCholesterol: int  221 175 224 217 139 198 NA
 $ SBPmmHg         : int  102 NA NA 90 86 114 NA 92 1
 $ DBPmmHg         : int  44 NA NA 42 42 74 NA 48 80
 $ BMIMetric       : num  18.4 18.7 21.6 20.6 18.7 ..
 $ BMIStatus       : int  1 2 2 2 2 2 2 2 1 2 ...
 $ Obesity         : int  1 1 1 1 1 1 1 1 1 1 ...
```

R Input

```
table(duplicated(
EmployeeBiometric.df$EmployeeID)) # Check for duplicates
```

R Output

```
FALSE
```

```
3945
```

These functions provide evidence that the object `EmployeeBiometric.df` seems to be initially correct:

- `EmployeeBiometric.df` is a dataframe, as demonstrated by using the `class()` function.
- The structure for each variable is initially identified by using the `str()` function. Whole numbers show as int (e.g., integer) and numbers in decimal format show as num (e.g., numeric).
- There is evidence that there are no duplicates at the individual level, with this assurance gained by using the `duplicated()` function. The `table()` function is wrapped around the `duplicated()` function to provide a simple read-out of the output. For this dataframe, FALSE is the only output to the `duplicated()` function and the number of FALSE listings is equal to the number of subjects in the dataframe. If there had been duplicates, numbers would have been provided for both FALSE (not a duplicate) and TRUE (a duplicate).

Recall that the Code Book shows data in their desired formats, which often requires some degree of recoding which has not yet occurred. Once there is agreement that the data were brought into R in correct original format, it is usually necessary to organize the data to some degree, to be sure that all data are in desired format, as identified in the Code Book:

The data for object variables `Gender`, `RaceEthnicity`, `BMIStatus`, and `Obesity` are currently viewed as integers, but they should be treated as if they were group-focused factor-type objects. A set of simple R-based actions can easily: (1) transform (e.g., recode) the integer-based object variables into a new enumerated factor format, and (2) apply natural language text labels (e.g., Female, Male) for the otherwise cryptic numeric codes (e.g., 1, 2).

Values for `AgeYears`, `TotalCholesterolmgdL`, `SBPmmHg`, and `DBPmmHg` are currently whole numbers and they are first treated in R as integers. A simple recode action will be used to put these values into numeric format, which may be desirable if any math-type actions are used with these object variables.

Values for the calculated object variable `BMIMetric` have a decimal format and show as numeric.

These transformations (e.g., recodes) of selected object variables are needed and the process, using R-based syntax, follows. There may be some unnecessary (perhaps redundant) actions with the following recode activities, but these

actions are purposely demonstrated to provide assurance that when the object variables are used, the values for each variable are in desired format:

R Input

```
str(EmployeeBiometric.df)      # structure before recoding
summary(EmployeeBiometric.df) # summary before recoding

EmployeeBiometric.df$EmployeeID <-
  as.factor(EmployeeBiometric.df$EmployeeID)

EmployeeBiometric.df$Gender <-
  factor(EmployeeBiometric.df$Gender,
  labels=c("Female", "Male"))
levels(EmployeeBiometric.df$Gender)
# NOTE: factor(...) and NOT as.factor (...)
```

R Output

```
[1] "Female" "Male"
```

R Input

```
EmployeeBiometric.df$AgeYears <-
  as.numeric(EmployeeBiometric.df$AgeYears)

EmployeeBiometric.df$RaceEthnicity <-
  factor(EmployeeBiometric.df$RaceEthnicity,
  labels=c("Asian", "Black", "Hispanic", "Other", "White"))
levels(EmployeeBiometric.df$RaceEthnicity)
# NOTE: factor(...) and NOT as.factor (...)
```

R Output

```
[1] "Asian"     "Black"      "Hispanic"   "Other"      "White"
```

R Input

```
EmployeeBiometric.df$TotalCholesterolmgdL <-
  as.numeric(EmployeeBiometric.df$TotalCholesterolmgdL)

EmployeeBiometric.df$SBPmmHg <-
```

```

as.numeric(EmployeeBiometric.df$SBPmmHg)

EmployeeBiometric.df$DBPmmHg <-
  as.numeric(EmployeeBiometric.df$DBPmmHg)

EmployeeBiometric.df$BMIMetric <-
  as.numeric(EmployeeBiometric.df$BMIMetric)

EmployeeBiometric.df$BMIStatus <-
  factor(EmployeeBiometric.df$BMIStatus,
  labels=c("Underweight", "Normal Weight", "Overweight",
  "Obese"))
levels(EmployeeBiometric.df$BMIStatus)
# NOTE: factor(...) and NOT as.factor (...)
```

R Output

```
[1] "Underweight"    "Normal Weight" "Overweight"
[4] "Obese"
```

R Input

```

EmployeeBiometric.df$Obesity <-
  factor(EmployeeBiometric.df$Obesity,
  labels=c("Not Obese", "Obese"))
levels(EmployeeBiometric.df$Obesity)
# NOTE: factor(...) and NOT as.factor (...)
```

R Output

```
[1] "Not Obese" "Obese"
```

Now that the recoding has been completed, including the recoding that is redundant but was done as a quality assurance measure, there is a heightened degree of confidence that all variables are in correct format. Now, it is best to again attach the data (which may also be a redundant action, but still useful) and to then use a series of functions to be sure once again that the data are correct.

R Input

```

getwd()                      # Identify the working directory
ls()                          # List objects
```

```

attach(EmployeeBiometric.df)      # Attach the data, for later use
nrow(EmployeeBiometric.df)       # List the number of rows
ncol(EmployeeBiometric.df)       # List the number of columns
dim(EmployeeBiometric.df)        # Dimensions of the data frame
names(EmployeeBiometric.df)      # Identify names
colnames(EmployeeBiometric.df)   # Show column names
head(EmployeeBiometric.df)       # Show the head
tail(EmployeeBiometric.df)       # Show the tail
str(EmployeeBiometric.df)        # Structure after recoding

```

R Output

```

'data.frame': 3945 obs. of 10 variables:
 $ EmployeeID       : Factor w/ 3945 levels "ID10005",
 $ Gender           : Factor w/ 2 levels "Female","Male"
 $ AgeYears         : num  34 67 41 57 35 40 40 34 40 50
 $ RaceEthnicity    : Factor w/ 5 levels "Asian","Black"
 $ TotalCholesterol: num  221 175 224 217 139 198 NA 205
 $ SBPmmHg          : num  102 NA NA 90 86 114 NA 92 120
 $ DBPmmHg          : num  44 NA NA 42 42 74 NA 48 80 72
 $ BMIMetric        : num  18.4 18.7 21.6 20.6 18.7 ...
 $ BMIStatus        : Factor w/ 4 levels "Underweight",...
 $ Obesity          : Factor w/ 2 levels "Not Obese",

```

R Input

```
summary(EmployeeBiometric.df) # Summary after recoding
```

R Output

	EmployeeID	Gender	AgeYears
ID10005:	1	Female:2078	Min. :21.0
ID10010:	1	Male :1867	1st Qu.:34.0
ID10016:	1		Median :47.0
ID10028:	1		Mean :47.7
ID10041:	1		3rd Qu.:61.0
ID10051:	1		Max. :79.0
(Other):	3939		
	RaceEthnicity	TotalCholesterolmgdL	SBPmmHg
Asian :	482	Min. : 78	Min. : 74
Black :	818	1st Qu.:151	1st Qu.:108
Hispanic:	904	Median :175	Median :118

Other : 117	Mean : 180	Mean : 122
White : 1624	3rd Qu.: 205	3rd Qu.: 132
	Max. : 380	Max. : 222
	NA's : 894	NA's : 209
DBPmmHg	BMI Metric	BMI Status
Min. : 34.0	Min. : 13.4	Underweight : 168
1st Qu.: 60.0	1st Qu.: 22.6	Normal Weight: 1437
Median : 70.0	Median : 26.4	Overweight : 1114
Mean : 69.3	Mean : 27.7	Obese : 1206
3rd Qu.: 78.0	3rd Qu.: 31.3	NA's : 20
Max. : 122.0	Max. : 70.1	
NA's : 217	NA's : 20	
Obesity		
Not Obese: 2719		
Obese : 1206		
NA's : 20		

7.3.1 Conduct a Visual Data Check Using Graphics (e.g., Figures)

Factor Object Variables

Although the barplot() function would likely be a first choice for preparing bar plots of the factor-type object variables Gender, RaceEthnicity, BMIStatus, and Obesity, it might be helpful to instead use the epiDisplay::tab1() function. The epiDisplay::tab1() function provides syntax that generates an attractive barplot with frequency counts placed over each bar and a frequency distribution table printed to the screen. Also, for factor-type object variables with missing data, the epiDisplay::tab1() function incorporates the presentation of these missing data into the barplot and the frequency distribution table, providing a complete sense of the data (Fig. 7.1).

R Input

```
install.packages("epiDisplay", dependencies = TRUE)
library(epiDisplay)          # Load the epiDisplay package.
help(package=epiDisplay)     # Show the information page.
sessionInfo()                # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
# Barplot and Frequencies of Gender
epiDisplay::tab1(EmployeeBiometric.df$Gender,
  main="Frequency of Gender",
```

```

ylab="Frequency",                      # The 1 of tab1 is the one
graph=TRUE,                            # numeric character and it
missing=TRUE,                          # is not the letter l.
bar.values=c("frequency"),            # Look at the frequency
horiz=TRUE,                            # distribution table printed
cex=0.85,                             # to the screen: Frequency,
cex.names=0.85,                        # Percent, and Cumulative
cex.lab=0.85, cex.axis=0.85, # Percent.
col= c("black", "red"))
# There are no missing data and it is only necessary to
# allow for the two breakout categories for Gender.

# Barplot and Frequencies of RaceEthnicity
epiDisplay::tab1(EmployeeBiometric.df$RaceEthnicity,
main="Frequency of Race Ethnicity", ylab="Frequency",
graph=TRUE, missing=TRUE, bar.values=c("frequency"),
horiz=TRUE, cex=0.85, cex.names=0.85, cex.lab=0.85,
cex.axis=0.85,
col= c("black", "red", "blue", "green", "cyan"))
# There are no missing data and it is only nececessary to
# allow for the five breakout categories for RaceEthnicity.

# Barplot and Frequencies of BMIStatus
epiDisplay::tab1(EmployeeBiometric.df$BMIStatus,
main="Frequency of BMI Status", ylab="Frequency",
graph=TRUE, missing=TRUE, bar.values=c("frequency"),
horiz=TRUE, cex=0.85, cex.names=0.85, cex.lab=0.85,
cex.axis=0.85,
col= c("black", "red", "blue", "green", "cyan"))
# There are missing data and it is nececessary to allow for
# the four breakout categories for BMIStatus and the few
# missing datapoints, seen as NA.

# Barplot and Frequencies of Obesity
epiDisplay::tab1(EmployeeBiometric.df$Obesity,
main="Frequency of Obesity", ylab="Frequency", graph=TRUE,
missing=TRUE, bar.values=c("frequency"), horiz=TRUE,
cex=0.85, cex.names=0.85, cex.lab=0.85, cex.axis=0.85,
col= c("black", "red", "blue"))
# There are missing data and it is nececessary to allow for
# the two breakout categories for Obesity and the few
# missing datapoints, seen as NA.

```

Numeric Object Variables

There are more than a few R-based graphical functions used to visualize data distribution patterns for the numeric-type object variables addressed in this

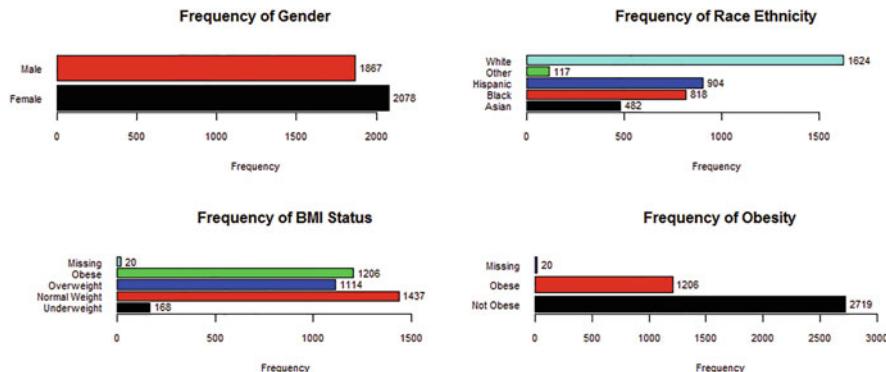


Figure 7.1: Distribution of factor-type object variables gender, RaceEthnicity, BMISStatus, and obesity

lesson: AgeYears, TotalCholesterolmgdL, SBPmmHg, DBPmmHg, and the calculated value for BMIMetric. Each function displays data distribution patterns in a specific manner. For this lesson, observe how multiple R-based graphical functions are incorporated into one comprehensive graphical image. Figures for the `EmployeeBiometric.df` object variables with numeric values (e.g., often called interval values or scale values) will be displayed in a 2 by 2 graphical grid. Each square in the grid uses a different R-based function to display data distribution:

- Histogram — Use the `hist()` function.
- Density Curve — The `plot()` function is wrapped around the `density()` function.
- Beanplot — Use the `beanplot::beanplot()` function, by loading the external package `beanplot` into the active R session.
- Quantile-Quantile Curve — Use the `qqnorm()` function.

Use the `install.packages()` function to load the external package `beanplot` into the active R session. Then, use the `library()` function to put the functions in this external package into use (Figs. 7.2, 7.3, and 7.4).

R Input

```
install.packages("beanplot", dependencies = TRUE)
library(beanplot)          # Load the beanplot package.
help(package=beanplot)      # Show the information page.
sessionInfo()               # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
```

```

hist(EmployeeBiometric.df$AgeYears, main="Age", col="red",
      breaks=100)
plot(density(EmployeeBiometric.df$AgeYears, na.rm=TRUE),
      main="Age", col="red") # na.rm argument is required
beanplot::beanplot(EmployeeBiometric.df$AgeYears, main="Age",
      col="red", what=c(1,1,1,0), overallline="mean", boxwex=0.75,
      horizontal=FALSE)
# Use a beanplot as an alternate to the more commonly used
# boxplot, to have a more comprehensive view of data
# distribution. Arguments that are mostly unique to the
# beanplot::beanplot() function, as used in this example,
# include:
# what=c(1,1,1,0) uses four Boolean options to control
# the plot, including (1) total average line, (2) beans,
# (3) bean average, and (4) beanlines.
# overallline="mean" shows a line that represents the
# mean, but "median" is also an option.
# boxwex=0.75 controls the aspect ratio of the output.
qqnorm(EmployeeBiometric.df$AgeYears, main="Age", col="red")

```

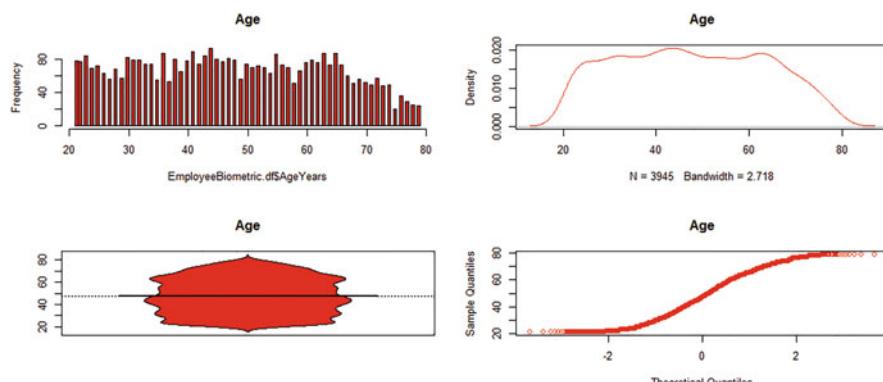


Figure 7.2: Distribution of numeric-type object variable age

R Input

```

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
hist(EmployeeBiometric.df$TotalCholesterolmgdL,
      main="Total Cholesterol (mg/dL)", col="red", breaks=100)
plot(density(EmployeeBiometric.df$TotalCholesterolmgdL,
      na.rm=TRUE), main="Total Cholesterol (mg/dL)", col="red")
# na.rm argument is required

```

```
beanplot::beanplot(EmployeeBiometric.df$TotalCholesterolmgdL,
  main="Total Cholesterol (mg/dL)", col="red", what=c(1,1,1,0),
  overallline="mean", boxwex=0.75, horizontal=FALSE)
qqnorm(EmployeeBiometric.df$TotalCholesterolmgdL,
  main="Total Cholesterol (mg/dL)", col="red")
```

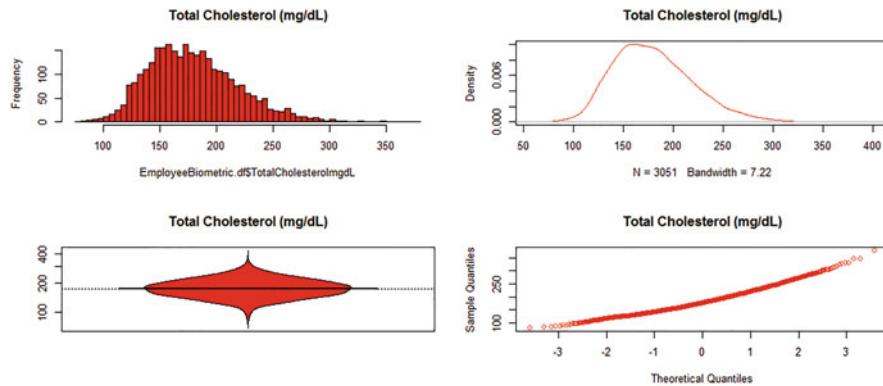


Figure 7.3: Distribution of numeric-type object variable cholesterol

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
hist(EmployeeBiometric.df$SBPmmHg,
  main="Systolic Blood Pressure (mmHg)", col="red", breaks=100)
plot(density(EmployeeBiometric.df$SBPmmHg, na.rm=TRUE),
  main="Systolic Blood Pressure (mmHg)", col="red")
# na.rm argument is required
beanplot::beanplot(EmployeeBiometric.df$SBPmmHg,
  main="Systolic Blood Pressure (mmHg)", col="red",
  what=c(1,1,1,0), overallline="mean", boxwex=0.75,
  horizontal=FALSE)
qqnorm(EmployeeBiometric.df$SBPmmHg,
  main="Systolic Blood Pressure (mmHg)", col="red")
```

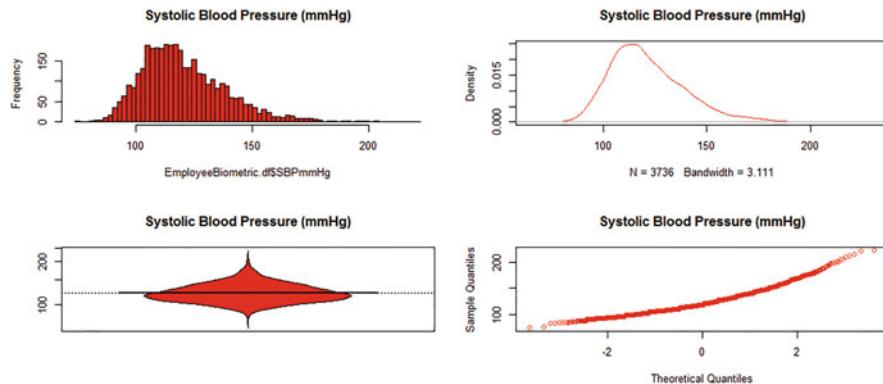


Figure 7.4: Distribution of numeric-type object variable systolic blood pressure

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
hist(EmployeeBiometric.df$DBPmmHg,
     main="Diastolic Blood Pressure (mmHg)", col="red",
     breaks=100)
plot(density(EmployeeBiometric.df$DBPmmHg, na.rm=TRUE),
     main="Diastolic Blood Pressure (mmHg)", col="red")
# na.rm argument is required
beanplot::beanplot(EmployeeBiometric.df$DBPmmHg,
     main="Diastolic Blood Pressure (mmHg)", col="red",
     what=c(1,1,1,0), overallline="mean", boxwex=0.75,
     horizontal=FALSE)
qqnorm(EmployeeBiometric.df$DBPmmHg,
     main="Diastolic Blood Pressure 1st Reading (mmHg)",
     col="red")
```

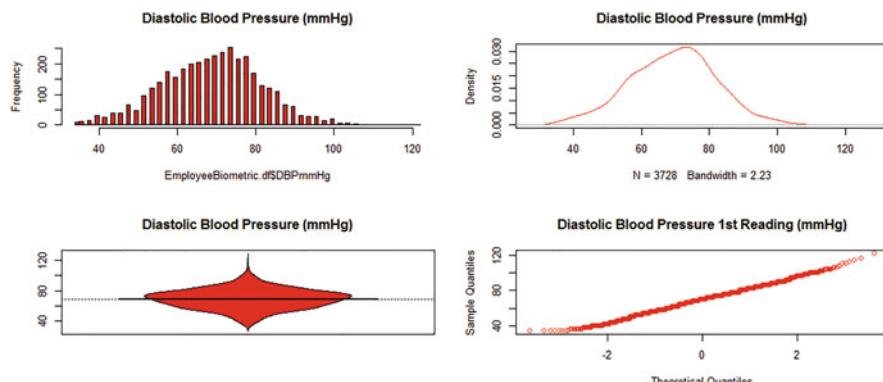


Figure 7.5: Distribution of numeric-type object variable diastolic blood pressure

R Input

```
par.ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
hist(EmployeeBiometric.df$BMIMetric,
     main="Body Mass Index: BMI Metric = kg/m^2", col="red",
     breaks=100)
plot(density(EmployeeBiometric.df$BMIMetric, na.rm=TRUE),
     main="Body Mass Index: BMI Metric = kg/m^2", col="red")
# na.rm argument is required
beanplot::beanplot(EmployeeBiometric.df$BMIMetric,
     main="Body Mass Index: BMI Metric = kg/m^2", col="red",
     what=c(1,1,1,0), overallline="mean", boxwex=0.75,
     horizontal=FALSE)
qqnorm(EmployeeBiometric.df$BMIMetric,
     main="Body Mass Index: BMI Metric = kg/m^2", col="red")
```

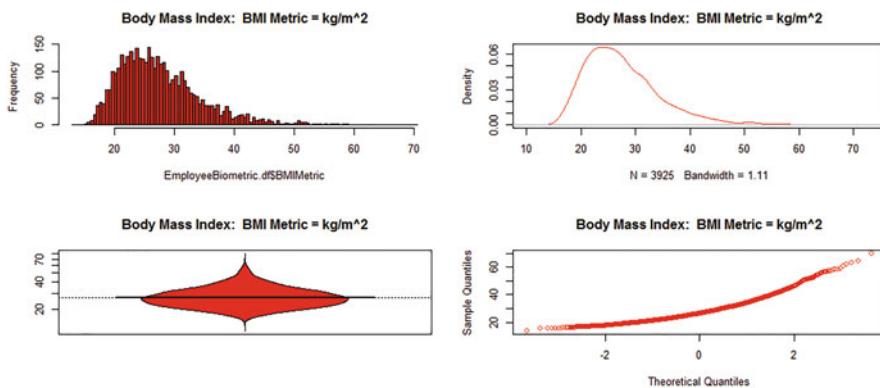


Figure 7.6: Distribution of numeric-type object variable body mass index

These figures, figures for both factor-type object variables and numeric-type object variables, provide a fairly good sense of the data. Other functions could have just as well be used to prepare the figures, with the `ggplot2::ggplot()` function a common choice, but do not overlook the wide variety of options available with R when producing graphics (Figs. 7.5 and 7.6).

7.3.2 Descriptive Statistics for Initial Analysis of the Data

The prior graphical presentations provide an initial sense of the wellness-focused data found in the `EmployeeBiometric.df` dataset. Even with guidance from the many figures, descriptive statistics are needed to gain a better understanding of the data and an understanding of the association, if any, between and among the variables and ultimately how future outcomes might be predicted based on known extant data.

Use the `summary()` function, again, to gain a broad view of the dataset. However, to better organize output, use the `summary()` function with [Row,Column] arguments so that one application of the `summary()` function is used against factor-type object variables and another application of the `summary()` function is used against numeric-type object variables:

R Input

```
summary(EmployeeBiometric.df[,c(2,4,9,10)])
# Column 1 ... EmployeeID was ignored
# Column 2 ... Gender
# Column 4 ... RaceEthnicity
# Column 9 ... BMIStatus
# Column 10 .. Obesity
# Factor object variables
```

R Output

Gender	RaceEthnicity	BMIStatus
Female:2078	Asian : 482	Underweight : 168
Male :1867	Black : 818	Normal Weight:1437
	Hispanic: 904	Overweight :1114
	Other : 117	Obese :1206
	White :1624	NA's : 20
 Obesity		
Not Obese:2719		
Obese :1206		
NA's : 20		

R Input

```
summary(EmployeeBiometric.df[,c(3,5,6,7,8)])
# Column 3 ... AgeYears
# Column 5 ... TotalCholesterolmgdL
# Column 6 ... SBPmmHg
# Column 7 ... DBPmmHg
# Column 8 ... BMIMetric
# Numeric object variables
```

R Output

AgeYears	TotalCholesterolmgdL	SBPmmHg
Min. :21.0	Min. : 78	Min. : 74
1st Qu.:34.0	1st Qu.:151	1st Qu.:108
Median :47.0	Median :175	Median :118
Mean :47.7	Mean :180	Mean :122
3rd Qu.:61.0	3rd Qu.:205	3rd Qu.:132
Max. :79.0	Max. :380	Max. :222
NA's :894	NA's :209	NA's :209

DBPmmHg	BMIMetric
Min. : 34.0	Min. :13.4
1st Qu.: 60.0	1st Qu.:22.6
Median : 70.0	Median :26.4
Mean : 69.3	Mean :27.7
3rd Qu.: 78.0	3rd Qu.:31.3
Max. :122.0	Max. :70.1
NA's :217	NA's :20

There are 10 object variables associated with `EmployeeBiometric.df`, but recall that the object variable `EmployeeID` is used for tracking purposes only, if needed. Of the remaining object variables, consider how frequency distributions tend to be the most common method for describing factor-type object variables (e.g., `Gender`, `RaceEthnicity`, `BMIStatus`, and `Obesity`) whereas descriptive statistics and measures of central tendency such as `Mean`, `Standard Deviation`, `Median`, etc., tend to be the most common method for describing numeric-type object variables (e.g., `AgeYears`, `TotalCholesterolmgdL`, `SBPmmHg`, `DBPmmHg`, and the calculated value for `BMIMetric`). The `summary()` function may not provide all desired descriptive statistics and measures of central tendency, but it is certainly a good starting point for this task.

Frequency Distributions of Factor-Type Object Variables

Going beyond use of the `summary()` function, from among the many R-based functions that could be used to easily generate a frequency distribution of a factor-type object variable, it may be best to merely revisit the `epiDisplay::tab1()` function, now with the `graph` argument set to `FALSE`. The frequency (e.g., `N`) and percentage for each breakout group (including missing data, when appropriate) are printed to the screen and this output can be easily copied from R and pasted into a word-processed document for later use.

R Input

```
epiDisplay::tab1(EmployeeBiometric.df$Gender,
graph=FALSE)
```

R Output

	Frequency	Percent	Cum. percent
Female	2078	52.7	52.7
Male	1867	47.3	100.0
Total	3945	100.0	100.0

R Input

```
epiDisplay::tab1(EmployeeBiometric.df$RaceEthnicity,
graph=FALSE)
```

R Output

	Frequency	Percent	Cum. percent
Asian	482	12.2	12.2
Black	818	20.7	33.0
Hispanic	904	22.9	55.9
Other	117	3.0	58.8
White	1624	41.2	100.0
Total	3945	100.0	100.0

R Input

```
epiDisplay::tab1(EmployeeBiometric.df$BMIStatus,
graph=FALSE)
```

R Output

	Frequency	%(NA+)	%(NA-)
Underweight	168	4.3	4.3
Normal Weight	1437	36.4	36.6
Overweight	1114	28.2	28.4

Obese	1206	30.6	30.7
NA's	20	0.5	0.0
Total	3945	100.0	100.0

R Input

```
epiDisplay::tab1(EmployeeBiometric.df$Obesity,
graph=FALSE)
```

R Output

	Frequency	%(NA+)	%(NA-)
Not Obese	2719	68.9	69.3
Obese	1206	30.6	30.7
NA's	20	0.5	0.0
Total	3945	100.0	100.0

Measures of Central Tendency for Numeric-Type Object Variables

The summary() function is also a frequent first choice for obtaining descriptive statistics of a numeric-type object variable. However, a limitation of the summary() function for numeric object variables is that Standard Deviation is not provided in the default output. To address this concern and the need for attention to Standard Deviation as a vital descriptive statistic, consider use of the RcmdrMisc::numSummary() function, which by default provides Mean, Standard Deviation, IQR (Interquartile Range, or Q3 minus Q1) Quartiles (0% (Minimum), 25%, 50% (Median), 75%, 100% (Maximum)) N, and NAs (the number of missing values, if any).

R Input

```
install.packages("RcmdrMisc", dependencies=TRUE)
library(RcmdrMisc)           # Load the RcmdrMisc package.
help(package=RcmdrMisc)      # Show the information page.
sessionInfo()                # Confirm all attached packages.

RcmdrMisc::numSummary(EmployeeBiometric.df$AgeYears)
```

R Output

```
mean      sd IQR 0% 25% 50% 75% 100%     n
47.6563 15.8221 27 21 34 47 61    79 3945
```

R Input

```
RcmdrMisc::numSummary(
  EmployeeBiometric.df$TotalCholesterolmgdL)
```

R Output

```
mean      sd IQR 0% 25% 50% 75% 100%     n   NA
179.835 39.9184 54 78 151 175 205 380 3051 894
```

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df$SBPmmHg)
```

R Output

```
mean      sd IQR 0% 25% 50% 75% 100%     n   NA
121.646 18.6069 24 74 108 118 132 222 3736 209
```

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df$DBPmmHg)
```

R Output

```
mean      sd IQR 0% 25% 50% 75% 100%     n   NA
69.3192 12.836 18 34 60 70 78 122 3728 217
```

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df$BMIMetric)
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
27.7057	7.12103	8.65	13.45	22.61	26.38	31.26	70.08	3925	20

The RcmdrMisc::numSummary() function can also be used to display measures of central tendency at the factor-type breakout group level for numeric-type object variables:

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df[,c("AgeYears",
  "TotalCholesterolmgdL", "SBPmmHg", "DBPmmHg", "BMIMetric")],
  groups=Gender) # Accept default printout
```

R Output

[Selected output (e.g., IQR) is not shown, to save space.]

Variable: AgeYears

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Female	47.6925	15.7969	27	21	34	47	61	79	2078	0
Male	47.6160	15.8542	27	21	34	47	61	79	1867	0

Variable: TotalCholesterolmgdL

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Female	179.768	40.5027	56.0	78	150	176	206.0	380	1628	450
Male	179.911	39.2533	52.5	85	151	175	203.5	348	1423	444

Variable: SBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Female	119.390	18.0147	20	82	108	116	128	222	1976	102
Male	124.178	18.9374	26	74	110	122	136	220	1760	107

Variable: DBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Female	68.2222	12.604	16	34	60	68	76	122	1971	107
Male	70.5498	12.985	18	34	62	72	80	114	1757	110

Variable: BMIMetric

	mean	sd	0%	25%	50%	75%	100%	n	NA
Female	27.6231	7.17061	15.41	22.435	26.18	31.18	64.15	2067	11
Male	27.7976	7.06625	13.45	22.750	26.56	31.29	70.08	1858	9

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df[,c("AgeYears",
  "TotalCholesterolmgdL", "SBPmmHg", "DBPmmHg", "BMIMetric")],
  groups=RaceEthnicity) # Accept default printout
```

R Output

[Selected output is not shown, to save space.]

Variable: AgeYears

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Asian	47.0353	14.7131	24	21	35	46	59	79	482	0
Black	48.2433	16.1025	28	21	34	49	62	79	818	0
Hispanic	46.8584	15.3455	26	21	35	46	61	79	904	0
Other	38.6068	13.3177	20	21	28	36	48	78	117	0
White	48.6410	16.2003	27	21	35	48	62	79	1624	0

Variable: TotalCholesterolmgdL

	mean	sd	0%	25%	50%	75%	100%	n	NA
Asian	178.809	40.5190	95	150	175.0	202.00	324	362	120
Black	181.892	40.7727	78	152	177.0	206.75	380	646	172
Hispanic	178.557	38.1856	82	150	174.0	202.50	309	687	217
Other	176.864	38.7806	109	147	171.5	206.00	292	88	29
White	179.979	40.3126	82	151	176.0	205.00	347	1268	356

Variable: SBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Asian	119.062	18.0030	24	82	106	116	130	222	454	28
Black	125.683	18.4081	24	90	112	122	136	200	769	49
Hispanic	122.948	18.4694	22	88	110	120	132	214	863	41
Other	122.000	18.0343	25	90	108	118	133	198	111	6
White	119.635	18.6154	24	74	106	116	130	220	1539	85

Variable: DBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Asian	67.6380	12.6124	18	38	58	68	76	104	453	29
Black	72.5312	12.9211	16	34	64	74	80	112	768	50
Hispanic	70.5128	13.0697	18	34	62	72	80	122	862	42
Other	68.8649	12.2121	16	34	62	70	78	100	111	6
White	67.5698	12.3939	18	34	58	68	76	116	1534	90

Variable: BMIMetric

	mean	sd	25%	50%	75%	100%	n	NA
Asian	27.9289	7.45770	22.2075	26.325	32.2225	58.34	476	6
Black	27.4193	6.73504	22.8300	26.250	30.8900	64.15	813	5
Hispanic	27.6545	6.99525	22.6600	26.375	31.3275	62.19	902	2
Other	28.5462	6.87024	23.8900	26.690	31.6400	51.24	117	0
White	27.7517	7.29454	22.5100	26.490	31.1300	70.08	1617	7

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df[,c("AgeYears",
  "TotalCholesterolmgdL", "SBPmmHg", "DBPmmHg", "BMIMetric")]),
  groups=BMISstatus) # Accept default printout
```

R Output

[Selected output is not shown, to save space.]

Variable: AgeYears

	mean	sd	25%	50%	75%	100%	n	NA
Underweight	45.9583	15.3289	32	46	59.25	77	168	0
Normal Weight	47.7537	15.8668	34	47	61.00	79	1437	0
Overweight	47.9282	15.7987	35	48	61.75	79	1114	0
Obese	47.4677	15.8322	34	47	60.75	79	1206	0

Variable: TotalCholesterolmgdL

	mean	sd	50%	75%	100%	n	NA
Underweight	179.354	39.8540	173.0	202.5	326	127	41
Normal Weight	179.461	41.0036	174.0	204.0	380	1088	349
Overweight	181.183	40.0782	177.0	207.0	347	871	243
Obese	179.254	38.6444	176.5	203.0	324	950	256

Variable: SBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Underweight	113.045	18.6707	16	82	102	110	118	208	157	11
Normal Weight	116.622	18.1826	20	76	104	114	124	220	1363	74
Overweight	123.802	18.5569	24	74	110	120	134	222	1058	56
Obese	126.915	17.1579	24	90	114	126	138	204	1145	61

Variable: DBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Underweight	63.3758	12.8100	18	36	54	64	72	116	157	11

Normal Weight	65.9750	12.3119	16	34	58	66	74	114	1358	79
Overweight	70.6199	12.5282	16	34	62	72	78	122	1055	59
Obese	72.9817	12.4347	16	34	64	74	80	108	1145	61

Variable: BMIMetric

	mean	sd	50%	75%	100%	n	NA
Underweight	17.4923	0.824308	17.58	18.1225	18.59	168	0
Normal Weight	22.1205	1.747179	22.28	23.6500	25.00	1437	0
Overweight	27.3041	1.395345	27.24	28.4375	30.00	1114	0
Obese	36.1543	6.022411	34.19	38.6600	70.08	1206	0

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df[,c("AgeYears",
  "TotalCholesterolmgdL", "SBPmmHg", "DBPmmHg", "BMIMetric")],
  groups=Obesity) # Accept default printout
```

R Output

[Selected output is not shown, to save space.]

Variable: AgeYears

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Not Obese	47.7142	15.8071	27.00	21	34	47	61.00	79	2719	0
Obese	47.4677	15.8322	26.75	21	34	47	60.75	79	1206	0

Variable: TotalCholesterolmgdL

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Not Obese	180.174	40.5399	54	78	151	175.0	205	380	2086	633
Obese	179.254	38.6444	52	82	151	176.5	203	324	950	256

Variable: SBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Not Obese	119.351	18.7503	22	74	106	116	128	222	2578	141
Obese	126.915	17.1579	24	90	114	126	138	204	1145	61

Variable: DBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Not Obese	67.7230	12.6746	18	34	58	68	76	122	2570	149
Obese	72.9817	12.4347	16	34	64	74	80	108	1145	61

Variable: BMIMetric

	mean	sd	0%	25%	50%	75%	100%	n
Not Obese	23.9583	3.37783	13.45	21.365	24.01	26.745	30.00	2719
Obese	36.1543	6.02241	30.01	31.740	34.19	38.660	70.08	1206

There are many more R-based functions available for further diagnostics about measured object variables such as Age, Total Cholesterol, Systolic Blood Pressure, Diastolic Blood Pressure, and Body Mass Index at the breakout level for factor-type object variables. However, the RcmdrMisc::numSummary() function is quite sufficient for most analyses of a large dataset. Notice also how the RcmdrMisc::numSummary() function generates output that is nicely organized and makes a good presentation if copied from R and pasted into a word-processed document.

To give another view of the descriptive statistics gained from use of the summary() function and the RcmdrMisc::numSummary() function, notice below how these outcomes are presented in graphical format, applying a beeswarm plot over a box plot (Figs. 7.7, 7.8, and 7.9).

R Input

```
install.packages("beeswarm")
library(beeswarm) # Load the beeswarm package.
help(package=beeswarm) # Show the information page.
sessionInfo() # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
boxplot(AgeYears ~ Gender, data = EmployeeBiometric.df,
        main = "Age (Years) by Gender", ylab = "Age (Years)",
        cex.axis=1.15, cex.lab=1.15)
beeswarm(AgeYears ~ Gender, data = EmployeeBiometric.df,
         col = c("red", "blue"), add = TRUE, spacing=0.10)
boxplot(AgeYears ~ RaceEthnicity, data = EmployeeBiometric.df,
        main = "Age (Years) by Race Ethnicity", ylab = "Age (Years)",
        cex.axis=1.15, cex.lab=1.15)
beeswarm(AgeYears ~ RaceEthnicity, data = EmployeeBiometric.df,
         col= c("black", "red", "blue", "green", "cyan"), add = TRUE,
         spacing=0.10)
boxplot(AgeYears ~ BMIStatus, data = EmployeeBiometric.df,
        main = "Age (Years) by Body Mass Index",
        ylab = "Age (Years)", cex.axis=1.15, cex.lab=1.15)
beeswarm(AgeYears ~ BMIStatus, data = EmployeeBiometric.df,
         col = c("black", "red", "blue", "green"), add = TRUE,
         spacing=0.10)
boxplot(AgeYears ~ Obesity, data = EmployeeBiometric.df,
        main = "Age (Years) by Obesity", ylab = "Age (Years)",
```

```
cex.axis=1.15, cex.lab=1.15)
beeswarm(AgeYears ~ Obesity, data = EmployeeBiometric.df,
  col = c("red", "blue"), add = TRUE, spacing=0.10)
# A Beeswarm Plot (a one-dimensional scatter plot) is placed
# over a Box Plot, to offer additional detail about the data.
```

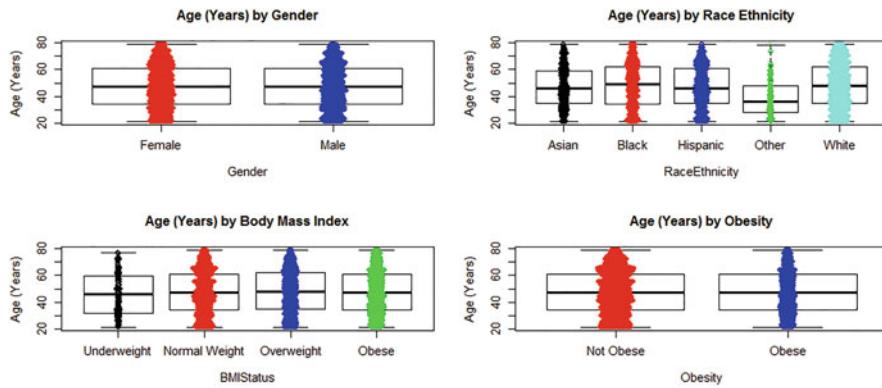


Figure 7.7: Distribution of age by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
boxplot(TotalCholesterolmgdL ~ Gender,
  data = EmployeeBiometric.df,
  main = "Total Cholesterol by Gender",
  ylab = "Total Cholesterol", cex.axis=1.15, cex.lab=1.15)
beeswarm(TotalCholesterolmgdL ~ Gender,
  data = EmployeeBiometric.df, col = c("red", "blue"),
  add = TRUE, spacing=0.10)
boxplot(TotalCholesterolmgdL ~ RaceEthnicity,
  data = EmployeeBiometric.df,
  main = "Total Cholesterol by Race Ethnicity",
  ylab = "Total Cholesterol", cex.axis=1.15, cex.lab=1.15)
beeswarm(TotalCholesterolmgdL ~ RaceEthnicity,
  data = EmployeeBiometric.df,
  col= c("black", "red", "blue", "green", "cyan"), add = TRUE,
  spacing=0.10)
boxplot(TotalCholesterolmgdL ~ BMIStatus,
  data = EmployeeBiometric.df,
```

```

main = "Total Cholesterol by Body Mass Index",
ylab = "Total Cholesterol", cex.axis=1.15, cex.lab=1.15)
beeswarm(TotalCholesterolmgdL ~ BMIStatus,
  data = EmployeeBiometric.df,
  col = c("black", "red", "blue", "green"), add = TRUE,
  spacing=0.10)
boxplot(TotalCholesterolmgdL ~ Obesity,
  data = EmployeeBiometric.df,
  main = "Total Cholesterol by Obesity",
  ylab = "Total Cholesterol", cex.axis=1.15, cex.lab=1.15)
beeswarm(TotalCholesterolmgdL ~ Obesity,
  data = EmployeeBiometric.df, col = c("red", "blue"),
  add = TRUE, spacing=0.10)

```

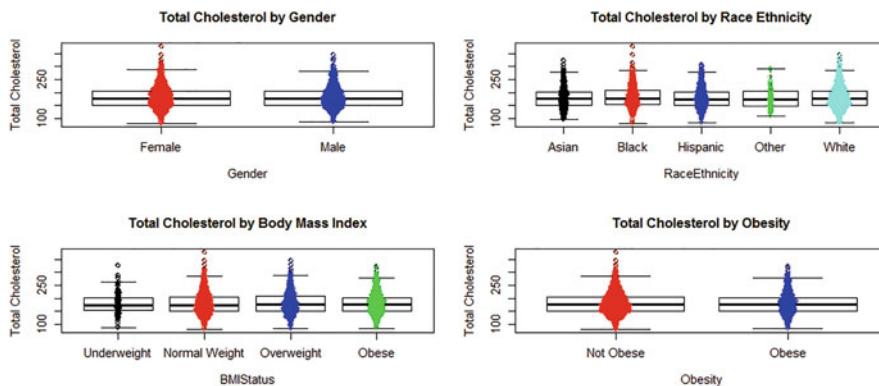


Figure 7.8: Distribution of cholesterol by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

R Input

```

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
boxplot(SBPmmHg ~ Gender, data = EmployeeBiometric.df,
  main = "Systolic Blood Pressure by Gender",
  ylab = "Systolic Blood Pressure", cex.axis=1.15,
  cex.lab=1.15)
beeswarm(SBPmmHg ~ Gender, data = EmployeeBiometric.df,
  col = c("red", "blue"), add = TRUE, spacing=0.10)
boxplot(SBPmmHg ~ RaceEthnicity, data = EmployeeBiometric.df,
  main = "Systolic Blood Pressure by Race Ethnicity",
  ylab = "Systolic Blood Pressure", cex.axis=1.15,
  cex.lab=1.15)
beeswarm(SBPmmHg ~ RaceEthnicity, data = EmployeeBiometric.df,

```

```

col= c("black", "red", "blue", "green", "cyan"), add = TRUE,
spacing=0.10)
boxplot(SBPmmHg ~ BMIStatus, data = EmployeeBiometric.df,
main = "Systolic Blood Pressure by Body Mass Index",
ylab = "Systolic Blood Pressure", cex.axis=1.15,
cex.lab=1.15)
beeswarm(SBPmmHg ~ BMIStatus, data = EmployeeBiometric.df,
col = c("black", "red", "blue", "green"), add = TRUE,
spacing=0.10)
boxplot(SBPmmHg ~ Obesity, data = EmployeeBiometric.df,
main = "Systolic Blood Pressure by Obesity",
ylab = "Systolic Blood Pressure", cex.axis=1.15,
cex.lab=1.15)
beeswarm(SBPmmHg ~ Obesity, data = EmployeeBiometric.df,
col = c("red", "blue"), add = TRUE, spacing=0.10)

```

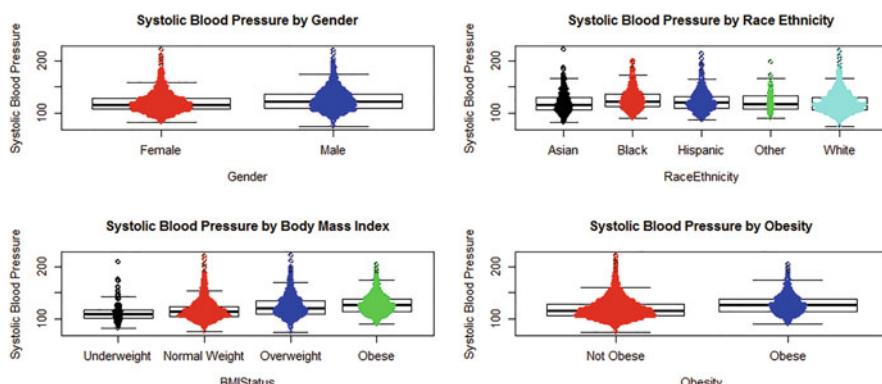


Figure 7.9: Distribution of systolic blood pressure by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

R Input

```

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
boxplot(DBPmmHg ~ Gender, data = EmployeeBiometric.df,
main = "Diastolic Blood Pressure by Gender",
ylab = "Diastolic Blood Pressure", cex.axis=1.15,
cex.lab=1.15)
beeswarm(DBPmmHg ~ Gender, data = EmployeeBiometric.df,
col = c("red", "blue"), add = TRUE, spacing=0.10)
boxplot(DBPmmHg ~ RaceEthnicity, data = EmployeeBiometric.df,
main = "Diastolic Blood Pressure by Race Ethnicity",

```

```

ylab = "Diastolic Blood Pressure", cex.axis=1.15,
cex.lab=1.15)
beeswarm(DBPmmHg ~ RaceEthnicity, data = EmployeeBiometric.df,
col= c("black", "red", "blue", "green", "cyan"), add = TRUE,
spacing=0.10)
boxplot(DBPmmHg ~ BMIStatus, data = EmployeeBiometric.df,
main = "Diastolic Blood Pressure by Body Mass Index",
ylab = "Diastolic Blood Pressure", cex.axis=1.15,
cex.lab=1.15)
beeswarm(DBPmmHg ~ BMIStatus, data = EmployeeBiometric.df,
col = c("black", "red", "blue", "green"), add = TRUE,
spacing=0.10)
boxplot(DBPmmHg ~ Obesity, data = EmployeeBiometric.df,
main = "Diastolic Blood Pressure by Obesity",
ylab = "Diastolic Blood Pressure", cex.axis=1.15,
cex.lab=1.15)
beeswarm(DBPmmHg ~ Obesity, data = EmployeeBiometric.df,
col = c("red", "blue"), add = TRUE, spacing=0.10)

```

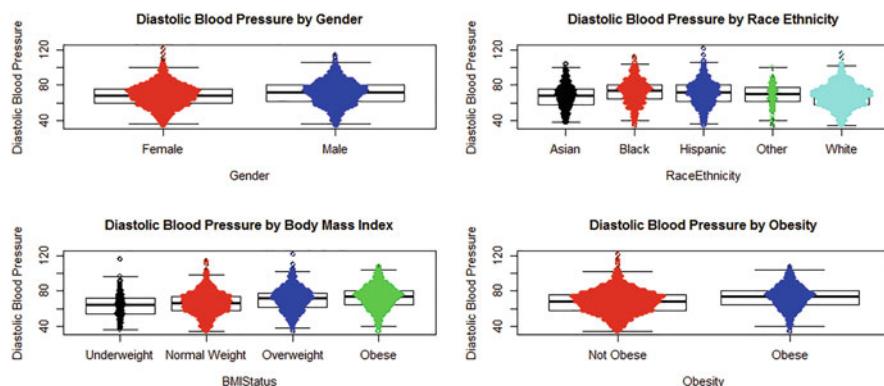


Figure 7.10: Distribution of diastolic blood pressure by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

R Input

```

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
boxplot(BMIMetric ~ Gender, data = EmployeeBiometric.df,
main = "Body Mass Index by Gender", ylab = "BMI Calculated",
cex.axis=1.15, cex.lab=1.15)
beeswarm(BMIMetric ~ Gender, data = EmployeeBiometric.df,
col = c("red", "blue"), add = TRUE, spacing=0.10)
boxplot(BMIMetric ~ RaceEthnicity, data = EmployeeBiometric.df,

```

```

main = "Body Mass Index by Race Ethnicity",
ylab = "BMI Calculated", cex.axis=1.15, cex.lab=1.15)
beeswarm(BMIMetric ~ RaceEthnicity,
  data = EmployeeBiometric.df,
  col= c("black", "red", "blue", "green", "cyan"), add = TRUE,
  spacing=0.10)
boxplot(BMIMetric ~ BMISatus, data = EmployeeBiometric.df,
  main = "BMI Group by BMI Calculated",
  ylab = "BMI Calculated", cex.axis=1.15, cex.lab=1.15)
beeswarm(BMIMetric ~ BMISatus, data = EmployeeBiometric.df,
  col = c("black", "red", "blue", "green"), add = TRUE,
  spacing=0.10)
boxplot(BMIMetric ~ Obesity, data = EmployeeBiometric.df,
  main = "Body Mass Index by Obesity",
  ylab = "BMI Calculated", cex.axis=1.15, cex.lab=1.15)
beeswarm(BMIMetric ~ Obesity, data = EmployeeBiometric.df,
  col = c("red", "blue"), add = TRUE, spacing=0.10)

```

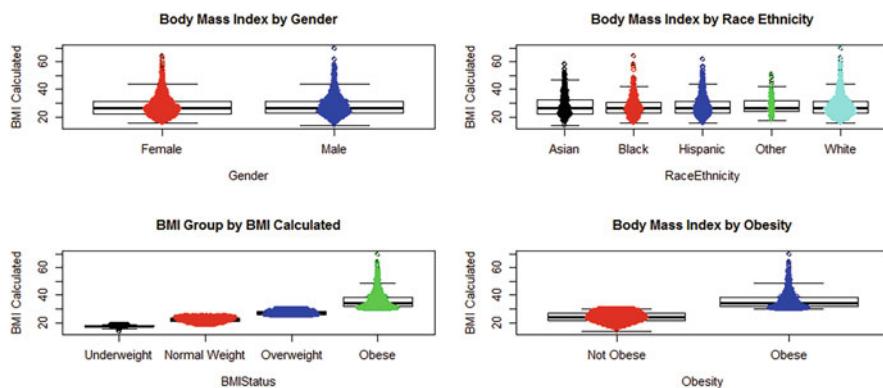


Figure 7.11: Distribution of body mass index by factor-type object variables gender, RaceEthnicity, BMISatus, and obesity

There are many R-based functions that could be used on this dataset to generate measures of central tendency and frequency distributions. The RcmdrMisc::numSummary() function is often a first choice to obtain measures of central tendency such as mean, sd, percentiles, etc.³ The epiDisplay::tab1() function is also valuable since it provides frequency distributions by break-outs of factor-type object variables as well as an accompanying bar chart, if desired. These functions all generate their many statistics in an attractive

³Note the inconsistency in this lesson and all other lessons for capitalization and abbreviation of terms such as Mean—mean, Standard Deviation—SD—standard deviation—sd, etc. It is common to see multiple means of expressing these constructs when reviewing the literature and this approach is purposely used here, to provide broad exposure to what will be seen when reviewing how these terms are expressed by others.

format that can be easily copied from the R session and pasted into a word-processed document, either in original format or formatted by placing some type of L^AT_EXwrapper around the text (e.g., The Hmisc::latex() function is a function that is often used to convert a R-generated object from text format into a format suitable for a L^AT_EXdocument.) (Figs. 7.10 and 7.11).

7.4 Quality Assurance, Data Distribution, and Tests for Normality

Quality assurance (QA) is by no means a simple activity where there is only one attempt to examine the data for quality issues, data distribution patterns, and examination of normality. Quality assurance needs to be inclusive throughout the entire research process, from the first ideas about the research project to the last proofreading of the final report. Attention has gone into the production of figures that offer a graphical sense of the data. Descriptive statistics have been explored from many perspectives. Summative descriptive statistics were provided for the five numeric object variables (e.g., AgeYears, TotalCholesterolmgdL, SBPmmHg, DBPmmHg, and BMIMetric) and for the four factor-type object variables (e.g., Gender, RaceEthnicity, BMIStatus, Obesity, with EmployeeID generally excluded other than to look for duplicates) associated with this lesson.

Quality assurance and data distribution have been examined through the use of multiple functions, to produce both graphical figures and numerical output. Following along with the notion that multiple R-based functions serve similar purposes, there are many statistical tests that provide sound judgment on data distribution and the degree to which there is adherence to or movement away from normality. A partial listing of these tests for normality includes:

- Anderson–Darling test for normality
- Cramer–von Mises test for normality
- Lilliefors (Kolmogorov–Smirnov) test for normality
- Pearson chi-square test for normality
- Shapiro test for normality

For this lesson the Shapiro test will be demonstrated, using the stats::shapiro.test() function.⁴ The stats package is one of the many packages obtained when

⁴For nearly all statistical tests, the Null Hypothesis is worded in a negative fashion and is typically stated as **There is no statistically significant difference between A and B in terms of C**. Somewhat different, the Null Hypothesis for a Shapiro Test of Normality is instead worded in the affirmative and is typically stated as **The data follow a normal distribution**. Give attention to this different approach to the way the Shapiro Test for Normality is worded when interpreting the p-values, significance levels, and outcomes from a Shapiro Test for Normality.

R is first downloaded—a core package. Although the `shapiro.test()` function will be used in this lesson, but it would have been just as accurate to use `stats::shapiro.test()` (e.g., `Package::Function()`) if there were a desire to be more formal.

It is common to focus on either a p-value of either $p \leq 0.05$ or $p \leq 0.01$ when considering normality tests, such as the Shapiro Test for Normality. Due to the way the Shapiro Test for Normality is worded, in the affirmative:

- Lower calculated p-values provide evidence against the null hypothesis. That is to say, if the calculated p-value is less than or equal to the criterion p-value (e.g., significance level), the rules-based decision is to reject the null hypothesis and to declare that the data do not follow a pattern of normal distribution.
- Higher calculated p-values fail to provide evidence against the null hypothesis. That is to say, if the calculated p-value is greater than the criterion p-value (e.g., significance level), the rules-based decision is to accept (e.g., fail to reject) the null hypothesis and to declare that the data seem to follow a pattern of normal distribution.

As demonstrated through this text, many different types of figures and text-based output are used to offer a sense of the data, overall. A key interest in this lesson when viewing the many figures and statistics on central tendency is the concern over normal distribution for numeric object variables, or `AgeYears`, `TotalCholesterolmgdL`, `SBPmmHg`, `DBPmmHg`, and `BMIMetric`. Normal distribution is important in that many inferential tests are only used, appropriately, if the underlying data exhibit normal distribution. Of course, compromised decisions are often made and it not uncommon to see the application of statistical tests that call for the use of normally distributed data, but with data that do not follow a normal distribution pattern—certainly not a perfect normal distribution pattern. The question then is how much deviation away from normal distribution is acceptable before a nonparametric approach to statistical analysis is necessary. A judgment call is then needed, but the Shapiro Test for Normality will at least provide a sense of distribution and from that outcome, whether it is accepted that the data exhibit a reasonable pattern of normal distribution or if normality of the data is questioned.

To address these first-level concerns, apply the `shapiro.test()` function, with no focus on breakouts of the many different factor-type object variables.

R Input

```
shapiro.test(EmployeeBiometric.df$AgeYears)
```

R Output

```
Shapiro-Wilk normality test  
p-value <0.0000000000000002
```

The calculated p-value is less than 0.05, providing evidence that normal distribution is not evident for the object variable AgeYears, overall.

R Input

```
shapiro.test(EmployeeBiometric.df$TotalCholesterolmgdL)
```

R Output

```
Shapiro-Wilk normality test  
p-value <0.0000000000000002
```

The calculated p-value is less than 0.05, providing evidence that normal distribution is not evident for the object variable TotalCholesterolmgdL, overall.

R Input

```
shapiro.test(EmployeeBiometric.df$SBPmmHg)
```

R Output

```
Shapiro-Wilk normality test  
p-value <0.0000000000000002
```

The calculated p-value is less than 0.05, providing evidence that normal distribution is not evident for the object variable SBPmmHg, overall.

R Input

```
shapiro.test(EmployeeBiometric.df$DBPmmHg)
```

R Output

```
Shapiro-Wilk normality test  
p-value = 0.0000000321
```

The calculated p-value is less than 0.05, providing evidence that normal distribution is not evident for the object variable DBPmmHg, overall.

R Input

```
shapiro.test(EmployeeBiometric.df$BMIMetric)
```

R Output

```
Shapiro-Wilk normality test
p-value <0.0000000000000002
```

The calculated p-value is less than 0.05, providing evidence that normal distribution is not evident for the object variable BMIMetric, overall (Fig. 7.12).

Use the qqnorm() function to visually confirm the degree of deviance away from normality of the measured object variables associated with this lesson.

R Input

```
par(ask=TRUE)
par(mfrow=c(2,3)) # 6 figures into a 2 row by 3 column grid
qqnorm(EmployeeBiometric.df$AgeYears, col="red",
       main="Normality of AgeYears - Overall")
qqline(EmployeeBiometric.df$AgeYears, col="blue")
qqnorm(EmployeeBiometric.df$TotalCholesterolmgdL, col="red",
       main="Normality of TotalCholesterolmgdL - Overall")
qqline(EmployeeBiometric.df$TotalCholesterolmgdL, col="blue")
qqnorm(EmployeeBiometric.df$SBPmmHg, col="red",
       main="Normality of SBPmmHg - Overall")
qqline(EmployeeBiometric.df$SBPmmHg, col="blue")
qqnorm(EmployeeBiometric.df$DBPmmHg, col="red",
       main="Normality of DBPmmHg - Overall")
qqline(EmployeeBiometric.df$DBPmmHg, col="blue")
qqnorm(EmployeeBiometric.df$BMIMetric, col="red",
       main="Normality of BMIMetric - Overall")
qqline(EmployeeBiometric.df$BMIMetric, col="blue")
```

The p-values gained from application of the shapiro.test() function and visual confirmation as seen in the Q-Q plots provide ample evidence that the numeric object variables AgeYears, TotalCholesterolmgdL, SBPmmHg, DBPmmHg, and BMIMetric do not follow a normal distribution pattern. A decision will be needed, later, to address this concern and subsequent selection of the appropriate tests, whether these object variables are used appropriately from a parametric perspective or if they are instead more appropriately used from

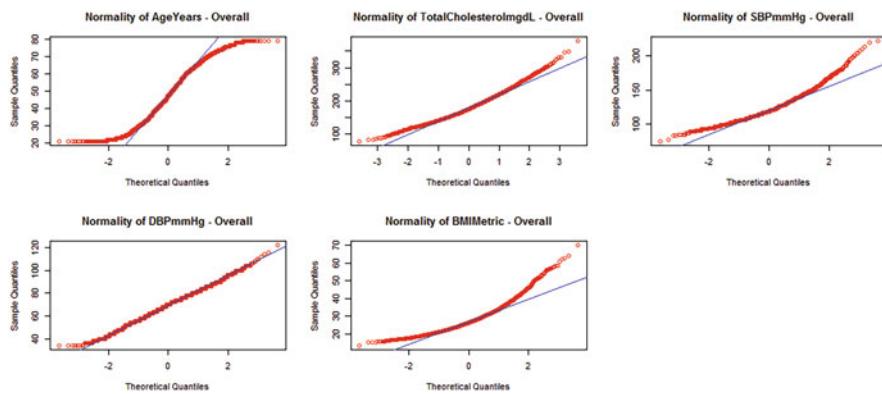


Figure 7.12: Normality of numeric-type object variables AgeYears, TotalCholesterolmgdL, SBPmmHg, DBPmmHg, and BMIMetric

a nonparametric perspective. It is not uncommon to reject an *either-or* approach to correlation analyses and to instead apply both Pearson's r (parametric) and Spearman's rho (nonparametric), but that approach is addressed in a later section of this lesson.

Although normal distribution is not evident at the overall level for each of the five numeric object variables in this lesson, does this statement apply equally to the factor-type breakouts of these five numeric object variables? Perhaps AgeYears does not follow normal distribution, overall, but is it possible that normal distribution is evident for Female participants, but not Male participants? The same type of question could be applied to all factor-type breakouts for each of the five numeric object variables.

The RVAideMemoire::byf.shapiro() function is an excellent choice to examine these questions on normality by breakouts. Observe below how the RVAideMemoire::byf.shapiro() function provides a simple and convenient summary of normality p-values by individual breakouts of the factor-type object variable.

R Input

```
install.packages("RVAideMemoire", dependencies=TRUE)
library(RVAideMemoire)      # Load the RVAideMemoire package.
help(package=RVAideMemoire) # Show the information page.
sessionInfo()               # Confirm all attached packages.

# Normality test of numeric object variables by Gender

RVAideMemoire::byf.shapiro(AgeYears ~ Gender,
                           data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: AgeYears by Gender
      W           p-value
Female 0.9661 < 0.0000000000000022 ***
Male   0.9633 < 0.0000000000000022 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(TotalCholesterolmgdL ~ Gender,
                           data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: TotalCholesterolmgdL by Gender
      W           p-value
Female 0.9806 0.00000000000047433 ***
Male   0.9739 0.0000000000002233 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(SBPmmHg ~ Gender,
                           data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: SBPmmHg by Gender
      W           p-value
Female 0.9318 < 0.0000000000000022 ***
Male   0.9653 < 0.0000000000000022 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(DBPmmHg ~ Gender,  
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests  
data: DBPmmHg by Gender  
      W      p-value  
Female 0.9945 0.000001046 ***  
Male   0.9964 0.0004106 ***  
---  
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(BMIMetric ~ Gender,  
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests  
data: BMIMetric by Gender  
      W      p-value  
Female 0.9206 < 0.00000000000000022 ***  
Male   0.9259 < 0.00000000000000022 ***  
---  
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
# Normality test of numeric object variables by RaceEthnicity  
  
RVAideMemoire::byf.shapiro(AgeYears ~ RaceEthnicity,  
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests  
data: AgeYears by RaceEthnicity
```

	W	p-value
Asian	0.9702	0.00000002483267275 ***
Black	0.9616	0.00000000000008035 ***
Hispanic	0.9648	0.00000000000005845 ***
Other	0.9336	0.00002059768431550 ***
White	0.9625	< 0.00000000000000022 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1		

R Input

```
RVAideMemoire::byf.shapiro(TotalCholesterolmgdL ~
  RaceEthnicity, data=EmployeeBiometric.df)
```

R Output

Shapiro-Wilk normality tests
 data: TotalCholesterolmgdL by RaceEthnicity

	W	p-value
Asian	0.9784	0.0000299780295813 ***
Black	0.9757	0.0000000073145531 ***
Hispanic	0.9783	0.0000000149272617 ***
Other	0.9642	0.01574 *
White	0.9777	0.0000000000004407 ***

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1		

R Input

```
RVAideMemoire::byf.shapiro(SBPmmHg ~ RaceEthnicity,
  data=EmployeeBiometric.df)
```

R Output

Shapiro-Wilk normality tests
 data: SBPmmHg by RaceEthnicity

	W	p-value
Asian	0.9415	0.00000000000222558 ***
Black	0.9551	0.0000000000001435 ***
Hispanic	0.9462	< 0.00000000000000022 ***
Other	0.9320	0.00002641595847946 ***

```
White      0.9448 < 0.0000000000000022 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(DBPmmHg ~ RaceEthnicity,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: DBPmmHg by RaceEthnicity
      W     p-value
Asian   0.9939 0.0657746 .
Black   0.9935 0.0020140 **
Hispanic 0.9942 0.0020703 **
Other   0.9736 0.0263387 *
White   0.9961 0.0005367 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(BMIMetric ~ RaceEthnicity,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: BMIMetric by RaceEthnicity
      W           p-value
Asian   0.9373    0.0000000000002847 ***
Black   0.9295 < 0.0000000000000022 ***
Hispanic 0.9266 < 0.0000000000000022 ***
Other   0.9217    0.0000038873197156 ***
White   0.9118 < 0.0000000000000022 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
# Normality test of numeric object variables by BMIStatus  
  
RVAideMemoire::byf.shapiro(AgeYears ~ BMIStatus,  
    data=EmployeeBiometric.df)
```

R Output

```

Shapiro-Wilk normality tests
data: AgeYears by BMIStatus
      W           p-value
Underweight 0.9526 0.0000193201320965490 ***
Normal Weight 0.9651 < 0.00000000000000022 ***
Overweight   0.9643 0.0000000000000006834 ***
Obese        0.9645 < 0.00000000000000022 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

R Input

```
RVAideMemoire::byf.shapiro(TotalCholesterolmgdL ~ BMIStatus,  
    data=EmployeeBiometric.df)
```

R Output

```

Shapiro-Wilk normality tests
data: TotalCholesterolmgdL by BMISstatus
      W          p-value
Underweight 0.9780      0.03646 *
Normal Weight 0.9710 0.00000000000006071 ***
Overweight   0.9785 0.00000000050969220 ***
Obese        0.9847 0.00000001972027389 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

R Input

```
RVAideMemoire::byf.shapiro(SBPmmHg ~ BMIStatus,  
    data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: SBPmmHg by BMIStatus
      W          p-value
Underweight 0.8420 0.00000000001006922 ***
Normal Weight 0.9122 < 0.00000000000000022 ***
Overweight   0.9447 < 0.00000000000000022 ***
Obese        0.9702 0.00000000000001384 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(DBPmmHg ~ BMIStatus,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: DBPmmHg by BMIStatus
      W          p-value
Underweight 0.9792 0.0180324 *
Normal Weight 0.9943 0.00005119 ***
Overweight   0.9947 0.0009071 ***
Obese        0.9944 0.0002697 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(BMIMetric ~ BMIStatus,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: BMIMetric by BMIStatus
      W          p-value
Underweight 0.9201      0.00000005606 ***
Normal Weight 0.9626 < 0.00000000000000022 ***
Overweight   0.9590 < 0.00000000000000022 ***
```

```
Obese           0.8329 < 0.0000000000000022 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
# Normality test of numeric object variables by Obesity

RVAideMemoire::byf.shapiro(AgeYears ~ Obesity,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: AgeYears by Obesity
      W                  p-value
Not Obese 0.9647 < 0.0000000000000022 ***
Obese     0.9645 < 0.0000000000000022 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(TotalCholesterolmgdL ~ Obesity,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: TotalCholesterolmgdL by Obesity
      W                  p-value
Not Obese 0.9754 < 0.0000000000000022 ***
Obese     0.9847       0.00000001972 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(SBPmmHg ~ Obesity,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: SBPmmHg by Obesity
      W           p-value
Not Obese 0.9298 < 0.0000000000000022 ***
Obese     0.9702  0.00000000000001384 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(DBPmmHg ~ Obesity,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: DBPmmHg by Obesity
      W           p-value
Not Obese 0.9953 0.000000308 ***
Obese     0.9944 0.0002697 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
RVAideMemoire::byf.shapiro(BMIMetric ~ Obesity,
  data=EmployeeBiometric.df)
```

R Output

```
Shapiro-Wilk normality tests
data: BMIMetric by Obesity
      W           p-value
Not Obese 0.9789 < 0.0000000000000022 ***
Obese     0.8329 < 0.0000000000000022 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Based on the p-values from these many Shapiro normality tests, it seems that normality is absent, overall and by breakouts of each factor-type object vari-

able. For additional confirmation, use the `ggplot2::ggplot()` function to produce QQ plots of the numeric object variables (e.g., `AgeYears`, `TotalCholesterolmgdL`, `SBPmmHg`, `DBPmmHg`, and `BMIIMetric`) by breakouts of each of the four factor-type object variables (e.g., `Gender`, `RaceEthnicity`, `BMIStatus`, and `Obesity`) (Fig. 7.13, 7.14, 7.15, and 7.16).

R Input

```

install.packages("ggplot2", dependencies=TRUE)
library(ggplot2)          # Load the ggplot2 package.
help(package=ggplot2)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("ggthemes", dependencies=TRUE)
library(ggthemes)          # Load the ggthemes package.
help(package=ggthemes)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("ggmosaic", dependencies=TRUE)
library(ggmosaic)          # Load the ggmosaic package.
help(package=ggmosaic)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("gridExtra", dependencies=TRUE)
library(gridExtra)          # Load the gridExtra package.
help(package=gridExtra)      # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("grid", dependencies=TRUE)
library(grid)               # Load the grid package.
help(package=grid)          # Show the information page.
sessionInfo()             # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)              # Load the scales package.
help(package=scales)         # Show the information page.
sessionInfo()             # Confirm all attached packages.

# QQ Plot AgeYears by Factor Breakouts

QQAgeYearsbyGender <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=AgeYears)) +
stat_qq(color="red") +

```

```

stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Gender) +
ggtitle("QQ - AgeYears")
QQAgeYearsbyRaceEthnicity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=AgeYears)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ RaceEthnicity) +
ggtitle("QQ - AgeYears")
QQAgeYearsbyBMISstatus <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=AgeYears)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ BMISstatus) +
ggtitle("QQ - AgeYears")
QQAgeYearsbyObesity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=AgeYears)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Obesity) +
ggtitle("QQ - AgeYears")

par(ask=TRUE); gridExtra::grid.arrange(
QQAgeYearsbyGender,
QQAgeYearsbyRaceEthnicity,
QQAgeYearsbyBMISstatus,
QQAgeYearsbyObesity, ncol=2)

```

R Input

```

# QQ Plot TotalCholesterolmgdL by Factor Breakouts

QQTotalCholesterolmgdLbyGender <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=TotalCholesterolmgdL)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Gender) +
ggtitle("QQ - TotalCholesterolmgdL")

```

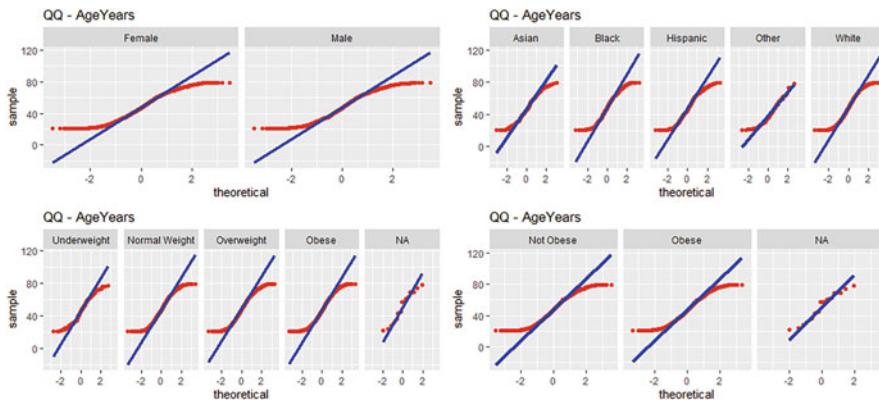


Figure 7.13: Normality of age by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

```

QQTotalCholesterolmgdLbyRaceEthnicity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=TotalCholesterolmgdL)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ RaceEthnicity) +
  ggtitle("QQ - TotalCholesterolmgdL")
QQTotalCholesterolmgdLbyBMISstatus <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=TotalCholesterolmgdL)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ BMISstatus) +
  ggtitle("QQ - TotalCholesterolmgdL")
QQTotalCholesterolmgdLbyObesity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=TotalCholesterolmgdL)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ Obesity) +
  ggtitle("QQ - TotalCholesterolmgdL")

par(ask=TRUE); gridExtra::grid.arrange(
  QQTotalCholesterolmgdLbyGender,
  QQTotalCholesterolmgdLbyRaceEthnicity,
  QQTotalCholesterolmgdLbyBMISstatus,
  QQTotalCholesterolmgdLbyObesity, ncol=2)

```

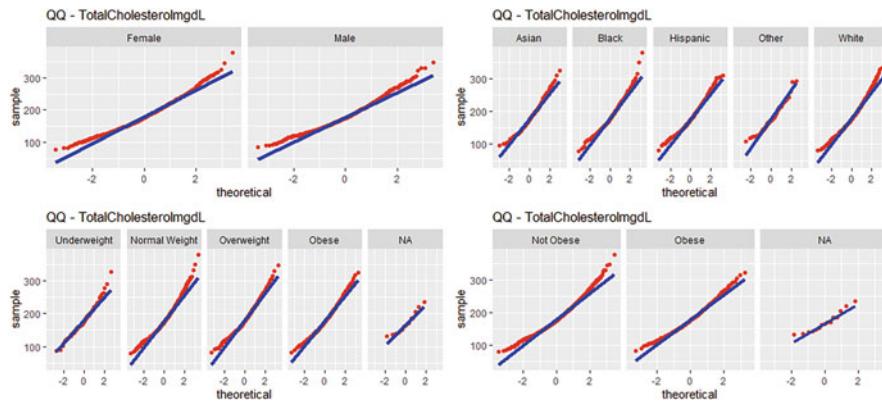


Figure 7.14: Normality of cholesterol by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

R Input

```
# QQ Plot SBPmmHg by Factor Breakouts

QQSBPmmHgbyGender <-
ggplot2::ggplot(EmployeeBiometric.df,
  aes(sample=SBPmmHg)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ Gender) +
  ggtitle("QQ - SBPmmHg")
QQSBPmmHgbyRaceEthnicity <-
ggplot2::ggplot(EmployeeBiometric.df,
  aes(sample=SBPmmHg)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ RaceEthnicity) +
  ggtitle("QQ - SBPmmHg")
QQSBPmmHgbyBMISatus <-
ggplot2::ggplot(EmployeeBiometric.df,
  aes(sample=SBPmmHg)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ BMISatus) +
  ggtitle("QQ - SBPmmHg")
QQSBPmmHgbyObesity <-
ggplot2::ggplot(EmployeeBiometric.df,
  aes(sample=SBPmmHg)) +
  stat_qq(color="red") +
```

```

stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Obesity) +
ggtitle("QQ - SBPmmHg")

par(ask=TRUE); gridExtra::grid.arrange(
QQSBPmmHgbyGender,
QQSBPmmHgbyRaceEthnicity,
QQSBPmmHgbyBMIStatus,
QQSBPmmHgbyObesity, ncol=2)

```

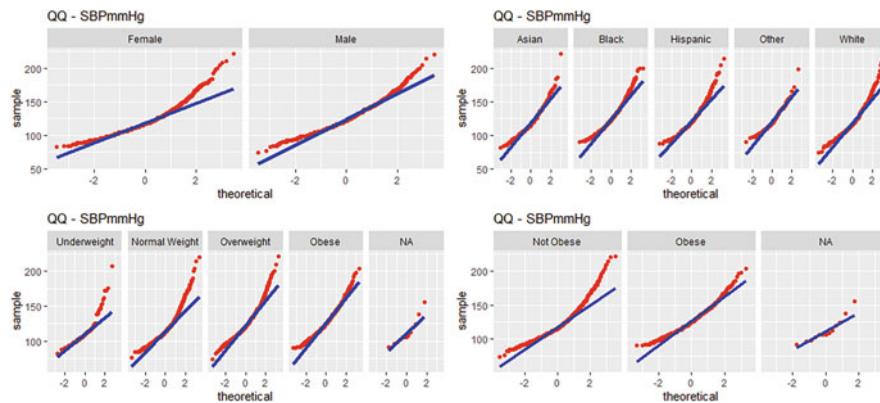


Figure 7.15: Normality of systolic blood pressure by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

R Input

```

# QQ Plot DBPmmHg by Factor Breakouts

QQDBPmmHgbyGender <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=DBPmmHg)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Gender) +
ggtitle("QQ - DBPmmHg")
QQDBPmmHgbyRaceEthnicity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=DBPmmHg)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ RaceEthnicity) +
ggtitle("QQ - DBPmmHg")

```

```

QQDBPmmHgbyBMISatus <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=DBPmmHg)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ BMISatus) +
ggtitle("QQ - DBPmmHg")

QQDBPmmHgbyObesity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=DBPmmHg)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Obesity) +
ggtitle("QQ - DBPmmHg")

par(ask=TRUE); gridExtra::grid.arrange(
QQDBPmmHgbyGender,
QQDBPmmHgbyRaceEthnicity,
QQDBPmmHgbyBMISatus,
QQDBPmmHgbyObesity, ncol=2)

```

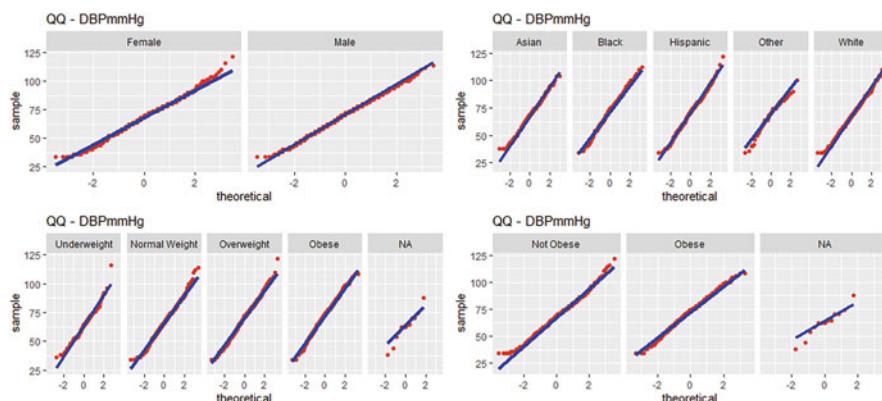


Figure 7.16: Normality of diastolic blood pressure by factor-type object variables gender, RaceEthnicity, BMISatus, and obesity

R Input

```

# QQ Plot BMIMetric by Factor Breakouts

QQBMIMetricbyGender <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=BMIMetric)) +

```

```

stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Gender) +
ggtitle("QQ - BMIMetric")
QQBMIMetricbyRaceEthnicity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=BMIMetric)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ RaceEthnicity) +
ggtitle("QQ - BMIMetric")
QQBMIMetricbyBMIMetric <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=BMIMetric)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ BMIStatus) +
ggtitle("QQ - BMIMetric")
QQBMIMetricbyObesity <-
ggplot2::ggplot(EmployeeBiometric.df,
aes(sample=BMIMetric)) +
stat_qq(color="red") +
stat_qq_line(color="blue", size=1.75) +
facet_grid(. ~ Obesity) +
ggtitle("QQ - BMIMetric")

par(ask=TRUE); gridExtra::grid.arrange(
QQBMIMetricbyGender,
QQBMIMetricbyRaceEthnicity,
QQBMIMetricbyBMIMetric,
QQBMIMetricbyObesity, ncol=2)

```

Using the Shapiro Normality Test against the numeric object variables, overall and by breakouts of the factor-type object variables, it is evident that normal distribution is a concern. The problem is especially evident at the tails of the QQ plots. Again, judgment will be needed to see if the data are appropriately subjected to a parametric approach to correlation analyses, a nonparametric approach to correlation analyses, or both approaches (e.g., parametric and nonparametric) to correlation analyses (Fig. 7.17).

For a different view of the data, generate a ggplot2-based figure of Obesity (a factor-type object variable) by BMIMetric (a numeric-type object variable), but with no missing (e.g., NA) values. Among the many ways to achieve this aim, a fairly simple way is to:

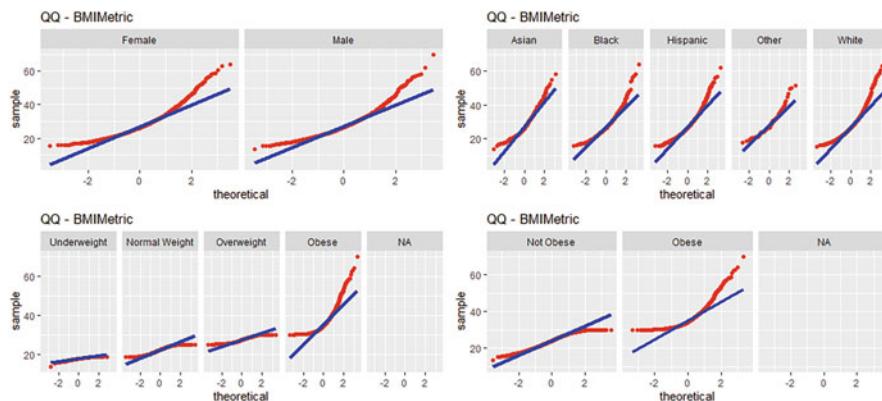


Figure 7.17: Normality of body mass index by factor-type object variables gender, RaceEthnicity, BMIStatus, and obesity

- Create a new dataframe (`EmployeeBiometric2.df`) that is at first equivalent to the original dataframe (`EmployeeBiometric.df`).
- Use `[!is.na(EmployeeBiometric2.df$BMIMetric),]` to remove rows that have a NA value for `EmployeeBiometric2.df$BMIMetric`.⁵

Once the new dataframe is completed, use the `ggplot2::ggplot()` function against the enumerated dataframe—now a dataframe of fewer rows given that the rows that had NA for BMIMetric have been removed.

R Input

```
EmployeeBiometric2.df <- EmployeeBiometric.df
# Create a new dataframe that is equivalent to the
# original dataframe and then remove rows that have a
# NA value for BMIMetric.

nrow(EmployeeBiometric.df)
# Confirm the number of rows of the original dataframe.
```

R Output

```
[1] 3945
```

⁵The tidyverse and more specifically the `tidyverse::drop_na()` function, with appropriate adjustments, could have been used to perhaps more easily drop rows that contain missing values. However, the method used in this lesson is better at showing the heuristics of this action. Again, with appropriate adjustments, do not overlook how the `complete.cases()` function could have also been used to drop rows with missing values, deploying a listwise deletion approach. Rarely is there ever one and only one way to achieve aims when using R.

R Input

```
nrow(EmployeeBiometric2.df)
# Confirm the number of rows of the enumerated dataframe
# before adjustment.
```

R Output

```
[1] 3945
```

R Input

```
EmployeeBiometric2.df <-
  EmployeeBiometric2.df[!is.na(
    EmployeeBiometric2.df$BMIMetric),]
# Remove rows that have NA values for BMIMetric in the
# now adjusted dataframe EmployeeBiometric2.df.

nrow(EmployeeBiometric2.df)
# Confirm the number of rows of the enumerated dataframe
# after adjustment.
```

R Output

```
[1] 3925
```

Present side-by-side Q-Q plots of BMIMetric faceted by Obesity from two perspectives, with NAs and without NAs. This type of presentation, presented here for BMIMetric by Obesity but possibly for all numeric by factor pairs, will be useful as a decision is made on how to account for missing data when presenting outcomes to others.

R Input

```
# Obesity (two breakouts and NAs) by BMIMetric
QQFacetObesityBMIMetricNAs <-
  ggplot2::ggplot(EmployeeBiometric.df,
    aes(sample=BMIMetric)) +
    stat_qq(color="red") +
    stat_qq_line(color="blue", size=1.75) +
    facet_grid(. ~ Obesity) +
    ggttitle("
```

```

BMIMetric QQ-Plot and QQ-Line by Obesity:
Account for NAs (Missing Data)\n") +
  labs(
    x = "\nTheoretical", y = "Body Mass Index Metric\n") +
  theme_bw()
# The dataframe for this figure is EmployeeBiometric.df
# and it is not EmployeeBiometric2.df

# Obesity (two breakouts and no NAs) by BMIMetric
QQFacetObesityBMIMetricNoNAs <-
  ggplot2::ggplot(EmployeeBiometric2.df,
    aes(sample=BMIMetric)) +
    stat_qq(color="red") +
    stat_qq_line(color="blue", size=1.75) +
    facet_grid(. ~ Obesity) +
    ggtitle("

BMIMetric QQ-Plot and QQ-Line by Obesity:
No NAs (Missing Data)\n") +
  labs(
    x = "\nTheoretical", y = "Body Mass Index Metric\n") +
  theme_bw()
# The dataframe for this figure is EmployeeBiometric2.df
# and it is not EmployeeBiometric.df

par(ask=TRUE); gridExtra::grid.arrange(
  QQFacetObesityBMIMetricNAs,
  QQFacetObesityBMIMetricNoNAs, ncol=2)

```

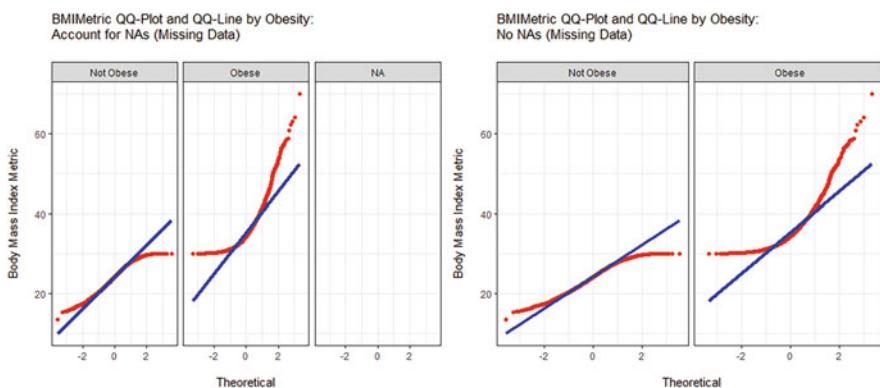


Figure 7.18: Normality of body mass index by obesity and accommodation for missing data

Quality Assurance has been attempted throughout this lesson by generating graphical images of the relationship between and among data and also by providing descriptive statistics and measures of central tendency. Further, normality has been addressed by examining distribution patterns of the measured object variables, overall, and then of the measured object variables by breakouts of the factor-type object variables (Fig. 7.18).

As a general statement, it was observed that normal distribution was not evident in the dataset associated with this lesson. Using $p \leq 0.05$ as the criterion value of acceptance or rejection of each normal distribution Null Hypothesis, overall and by breakouts, the numeric object variables did not follow normal distribution. This finding must be considered when deciding which tests to use for estimating measures of association.⁶ Should a Pearson's r, which is based on a parametric approach toward the data, be used—only? Should a Spearman's rho, which is based on a nonparametric approach toward the data, be used—only? Should both a parametric and nonparametric approach be used to provide estimates of association, with each approach (parametric or nonparametric) confirming or contrasting the other? Ultimately, the aim of these questions is to determine if the association outcomes are in parity, whether using a parametric approach or a nonparametric approach.

7.5 Statistical Test(s)

The terms correlation and association are often used interchangeably, as if they were synonyms.⁷ Regardless of which term is used, estimates of correlation have value based on the premise that *Past behavior is the best predictor of future behavior*. That is to say, if past behavior is known, then correlation analyses may offer a forward-looking view (e.g., a prediction) of future behavior.

Estimates of correlation and association have had a prominent role in statistical analyses since the algorithms were first developed in the late 1800s and early 1900s. However, the concept of how correlation and association can be used for good effect is now even more important since the evolving use of *Big Data*, which has become quite prevalent in business, health care, government services, etc. Even minor associations between seemingly disassociated variables, when discovered and applied iteratively against large numbers of transactions, can be used to improve product development and delivery logistics (e.g., business), patient recovery protocols and therapies (e.g., health care), crop yields and livestock efficiencies (e.g., agriculture), improved deployment of services to taxpayers and municipal residents (e.g., government), etc.

⁶It is common to use the term *estimates* of correlation.

⁷Search on each term to see why *correlation* and *association* are not synonyms, even if many use each term as if they were, and how context has a major role in the appropriate use of each term.

However correlation and association estimates are attempted, it is crucial to once again recall the often repeated phrase *Correlation does not imply causation*. It may be determined that there is an association between Variable X and Variable Y, but that does not mean that Variable X causes Variable Y nor does it mean that Variable Y causes Variable X—it only means that there is an observed degree of association between Variable X and Variable Y.

As a brief reminder about using R and all other software programs used for statistical analysis, correlation analyses can be conducted on a one-by-one basis, determining the association of one variable in relation to another variable. There are ways, however, to conduct multiple correlations at the same time, improving efficiencies of production, presentation, and interpretation. Observe how Pearson's r (parametric) and Spearman's rho (nonparametric) coefficients of correlation are determined, using simple R-based syntax. Then, farther along, look at the way multiple correlation analyses are conducted at the same time, improving efficiencies.

7.5.1 Correlation Using Pearson's r and Spearman's rho

R Input

```
# One-by-one parametric approach to correlation estimates and
# correlation testing

cor(EmployeeBiometric.df$SBPmmHg,
   EmployeeBiometric.df$DBPmmHg,
   use="complete.obs",      # Need to account for NA
   method=c("pearson"))    # Parametric
# Use Pearson's r to obtain an estimate of the
# association between two object variables --
# a parametric approach
```

R Output

```
[1] 0.500096
```

R Input

```
cor.test(EmployeeBiometric.df$SBPmmHg,
          EmployeeBiometric.df$DBPmmHg,
          use="complete.obs",      # Need to account for NA
          method=c("pearson"))    # Parametric
# Observe the calculated p-value to see if the
```

```
# correlation is statistically significant and
# then note how the output ends with the
# coefficient of correlation.
# Use Pearson's r to obtain an estimate of the
# association between two object variables --
# a parametric approach
```

R Output

```
Pearson's product-moment correlation
data: EmployeeBiometric.df$SBPmmHg and
EmployeeBiometric.df$DBPmmHg
p-value <0.000000000000002
cor
0.500096
```

R Input

```
# One-by-one nonparametric approach to correlation estimates
# and correlation testing

cor(EmployeeBiometric.df$SBPmmHg,
EmployeeBiometric.df$DBPmmHg,
use="complete.obs",      # Need to account for NA
method=c("spearman"))   # Nonparametric
# Use Spearman's rho to obtain an estimate of
# the association between two object variables
# -- a nonparametric approach
```

R Output

```
[1] 0.518411
```

R Input

```
cor.test(EmployeeBiometric.df$SBPmmHg,
EmployeeBiometric.df$DBPmmHg,
use="complete.obs",      # Need to account for NA
method=c("spearman"))   # Nonparametric
# Observe the calculated p-value to see if the
# correlation is statisticall significant and
```

```
# then note how the output ends with the  
# coefficient of correlation.  
# Use Spearman's rho to obtain an estimate of  
# the association between two object variables  
# -- a nonparametric approach
```

R Output

```
Spearman's rank correlation rho  
data: EmployeeBiometric.df$SBPmmHg and  
EmployeeBiometric.df$DBPmmHg  
p-value <0.0000000000000002  
sample estimates: rho 0.518411
```

When viewing the syntax for these two estimates of association (e.g., Pearson's r for when data are viewed from a parametric perspective and Spearman's rho for when data are viewed from a nonparametric perspective) between Systolic Blood Pressure (SBPmmHg) and Diastolic Blood Pressure (DMPmmHg), two things deserve immediate attention:

- There is a great deal of parity in the calculated estimate of correlation, with the parametric Pearson's $r = 0.500096$ and the nonparametric Spearman's $\rho = 0.518411$.
- The time-on-task to complete, review, and understand the many one-by-one correlations, using either the `cor()` function or the `cor.test()` function, would be quite demanding. A dataset with multiple numeric variables would likely call for 5, 15, 25, 35, etc., individual correlation estimates—a time-consuming task if all were attempted one-by-one.

Given the need for some degree of efficiency in how the many correlation estimates are obtained, consider using the existing dataframe (`EmployeeBiometric.df`), but dropping all factor-type object variables and keeping only those numeric-type object variables of interest for correlation analyses.

Review the dataframe `EmployeeBiometric.df` (the original dataframe) using the `str()` function:

R Input

```
str(EmployeeBiometric.df)
```

R Output

```
'data.frame': 3945 obs. of 10 variables:
 $ EmployeeID       : Factor w/ 3945 levels "ID10005",
 $ Gender           : Factor w/ 2 levels "Female","Male"
 $ AgeYears         : num  34 67 41 57 35 40 40 34 40 50
 $ RaceEthnicity    : Factor w/ 5 levels "Asian","Black"
 $ TotalCholesterolmgdL: num  221 175 224 217 139 198 NA
 $ SBPmmHg          : num  102 NA NA 90 86 114 NA 92 120
 $ DBPmmHg          : num  44 NA NA 42 42 74 NA 48 80 72
 $ BMIMetric        : num  18.4 18.7 21.6 20.6 18.7 ...
 $ BMIStatus        : Factor w/ 4 levels "Underweight",.
 $ Obesity          : Factor w/ 2 levels "Not Obese",
```

Create a new dataframe, `EmployeeBiometric3.df`, recalling in this naming scheme how `EmployeeBiometric2.df` was previously used to create an enumerated dataframe, and then remove those columns that represent factor-type object variables by using the `subset()` function:

R Input

```
EmployeeBiometric3.df <- EmployeeBiometric.df
# Create a new dataframe (EmployeeBiometric3.df) that
# is equivalent to the existing dataframe
# (EmployeeBiometric.df)

EmployeeBiometric3.df <- subset(EmployeeBiometric3.df,
 select=-c(EmployeeID, Gender, RaceEthnicity, BMIStatus,
 Obesity))
# Remove the columns from EmployeeBiometric3.df that
# represent factor-type data,
# Notice immediately after select= the use of -c when
# using the select argument, with the - symbol used to
# select out (e.g., deselect) the named columns.

str(EmployeeBiometric3.df)
```

R Output

```
'data.frame': 3945 obs. of 5 variables:
 $ AgeYears         : num  34 67 41 57 35 40 40 34 40 50
 $ TotalCholesterolmgdL: num  221 175 224 217 139 198 NA 205
 $ SBPmmHg          : num  102 NA NA 90 86 114 NA 92 120
```

```
$ DBPmmHg           : num  44 NA NA 42 42 74 NA 48 80 72
$ BMIMetric         : num  18.4 18.7 21.6 20.6 18.7
# EmployeeBiometric3.df now consists only of numeric
# object variables, with all factor-type object variables
# removed using the subset() function and select=-c
# argument. There are now five variables, not 10.
```

Anticipating format-type problems when a correlation matrix is presented graphically, it is often desirable to adjust the column names. Typically, long column names are shortened.

R Input

```
colnames(EmployeeBiometric3.df) <-
  c("Age", "Cholesterol", "SBP", "DBP", "BMI")
# Columns names are shortened to accommodate the way
# they will show in later graphical output.

attach(EmployeeBiometric3.df) # Attach the new dataframe

str(EmployeeBiometric3.df)
# Apply the str() function after changing column names.
```

R Output

```
'data.frame': 3945 obs. of 5 variables:
 $ Age        : num  34 67 41 57 35 40 40 34 40 50
 $ Cholesterol: num  221 175 224 217 139 198 NA 205 87 149
 $ SBP        : num  102 NA NA 90 86 114 NA 92 120 126
 $ DBP        : num  44 NA NA 42 42 74 NA 48 80 72
 $ BMI        : num  18.4 18.7 21.6 20.6 18.7
```

Now that the dataframe `EmployeeBiometric3.df` contains only those numeric object variables of interest, apply the `cor()` function again, against the newly created dataframe—a dataframe of numeric-type object variables only. It is necessary to account for missing data, which is problematic with correlation estimates since comparisons between Variable X and Variable Y depend on two datapoints for each comparison, one datapoint for Variable X and another datapoint for Variable Y. Although undesirable, in real-world research missing data are the norm, not the exception.

Observe also how it is necessary to declare the use of either Pearson's r or Spearman's rho, accounting for how the data are viewed, from either a parametric

perspective (e.g., Pearson's r) or a nonparametric perspective (e.g., Spearman's rho). Go back to the prior section where normal distribution was examined to gain a better perspective on this issue and why attention needs to be directed to distribution patterns.

R Input

```
round(cor(EmployeeBiometric3.df,
use="complete.obs",           # Need to account for NA
method=c("pearson")),2)      # Nonparametric
# Generate Pearson's r correlation coefficients
# for multiple variables and wrap the round()
# function around the cor() function so that
# output displays the correlation coefficients at
# two places to the right of the decimal point,
# thus the reason for ,2.
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.00	0.03	-0.01	-0.02	0.00
Cholesterol	0.03	1.00	-0.01	0.02	0.01
SBP	-0.01	-0.01	1.00	0.49	0.24
DBP	-0.02	0.02	0.49	1.00	0.24
BMI	0.00	0.01	0.24	0.24	1.00

R Input

```
round(cor(EmployeeBiometric3.df,
use="complete.obs",           # Need to account for NA
method=c("spearman")),2)    # Nonparametric
# Generate Spearman's rho correlation
# coefficients for multiple variables and wrap
# the round() function around the cor() function
# so that output displays the correlation
# coefficients at two places to the right of the
# decimal point, thus the reason for ,2.
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.00	0.03	0.00	-0.02	-0.01
Cholesterol	0.03	1.00	-0.01	0.02	0.01

SBP	0.00	-0.01	1.00	0.52	0.31
DBP	-0.02	0.02	0.52	1.00	0.26
BMI	-0.01	0.01	0.31	0.26	1.00

Similar to nearly all R-based actions, there are many functions in external packages that address correlation, offering different views on output and related statistics. From among the many possible selections, the Hmisc::rcorr(as.matrix()) functions should be considered. Observe how the as.matrix() function is wrapped around the dataframe, which is needed for applying the Hmisc::rcorr() function, when data are otherwise organized in a dataframe, not a matrix.

R Input

```
install.packages("Hmisc", dependencies=TRUE)
library(Hmisc)                                # Load the Hmisc package.
help(package=Hmisc)                            # Show the information page.
sessionInfo()                                 # Confirm all attached packages.

Hmisc::rcorr(as.matrix(EmployeeBiometric3.df),
             type=c("pearson"))
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.00	0.04	-0.02	-0.02	0.00
Cholesterol	0.04		1.00	-0.01	0.02
SBP	-0.02		-0.01	1.00	0.50
DBP	-0.02		0.02	0.50	1.00
BMI	0.00		0.01	0.25	0.25
					1.00
n					
	Age	Cholesterol	SBP	DBP	BMI
Age	3945		3051	3736	3728
Cholesterol	3051		3051	2894	2886
SBP	3736		2894	3736	3728
DBP	3728		2886	3728	3715
BMI	3925		3036	3723	3715
					3925
P					
	Age	Cholesterol	SBP	DBP	BMI
Age		0.0375		0.2893	0.1792
Cholesterol	0.0375			0.5110	0.3828
SBP	0.2893	0.5110		0.0000	0.0000

DBP	0.1792	0.3828	0.0000	0.0000
BMI	0.8417	0.7192	0.0000	0.0000

R Input

```
Hmisc::rcorr(as.matrix(EmployeeBiometric3.df),  
type=c("spearman"))
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.00	0.03	-0.01	-0.02	0.00
Cholesterol	0.03	1.00	-0.01	0.02	0.00
SBP	-0.01	-0.01	1.00	0.52	0.32
DBP	-0.02	0.02	0.52	1.00	0.27
BMI	0.00	0.00	0.32	0.27	1.00

n

	Age	Cholesterol	SBP	DBP	BMI
Age	3945	3051	3736	3728	3925
Cholesterol	3051	3051	2894	2886	3036
SBP	3736	2894	3736	3728	3723
DBP	3728	2886	3728	3728	3715
BMI	3925	3036	3723	3715	3925

P

	Age	Cholesterol	SBP	DBP	BMI
Age	0.0657		0.6721	0.2951	0.8651
Cholesterol	0.0657		0.5515	0.3028	0.7978
SBP	0.6721	0.5515		0.0000	0.0000
DBP	0.2951	0.3028	0.0000		0.0000
BMI	0.8651	0.7978	0.0000	0.0000	

Output of the Hmisc::rcorr(as.matrix()) functions includes the correlation coefficient, N, and p-value (e.g., significance) for each variable-by-variable comparison. Each section of output has value and all three (e.g., correlation coefficient, N, and p-value) should be reviewed.

There are slight differences in the p-values, depending on whether Pearson's r or Spearman's rho is used to estimate correlation. Although there is a fair degree of parity in general outcomes, the diligent researcher will go back to the figures (QQ plots and QQ analyses) and normality statistics (Shapiro-Wilk normality test) to make individual judgment on whether the data should be viewed from a

parametric perspective or a nonparametric perspective. Of course, the decision on how the data are viewed will be prominently mentioned in the methods and summary section when a final report is prepared and only by peer review will there be final consensus on appropriate conclusions and later applications of outcomes.

There are many ways to calculate correlation coefficients (e.g., either Pearson's r or Spearman's rho), N, and p-values. To be consistent with this text and the way lessons are presented, prepare a visualization of correlation trends between and among the many numeric variables. To achieve this aim, first consider a brute force display of preparing correlation displays for Variable X and Variable Y on a one-by-one bases. Note below one attempt at this action and then consider the complexity of how multiple correlation figures are prepared, one-by-one (Fig. 7.19).

R Input

```
WellnessSBPAge <-
ggplot2::ggplot(EmployeeBiometric3.df,
  aes(x=SBP, y=Age)) +
  geom_point(size=2, shape=19) +
  geom_smooth(method=lm) +
  theme_bw()
# Correlation of SBP by Age

WellnessSBPCholesterol <-
ggplot2::ggplot(EmployeeBiometric3.df,
  aes(x=SBP, y=Cholesterol)) +
  geom_point(size=2, shape=19) +
  geom_smooth(method=lm) +
  theme_bw()
# Correlation of SBP by Cholesterol

WellnessSBPDBP <-
ggplot2::ggplot(EmployeeBiometric3.df,
  aes(x=SBP, y=DBP)) +
  geom_point(size=2, shape=19) +
  geom_smooth(method=lm) +
  theme_bw()
# Correlation of SBP by DBP

WellnessSBPBMI <-
ggplot2::ggplot(EmployeeBiometric3.df,
  aes(x=SBP, y=BMI)) +
```

```

geom_point(size=2, shape=19) +
geom_smooth(method=lm) +
theme_bw()
# Correlation of SBP by BMI

par(ask=TRUE); gridExtra::grid.arrange(
  WellnessSBPAge, WellnessSBPCholesterol,
  WellnessSBPDBP, WellnessSBPBMI, ncol=2)

```

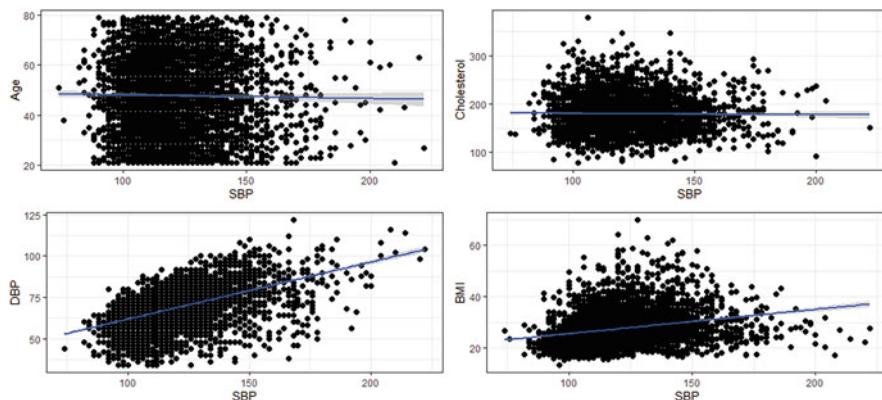


Figure 7.19: Brute force one-by-one display of correlation: SBP by Age, SBP by cholesterol, SBP by DBP, and SBP by BMI

By looking at this figure, there seems to be little to no correlation between SBP and Age and SBP and Cholesterol. However, there does seem to be an association between SBP and DBP and SBP and BMI. Then, review the correlation matrix and confirm that the figures tend to agree with the previously obtained correlation coefficients:

		Age	Cholesterol	SBP	DBP	BMI
# Pearson's r	SBP	-0.02	-0.01	1.00	0.50	0.25
# Spearman's rho	SBP	-0.01	-0.01	1.00	0.52	0.32

This back-and-forth comparison of figures and calculated correlation coefficients is not overly efficient and presents inherent tracking problems. To address this concern, use the `PerformanceAnalytics::chart.Correlation()` function to prepare a correlation matrix that combines graphics, calculated statistics, and the use of asterisks to identify significance levels. As an advance caution, be patient when downloading the `PerformanceAnalytics` package (Figs. 7.20 and 7.21).

R Input

```
install.packages("PerformanceAnalytics", dependencies=TRUE)
library(PerformanceAnalytics)      # Load the package.
help(package=PerformanceAnalytics) # Information page.
sessionInfo()                     # Confirm attached packages.

par.ask=TRUE)
PerformanceAnalytics::chart.Correlation(EmployeeBiometric3.df,
method=c("pearson"), histogram=TRUE, pch = 19)
```

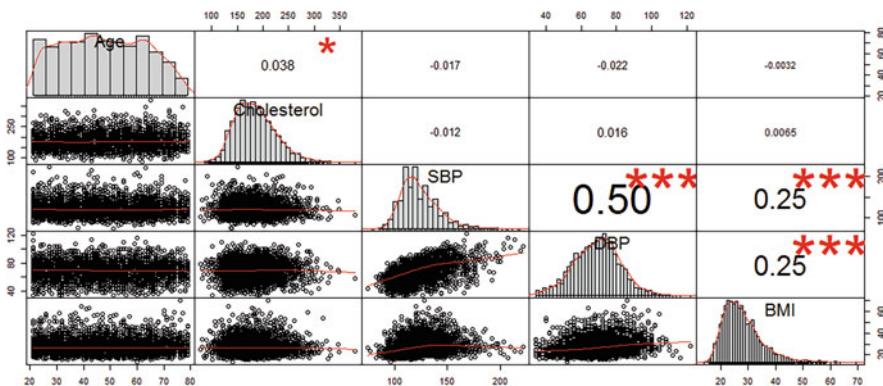


Figure 7.20: Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Pearson’s r

R Input

```
par.ask=TRUE)
PerformanceAnalytics::chart.Correlation(EmployeeBiometric3.df,
method=c("spearman"), histogram=TRUE, pch = 19)
```

Whether using Pearson’s r (e.g., parametric) or Spearman’s rho (e.g., nonparametric) as the selection for method, observe the consistency of outcomes, but now with outcomes presented in graphical format as well as statistical format. Observe the standard means by which statistical significance is marked, using asterisks: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Using these codes, it is evident that the degree of association between SBP:DBP, SBP:BMI, and DBP:BMI are all unequivocally statistically significant, with each comparison showing three *** symbols. In contrast, the degree of statistical significance for Cholesterol:Age is marked by one * symbol. The scatterplots and fit lines, which accompany the figure, also reinforce these outcomes. With these outcomes, it is more than likely that the summary section will focus

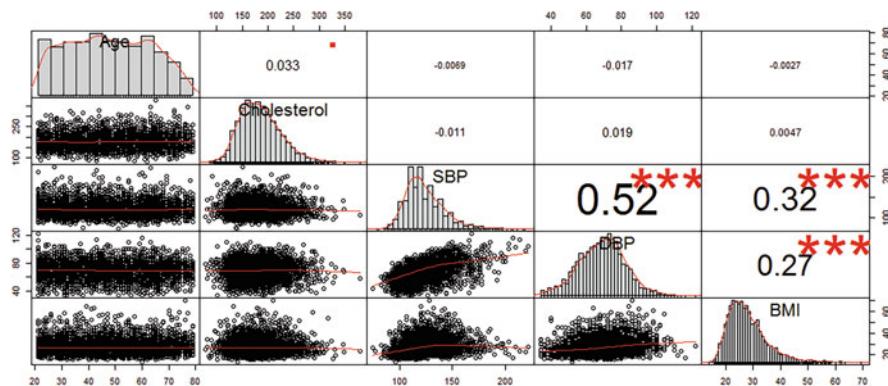


Figure 7.21: Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Spearman’s rho

mostly on how Blood Pressure, whether Systolic Blood Pressure or Diastolic Blood Pressure, has a degree of association with Body Mass Index.

To be efficient with the use of syntax and to reduce keying, create the object variable `WellnessCorPearson` and `WellnessCorSpearman`, each of which represents a correlation matrix:

R Input

```
WellnessCorPearson <- cor(EmployeeBiometric3.df,
  use="complete.obs", method=c("pearson"))

round(WellnessCorPearson, 4)
# Use the round() function to improve clarity,
# going to four places to the right of the
# decimal point.
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.0000	0.0313	-0.0145	-0.0211	-0.0045
Cholesterol	0.0313	1.0000	-0.0148	0.0158	0.0136
SBP	-0.0145	-0.0148	1.0000	0.4903	0.2385
DBP	-0.0211	0.0158	0.4903	1.0000	0.2379
BMI	-0.0045	0.0136	0.2385	0.2379	1.0000

R Input

```
WellnessCorSpearman <- cor(EmployeeBiometric3.df,
  use="complete.obs", method=c("spearman"))

round(WellnessCorSpearman, 4)
# Use the round() function to improve clarity,
# going to four places to the right of the
# decimal point.
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.0000	0.0271	-0.0030	-0.0181	-0.0139
Cholesterol	0.0271	1.0000	-0.0142	0.0184	0.0121
SBP	-0.0030	-0.0142	1.0000	0.5163	0.3116
DBP	-0.0181	0.0184	0.5163	1.0000	0.2591
BMI	-0.0139	0.0121	0.3116	0.2591	1.0000

Consider the `corrplot::corrplot()` function, which allows for many different ways to present the correlation matrix. Observe how different shapes and shading are used to communicate if there is a significant correlation and the degree of correlation. Focus on the figures and determine which meets immediate needs in terms of conveying outcomes to the intended audience. A correlation matrix, with exact correlation coefficients, is also produced along with each figure.

R Input

```
install.packages("corrplot", dependencies=TRUE)
library(corrplot) # Load the corrplot package.
help(package=corrplot) # Show the information page.
sessionInfo() # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(2,3)) # 6 figures into a 2 row by 3 column grid
corrplot::corrplot(WellnessCorPearson,
  method="square", type="upper", order="original",
  tl.srt=30, number.font=2, title="\nPearson")
# Squares indicate significant correlation(s)
corrplot::corrplot(WellnessCorPearson,
  method="color", type="upper", order="original",
  tl.srt=30, number.font=2, title="\nPearson")
# Shading indicates significant correlation(s)
```

```

corrplot::corrplot(WellnessCorPearson,
  method="number", type="upper", order="original",
  tl.srt=30, number.font=2, title="\nPearson")
# Correlation coefficients are produced for
# significant correlations
corrplot::corrplot(WellnessCorSpearman,
  method="square", type="upper", order="original",
  tl.srt=30, number.font=2, title="\nSpearman")
# Squares indicate significant correlation(s)
corrplot::corrplot(WellnessCorSpearman,
  method="color", type="upper", order="original",
  tl.srt=30, number.font=2, title="\nSpearman")
# Shading indicates significant correlation(s)
corrplot::corrplot(WellnessCorSpearman,
  method="number", type="upper", order="original",
  tl.srt=30, number.font=2, title="\nSpearman")
# Correlation coefficients are produced for
# significant correlations
#
# Adjust the angle of column headers as needed,
# using the tl.srt argument.

```

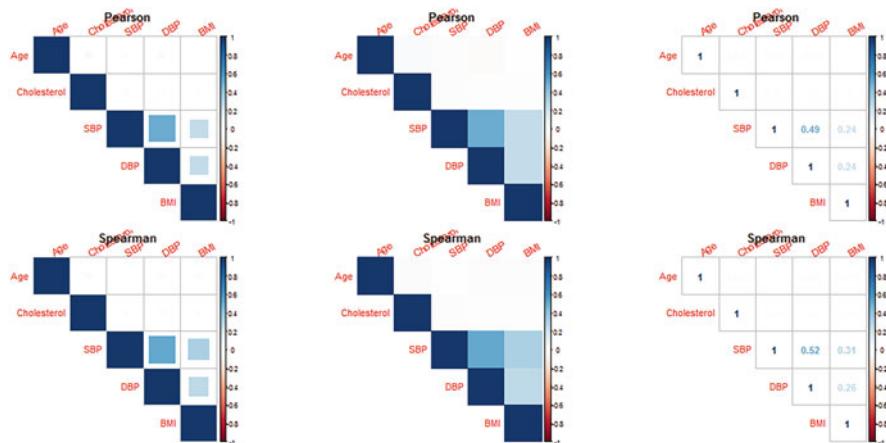


Figure 7.22: Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Pearson’s r and Spearman’s rho—1

Finally, for purposes of visualization, consider use of the GGally package and a few functions in this package, such as the `GGally::ggcorr()` function and the `GGally::ggpairs()` function. The output may be somewhat redundant to what has been seen previously in this lesson, but the output is linked to the `ggplot2` package and it is now well-received and easily recognized as a contemporary R-based tool (Figs. 7.22, 7.23, and 7.24).

R Input

```
install.packages("GGally", dependencies=TRUE)
library(GGally) # Load the GGally package.
help(package=GGally) # Show the information page.
sessionInfo() # Confirm all attached packages.

ggcorrPearson <-
GGally::ggcorr(EmployeeBiometric3.df, palette = "RdBu",
label = TRUE, method=c("complete.obs", "pearson"),
name="Pearson")

ggcorrSpearman <-
GGally::ggcorr(EmployeeBiometric3.df, palette = "RdBu",
label = TRUE, method=c("complete.obs", "spearman"),
name="Spearman")

par(ask=TRUE); gridExtra::grid.arrange(
ggcorrPearson, ggcorrSpearman, ncol=2)
```

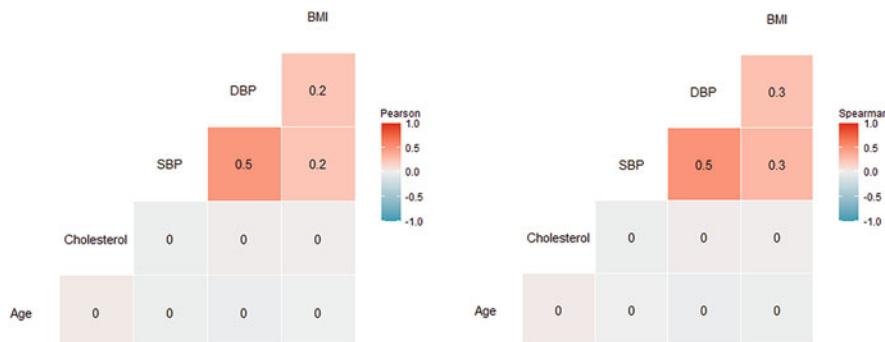


Figure 7.23: Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI—Pearson's r and Spearman's rho—2

R Input

```
par(ask=TRUE)
GGally::ggpairs(EmployeeBiometric3.df,
columns=1:ncol(EmployeeBiometric3.df), title=
"Correlation Matrix of Age, Cholesterol, SBP, DBP, and BMI",
upper=list(continuous = wrap("cor", size = 7)))
```

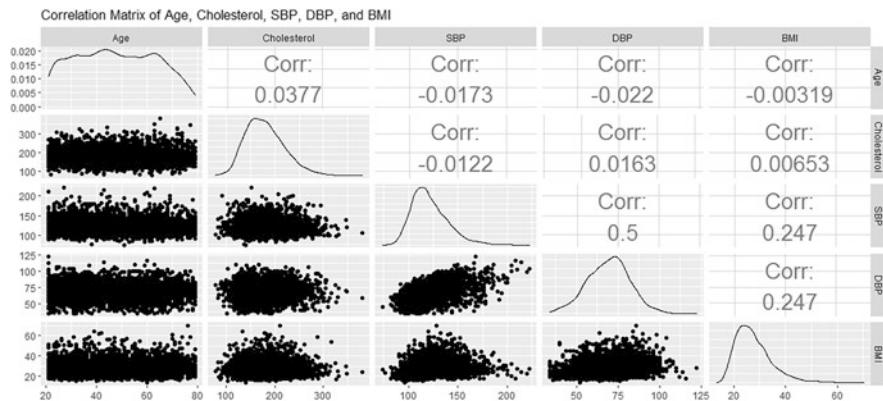


Figure 7.24: Correlation matrix of numeric-type object variables age, cholesterol, SBP, DBP, and BMI

Summary to Correlation Coefficients for Age, Cholesterol, SBP, DBP, and BMI

The degree of association between Age, Cholesterol, SBP, DBP, and BMI has been generated in both text format and in graphical format. Estimates of correlation have been provided for both Pearson's r (parametric) and Spearman's ρ (nonparametric). The outcomes of importance are that:

- There is a statistically significant ($p \leq 0.05$) association between Systolic Blood Pressure and Diastolic Blood Pressure (Pearson's $r = 0.49$ and Spearman's $\rho = 0.52$).
- There is a statistically significant ($p \leq 0.05$) association between Systolic Blood Pressure and Body Mass Index (Pearson's $r = 0.24$ and Spearman's $\rho = 0.31$).
- There is a statistically significant ($p \leq 0.05$) association between Diastolic Blood Pressure and Body Mass Index (Pearson's $r = 0.24$ and Spearman's $\rho = 0.26$).

For all other pairings, the correlation coefficients approximate zero, indicating that there is no association, certainly no actionable association.

7.5.2 Linear Regression Using a Single Predictor Variable

It has been determined that there is a statistically significant association between SBP and DBP, SBP and BMI, and DBP and BMI. In terms of practical implementation, recall that SBP and DBP are both gained from the mildly intrusive practice of placing a sphygmomanometer (e.g., blood pressure monitor) over an arm and for many patients causing discomfort as the associated cuff is inflated.

In contrast, a reasonable BMI measurement can be gained by using a home bathroom scale and a simple tape measure. Weight (kg) and height (m) are measured using the metric scale (kg = kilogram and m = meters) and the following algorithm is then applied:

$$\text{Body Mass Index (BMI Metric)} = \frac{\text{kg}}{\text{m}^2}$$

Using this process, Body Mass Index can be obtained in only a minute or two, as opposed to blood pressure monitoring, which may take a week or more for scheduling by a health care professional. Further, Body Mass Index measures are easily obtained and do not require the services of trained medical personnel for access to and use of required devices.

Knowing that there is an association between BMI and the two measures for blood pressure (SBP and DBP), is it possible to use known BMI values to obtain a reasonable estimate of blood pressure, at least as an early on tool in a wellness inventory? Look at the way R can be used to achieve this aim, using a simple linear regression process, which is based on prior correlation estimates:

Prediction of Systolic Blood Pressure Using Body Mass Index

R Input

```
cor(EmployeeBiometric.df$BMIMetric,  
     EmployeeBiometric.df$SBPmmHg,  
     use="complete.obs",      # Need to account for NA  
     method=c("pearson"))    # Parametric  
     # Use Pearson's r to obtain an estimate of the  
     # association between two object variables --  
     # a parametric approach
```

R Output

```
[1] 0.247134
```

R Input

```
lm(SBPmmHg ~ BMIMetric, data=EmployeeBiometric.df)  
# Build a linear regression model using the full  
# dataset, in original format.
```

R Output

```
Call:
lm(formula = SBPmmHg ~ BMIMetric, data = EmployeeBiometric.df)

Coefficients:
(Intercept)    BMIMetric
          103.754        0.647
```

From this output, gained by applying the lm() function against SBP and BMI, it is possible to prepare the prediction equation. The theoretical prediction equation for this scenario is:

$$SBPmmHg = \text{Intercept} + (\beta * \text{BMIMetric})$$

Then, as an example, apply this prediction equation for when a BMI of 25.00 is obtained, to see what the predicted SBP value will be:

```
# SBPmmHg = 103.754 + (0.647 * 25.00)
# SBPmmHg = 103.754 + 16.175
# SBPmmHg = 119.929
```

Is it reasonable to think that an individual with a Body Mass Index of approximately 25.00 will have a systolic Blood Pressure reading of 119.929? Apply either the RcmdrMisc::numSummary() function or the summary() function and purposely select only those subjects who had a Body Mass Index of approximately 25.00, selecting for BMIs that ranged from BMI 24.95 to BMI 25.05.⁸

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df$SBPmmHg [
  EmployeeBiometric.df$BMIMetric >= 24.95 &
  EmployeeBiometric.df$BMIMetric <= 25.05])
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
117.692	13.5846	18.5	100	107	115	125.5	152	26	23

⁸There are many R-based tools that could be used to achieve Boolean-type selections. The process demonstrated in this lesson was purposely selected to reinforce the heuristics of how data are selected from a large dataset.

R Input

```
summary(EmployeeBiometric.df$SBPmmHg[
  EmployeeBiometric.df$BMIMetric >= 24.95 &
  EmployeeBiometric.df$BMIMetric <= 25.05])
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
100	107	115	118	126	152	23

Note how subjects with a BMI that ranged from 24.95 to 25.05, approximating a BMI of 25.00, had a mean SBP reading of 118—certainly in range to 119.929. Further, this estimated SBP reading was obtained at a far lower cost and equally much faster than if medical personnel using a sphygmomanometer were involved in obtaining blood pressure readings. Yes, it might be better to use the services of medical personnel, but this method (e.g., Linear Regression Using a Single Predictor Variable) of using a known value that is easily obtained (BMI) provides a reasonably accurate sense of a value that is more difficult to obtain and at far greater cost (SBPmmhg) in terms of services by medical personnel and access to a specialized device.

In the same way that Body Mass Index was used in a linear regression equation to obtain a reasonable estimate of Systolic Blood Pressure, apply the same process to Diastolic Blood Pressure and see if the calculated value approximates expected values:

R Input

```
cor(EmployeeBiometric.df$BMIMetric,
  EmployeeBiometric.df$DBPmmHg,
  use="complete.obs",      # Need to account for NA
  method=c("pearson"))    # Parametric
# Use Pearson's r to obtain an estimate of the
# association between two object variables --
# a parametric approach
```

R Output

```
[1] 0.247427
```

R Input

```
lm(DBPmmHg ~ BMIMetric, data=EmployeeBiometric.df)
# Build a linear regression model using the full
# dataset, in original format
```

R Output

```
Call:
lm(formula = DBPmmHg ~ BMIMetric, data = EmployeeBiometric.df)

Coefficients:
(Intercept)      BMIMetric
      56.972        0.446
```

From this output, gained by applying the lm() function against DBP and BMI, it is possible to prepare the prediction equation. The theoretical prediction equation for this scenario is:

$$\text{DBPmmHg} = \text{Intercept} + (\beta * \text{BMIMetric})$$

Then, apply this prediction equation for when a BMI of 25.00 is obtained, to see what the predicted SBP value will be:

```
# DBPmmHg = 56.972 + (0.446 * 25.00)
# DBPmmHg = 56.972 + 11.15
# DBPmmHg = 68.122
```

Is it reasonable to think that an individual with a Body Mass Index of approximately 25.00 will have a Diastolic Blood Pressure reading of 68.122? Apply either the RcmdrMisc::numSummary() function or the summary() function and purposely select only those subjects who had a Body Mass Index of approximately 25.00, selecting for BMIs that ranged from BMI 24.95 to BMI 25.05.

R Input

```
RcmdrMisc::numSummary(EmployeeBiometric.df$DBPmmHg [
  EmployeeBiometric.df$BMIMetric >= 24.95 &
  EmployeeBiometric.df$BMIMetric <= 25.05])
```

R Output

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
	67.0769	9.76902	12	44	62	67	74	88	26	23

R Input

```
summary(EmployeeBiometric.df$DBPmmHg[
  EmployeeBiometric.df$BMIMetric >= 24.95 &
  EmployeeBiometric.df$BMIMetric <= 25.05])
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
44.0	62.0	67.0	67.1	74.0	88.0	23

Subjects with a BMI that ranged from 24.95 to 25.05, approximating a BMI of 25.00, had a mean DBP reading of 67—certainly in range to 68.122. This estimated blood pressure reading (DBP) was easily obtained at a far lower cost than scheduling a test with medical personnel.

Once again, consider the frequently cited statement *Past behavior is the best predictor of future behavior*. Linear regression builds on this premise. Using correlation as a starting point, when linear regression is applied correctly using object variables that have a reasonable degree of association, it may be possible to obtain predicted values for object variables with ease and at a low cost. The key to correct application of linear regression is that there should be some degree of acceptable correlation (e.g., association) between Variable X and Variable Y so that in the future, Variable X can be used to predict the value of Variable Y and conversely Variable Y can be used to predict the value of Variable X.

7.5.3 Linear Regression Using Multiple Predictor Variables

From a different perspective, now using two predictor variables and going beyond simple linear regression, look at the way multiple linear regression is structured—using Systolic Blood Pressure and Diastolic Blood Pressure to predict Body Mass Index. Again, the lm() function plays a major role in this prediction activity:

R Input

```
lm(formula = BMIMetric ~ SBPmmHg + DBPmmHg,
  data=EmployeeBiometric.df)
# Build a linear regression model using the full
# dataset, in original format
```

R Output

```

Call:
lm(formula = BMIMetric ~ SBPmmHg + DBPmmHg,
data = EmployeeBiometric.df)

Coefficients:
            (Intercept)      SBPmmHg       DBPmmHg
              13.7115        0.0629        0.0916

```

From this output, gained by applying the lm() function against BMIMetric as well as the predictor variables SBPmmHg and DBPmmHg, it is possible to prepare the multiple linear regression prediction equation. The theoretical prediction equation for this scenario is:

$$\text{BMIMetric} = \text{Intercept} + (\beta_1 * \text{SBPmmHg}) + (\beta_2 * \text{DBPmmHg})$$

Apply this prediction equation for when a Systolic Blood Pressure of 124.00 and a DBP of 76.00 are obtained, to see what the predicted Body Mass Index value will be:

```

# BMIMetric = 13.7115 + (0.0629 * 124.00) + (0.0916 * 76.00)
# BMIMetric = 13.7115 + (7.7996) + (6.9616)
# BMIMetric = 28.4727

```

Is it reasonable to think that an individual with 124 Systolic Blood Pressure and a 76 Diastolic Blood Pressure will have a Body Mass Index of approximately 28.50? Apply the RcmdrMisc::numSummary() function against subjects who are reasonably in range for the selected blood pressure readings.

R Input

```

RcmdrMisc::numSummary(EmployeeBiometric.df$BMIMetric[
  EmployeeBiometric.df$SBPmmHg >= 123.95 &
  EmployeeBiometric.df$SBPmmHg <= 124.05 &
  EmployeeBiometric.df$DBPmmHg >= 75.95 &
  EmployeeBiometric.df$DBPmmHg <= 76.05])

```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
30.2987	6.05675	4.39	19.77	28.275	30.3	32.665	44.39	15	209

Patients with a SBPmmHg of approximately 124 and a DBPmmHg of approximately 76 had a mean BMIMetric of 30.30—in range to a BMIMetric of 28.50.

And as mentioned before, this predicted Body Mass Index statistic was obtained in a fairly simple manner using multiple linear regression and values in the `EmployeeBiometric.df` dataset for object variables Systolic Blood Pressure and Diastolic Blood Pressure. Multiple linear regression is a valuable tool that is used quite frequently in biostatistics, where interval data with known values are used for prediction purposes against other variables.

7.5.4 Ordinal Logistic Regression

`BMIStatus` consists of four ordinal breakout values: Underweight, Normal Weight, Overweight, and Obese. There is a degree of order to these four breakout groups, but the ordering is by no mean interval. Instead, the cutpoint values for these four breakout groups (review the Code Book) are convenient markers of where one breakout group ends and another breakout group begins in terms of different views toward Body Mass Index values:

```
# BMIStatus - BMI Collapsed Into Four Groups      #
# Factor (e.g., ordinal)                         #
#                                                 #
# 1  BMI <= 18.599           Underweight       #
# 2  BMI >= 18.600 and <= 24.999  Normal Weight #
# 3  BMI >= 25.000 and <= 29.999  Overweight    #
# 4  BMI >= 30.000           Obese            #
```

The purpose of ordinal logistic regression is to gain a sense of the odds ratio and subsequently the probability of the predicted values for SBP and DBP relative to each of the four `BMIStatus` breakout groups. As a reminder, review the mean and median (50th percentile) of SBP and DBP, using the `RcmdrMisc::numSummary()` function, for each of the `BMIStatus` breakouts.

R Input

```
RcmdrMisc::numSummary(
  EmployeeBiometric.df[,c("SBPmmHg", "DBPmmHg")],
  groups=BMIStatus)      # Accept default printout
```

R Output

Variable: SBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Underweight	113.045	18.6707	16	82	102	110	118	208	157	11
Normal Weight	116.622	18.1826	20	76	104	114	124	220	1363	74
Overweight	123.802	18.5569	24	74	110	120	134	222	1058	56
Obese	126.915	17.1579	24	90	114	126	138	204	1145	61

Variable: DBPmmHg

	mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
Underweight	63.3758	12.8100	18	36	54	64	72	116	157	11
Normal Weight	65.9750	12.3119	16	34	58	66	74	114	1358	79
Overweight	70.6199	12.5282	16	34	62	72	78	122	1055	59
Obese	72.9817	12.4347	16	34	64	74	80	108	1145	61

A well-defined figure will help with a better understanding of these descriptive statistics of SBP and DBP for the four breakouts of BMIStatus. Among many possible selections, use the `gplots:::plotmeans()` function to further understand the relationship between the two blood pressure measures (e.g., SBPmmHg and DBPmmHg) and Body Mass Index breakout groups (e.g., BMIStatus) (Fig. 7.25).

R Input

```
install.packages("gplots", dependencies=TRUE)
library(gplots)                      # Load the gplots package.
help(package=gplots)                  # Show the information page.
sessionInfo()                        # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
gplots:::plotmeans(EmployeeBiometric.df$SBPmmHg ~
  EmployeeBiometric.df$BMIStatus,
  main="Mean Systolic Blood Pressure by Body Mass Index
Breakout Groups",
  pch=22, cex=10, lwd=3, col="red",      # Symbol, size, line
  ylim=c(105,135),                      # Y axis scale
  cex.axis=0.75,                         # Axis font size
  xlab="Body Mass Index Breakout Groups",
  ylab="Mean Systolic Blood Pressure (mmHg)")
gplots:::plotmeans(EmployeeBiometric.df$DBPmmHg ~
  EmployeeBiometric.df$BMIStatus,
  main="Mean Diastolic Blood Pressure by Body Mass Index
Breakout Groups",
  pch=22, cex=10, lwd=3, col="blue",     # Symbol, size, line
  ylim=c(60,80),                        # Y axis scale
  cex.axis=0.75,                         # Axis font size
  xlab="Body Mass Index Breakout Groups",
  ylab="Mean Diastolic Blood Pressure (mmHg)")
```

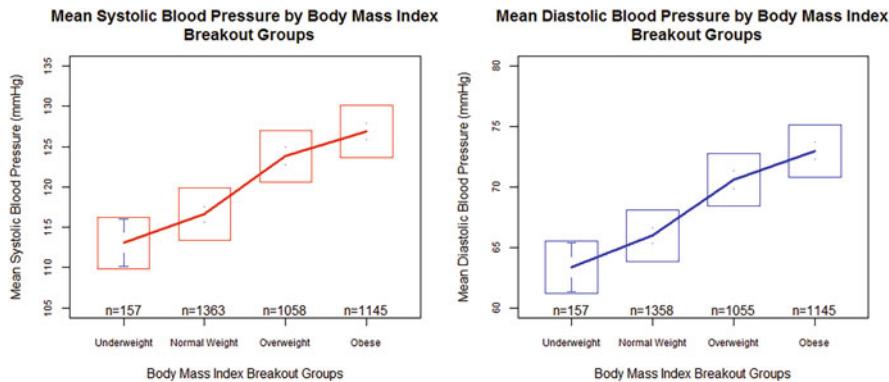


Figure 7.25: Mean systolic blood pressure by body mass index breakout groups and mean diastolic blood pressure by body mass index breakout groups

Notice the upward trend of the mean for both SBP and DBP as the BMIS-status changes from Underweight to Normal Weight, from Normal Weight to Overweight, and from Overweight to Obese. Perhaps what is most interesting about this figure is that the general trend is quite consistent for both SBP and DBP, but that might be expected considering the association between SBP and DBP. Recall also that there is a statistically significant ($p \leq 0.05$) association between Systolic Blood Pressure and Diastolic Blood Pressure (Pearson's $r = 0.49$ and Spearman's rho = 0.52).

Careful review of the descriptive statistics reinforces how there are missing values (e.g., NA's) for both Systolic Blood Pressure and Diastolic Blood Pressure. However, correlation (and subsequently regression) calls for comparison of paired values, the association between X and Y. It is necessary to account for missing values (e.g., values that show as NA).⁹ Perhaps the easiest way to achieve this aim is to use the `complete.cases()` function, with appropriate arguments, to create a new dataset (`EmployeeBiometric4.df`) that has no missing data since cases with missing data must otherwise be dropped for regression to continue.

R Input

```
EmployeeBiometric4.df <-
  EmployeeBiometric.df[complete.cases(EmployeeBiometric.df), ]
```

⁹There are functions in external R packages, such as the `amelia` package, that can be used to impute missing values with an estimate. The imputation of missing values (e.g., the replacement of missing values with an estimate) is a practice that some researchers occasionally use, but its application is not overly frequent and imputation of missing values is totally beyond the purpose of this lesson. As a sidebar comment, the `amelia` package was named for Amelia Mary Earhart, the famous aviatrix who went missing in 1937 while flying over the Pacific Ocean.

```
# Be sure to notice the Row-by-Column ending to the use of
# complete.cases(), or ,] in this example.
```

```
attach(EmployeeBiometric4.df) # Attach the dataframe
nrow(EmployeeBiometric.df) # N rows - original dataframe
```

R Output

```
[1] 3945
```

R Input

```
nrow(EmployeeBiometric4.df) # N rows - adjusted dataframe
```

R Output

```
[1] 2877
```

Use the summary() function against EmployeeBiometric4.df to confirm the descriptive statistics and that all missing values for SBP and DBP are removed from this new dataframe. Wrap the round() function around the summary() function to make all output simple and easy to read.

R Input

```
round(summary(EmployeeBiometric4.df$SBPmmHg, na.rm=TRUE))
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
74	108	118	122	132	222

R Input

```
round(summary(EmployeeBiometric4.df$DBPmmHg, na.rm=TRUE))
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
34	60	70	69	78	110

No NAs (e.g., missing data) show after using the `summary()` function against the new dataframe `EmployeeBiometric4.df`. It should now be possible to use syntax over a few step-by-step processes to gain a sense of ordinal regression for `BMIStatus`, using `SBPmmHg` and `DBPmmHg` as known predictor variables. In an attempt to have a full understanding of the ordered `BMIStatus` breakout groups and their relationship to Systolic Blood Pressure and Diastolic Blood Pressure, ordered logistic regression is used to eventually:

- Calculate the Odds Ratio values for both Systolic Blood Pressure and Diastolic Blood Pressure in terms of the four Body Mass Index breakout groups.
- Prepare a graphical summary of probability for both Systolic Blood Pressure and Diastolic Blood Pressure values as Body Mass Index progresses from Underweight to Normal Weight, from Normal Weight to Overweight, and from Overweight to Obese.

The `MASS::polr()` function has a major role in this set of actions, in an effort to conduct ordinal logistic regression. There are of course other R-based packages that include functions related to regression analyses, including ordinal logistic regression, but syntax for the `MASS::polr()` function is fairly straight-forward and may be the most convenient R function for a first attempt at this otherwise complex construct.

R Input

```
install.packages("MASS", dependencies = TRUE)
library(MASS)                      # Load the MASS package.
help(package=MASS)                  # Show the information page.
sessionInfo()                      # Confirm all attached packages.

# Step 1 Ordinal Logistic Regression: Build an Ordinal
# Logistic Regression Prediction Model

BMIBreakouts.plr <- MASS::polr(BMIStatus ~ SBPmmHg + DBPmmHg,
                                 data=EmployeeBiometric4.df, Hess=TRUE)
# Hess (the observed information matrix) is needed to allow
# later use of the summary() function against the derived
# object variable BMIBreakouts.plr.

summary(BMIBreakouts.plr)
```

R Output

Coefficients:

	Value	Std. Error	t value
SBPmmHg	0.0178	0.00221	8.04
DBPmmHg	0.0236	0.00312	7.57

Intercepts:

	Value	Std. Error	t value
Underweight Normal Weight	0.536	0.264	2.030
Normal Weight Overweight	3.369	0.260	12.945
Overweight Obese	4.631	0.267	17.330

R Input

```
# Step 2 Ordinal Logistic Regression: Extract Coefficients
BMIBreakouts.Coefficients <- coef(summary(BMIBreakouts.plr))
BMIBreakouts.Coefficients
```

R Output

	Value	Std. Error	t value
SBPmmHg	0.0177786	0.00221078	8.04182
DBPmmHg	0.0236051	0.00311774	7.57121
Underweight Normal Weight	0.5362382	0.26421553	2.02955
Normal Weight Overweight	3.3687721	0.26022750	12.94549
Overweight Obese	4.6310873	0.26723338	17.32975

R Input

```
# Step 3 Ordinal Logistic Regression: Calculate Probability
# Values

BMIBreakouts.p <-
  pnorm(abs(BMIBreakouts.Coefficients[, "t value"]),
    lower.tail=FALSE * 2)

format(BMIBreakouts.p, scientific=TRUE)
# Wrap the format() function around BMIBreakouts.p to put
# the output into Scientific Notation (e.g., E-notation),
```

```
# to accommodate the extreme number of places to the right
# of the decimal point.
```

R Output

SBPmmHg	4.42585e-16
DBPmmHg	1.84882e-14
Underweight Normal Weight	2.12012e-02
Normal Weight Overweight	1.24587e-38
Overweight Obese	1.40275e-67

R Input

```
# Step 4 Ordinal Logistic Regression: Create a Combined
# Table of Coefficients and Probabilities

BMIBreakouts.CoefficientsAndpvalues <-
  cbind(BMIBreakouts.Coefficients,"p value" = BMIBreakouts.p)

format(BMIBreakouts.CoefficientsAndpvalues, scientific=TRUE)
# The table will be slightly edited to allow for
# better presentation, but refer to the original
# screen output if needed.
```

R Output

	Value	p value
SBPmmHg	1.77786e-02	4.42585e-16
DBPmmHg	2.36051e-02	1.84882e-14
Underweight Normal Weight	5.36238e-01	2.12012e-02
Normal Weight Overweight	3.36877e+00	1.24587e-38
Overweight Obese	4.63109e+00	1.40275e-67

R Input

```
# Step 5 Ordinal Logistic Regression: Prepare Confidence
# Intervals

BMIBreakouts.ci <- confint(BMIBreakouts.plr)
# Allow for a delay during profiling.

BMIBreakouts.ci
```

R Output

```
2.5 %    97.5 %
SBPmmHg 0.0134168 0.0220808
DBPmmHg 0.0175603 0.0297878
```

R Input

```
# Step 6 Ordinal Logistic Regression: Prepare an Odds Ratio
# Table for the Predictor Variables

BMIBreakouts.OddsRatio <-
  exp(cbind(OddsRatio = coef(BMIBreakouts.plr)))

BMIBreakouts.OddsRatio
```

R Output

OddsRatio
SBPmmHg 1.01794
DBPmmHg 1.02389

The OddsRatio statistics ($SBP_{mmHg} = 1.01794$ and $DBP_{mmHg} = 1.02389$) are more appropriately viewed as Proportional Odds Ratios. It is well-beyond the purpose of this text to offer explicit detail on the concept of Odds Ratio, but a brief summary will help for immediate purposes. For continuous (e.g., interval) numeric object variables SBP_{mmHg} and DBP_{mmHg} :

- When a subject's SBP_{mmHg} moves one unit, such as from $SBP = 122$ to $SBP = 123$ or $SBP = 134$ to $SBP = 135$, the odds of moving from Underweight to Normal Weight, from Normal Weight to Overweight, and from Overweight to Obese are multiplied by 1.01794.
- When a subject's DBP_{mmHg} moves one unit, such as from $DBP = 70$ to $DBP = 71$ or $DBP = 86$ to $DBP = 87$, the odds of moving from Underweight to Normal Weight, from Normal Weight to Overweight, and from Overweight to Obese are multiplied by 1.02389.

The immediate impact of this finding, and in biostatistics in particular, is that even small changes in one variable can result in immediate changes in other variables. Look above and consider how weight gain (ostensibly, weight gain as a result of less than ideal nutrition and insufficient exercise since the patients in this lesson are all adults 21 years and older and are likely at full height) can easily result in increased blood pressure readings, both Systolic Blood Pressure

and Diastolic Blood Pressure, and then think of the known impact of increased blood pressure on concerns for wellness.

As useful as Odds Ratios may be, there are many individuals who will have difficulty understanding this concept. Far too often and certainly in mass media, the wording associated with Odds Ratio statistics is often incorrect, with terms such as probability and likelihood used even though these constructs are not the same as Odds Ratio. With this concern, add additional object variables to the EmployeeBiometric4.df dataframe, now including a set of probability statistics for each of the thousands of subjects. Then use these probability statistics to prepare figures that may be more readily understandable to the public.

R Input

```
EmployeeBiometric4.df <- cbind(EmployeeBiometric4.df,
  predict(BMIBreakouts.plr, EmployeeBiometric4.df,
  type = "probs"))
# Modify the dataframe EmployeeBiometric4.df, adding four
# new columns, probability statistics for each of the four
# BMISstatus breakout groups: Underweight, Normal Weight,
# Overweight, and Obese.

attach(EmployeeBiometric4.df)
# Attach the data to the modified dataframe

str(EmployeeBiometric4.df)
```

R Output

```
# 'data.frame': 2877 obs. of 14 variables:
# $ EmployeeID : Factor w/ 3945 levels "ID10005",
# $ Gender      : Factor w/ 2 levels "Female","Male"
# $ AgeYears    : num 34 57 35 40 34 40 50 60 32 ...
# $ RaceEthnicity: Factor w/ 5 levels "Asian","Black"
# $ TotalCholesterolmgdL: num 221 217 139 198 205 87 149 ...
# $ SBPmmHg     : num 102 90 86 114 92 120 126 96 ...
# $ DBPmmHg     : num 44 42 42 74 48 80 72 52 48 ...
# $ BMIMetric   : num 18.4 20.6 18.7 19.7 18.7 ...
# $ BMISstatus  : Factor w/ 4 levels "Underweight"
# $ Obesity     : Factor w/ 2 levels "Not Obese","Obese"
# $ Underweight : num 0.0898 0.1135 0.1209 0.0378 ...
# $ Normal Weight: num 0.537 0.572 0.579 0.362 0.5 ...
# $ Overweight  : num 0.229 0.2 0.192 0.302 0.22 ...
# $ Obese       : num 0.144 0.115 0.108 0.298 0.1 ...
```

Notice how there is a space between Normal and Weight for the new object variable Normal Weight. This space may be problematic for R. Normal Weight is the 12th column in the `EmployeeBiometric4.df` dataframe, so rename the 12th column to a single word—Normal.

R Input

```
names(EmployeeBiometric4.df)[12] <- "Normal"
attach(EmployeeBiometric4.df)
names(EmployeeBiometric4.df)

match("Normal", names(EmployeeBiometric4.df))
# Confirm column numbering.
# The match() function is part of the base
# package.
```

R Output

```
[1] 12
```

It is now possible to plot the probability of being Underweight, Normal (e.g., Normal Weight), Overweight, and Obese for Systolic Blood Pressure. This type of outcome, a graphical presentation, will be helpful and perhaps better communicate findings than merely provide Odds Ratio statistics (Fig. 7.26).

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
plot(EmployeeBiometric4.df$Underweight,
      EmployeeBiometric4.df$SBPmmHg,
      main="Probability of Underweight by SBP",
      col="red", xlim=c(0.00,1.00),
      xlab="Probability (0.00 to 1.00)", ylab="SBP")
plot(EmployeeBiometric4.df$Normal,
      EmployeeBiometric4.df$SBPmmHg,
      main="Probability of Normal Weight by SBP",
      col="red", xlim=c(0.00,1.00),
      xlab="Probability (0.00 to 1.00)", ylab="SBP")
plot(EmployeeBiometric4.df$Overweight,
      EmployeeBiometric4.df$SBPmmHg,
      main="Probability of Overweight by SBP",
      col="red", xlim=c(0.00,1.00),
```

```
xlab="Probability (0.00 to 1.00)", ylab="SBP")
plot(EmployeeBiometric4.df$Obese,
      EmployeeBiometric4.df$SBPmmHg,
      main="Probability of Obese by SBP",
      col="red", xlim=c(0.00,1.00),
      xlab="Probability (0.00 to 1.00)", ylab="SBP")
```

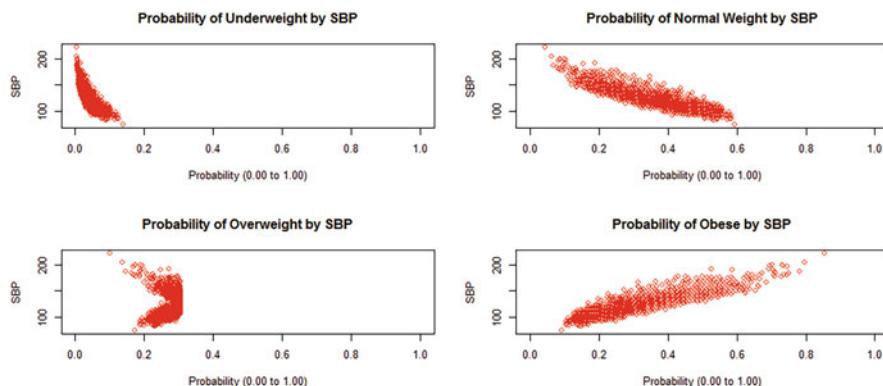


Figure 7.26: Probability of body mass index breakout group assignment by systolic blood pressure

From this figure, it is possible to make a few simple statements about Body Mass Index breakout group membership, probability, and Systolic Blood Pressure values:

- The probability of being Underweight increases as Systolic Blood Pressure decreases.
- The probability of being Normal Weight increases as Systolic Blood Pressure decreases.
- There does not seem to be a clear trend in the probability of being Overweight in terms of Systolic Blood Pressure values, either low Systolic Blood Pressure or high Systolic Blood Pressure.
- The probability of being Obese increases as Systolic Blood Pressure increases.

It is also possible to plot the probability of being Underweight, Normal (e.g., Normal Weight), Overweight, and Obese for Diastolic Blood Pressure. This type of outcome, a graphical presentation, will be helpful and perhaps better communicate findings than merely providing Odds Ratio statistics, which many find more helpful than numeric statistics (Fig. 7.27).

R Input

```

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
plot(EmployeeBiometric4.df$Underweight,
      EmployeeBiometric4.df$DBPmmHg,
      main="Probability of Underweight by DBP",
      col="red", xlim=c(0.00,1.00),
      xlab="Probability (0.00 to 1.00)", ylab="DBP")
plot(EmployeeBiometric4.df$Normal,
      EmployeeBiometric4.df$DBPmmHg,
      main="Probability of Normal Weight by DBP",
      col="red", xlim=c(0.00,1.00),
      xlab="Probability (0.00 to 1.00)", ylab="DBP")
plot(EmployeeBiometric4.df$Overweight,
      EmployeeBiometric4.df$DBPmmHg,
      main="Probability of Overweight by DBP",
      col="red", xlim=c(0.00,1.00),
      xlab="Probability (0.00 to 1.00)", ylab="DBP")
plot(EmployeeBiometric4.df$Obese,
      EmployeeBiometric4.df$DBPmmHg,
      main="Probability of Obese by DBP",
      col="red", xlim=c(0.00,1.00),
      xlab="Probability (0.00 to 1.00)", ylab="DBP")

```

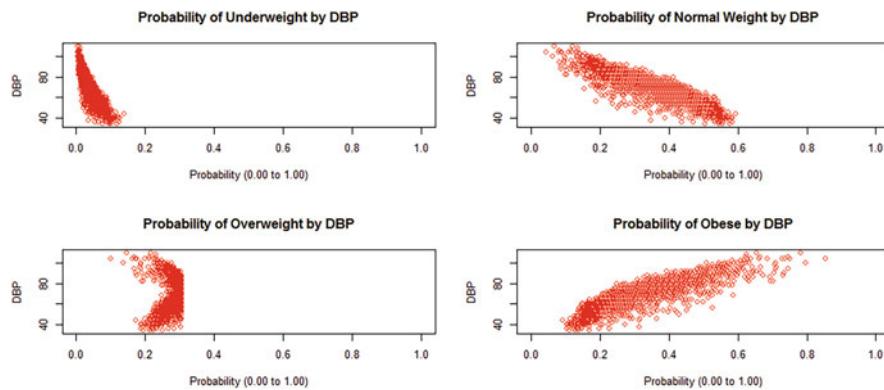


Figure 7.27: Probability of body mass index breakout group assignment by diastolic blood pressure

From this figure, it is possible to make a few simple statements about Body Mass Index breakout group membership, probability, and Diastolic Blood Pressure values:

- The probability of being Underweight increases as Diastolic Blood Pressure decreases.
- The probability of being Normal Weight increases as Diastolic Blood Pressure decreases.
- There does not seem to be a clear trend in the probability of being Overweight in terms of Diastolic Blood Pressure values, either low Diastolic Blood Pressure or high Diastolic Blood Pressure.
- The probability of being Obese increases as Diastolic Blood Pressure increases.

Go back to the statement *Correlation does not imply causation*. Given the data available in the dataframe associated with this study it cannot be stated that a change in blood pressure readings (either an increase or decrease) *causes* a change in Body Mass Index. It can be stated, however, that there is an *association* between blood pressure (both Systolic and Diastolic) and Body Mass Index. Ordinal logistic regression helps provide a sense of that association, possibly using terms (e.g., Underweight, Normal Weight, Overweight, and Obese) and graphical presentation methods that are easy for the public to understand.

7.5.5 Binary Logistic Regression

Measurements for height and weight, applied against a defined formula (e.g., algorithm), can be used easily to calculate Body Mass Index (BMI):

$$\text{Body Mass Index (BMI Metric Scale)} = \frac{\text{kg}}{\text{m}^2}$$

When measurements are completed with accuracy and the formula is calculated correctly, Body Mass Index represents an interval scale and there should be a fair degree of reliability and validity on the use of BMI statistics by the professional community.

However, there are those in the public who do not have sufficient background in the health sciences to know the fine points of just what Body Mass Index (BMI) statistics mean. What does it mean to have a BMI of 19.786 compared to a BMI of 25.529? Saying that, it is common to see interval BMI statistics collapsed into an ordinal scale, using terms such as Underweight, Normal Weight, Overweight, and Obese. When put into this ordinal scale, the precision of exact BMI statistics, based on an interval scale, is lost (or at least diminished), but at least this ordered listing and the use of common terms have value in that it can be understood by many.

There may be a degree of confusion on the fine points of this ordered listing. As an example, just where on the BMI scale does Normal Weight end and Overweight begin? How were these cutpoints determined and by whom? To accommodate this concern, it is not uncommon to see the four breakout groups

related to BMI (e.g., Underweight, Normal Weight, Overweight, and Obese) collapsed again, but into two breakouts: Not Obese and Obese. Again, the granularity of a defined interval scale (e.g., BMI) is lost but the gain is that more people can understand the classification scheme of a simple binary condition (e.g., Not Obese v Obese), similar to the way the meaning of binary condition terms such as Alive v Dead, Fail v Pass, Continue v Stop are evident to the public.

Given this background on why binary conditions are often appropriate, the purpose of binary logistic regression in this lesson is to gain a sense of the odds ratio and subsequently the probability of the predicted values for SBP and DBP for each of the two Obesity breakouts (e.g., Not Obese v Obese). As a reminder, review the mean and median (50th percentile) of SBP and DBP, using the RcmdrMisc::numSummary() function, for each Obesity breakout.

R Input

```
RcmdrMisc::numSummary(
  EmployeeBiometric4.df[,c("SBPmmHg", "DBPmmHg")],
  groups=Obesity)           # Accept default printout
```

R Output

		Variable: SBPmmHg	mean	sd	IQR	0%	25%	50%	75%	100%	n
	Not Obese	119.533	18.5322	24	74	106	116	130	222	1968	
	Obese	126.603	16.9023	24	90	114	126	138	198	909	

		Variable: DBPmmHg	mean	sd	IQR	0%	25%	50%	75%	100%	n
	Not Obese	67.7988	12.6266	18	34	58	68	76	110	1968	
	Obese	72.7547	12.4454	16	34	64	74	80	108	909	

The upward trend of the mean and median (e.g., 50th percentile) for both SBP and DBP is quite noticeable when Body Mass Index statistics change from Not Obese to Obese. If it helps to visualize this trend, use the gplots::plotmeans() function to confirm this trend (Fig. 7.28).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
gplots::plotmeans(EmployeeBiometric4.df$SBPmmHg ~
  EmployeeBiometric4.df$Obesity,
```

```

main="Mean Systolic Blood Pressure by Obesity",
pch=22, cex=10, lwd=3, col="red",      # Symbol, size, line
ylim=c(115,130),                      # Y axis scale
cex.axis=0.75,                          # Axis font size
xlab="Obesity",
ylab="Mean Systolic Blood Pressure (mmHg)")
gplots:::plotmeans(EmployeeBiometric.df$DBPmmHg ~
EmployeeBiometric.df$Obesity,
main="Mean Diastolic Blood Pressure by Obesity",
pch=22, cex=10, lwd=3, col="blue",      # Symbol, size, line
ylim=c(65,75),                         # Y axis scale
cex.axis=0.75,                          # Axis font size
xlab="Obesity",
ylab="Mean Diastolic Blood Pressure (mmHg)")

```

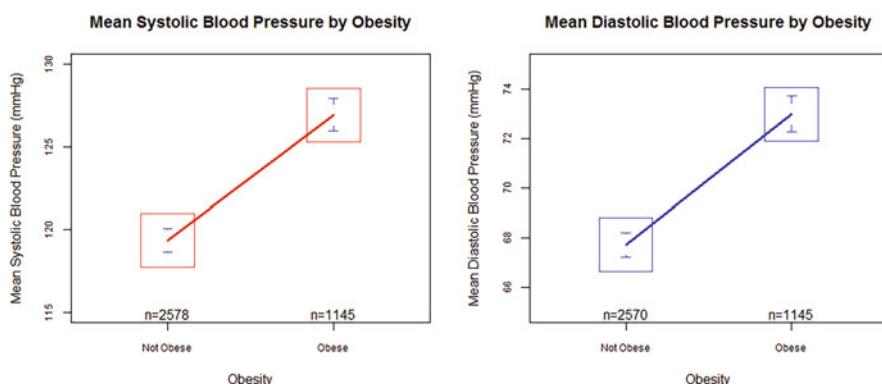


Figure 7.28: Mean systolic blood pressure by obesity breakout groups and mean diastolic blood pressure by obesity breakout groups

Recalling concern about the way correlation and regression require a value for both X and Y, the previously created dataframe `EmployeeBiometric4.df` will be used to conduct the binary logistic regression. Missing data should not be a concern using this dataframe.

Similar to the process for Ordinal Logistic Regression, syntax is now used for a few step-by-step processes to gain a sense of binary logistic regression for Obesity, using SBPmmHg and DBPmmHg as known predictor variables. In an attempt to have a full understanding of the two Obesity breakout groups (e.g., Not Obese v Obese) and their relationship to Systolic Blood Pressure and Diastolic Blood Pressure, binary logistic regression is used to eventually:

- Calculate the Odds Ratio values for both Systolic Blood Pressure and Diastolic Blood Pressure in terms of the two Obesity breakout groups.

- Prepare a graphical summary of probability for both Systolic Blood Pressure and Diastolic Blood Pressure values as Body Mass Index progresses from Not Obese to Obese.

The `glm()` function found in the `stats` package from when R is first downloaded has a major role in this set of actions, in an effort to conduct binary logistic regression. There are of course other R-based packages that include functions related to regression analyses, including binary logistic regression, but the syntax shown below is fairly straight-forward and may be the most convenient for a first attempt at this otherwise complex construct.

R Input

```
# Step 1 Binary Logistic Regression: Build a Binary Logistic
# Regression Prediction Model

Obesity.glm <- glm(Obesity ~ SBPmmHg + DBPmmHg,
  data=EmployeeBiometric4.df,
  family="binomial")
# Use the glm() function to develop a model
# for the binary object variable Obesity (1 =
# Not Obese and 2 = Obese) in view of SBPmmHg
# and DBPmmHg:
# 1 BMI <= 29.999 Not Obese
# 2 BMI >= 30.000 Obese

summary(Obesity.glm)
# Apply the summary() function against the
# object Obesity.glm.
```

R Output

[Selected output is not shown, to save space.]

Call:

```
glm(formula = Obesity ~ SBPmmHg + DBPmmHg, family = "binomial",
  data = EmployeeBiometric4.df)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.764	-0.878	-0.736	1.322	2.018

Coefficients:

z value	Pr(> z)
---------	----------

```
(Intercept) -13.12 < 0.000000000000002 ***
SBPmmHg      5.46       0.0000000481 ***
DBPmmHg      5.77       0.0000000081 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

R Input

```
# Step 2 Binary Logistic Regression: Prepare an Odds Ratio
# Table for the Predictor Variables

exp(coef(Obesity.glm))
# Obtain Odds Ratios for SBPmmHg and DBPmmHg by wrapping
# the exp() function (e.g., a logarithm function) around
# the coef() function (e.g., a generic function that
# yields model coefficients)
```

R Output

(Intercept)	SBPmmHg	DBPmmHg
0.0187415	1.0138574	1.0217749

The `exp()` function, wrapped around the `coef()` function, results in proportional Odds Ratio statistics for SBPmmHg and DBPmmHg and the binary object variable `Obesity` (e.g., original codes are 1 = Not Obese and 2 = Obese). As mentioned in the discussion on Ordinal Logistic Regression, it is well-beyond the purpose of this text to offer explicit detail on Odds Ratios, what they mean, and detail on how they are used, but a brief summary will help for immediate purposes. For continuous (e.g., interval) numeric object variables SBPmmHg and DBPmmHg in view of Odds Ratio statistics for the factor-type object variable `Obesity`:

- When a subject's SBPmmHg moves one unit, such as from SBP = 122 to SBP = 123 or SBP = 134 to SBP = 135, the odds of moving from Not Obese to Obese are multiplied by 1.0138574.
- When a subject's DBPmmHg moves one unit, such as from DBP = 70 to DBP = 71 or DBP = 86 to DBP = 87, the odds of moving from Not Obese to Obese are multiplied by 1.0217749.

A visualization of the upward trend may be helpful, where blood pressure measurements are placed on the X axis and the probability of moving from Not Obese to Obese is placed on the Y axis (Fig. 7.29).

R Input

```

ObesitybySystolicProbability <-
ggplot2::ggplot(EmployeeBiometric4.df,
  aes(SBPmmHg, as.numeric(Obesity)-1)) +
  stat_smooth(method="glm", formula=y~x, size=3) +
  ggtitle("Probability of Obesity by Systolic Blood Pressure (mmHg)\n") +
  labs(
    x = "\nSystolic Blood Pressure (mmHg)",
    y = "Probability of Obesity (0.00 to 1.00)\n") +
  scale_x_continuous(labels=scales::comma, limits=c(0,250),
    breaks=seq(0,250, by=25)) +
  scale_y_continuous(limits=c(0,1), expand=c(0,0)) +
  theme_bw()
# Remember that Obesity is coded as 1 and 2, thus the reason
# why as.numeric(Obesity)-1 is used, to force the binary
# values to 0 and 1.

ObesitybyDiastolicProbability <-
ggplot2::ggplot(EmployeeBiometric4.df,
  aes(DBPmmHg, as.numeric(Obesity)-1)) +
  stat_smooth(method="glm", formula=y~x, size=3) +
  ggtitle("Probability of Obesity by Diastolic Blood Pressure (mmHg)\n") +
  labs(
    x = "\nDiastolic Blood Pressure (mmHg)",
    y = "Probability of Obesity (0.00 to 1.00)\n") +
  scale_x_continuous(labels=scales::comma, limits=c(0,250),
    breaks=seq(0,250, by=25)) +
  scale_y_continuous(limits=c(0,1), expand=c(0,0)) +
  theme_bw()
# In both figures, note the common scale for the X axis
# (e.g., SBP and DBP)and the Y axis (e.g., Probability of
# Obesity), to allow convenient side-by-side comparisons.

gridExtra::grid.arrange(
  ObesitybySystolicProbability,
  ObesitybyDiastolicProbability, ncol=2)

```

The probability of being obese (e.g., Obesity is defined as a Body Mass Index that is greater than or equal to 30.00) increases as Systolic Blood Pressure increases and as Diastolic Blood Pressure increases. Once again, the Binary Logistic Regression outcomes provide evidence on how an increase in Body Mass Index (ostensibly, weight gain as a result of less than ideal nutrition and

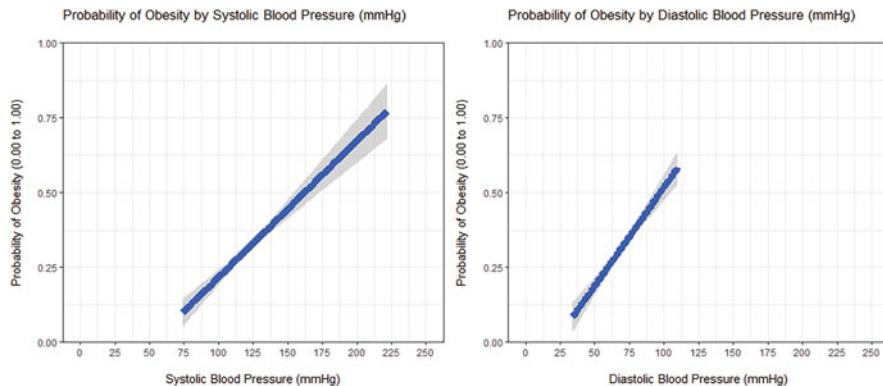


Figure 7.29: Probability of obesity by systolic blood pressure and probability of obesity by diastolic blood pressure

insufficient exercise since the patients in this lesson are all adults 21 years and older and are likely at full height) impacts the propensity for Obesity, which is in turn associated with a propensity for increased blood pressure measurements, both Systolic Blood Pressure and Diastolic Blood Pressure.

Go back to the statement that *Correlation does not imply causation*. Given the data available in the dataframe associated with this study it cannot be stated that a change in blood pressure readings (either an increase or decrease) *causes* a change in Body Mass Index. It can be stated, however, that there is an *association* between blood pressure (both Systolic and Diastolic) and Body Mass Index. Both Ordinal Logistic Regression and Binary Logistic Regression help provide a further sense of that association, beyond the use of Pearson's r or Spearman's rho correlation coefficients.

7.6 Summary of Outcomes

Correlation (e.g., association) is based on two well-accepted statements, statements that apply to the biological sciences, physical sciences, social sciences, economics, etc.:

- *Past behavior is the best predictor of future behavior.*
- *Correlation does not imply causation.*

A dataset related to a wellness program for thousands of subjects, with all subjects employees of a company that provided access to health-related services, served as the data resource for this lesson. The data are inspired by real-world outcomes, but it cannot be ignored that the data are best viewed as part of a teaching dataset in an attempt to reach desired outcomes, for demonstration purposes. In final form, there were ten object variables specific to this lesson:

- Factor-type object variables

- EmployeeID
- Gender
- RaceEthnicity
- BMIStatus
- Obesity

- Numeric-type object variables

- AgeYears
- TotalCholesterolmgdL
- SBPmmHg
- DBPmmHg
- BMIMetric

A series of Quality Assurance actions were applied against the dataset, to know more about the data and to determine if the data were fit for correlation analyses and if so, which type of analyses (e.g., parametric or nonparametric) to use. Various functions were used to put data into graphical format to examine relationships between and among the data. Data were also examined by applying functions that generated descriptive statistics such as frequency distributions and measures of central tendency such as mean, standard deviation, median, etc. Numeric data were then subjected to normality tests, overall and by breakouts of factor-type object variables.

Discovering that there were reasonable concerns about the normality of numeric object variables, it was decided to conduct the correlation analyses from both a parametric perspective (e.g., Pearson's r) and a nonparametric perspective (e.g., Spearman's rho). There was a general finding of parity in calculated p-values and correlation coefficients for both approaches, parametric or nonparametric. Overall, there is a high degree of confidence for the statements:

- There is a statistically significant ($p \leq 0.05$) association between Systolic Blood Pressure and Diastolic Blood Pressure (Pearson's $r = 0.49$ and Spearman's $\rho = 0.52$).
- There is a statistically significant ($p \leq 0.05$) association between Systolic Blood Pressure and Body Mass Index (Pearson's $r = 0.24$ and Spearman's $\rho = 0.31$).
- There is a statistically significant ($p \leq 0.05$) association between Diastolic Blood Pressure and Body Mass Index (Pearson's $r = 0.24$ and Spearman's $\rho = 0.26$).

After viewing these general trends, additional analyses closely aligned with prediction were attempted, with demonstrations on:

- Linear regression using a single predictor variable
- Linear regression using multiple predictor variables
- Ordinal logistic regression
- Binary logistic regression

These predictive approaches toward discovery of broad trends from the data provided a basis for findings that go beyond simple correlation analyses:

- The probability of being Underweight increases as both Systolic and Diastolic Blood Pressure decrease.
- The probability of being Normal Weight increases as both Systolic and Diastolic Blood Pressure decrease.
- There does not seem to be a clear trend in the probability of being Overweight in terms of Systolic and Diastolic Blood Pressure values.
- The probability of being Obese increases as both Systolic and Diastolic Blood Pressure increase.

Offering even greater granularity of findings, the binary logistic regression resulted in the calculation of odds ratios, detailing the change in Body Mass Index that can be expected with simple changes in Systolic and Diastolic Blood Pressure:

- When a subject's SBPmmHg moves one unit, such as from SBP = 122 to SBP = 123 or SBP = 134 to SBP = 135, the odds of moving from Not Obese to Obese are multiplied by 1.0138574.
- When a subject's DBPmmHg moves one unit, such as from SDP = 70 to DBP = 71 or DBP = 86 to DBP = 87, the odds of moving from Not Obese to Obese are multiplied by 1.0217749.

A constant reminder of correlation and regression is the need to avoid any attempt to infer causation (e.g., cause-and-effect).¹⁰ X may be correlated (e.g., associated) with Y, but that does not mean that X causes Y. Consider the well-accepted observation in education that family income is a predictor of grade point average (GPA) in the first year of college. That does not mean that a rise in family income immediately results in an increase in GPA. Family income does not *cause* GPA. Class attendance, time-on-task and diligence with homework and study, and performance on exams contribute to GPA. Instead, the two

¹⁰Search on the term *spurious correlations*. There are many Web sites that list correlations that exist between X and Y, but there is simply no reason to think that there is any linkage between the two variables.

variables (family income and GPA) are merely correlated to each other. This example from education should be recalled when applying the use of correlation and regression to biostatistics. Causation cannot be assumed even when selected variables are used to build a future prediction equation.

7.7 Addendum 1: Multiple Regression

As shown in this lesson, data from the past can be used to predict outcomes for the future. Given the extreme importance of prediction in biostatistics, another example of multiple regression, using a different dataset, will be helpful to see how existing data are used to predict future outcomes.¹¹ These outcomes support informed decisions that can now be made with some degree of confidence as compared to decisions that are instead based on best guess, which is clearly not a good practice.

The data for this addendum on multiple regression address the vigor of livestock in a commercial operation. Before the regression analyses are attempted, apply the different actions associated with these lessons: import data into R, examine the data graphically, determine measures of central tendency, and address normality. The focus of this addendum, in abbreviated format compared to the front matter in this lesson, is on the processes needed to use multiple regression successfully.

R Input

```
LStockVg.df <- read.table (file =
  "LivestockVigor.csv",
  header = TRUE,
  sep = ",")                                # Import the csv file

getwd()                                         # Identify the working directory
ls()                                              # List objects
attach(LStockVg.df)                            # Attach the data, for later use
str(LStockVg.df)                               # Identify structure
nrow(LStockVg.df)                             # List the number of rows
ncol(LStockVg.df)                             # List the number of columns
```

¹¹When practicing with the data in this Addendum 1, give attention, again, to the emphasis in this lesson on linear regression using multiple predictor variables. It is important to learn about correlation and the possible association between two separate variables. It is important to learn about linear regression using a single predictor variable. However, the biological sciences nearly always involve multiple variables and their possible interplay with each other. This addendum provides additional guidance on multiple regression, going beyond what was previously presented in the front matter of this lesson. Indeed, look at the way this addendum addresses an algorithmic approach to regression and the automatic identification (and removal) of spurious predictor variables (e.g., false friends).

```

dim(LStockVg.df)                      # Dimensions of the data frame
names(LStockVg.df)                     # Identify names
colnames(LStockVg.df)                  # Show column names
rownames(LStockVg.df)                  # Show row names
head(LStockVg.df)                     # Show the head
tail(LStockVg.df)                     # Show the tail
# LStockVg.df                         # Show the entire data frame
summary(LStockVg.df)                  # Summary statistics

```

By completing these actions, an object called `LStockVg.df` has been created. This R-based object is a dataframe and it consists of the data originally included in the file `LivestockVigor.csv`, a comma-separated .csv file. To avoid possible conflicts, make sure that there are no prior R-based objects called `LStockVg.df` in the active R session.

R Input

```

#####
# Code Book for LStockVg.df          #
#####
#
# Subject ..... Factor (e.g., nominal) #
#           A unique ID ranging from S0001 onward #
#
# WeightInitial ..... Numeric (e.g., interval) #
#           Initial feeder stock weight, with #
#           typical weight about 35 lbs #
#
# WeightFinish ..... Numeric (e.g., interval) #
#           Finished feeder stock weight, with #
#           typical weight about 250 lbs #
#
# VigorInitial ..... Numeric (e.g., interval) #
#           A measure of general health and appearance, #
#           ranging from > 0.00 to 10.00 #
#
# Vigor100Lbs ..... Numeric (e.g., interval) #
#           A measure of general health and appearance, #
#           ranging from > 0.00 to 10.00 #
#
# Vigor200Lbs ..... Numeric (e.g., interval) #
#           A measure of general health and appearance, #
#           ranging from > 0.00 to 10.00 #

```

```

#                                     #
# VigorFinish ..... Numeric (e.g., interval) #
#           A measure of general health and appearance, #
#           ranging from > 0.00 to 10.00 #
#####
#####
```

From among the many variables used in this addendum, never forget that the livestock manager needs to be practical and make decisions based on eventual profits. Consider the four measures for vigor: VigorInitial, Vigor100Lbs, Vigor200Lbs, VigorFinish. The desired weight at time of sale, when the live-stock are taken to an abattoir for slaughter, is about 250 pounds. The manager has the best opportunity to influence finished weight by applying sound live-stock practices at the beginning (when VigorInitial is measured) and early on (when Vigor100Lbs is measured). Management options that influence future finished weight at 250 pounds are limited when Vigor200Lbs is measured and of course practices are even more limited when VigorFinish is measured.

7.7.1 Hand-Calculate Multiple Regression

Given the importance of early intervention when working with livestock, construct a model for VigorEarly, which takes into account management practices up to the time Vigor100Lbs is measured. Look at how two (or more) variables can be manually entered into the construction of a prediction equation for \hat{Y} , commonly referred to as *Y-hat*, the predicted outcome value of a regression equation.

R Input

```

Fit.WeightFinish.by.VigorEarly <- lm(WeightFinish ~
  VigorInitial + Vigor100Lbs,
  data=LStockVg.df)
# Note how this model includes two predictors and
# enumeration of a new concept, VigorEarly

anova(Fit.WeightFinish.by.VigorEarly)
# Confirm significance
```

R Output

[Selected output is not shown, to save space.]

Analysis of Variance Table

```
Response: WeightFinish
          Df  Mean Sq F value    Pr(>F)
VigorInitial     1    21214   175.7 <0.0000000000000002 ***
Vigor100Lbs      1    43447   359.8 <0.0000000000000002 ***
Residuals       3296      121
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

R Input

```
summary(Fit.WeightFinish.by.VigorEarly) # Prediction equation
# Obtain statistics needed for prediction equation
```

R Output

```
ssl
[Selected output is not shown, to save space.]

Call:
lm(formula = WeightFinish ~ VigorInitial + Vigor100Lbs,
data = LStockVg.df)

Coefficients:
            Std. Error t value    Pr(>|t|)
(Intercept) 4.741     31.47 < 0.0000000000000002 ***
VigorInitial 0.482      6.44     0.00000000014 ***
Vigor100Lbs  0.448     18.97 < 0.0000000000000002 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Before \hat{Y} is calculated, for what is now a multiple regression equation, apply the coefficients() function against Fit.WeightFinish.by.VigorEarly, the enumerated lm-type object, to gain a better sense of the model.

R Input

```
coefficients(Fit.WeightFinish.by.VigorEarly)
```

R Output

```
(Intercept) VigorInitial Vigor100Lbs
149.18433     3.10041      8.49965
```

Based on this output, the prediction equation can now be calculated. The process, similar to what was seen earlier in this lesson, is presented in simple format as.¹²

$$\text{Y-hat} = a + b(x) + b(y)$$

$$\begin{aligned}\text{WeightFinish} &= 149.18433 + \text{VigorInitial}(3.10041) \\ &\quad + \text{Vigor100Lbs}(8.49965)\end{aligned}$$

Assume that an individual animal had a VigorInitial value of 8.73 and a Vigor100Lbs value of 8.63. What is the predicted value for WeightFinish? Apply the multiple regression prediction equation:

$$\text{WeightFinish} = 149.18433 + (8.73 * 3.10041) + (8.63 * 8.49965)$$

$$\text{WeightFinish} = 149.18433 + 27.0666 + 73.352$$

$$\text{WeightFinish} = 249.603$$

The livestock manager now knows that for this large-scale feeder operation, an animal with a VigorInitial value of 8.73 and a Vigor100Lbs value of 8.63 will yield a finishing weight (WeightFinish = 249.603) of approximately 250 pounds—the desired finishing weight.

Although it may be a bit redundant, apply the cor() function against VigorInitial and Vigor100Lbs to see the relationship between these two variables and then reinforce the outcome with a visual image of the correlation (Fig. 7.30).

R Input

```
cor(LStockVg.df$VigorInitial, LStockVg.df$Vigor100Lbs,
  use="complete.obs", method=c("pearson"))
```

R Output

```
[1] 0.321438
```

¹²Notice the difference in how the prediction equation is presented, where it is slightly different than what was presented previously. As an example, the character **a** shows where previously the term **Intercept** was used. The process for application of the prediction equation remains the same and this alternate presentation was purposely shown to reinforce the variety in how prediction equations are displayed.

R Input

```
par(ask=TRUE)
plot(LStockVg.df$VigorInitial, LStockVg.df$Vigor100Lbs,
     xlab="Vigor at Purchase (Approximately 35 Pounds)",
     ylab="Vigor at 100 Pounds",
     font.lab=2, font.axis=2, cex.lab=1.15, cex.main=1.15,
     pch=c(19), # solid circle
     main="Scatter Plot of Vigor at Purchase (X) by Vigor at
     100 Pounds (Y) with Added Regression Line")
abline(lm(Vigor100Lbs ~ VigorInitial, data =LStockVg.df),
       lwd=3, col="red")
```

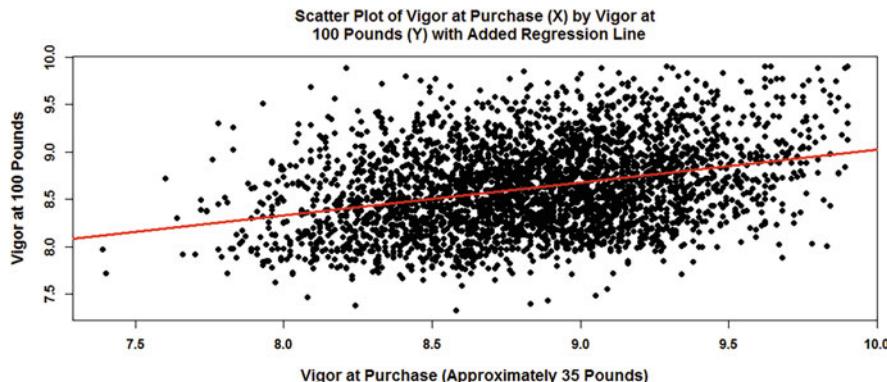


Figure 7.30: Scatterplot of two variables (vigor at purchase by vigor at 100 pounds) with added regression line

As is evident from a finding of Pearson's $r = 0.321438$ and then reinforced graphically, there is an upward trend in the association between vigor at time of purchase and vigor when livestock reach 100 pounds. The fit line adds to understanding of general trends.

7.7.2 Minimal Adequate Model (MAM) for Regression

Multiple regression is not restricted to the use of only two predictors (VigorInitial and Vigor100Lbs, shown above). Below, notice how all four indicators of vigor (VigorInitial, Vigor100Lbs, Vigor200Lbs, and VigorFinish) are introduced into a prediction equation. A Minimal Adequate Model (MAM) approach will be used for this approach at multiple regression.

R Input

```
Fit.Model.WeightFinish.Vigor <- lm(WeightFinish ~
  VigorInitial + Vigor100Lbs + Vigor200Lbs + VigorFinish,
  data=LStockVg.df)

summary(Fit.Model.WeightFinish.Vigor)
```

R Output

[Selected output is not shown, to save space.]

Call:

```
lm(formula = WeightFinish ~ VigorInitial + Vigor100Lbs +
  Vigor200Lbs + VigorFinish, data = LStockVg.df)
```

Coefficients:

	Std. Error	t value	Pr(> t)
(Intercept)	5.267	28.48	< 0.0000000000000002 ***
VigorInitial	0.485	6.49	0.0000000000097 ***
Vigor100Lbs	1.102	8.68	< 0.0000000000000002 ***
Vigor200Lbs	1.897	-1.40	0.16
VigorFinish	1.572	0.92	0.36

Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1		

R Input

```
summary.aov(Fit.Model.WeightFinish.Vigor)
```

R Output

[Selected output is not shown, to save space.]

	Df	Mean Sq	F value	Pr(>F)
VigorInitial	1	21197	175.67	< 0.0000000000000002 ***
Vigor100Lbs	1	43088	357.08	< 0.0000000000000002 ***
Vigor200Lbs	1	139	1.15	0.28
VigorFinish	1	103	0.85	0.36
Residuals	3288	121		

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

15 observations deleted due to missingness

The p-value for VigorFinish exceeds 0.05 and is the greatest of the two object variables (e.g., Vigor200Lbs and VigorFinish) that exceed a p-value of 0.05. Remove VigorFinish from the model, to see the effect of this action on model building and then attempt the same set of actions, but now with the object variable VigorFinish removed from the model.

R Input

```
Fit.Model.WeightFinish.Vigor2 <- update(
  Fit.Model.WeightFinish.Vigor, .~.-VigorFinish,
  data=LStockVg.df)
# Note the . character that goes before the ~ character and
# the . character then comes after the ~ character. The .
# character means "same." Using this syntax, VigorFinish
# is removed from the model by placing a - character in
# front of the object variable that is to be removed.

summary(Fit.Model.WeightFinish.Vigor2)
```

R Output

[Selected output is not shown, to save space.]

Call:

```
lm(formula = WeightFinish ~ VigorInitial + Vigor100Lbs +
Vigor200Lbs, data = LStockVg.df)
```

Coefficients:

	Std. Error	t value	Pr(> t)
(Intercept)	5.098	29.67 < 0.0000000000000002 ***	
VigorInitial	0.484	6.55 0.000000000006 ***	
Vigor100Lbs	1.102	8.67 < 0.0000000000000002 ***	
Vigor200Lbs	1.256	-1.07 0.28	
<hr/>			

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R Input

```
summary.aov(Fit.Model.WeightFinish.Vigor2)
```

R Output

[Selected output is not shown, to save space.]

	Df	Mean Sq	F value	Pr(>F)
VigorInitial	1	21197	175.68 <0.0000000000000002	***
Vigor100Lbs	1	43088	357.10 <0.0000000000000002	***
Vigor200Lbs	1	139	1.15	0.28
Residuals	3289	121		
<hr/>				
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1				
15 observations deleted due to missingness				

Vigor200Lbs remains at a p-value level greater than 0.05. Remove the remaining object variable for which there is no significance ($p \leq 0.05$), or Vigor200Lbs in this example.

R Input

```
Fit.Model.WeightFinish.Vigor3 <- update(
  Fit.Model.WeightFinish.Vigor2, .~.-Vigor200Lbs,
  data=LStockVg.df)

summary(Fit.Model.WeightFinish.Vigor3)
```

R Output

[Selected output is not shown, to save space.]

Call:			
lm(formula = WeightFinish ~ VigorInitial + Vigor100Lbs,			
data = LStockVg.df)			
<hr/>			
Coefficients:			
	Std. Error	t value	Pr(> t)
(Intercept)	4.741	31.47 < 0.0000000000000002	***
VigorInitial	0.482	6.44 0.00000000014	***
Vigor100Lbs	0.448	18.97 < 0.0000000000000002	***
<hr/>			
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1			

R Input

```
summary.aov(Fit.Model.WeightFinish.Vigor3)
```

R Output

[Selected output is not shown, to save space.]

	Df	Mean Sq	F value	Pr(>F)
VigorInitial	1	21214	176 <0.0000000000000002	***
Vigor100Lbs	1	43447	360 <0.0000000000000002	***
Residuals	3296	121		

Signif. codes:	0	'***'	0.001 '**'	0.01 '*'
			0.05 '.'	0.1 ' '
			1	
9 observations deleted due to missingness				

And now all remaining predictors (VigorInitial and Vigor100Lbs) are significant. Use these predictors, as needed, for practical considerations in development of a prediction equation and possible attention to management practices. Biostatistics is not divorced from the economics of commercial production and the two can easily complement each other.

7.7.3 Stepwise Regression

As always with R, there is generally more than one way of attempting an analysis. Note how a stepwise approach is used, as opposed to the earlier minimal adequate model (MAM). Because missing values impact stepwise regression and because there are only a few missing values in this dataset of 3308 subjects, use the na.omit() function to remove all cases with missing data.

R Input

```
nrow(LStockVg.df)
# Display the number of rows before
# the dataset is adjusted.
```

R Output

```
[1] 3308
```

R Input

```
LStockVg2.df <- na.omit(LStockVg.df)
# Remove all cases with missing values

nrow(LStockVg2.df)
# Display the number of rows after
# the dataset is adjusted.
```

R Output

```
[1] 3293
```

Note the change from 3308 rows (`LStockVg.df`) to 3293 rows (`LStockVg2.df`). With the dataset now in correct format, apply the `step()` function to conduct the stepwise regression. Again, it is beyond the purpose of this introductory lesson on R to go into too much detail on the particulars of regression and differences in methods, such as the difference (if any) between a forward stepwise regression, backward stepwise regression, etc. Fortunately, there is an abundance of available resources. Take advantage of these resources.

Construct the object `Fit.Model.WeightFinish.Vigor2` again, incorporating all four vigor measures. Recall that as opposed to the prior construction of this model, the few cases with missing values have been removed.

R Input

```
Fit.Model.WeightFinish.Vigor2 <- lm(WeightFinish ~
VigorInitial + Vigor100Lbs + Vigor200Lbs + VigorFinish,
data=LStockVg2.df)
# Reminder: all cases with missing data were removed as
# LStockVg2.df was created.

summary(Fit.Model.WeightFinish.Vigor2)
```

R Output

[Selected output is not shown, to save space.]

Call:
`lm(formula = WeightFinish ~ VigorInitial + Vigor100Lbs +
Vigor200Lbs + VigorFinish, data = LStockVg2.df)`

Coefficients:

	Std. Error	t value	Pr(> t)
(Intercept)	5.267	28.48 < 0.0000000000000002	***
VigorInitial	0.485	6.49 0.0000000000097	***
Vigor100Lbs	1.102	8.68 < 0.0000000000000002	***
Vigor200Lbs	1.897	-1.40 0.16	
VigorFinish	1.572	0.92 0.36	

Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1		

R Input

```
summary.aov(Fit.Model.WeightFinish.Vigor2)
```

R Output

[Selected output is not shown, to save space.]

	Df	Mean Sq	F value	Pr(>F)
VigorInitial	1	21197	175.67 < 0.0000000000000002	***
Vigor100Lbs	1	43088	357.08 < 0.0000000000000002	***
Vigor200Lbs	1	139	1.15 0.28	
VigorFinish	1	103	0.85 0.36	
Residuals	3288	121		

Signif. codes:	0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1			

With the model in correct format, apply the step() function to begin the step-wise regression process.¹³ For this example, use a backward stepwise regression. Notice, below, how the model steps (in a backward fashion) until only those object variables of importance to the construction of a prediction equation remain. The coefficients are then shown for those remaining variables, VigorInitial and Vigor100Lbs as will be seen later.

R Input

```
step(Fit.Model.WeightFinish.Vigor2, direction="backward")
```

¹³The step() function is associated with the stats package.

R Output

Start: AIC=15788.4
 WeightFinish ~ VigorInitial + Vigor100Lbs + Vigor200Lbs +
 VigorFinish

	Df	Sum of Sq	RSS	AIC
- VigorFinish	1	103	396853	15787
- Vigor200Lbs	1	237	396988	15788
<none>			396751	15788
- VigorInitial	1	5086	401837	15828
- Vigor100Lbs	1	9086	405837	15861

Step: AIC=15787.3
 WeightFinish ~ VigorInitial + Vigor100Lbs + Vigor200Lbs

	Df	Sum of Sq	RSS	AIC
- Vigor200Lbs	1	139	396992	15786
<none>			396853	15787
- VigorInitial	1	5179	402033	15828
- Vigor100Lbs	1	9066	405920	15860

Step: AIC=15786.4
 WeightFinish ~ VigorInitial + Vigor100Lbs

	Df	Sum of Sq	RSS	AIC
<none>			396992	15786
- VigorInitial	1	5065	402058	15826
- Vigor100Lbs	1	43088	440080	16124

Call:

```
lm(formula = WeightFinish ~ VigorInitial + Vigor100Lbs,
  data = LStockVg2.df)
```

Coefficients:

(Intercept)	VigorInitial	Vigor100Lbs
149.23	3.12	8.47

Go back to the prior example, where VigorInitial was 8.73 and Vigor100Lbs was 8.63. What is the predicted value for WeightFinish? From the output of this backward stepwise regression it is now known that calculation for \hat{Y} and the prediction equation that best fits the model is:

$\hat{Y} = a + b(x) + b(y)$

`WeightFinish = 149.23 + VigorInitial(3.12) + Vigor100Lbs(8.47)`

`WeightFinish = 149.23 + (8.73 * 3.12) + (8.63 * 8.47)`

`WeightFinish = 149.23 + 27.2376 + 73.0961`

`WeightFinish = 249.564`

Compare the similarity of this prediction equation, generated through the use of a backward stepwise regression (all cases with missing data were removed from the dataset) to the prior prediction equation that was manually prepared and may have included cases that had missing data in the dataset. The differences between the two equations are slight.

From a practical viewpoint, the importance of this stepwise regression is that the manager of the livestock finishing operation can influence final weight early on, up to the time weights approach 100 pounds. At 200 pounds and beyond the manager would have less direct influence on attainment of final weight. Early intervention yields desired results regarding finishing weight of the livestock.

As an interesting value-added exercise, use the `predict()` function to determine the model-based prediction of final weight for an animal with a vigor value of 8.00 for all four vigor measures: `VigorInitial`, `Vigor100Lbs`, `Vigor200Lbs`, and `VigorFinish`.

R Input

```
predict(Fit.Model.WeightFinish.Vigor, list(VigorInitial=8.00,  
Vigor100Lbs=8.00, Vigor200Lbs=8.00, VigorFinish=8.00))
```

R Output

```
1  
242.026
```

Given these parameters, the animal would have a finished weight of 242.026 pounds, which is below the desired finished weight of 250 pounds. Now look at what happens if `VigorInitial` and `Vigor100Lbs` are both increased to 8.55 while `Vigor200Lbs` and `VigorFinish` remain 8.00.

R Input

```
predict(Fit.Model.WeightFinish.Vigor, list(VigorInitial=8.55,  
Vigor100Lbs=8.55, Vigor200Lbs=8.00, VigorFinish=8.00))
```

R Output

```
1  
249.017
```

By deploying management practices early on and in turn improving vigor at the beginning of the finishing operation, it is possible to approach desired weight (250 pounds) in this hypothetical example.

Correlation and regression are focused on groups and not specific individuals. Even so, it is interesting to adjust predictors and to use this exercise as a *What-If* tool in model building for future desired results. This limited exercise may help the livestock finishing operator justify the purchase cost of feeder stock that meet or exceed $\text{VigorInitial} = 8.55$ and to also apply management practices to be sure that $\text{Vigor100Lbs} = 8.85$ or greater. Again, this is merely one example of how biostatistics can contribute to both better knowledge of the biological sciences and also contribute to improved management practices and eventual return-on-investment.

7.8 Addendum 2: Likelihood and Odds Ratio

Clarity of terms is important in biostatistics and far too often terms related to correlation, association, and prediction are often used incorrectly. To have a good understanding of how correlation is used as an underlying foundation for concepts such as likelihood, odds ratio, and eventually prediction it is necessary to first come to grips with the term probability.

Perhaps the best way to describe probability in this lesson is to use R and model a coin toss, for the binary outcome of either heads or tails. If an honest (e.g., unaltered) coin is tossed correctly, then it is reasonable to think that the probability of heads is one out of two and the probability of tails is also one out of two. But, how is it possible to explain how someone can toss a coin and get either heads or tails five, seven, nine, etc., times in a row, which occasionally happens?¹⁴ With a limited number of coin tosses these oddities are always possible, but what happens when a coin is tossed 100, 1000, 100,000, or 1,000,000 times?

For this addendum, R is used to simulate 1,000,000 coin tosses. What is the probability that a heads will show 500,000 times and that tails will show 500,000 times?

¹⁴Search on the terms *Monte Carlo Fallacy* or *Gambler's Fallacy* to learn more about this occasional event, whether involving a coin toss, game of roulette, etc.

R Input

```
base::set.seed(8)                      # Set the seed.  
# See prior lessons why the set.seed() function  
# is used to provide consistent outcomes when  
# constructing user-created data and equally  
# why 8 was used and not, 3, 123, or some other  
# number for the seed.  
  
#####  
# Code Book for CoinToss #  
#####  
#  
# Heads ..... 0      #  
# Tails ..... 1      #  
#####  
  
CoinToss <- sample(0:1, 1000000, replace=TRUE)  
  
table(CoinToss)
```

R Output

```
CoinToss  
0      1  
500019 499981
```

R Input

```
# Determine the probability of heads and tails from one  
# million coin tosses.  
  
ProbHeads <- sum((CoinToss==0) / length(CoinToss))  
format(ProbHeads, nsmall=10)
```

R Output

```
[1] "0.5000190000"
```

R Input

```
ProbTails <- sum((CoinToss==1) / length(CoinToss))  
format(ProbTails, nsmall=10)
```

R Output

```
[1] "0.4999810000"
```

R Input

```
ProbHeadsTails <- sum(ProbHeads + ProbTails)  
format(ProbHeadsTails, nsmall=10)
```

R Output

```
[1] "1.0000000000"
```

The use of 0s and 1s was needed at first, but now put all of this output to the screen into more meaningful language.

R Input

```
CoinToss <-  
  factor(CoinToss,  
  labels=c("Heads", "Tails"))  
levels(CoinToss)  
# NOTE: factor(...) and NOT as.factor (...)
```

R Output

```
[1] "Heads" "Tails"
```

R Input

```
CoinTossTable <- table(CoinToss)  
CoinTossTable
```

R Output

```
CoinToss
  Heads   Tails
 500019 499981
```

Use the epiDisplay::tab1() function to produce a simple barplot and confirming screen print of frequency distribution and percentages for the binary outcome of heads or tails from one million coin tosses (Fig. 7.31).

R Input

```
epiDisplay::tab1(CoinToss, decimal=4, sort.group=FALSE,
  cum.percent=TRUE, graph=TRUE, missing=TRUE,
  bar.values=c("percent"), horiz=TRUE, cex=1.15,
  cex.names=1.15, cex.lab=1.15, cex.axis=1.15,
  main="Percentage of Heads or Tails: One Million Coin Tosses",
  xlab="Percentage", col= c("red", "black"), gen=TRUE)
```

R Output

	Frequency	Percent	Cum. percent
Heads	500019	50.0019	50.0019
Tails	499981	49.9981	100.0000
Total	1000000	100.0000	100.0000

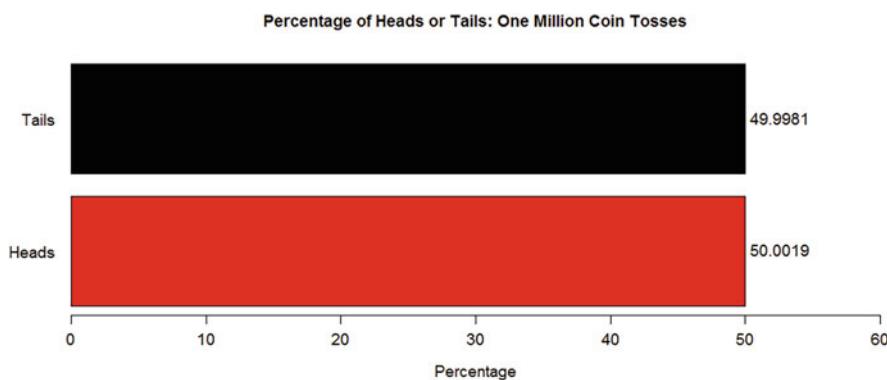


Figure 7.31: Percentage distribution of coin tosses

As seen in the screen print and also in the accompanying figure, after one million tosses the simulated mechanical activity of a coin toss was in near parity

with the expectation that nearly one-half of all outcomes resulted in heads and nearly one-half of all outcomes resulted in tails. Be sure to observe use of the term mechanical activity when referring to this coin toss demonstration. In biostatistics, however, events do not always follow along in such a mechanical fashion.¹⁵

Probability, Odds, Odds Ratio, and Binary Logistic Regression

With this brief review on a simulated mechanical activity such as a coin toss, more information with implications for biostatistics on the terms probability, odds, and odds ratio may be useful for those who are not acquainted with the predictive capabilities of Binary Logistic Regression. The terms probability, odds, and odds ratio are related to each other, but they are not synonymous. Saying that, consider how probability becomes far more complicated when dealing with human and other biological subjects. As an example, the Leicester City Football Club was the unexpected winner of the Premier League 2015–16 season—unexpected at least according to bets placed before start of the season. Or, an example from baseball will also help explain the differences between these terms, using the Florida Marlins (National League) and their unexpected 2003 World Series championship over the New York Yankees (American League).

Probability

Similar to all other teams in Major League Baseball, in 2003 the Florida Marlins would either win the World Series or the Florida Marlins would not win the World Series.¹⁶ Although this statement is presented as a binary (e.g., 50-50) outcome, at the beginning of Spring Training in 2003, approximately 6 weeks before the season started, few (if anyone) would have stated that there was a 50% probability that the Florida Marlins would win the 2003 World Series. Championships are not picked at random or by the simulated mechanical toss of a coin but are instead the result of purposeful outcomes over months of game-by-game results.

Odds

In 2003 there were 30 teams in Major League Baseball. The Florida Marlins were one of 16 teams in the National League and there were 14 teams in the American League, prior to the 2013 realignment of both leagues at 15 teams each. Although the Florida Marlins were one of 16 teams in the National League

¹⁵If time and interest allow, go beyond the probability of a simple coin toss simulation and instead look into the probability of obtaining either a red or a black when playing roulette. First, study on the difference between the number of zeros on the roulette wheel, where there are two zeros on an American roulette wheel and one zero for standard roulette wheels. Then, study on how many opportunities there are for a roulette ball to land on red and how many opportunities there are for the roulette ball to land on black. This activity should be far more challenging than the simple binary activity of Heads or Tails when tossing a coin.

¹⁶Immediately, note how the expression *would not win the World Series* is not the same as saying *would lose the World Series*.

and one of these teams would go on to play in the World Series, at the beginning of Spring Training in 2003 few would have said that the odds of the Florida Marlins winning the National League championship and going on to the World Series were 1 out of 16. Even fewer would have stated at the beginning of Spring Training in 2003 that the odds of the Florida Marlins going on to win the World Series were 1 out of 30, given that one of 30 teams must win. An extensive search failed to find any pre-season record of legal bets that the Florida Marlins would win the World Series in 2003.

Odds Ratio

Following along with this scenario, consider the probability, at the beginning of Spring Training in 2003, that the Florida Marlins would play against and then beat the New York Yankees in the 2003 World Series. This event would have to take into consideration the odds that the Florida Marlins would win the 2003 National League championship (e.g., pennant) and equally the odds that the New York Yankees would win the 2003 American League championship (e.g., pennant), with the two league champions then going on the play in the World Series. In this scenario, there is a comparison of one set of odds to another set of odds. Even the most adventurous Las Vegas odds-maker (e.g., tout) would have had trouble conceptualizing this match-up in the 2003 World Series. Odds ratios (more specifically, odds ratio statistics) are reported in Binary Logistic Regression, going well-beyond probability and simple statements of odds.¹⁷

Binary Logistic Regression

A manual R-based example of how a hypothetical odds ratio statistic is calculated may help. In this hypothetical example of biological subjects, there was a total of 107,081 subjects. Note the breakout N for each of the four sub-groups, for Event (e.g., Failed v Passed) and Group (e.g., East v West).

Hypothetical Example of Event Completion (Fail v Passed) by Group (East v West)

	Group	
Event	East	West
Failed	13169	08755
Passed	36238	48919

¹⁷Immediately before the beginning of 2003 Spring Training, the Florida Marlins were predicted to finish last in the National League Eastern Division. As late as mid-season, legal sports betting sites only offered 100 to 1 odds that the Florida Marlins would win the 2003 World Series. There is no record of any odds offered at that time or earlier that the Florida Marlins would ever face the New York Yankees and go on to win the World Series.

R Input

```
EventGroup.tbl <- as.table(
  epiDisplay::make2x2(48919, 08755, 36238, 13169)
)
names(dimnames(EventGroup.tbl)) <- c("Event", "Group")
rownames(EventGroup.tbl) <- c("Failed", "Passed")
colnames(EventGroup.tbl) <- c("East", "West")

EventGroup.tbl
```

R Output

	Group	
Event	East	West
Failed	13169	8755
Passed	36238	48919

R Input

```
str(EventGroup.tbl)
```

R Output

```
'table' num [1:2, 1:2] 13169 36238 8755 48919
- attr(*, "dimnames")=List of 2
..$ Event: chr [1:2] "Failed" "Passed"
..$ Group: chr [1:2] "East" "West"
```

R Input

```
epiDisplay::cc(cctable=EventGroup.tbl,
  main="Odds Ratio of Event by Group",
  xlab="Group")
```

R Output

	Group		Total
Event	East	West	Total
Failed	13169	8755	21924
Passed	36238	48919	85157

Total 49407 57674 107081

OR = 2.03

95% CI = 1.97, 2.09

Chi-squared = 2151.63, 1 d.f., P value = 0

Fisher's exact test (2-sided) P value = 0

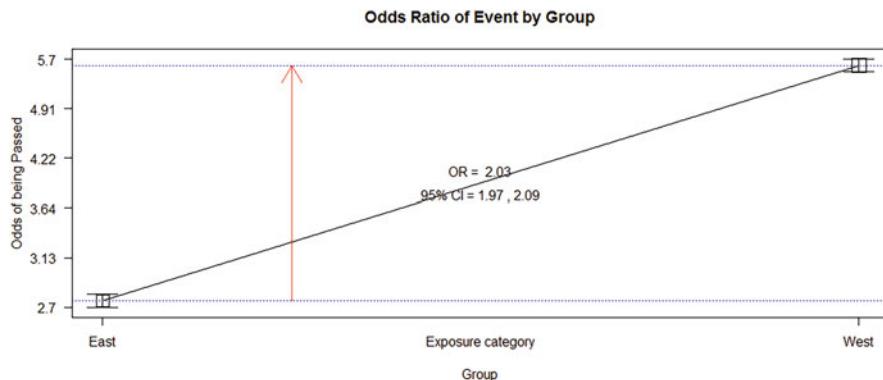


Figure 7.32: Display of odds ratio—1

The Odds Ratio (e.g., OR as shown above) is 2.03. If helpful, look at the way this statistic is hand-calculated to gain a complete sense of this vital finding (Fig. 7.32).

Odds of Event = Failed if Group = East
(13,169/36,238) or 0.363403

Odds of Event = Failed if Group = West
(8,755/48,919) or 0.178969

Odds Ratio = (0.363403/0.178969) or 2.03054

Give special attention to the accompanying Odds Ratio figure, the scale on the left side of the Odds Ratio figure, and the odds that subjects will pass, based on group membership:

- Odds of Passed is slightly more than 2.7 for East group members.
- Odds of Passed is slightly less than 5.7 for West group members.

Assuming that Passed is desirable, West group members clearly had more desirable outcomes than East group members. However, going beyond simple frequency percentages, it is now possible to offer a degree of prediction for Event breakouts (e.g., Failed v Passed) for Group breakouts (e.g., East v West).

Calculate Odds Ratio for Systolic Blood Pressure by Obesity

After viewing this hypothetical example of hand-calculation odds ratio statistics, go back to the front matter in this lesson and review blood pressure descriptive statistics from the wellness inventory. Consider specifically the odds ratio of changing status from Not Obese to Obese based on Systolic Blood Pressure. Then, after looking at Systolic Blood Pressure, approach this process again, but now focusing on the odds ratio of changing status from Not Obese to Obese based on Diastolic Blood Pressure. For both measures (Systolic Blood Pressure and Diastolic Blood Pressure), use the dataframe `EmployeeBiometric4.df` to avoid any concerns about missing data.

Determine the mean for Systolic Blood Pressure, as a rational cutpoint for the creation of two groups, one group Less Than (LT) mean SBP and the other group Greater Than or Equal To (GTE) mean SBP.

R Input

```
mean(EmployeeBiometric4.df$SBP)
```

R Output

```
[1] 121.766
```

R Input

```
EmployeeBiometric4SBPLT122.df <-
base::subset(EmployeeBiometric4.df,
EmployeeBiometric4.df$SBP < 122)
# EmployeeBiometric4SBPLT122.df is a subset
# of EmployeeBiometric4.df and consists of
# only those subjects with a SBPmmHg of
# less than (LT) 122, the rounded mean for
# SBPmmHg.
```

```
EmployeeBiometric4SBPGTE122.df <-
base::subset(EmployeeBiometric4.df,
EmployeeBiometric4.df$SBP >= 122)
# EmployeeBiometric4SBPGTE122.df is a subset
# of EmployeeBiometric4.df and consists of
# only those subjects with a SBPmmHg greater
# than or equal to (GTE) 122, the rounded
# mean for SBPmmHg.
```

```
summary(EmployeeBiometric4$Obesity)
```

R Output

Not Obese	Obese
1219	381

R Input

```
summary(EmployeeBiometric4$Obesity)
```

R Output

Not Obese	Obese
749	528

R Input

```
ObeseMeanSBP122.tbl <- as.table(  
  epiDisplay::make2x2(0528, 0749, 0381, 1219)  
)  
  
names(dimnames(ObeseMeanSBP122.tbl)) <-  
  c("Obesity", "Mean SBP")  
rownames(ObeseMeanSBP122.tbl) <-  
  c("Not Obese", "Obese")  
colnames(ObeseMeanSBP122.tbl) <-  
  c("SBPLT122", "SBPGTE122")  
  
ObeseMeanSBP122.tbl
```

R Output

Mean SBP		
Obesity	SBPLT122	SBPGTE122
Not Obese	1219	749
Obese	381	528

R Input

```
epiDisplay::cc(cctable=ObeseMeanSBP122.tbl,
  main="Odds Ratio of Obesity by SBP Mean",
  xlab="Group")
```

R Output

		Mean SBP		Total
Obesity		SBPLT122	SBPGTE122	
Not Obese		1219	749	1968
Obese		381	528	909
Total		1600	1277	2877

OR = 2.26

95% CI = 1.92, 2.65

Chi-squared = 101.03, 1 d.f., P value = 0

Fisher's exact test (2-sided) P value = 0

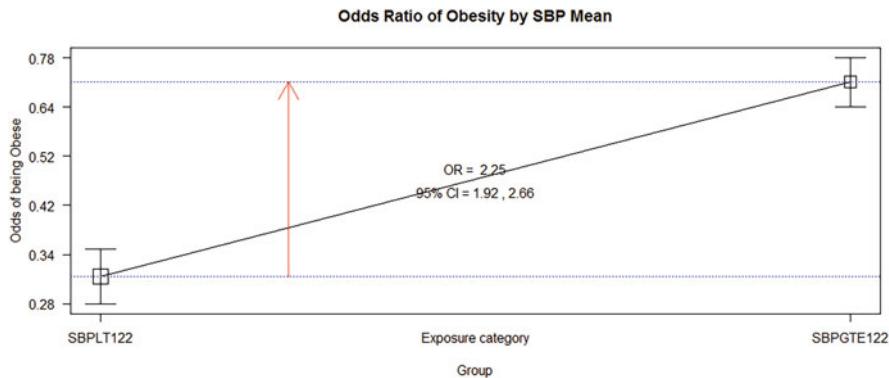


Figure 7.33: Display of odds ratio—2

The Odds Ratio (e.g., OR as shown above) is 2.26. If helpful, look at the way this statistic is hand-calculated to gain a complete sense of this vital finding (Fig. 7.33).

Odds of Obesity = Not Obese if Mean SBP < 122
(1219/381) or 3.19948

Odds of Obesity = Not Obese if Mean SBP >= 122
(749/528) or 1.41856

Odds Ratio = (3.19948/1.41856) or 2.25544

Give special attention to the accompanying Odds Ratio figure and how calculated odds can be used to predict that subjects will become obese, based on knowing Systolic Blood Pressure:

- The odds of becoming obese is approximately 0.30 for subjects with a Systolic Blood Pressure that is less than 122.
- In contrast, the odds of becoming obese is approximately 0.70 for subjects with a Systolic Blood Pressure that is greater than or equal to 122.

Calculate Odds Ratio for Diastolic Blood Pressure by Obesity

Determine the mean for Diastolic Blood Pressure, as a rational cutpoint for the creation of two groups, one group Less Than (LT) mean DBP and the other group Greater Than or Equal To (GTE) mean DBP.

R Input

```
mean(EmployeeBiometric4.df$DBP)
```

R Output

```
[1] 69.3646
```

R Input

```
EmployeeBiometric4DBPLT69.df <-
base::subset(EmployeeBiometric4.df,
EmployeeBiometric4.df$DBP < 69)
# EmployeeBiometric4DBPLT69.df is a subset
# of EmployeeBiometric4.df and consists of
# only those subjects with a DBPmmHg of
# less than (LT) 69, the rounded mean for
# DBPmmHg.
```

```
EmployeeBiometric4DBPGTE69.df <-
base::subset(EmployeeBiometric4.df,
EmployeeBiometric4.df$DBP >= 69)
# EmployeeBiometric4DBPGTE69.df is a subset
# of EmployeeBiometric4.df and consists of
# only those subjects with a DBPmmHg greater
# than or equal to (GTE) 69, the rounded
# mean for DBPmmHg.
```

```
summary(EmployeeBiometric4DBPLT69.df$Obesity)
```

R Output

Not Obese	Obese
1031	328

R Input

```
summary(EmployeeBiometric4DBPGTE69.df$Obesity)
```

R Output

Not Obese	Obese
937	581

R Input

```
ObeseMeanDBP69.tbl <- as.table(
  epiDisplay::make2x2(0581, 0937, 0328, 1031)
)

names(dimnames(ObeseMeanDBP69.tbl)) <-
  c("Obesity", "Mean DBP")
rownames(ObeseMeanDBP69.tbl) <-
  c("Not Obese", "Obese")
colnames(ObeseMeanDBP69.tbl) <-
  c("DBPLT69", "DBPGTE69")

ObeseMeanDBP69.tbl
```

R Output

	Mean DBP	
Obesity	DBPLT69	DBPGTE69
Not Obese	1031	937
Obese	328	581

R Input

```
epiDisplay::cc(cctable=ObeseMeanDBP69.tbl,
  main="Odds Ratio of Obesity by DBP Mean",
  xlab="Group")
```

R Output

```

Mean DBP
Obesity      DBPLT69 DBPGTE69 Total
  Not Obese    1031      937  1968
  Obese        328       581  909
  Total        1359     1518 2877

OR = 1.95
95% CI = 1.66, 2.29
Chi-squared = 66.32, 1 d.f., P value = 0
Fisher's exact test (2-sided) P value = 0

```

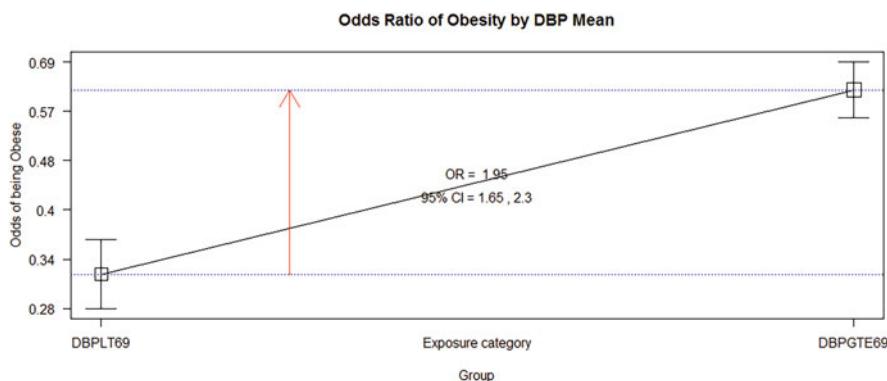


Figure 7.34: Display of odds ratio—3

The Odds Ratio (e.g., OR as shown above) is 1.95. If helpful, look at the way this statistic is hand-calculated to gain a complete sense of this vital finding (Fig. 7.34).

Odds of Obesity = Not Obese if Mean DBP < 69
(1031/328) or 3.14329

Odds of Obesity = Not Obese if Mean DBP >= 69
(937/581) or 1.61274

Odds Ratio = (3.14329/1.61274) or 1.94904

Give special attention to the accompanying figure and how calculated odds can be used to predict that subjects will become obese, based on knowing Diastolic Blood Pressure:

- The odds of becoming obese is approximately 0.32 for subjects with a Diastolic Blood Pressure that is less than 69.
- In contrast, the odds of becoming obese is approximately 0.68 for subjects with a Diastolic Blood Pressure that is greater than or equal to 69.

7.9 Addendum 3: Parametric v Nonparametric

Normality tests and QQ plots were demonstrated in the front matter of this lesson. The numeric object variables were examined overall and by breakouts of the factor-type object variables. These actions are an attempt to address data distribution patterns and inquiries into the selection of either a parametric or nonparametric approach to correlation analyses.

There were reasonable concerns about normality for the numeric object variables AgeYears, TotalCholesterolmgdL, SBPmmHg, DBPmmHg, and BMIMetric. In an abundance of caution, it was decided to approach correlational analyses of these variables from both a parametric and nonparametric perspective, largely to see if there were any practical differences in observed correlation coefficients from both perspectives.

As stated earlier, when conducting a correlation analysis it is necessary to declare the use of either Pearson's r for a parametric view of the data or Spearman's rho for a nonparametric view of the data. The prior section on normal distribution, in the front matter of this lesson, provides a rationale for why this issue is important as a Quality Assurance issue.

R Input

```
round(cor(EmployeeBiometric3.df,
  use="complete.obs",           # Need to account for NA
  method=c("pearson")),2)      # Nonparametric
# Generate Pearson's r correlation coefficients
# for multiple variables and wrap the round()
# function around the cor() function so that
# output displays the correlation coefficients at
# two places to the right of the decimal point.
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.00	0.03	-0.01	-0.02	0.00
Cholesterol	0.03		1.00	-0.01	0.02
SBP	-0.01		-0.01	1.00	0.49
DBP	-0.02		0.02	0.49	1.00
BMI	0.00		0.01	0.24	0.24

R Input

```
round(cor(EmployeeBiometric3.df,
use="complete.obs",           # Need to account for NA
method=c("spearman")),2) # Nonparametric
# Generate Spearman's rho correlation
# coefficients for multiple variables and wrap
# the round() function around the cor() function
# so that output displays the correlation
# coefficients at two places to the right of the
# decimal point.
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.00	0.03	0.00	-0.02	-0.01
Cholesterol	0.03		1.00	-0.01	0.02
SBP	0.00		-0.01	1.00	0.52
DBP	-0.02		0.02	0.52	1.00
BMI	-0.01		0.01	0.31	0.26

As a new topic to this lesson, the Kendall Rank Correlation Coefficient or Kendall's tau is also used for correlation analyses. Kendall's tau is another nonparametric test, using a rank-based approach to correlation. Spearman's rho is usually the first choice for nonparametric correlation analyses, but Kendall's tau may be useful for some occasions.

R Input

```
round(cor(EmployeeBiometric3.df,
use="complete.obs",           # Need to account for NA
method=c("kendall")),2) # Nonparametric
# Generate Kendall's tau correlation coefficients
# for multiple variables and wrap the round()
# function around the cor() function so that
# output displays the correlation coefficients at
# two places to the right of the decimal point.
```

R Output

	Age	Cholesterol	SBP	DBP	BMI
Age	1.00	0.02	0.00	-0.01	-0.01
Cholesterol	0.02		1.00	-0.01	0.01

SBP	0.00	-0.01	1.00	0.38	0.21
DBP	-0.01	0.01	0.38	1.00	0.18
BMI	-0.01	0.01	0.21	0.18	1.00

It is beyond the purpose of this lesson to go into a long discussion on differences between Spearman's rho and Kendall's tau. Spearman's rho is certainly more widely used when conducting a nonparametric rank-based correlation analysis, but it is often prudent to apply Kendall's tau against the data as a redundant Quality Assurance check. As seen in the examples, there is some degree of difference between output when using Spearman's rho and Kendall's tau, but are there any *practical* differences in terms of eventual inferences and application of outcomes? As always, multiple approaches to data analysis have value in that Quality Assurance is a never ending goal.

7.10 Addendum 4: Additional Practice Datasets

There is only one dataset for this addendum since the data are quite comprehensive, with multiple variables that support a wide variety of analyses, and with varying degrees of normality (or deviation away from normality) for the numeric variables. Similar to what was provided in the front matter of this lesson and all other lessons, follow along with standard practices: import the data, consider the Code Book (provided below), generate figures that explain relationships between and among the data, address descriptive statistics and measures of central tendency, look at the issue of normality for numeric variables overall as well as normality by factor-type object variable breakouts, conduct the analyses presented in this addendum and other analyses that may be of interest, and prepare a summary that provides guidance on outcomes.¹⁸

Note In the front matter to this lesson, the syntax is presented and is then immediately followed in most cases by either a copy of screen output or an accompanying figure. That approach is not used in all parts of this addendum. View this addendum as practice homework-type bonus materials. Syntax is presented and descriptive text goes along with the syntax. However, screen output and figures are often excluded and when they show it is only to offer mid-point guidance that analyses follow along in a correct manner. Either key the syntax in this addendum or copy and paste it into an editor—whatever

¹⁸Syntax is provided throughout this addendum, but of course this syntax is only a suggestion. Experiment and take other approaches to how the data can be analyzed and outcomes presented by using other functions and other arguments. For the first part of this addendum the syntax is presented, but the screen output is generally not presented since the output is straight-forward and often a review of activities from prior lessons. However, the second part of this addendum includes both syntax and some output since the focus is on the way correlation and regression serve as a basis for prediction, which can be challenging without extensive experience. Use this addendum as a confidence-building resource on how R is used with increasingly complex analyses.

is feasible and most convenient. And then, to use the common expression, Practice–Practice–Practice!

The Code Book for the data in this addendum is generally descriptive and otherwise only a minimal amount of information about the data has been shared, purposely:

- The data are related to rehabilitation services provided to human subjects, but the nature of the rehabilitation program is unknown.
- A few descriptors are known, such as EndingYear, Gender, and RaceEthnic.
- Subjects are asked, but not required, to participate in two eligibility screening tests prior to the beginning of rehabilitation activities: ScreeningScore1 and ScreeningScore2.
- Subjects are asked to participate in evaluations periodically over the first year: Score03Months, Score06Months, Score12Months. At the end of the first year they are asked to participate in a formative Fail v Pass assessment of progress (Event12MonthsFailPass). A failing mark at 12 months does not mean that services are terminated and a passing mark does not mean the program has been completed.
- Subjects are asked to participate in an additional evaluation during their 18th month (Score18Months).
- Sometime after the 18th month, on a subject-by-subject basis, there is a determination of subject readiness to exit the rehabilitation program. At that time, subjects are asked to participate in an additional evaluation (ScoreExit) and an ending Fail v Pass assessment of progress (EventExitFailPass).
- As an ending activity, the total number of months needed for exit are marked (MonthsToComplete). A successful exit is not assured, subjects are free to leave the program at any time, and they can also be terminated from the program if future expectations for progress are questioned, or for other reasons.

R Input

```
Rehabilitation.df <- read.table(file = "Rehabilitation.csv",
  header = TRUE, sep = ",") # Import the .csv file

getwd()                      # Identify the working directory
ls()                          # List objects
attach(Rehabilitation.df)    # Attach the data, for later use
```

```

str(Rehabilitation.df)          # Identify structure
nrow(Rehabilitation.df)         # List the number of rows
ncol(Rehabilitation.df)         # List the number of columns
dim(Rehabilitation.df)          # Dimensions of the data frame
names(Rehabilitation.df)        # Identify names
colnames(Rehabilitation.df)     # Show column names
rownames(Rehabilitation.df)     # Show row names
head(Rehabilitation.df)         # Show the head
tail(Rehabilitation.df)         # Show the tail
# Rehabilitation.df             # Show the entire data frame
summary(Rehabilitation.df)      # Summary statistics

```

By completing this action, an object called `Rehabilitation.df` has been created. This R-based object is a dataframe and it consists of the data originally included in the file `Rehabilitation.csv`, a comma-separated .csv file. To avoid possible conflicts, make sure that there are no prior R-based objects called `Rehabilitation.df` in the active R session.

R Input

```

#####
# Code Book for Rehabilitation.df          #
#####
#                                         #
# EndingYear           Factor (e.g., nominal or ordinal) #
#                                         Range: 2007 to 2019 #
#                                         #
# Gender                Factor (e.g., nominal) #
#                                         Female and Male #
#                                         #
# RaceEthnic            Factor (e.g., nominal) #
#                                         Standard naming practices that combine #
#                                         race and ethnicity #
#                                         #
# ScreeningScore1       Numeric, range 0 to 200 #
# ScreeningScore2       Numeric, range 0.00 to 5.00 #
# Score03Months         Numeric, range 0.00 to 5.00 #
# Score06Months         Numeric, range 0.00 to 5.00 #
# Score12Months         Numeric, range 0.00 to 5.00 #
# Score18Months         Numeric, range 0.00 to 5.00 #
# ScoreExit              Numeric, range 0.00 to 5.00 #
#                                         #
# Event12MonthsFailPass Nominal, 0 = Fail 1 = Pass #

```

```
# EventExitFailPass      Nominal, 0 = Fail 1 = Pass #
#
# MonthsToComplete     Interval, 18 months onward #
#####
#####
```

It must be mentioned that there are missing data in `Rehabilitation.csv`. The reasons for missing data are unknown to the researcher.

Guided Direction: Prepare a set of figures needed to have full understanding of the data, including frequency distributions of the factor-type object variables. As an aid, consider as one of many possible examples a set of figures as simple as bar charts for each factor-type object variable, but other R-based functions should also be explored:

R Input

```
par.ask=TRUE)
barplot(table((as.factor(Rehabilitation.df$EndingYear))),
       col="red", main="Ending Year")
par.ask=TRUE)
barplot(table((Rehabilitation.df$Gender)), col="red",
       main="Gender")
par.ask=TRUE)
barplot(table((Rehabilitation.df$RaceEthnic)), col="red",
       main="RaceEthnic")
par.ask=TRUE)
barplot(table((as.factor(
  Rehabilitation.df$Event12MonthsFailPass))), col="red",
       main="Event12MonthsFailPass")
par.ask=TRUE)
barplot(table((as.factor(
  Rehabilitation.df$EventExitFailPass))), col="red",
       main="EventExitFailPass")
par.ask=TRUE)
barplot(table((as.factor(
  Rehabilitation.df$MonthsToComplete))), col="red",
       main="MonthsToComplete")
# It may be appropriate to view MonthsToComplete as a
# factor-type object variable due to the somewhat
# unique distribution pattern and allowance for
# extended rehabilitation program participation.
#
# The as.factor() function is needed for object
```

```
# variables that are numeric, but treated as factors.
```

Guided Direction: Prepare a set of figures needed to have full understanding of the data, including calculations of descriptive statistics and measures of central tendency of the numeric object variables. As an aid, consider as one of many possible examples a set of figures as simple as density plots for each measured variable, but other R-based functions should also be explored:

R Input

```
par(ask=TRUE); plot(density(Rehabilitation.df$ScreeningScore1,
  na.rm=TRUE), col="red", lwd=4, main="ScreeningScore1")
par(ask=TRUE); plot(density(Rehabilitation.df$ScreeningScore2,
  na.rm=TRUE), col="red", lwd=4, main="ScreeningScore2")
par(ask=TRUE); plot(density(Rehabilitation.df$Score03Months,
  na.rm=TRUE), col="red", lwd=4, main="Score03Months")
par(ask=TRUE); plot(density(Rehabilitation.df$Score06Months,
  na.rm=TRUE), col="red", lwd=4, main="Score06Months")
par(ask=TRUE); plot(density(Rehabilitation.df$Score12Months,
  na.rm=TRUE), col="red", lwd=4, main="Score12Months")
par(ask=TRUE); plot(density(Rehabilitation.df$Score18Months,
  na.rm=TRUE), col="red", lwd=4, main="Score18Months")
par(ask=TRUE); plot(density(Rehabilitation.df$ScoreExit,
  na.rm=TRUE), col="red", lwd=4, main="ScoreExit")
par(ask=TRUE); plot(density(Rehabilitation.df$MonthsToComplete,
  na.rm=TRUE), col="red", lwd=4, main="MonthsToComplete")
```

Guided Direction: Prepare a comprehensive set of descriptive statistics and measures of central tendency of both the factor-type and numeric object variables. Start with the `summarytools` package and functions available in this package, but there are many possible ways to achieve this aim and try a few, to look for consistency in outcomes.

R Input

```
install.packages("summarytools", dependencies=TRUE)
library(summarytools)                      # Load the package.
help(package=summarytools)                 # Information page.
sessionInfo()                            # Confirm attached packages.

# Factor-type object variables
```

```
summarytools::freq(Rehabilitation.df[, 1:3])
# Search for R package summarytools to learn about the
# many functions available in this highly versatile
# package, for both factor-type object variables,
# numeric variables, and also for a detailed and highly
# attrractive summary of the entire dataframe.
#
# Generate frequency distributions for object variables
# in columns 1 to 3.

summarytools::ctable(Rehabilitation.df$EndingYear,
Rehabilitation.df$Gender, prop="r", method="render")
# Generate a Row by Column crosstab: EndingYear by
# Gender.

summarytools::ctable(Rehabilitation.df$EndingYear,
Rehabilitation.df$RaceEthnic, prop="r", method="render")
# Generate a Row by Column crosstab: EndingYear by
# RaceEthnic.

summarytools::ctable(Rehabilitation.df$RaceEthnic,
Rehabilitation.df$Gender, prop="r", method="render")
# Generate a Row by Column crosstab: RaceEthnic by
# Gender.
#
# Hint: Given a choice, make the object variable with
# the greatest number of breakout groups the Row and the
# object variable with the fewest number of breakout
# groups the Column. This practice improves placement
# when the output is copied into a word-processed
# document.

# Numeric object variables

summarytools::descr(Rehabilitation.df[, 4:10],
stats=c("mean", "sd", "min", "max", "q1", "med", "q3",
"n.valid", "pct.valid"), transpose=TRUE, headings=TRUE)
# Generate descriptive statistics for numeric object
# variables in columns 4 to 10.

# Overview of the entire dataset

summarytools::view(summarytools::dfSummary(Rehabilitation.df))
# Give attention to the pathway of the .html output file, if
# needed for later placement into a report.
```

Guided Direction: Prepare a set of QQ plots and tests of normality of the numeric object variables. As an aid, look below at the way inquiries into normality have been addressed.

R Input

```
par(asq=TRUE); qqnorm(Rehabilitation.df$ScreeningScore1,
  main="ScreeningScore1")
par(asq=TRUE); qqnorm(Rehabilitation.df$ScreeningScore2,
  main="ScreeningScore2")
par(asq=TRUE); qqnorm(Rehabilitation.df$Score03Months,
  main="Score03Months")
par(asq=TRUE); qqnorm(Rehabilitation.df$Score06Months,
  main="Score06Months")
par(asq=TRUE); qqnorm(Rehabilitation.df$Score12Months,
  main="Score12Months")
par(asq=TRUE); qqnorm(Rehabilitation.df$Score18Months,
  main="Score18Months")
par(asq=TRUE); qqnorm(Rehabilitation.df$ScoreExit,
  main="ScoreExit")
par(asq=TRUE); qqnorm(Rehabilitation.df$MonthsToComplete,
  main="MonthsToComplete")

shapiro.test(Rehabilitation.df$ScreeningScore1)
shapiro.test(Rehabilitation.df$ScreeningScore2)
shapiro.test(Rehabilitation.df$Score03Months)
shapiro.test(Rehabilitation.df$Score06Months)
shapiro.test(Rehabilitation.df$Score12Months)
shapiro.test(Rehabilitation.df$Score18Months)
shapiro.test(Rehabilitation.df$ScoreExit)
shapiro.test(Rehabilitation.df$MonthsToComplete)
```

Guided Direction: Do not immediately dismiss the shapiro.test() function results.¹⁹ As an example, look at the Shapiro Test p-value forScore03Months. Does the p-value, at first glance, deserve further attention? Is it possible that distribution is not consistent between the two breakouts, Female and Male? Diligent researchers go beyond first-level statistics and instead look for outcomes that are not always immediately obvious. Observe how normality for Score03Months is examined by Gender (e.g., both statistically and graphically), which yields results that demand attention. Repeat this type of analysis and

¹⁹Search the literature to see how normality is viewed when samples are large, such as the numeric object variables in Rehabilitation.df, where there are thousands of datapoints for each numeric object variable. Does a large N mitigate concerns about normality?

presentation for all possible configurations of normality of the numeric object variable by factor-type object variable breakouts.

R Input

```
RVAideMemoire::byf.shapiro(Score03Months ~ Gender,  
  data=Rehabilitation.df)  
  
ggplot2::ggplot(Rehabilitation.df,  
  aes(sample=Score03Months)) +  
  stat_qq(color="red") +  
  stat_qq_line(color="blue", size=1.75) +  
  facet_grid(. ~ Gender) +  
  ggtitle("QQ - Score03Months")
```

7.10.1 Data with Normal Distribution Patterns

Guided Direction: There seems to be an acceptable degree of normality for the Gender breakouts of Score03Months. As an interesting exercise, conduct a Student's t-Test for Independent Samples for these two object variables.

R Input

```
t.test(Rehabilitation.df$Score03Months ~ # Measured variable  
  Rehabilitation.df$Gender, # Grouping variable  
  alternative="two.sided", # Two-sided t-Test  
  paired=FALSE, # Independent samples  
  na.rm=TRUE, # Missing data  
  var.equal=TRUE) # Equal variance
```

Guided Direction: Using a Student's t-Test for Independent Samples, a parametric test that is based on the assumption that data exhibit a reasonable semblance of normal distribution, it was determined that there is no statistically significant difference ($p \leq 0.05$) between Females and Males in regard to Score03Months. Examine by Gender relationships for all other numeric object variables, using Student's t-Test for Independent Samples, to see overall trends between Females and Males in terms of progress in the rehabilitation program.

Guided Direction: Going into further analysis of the data, use Student's t-Test for Matched Pairs and compare Score03Months to Score06Months and all other appropriate test-retest comparisons. Using $p \leq 0.05$ as the criterion p-value for this matched pairs comparison, is there a statistically significant difference between scores at 3 months and scores at 6 months, between scores at 6 months and scores at 12 months, etc.?

R Input

```
t.test(Rehabilitation.df$Score03Months,      # Measured variable
       Rehabilitation.df$Score06Months,        # Measured variable
       paired=TRUE,                          # Matched pairs
       na.rm=TRUE)                         # Missing data
```

Guided Direction: Comparisons of numeric object variables by Gender involves only two breakouts: Female and Male. Use Oneway ANOVA to examine relationships by RaceEthnic, or six breakouts in this dataset: Asian, Black, Hispanic, Indigenous, Unidentified, and White. A typical example of this type of inquiry follows, but there are many more opportunities on the use of Oneway ANOVA for these data on rehabilitation services and periodic scores of progress. Explore these many relationships to look for over-time (e.g., 3 months, 6 months, 12 months, 18 months, and exit) trends. For each score time-period, which RaceEthnic groups are in common and which RaceEthnic groups show difference at $p \leq 0.05$? Use Tukey's HSD (Honestly Significant Difference) for all possible pairwise mean comparisons and complement these many analyses with a complete set of ggplot2-based figures.

R Input

```
TukeyHSD((aov(Score03Months ~ RaceEthnic,
               data=Rehabilitation.df)), conf.level=0.95)

par(ask=TRUE)
ggplot2::ggplot(data=Rehabilitation.df,
                 aes(x=RaceEthnic, y=Score03Months, fill=RaceEthnic)) +
  geom_boxplot() +
  stat_summary(fun.y=mean, color="black", geom="point",
               shape=18, size=2, show.legend=FALSE) +
  stat_summary(fun.y=mean, color="black", geom="text",
               size=6, show.legend=FALSE, vjust=1.50,
               aes(label=round(..y.., digits=1))) +
  labs(title=
    "Mean Rehabilitation Score at 3 Months by Race-Ethnicity",
```

```

caption="Mean shows as a small black dot inside the box
and median shows as a solid horizontal line.\n",
x="\nRace-Ethnicity",
y="Rehabilitation Score at 3 Months (Scale = 0 to 5)\n" +
scale_y_continuous(limits=c(1,5), breaks=seq(1,5,0.5)) +
theme(plot.caption=element_text(face="bold", size=08,
hjust=0.5, color="black")) +
theme(legend.position="none") +
theme_bw()

```

Guided Direction: Twoway ANOVA should also be considered, now focusing on Gender and RaceEthnic by Score03Months. Although the Twoway ANOVA output is helpful, the plot() function will most likely supplement an understanding of Gender and RaceEthnic breakout groups that are in common and those that are not in common ($p \leq 0.05$). Use the data to examine the many other possible Twoway ANOVA comparisons supported by the data in Rehabilitation.df. Complement each analysis with an accompanying figure.

R Input

```

install.packages("agricolae")
library(agricolae)           # Load the agricolae package.
help(package=agricolae)       # Show the information page.
sessionInfo()                 # Confirm all attached packages.

TwowayGR3Months <-
aov(Score03Months ~ Gender * RaceEthnic,
data=Rehabilitation.df)
# Twoway ANOVA for G(ender) and R(aceEthnic) --
# TwowayGR3Months

summary(TwowayGR3Months)
# Is there a statistically significant difference (p <= 0.05)
# in Score03Months by Gender? Is there a statistically
# significant difference (p <= 0.05) in Score03Months by
# RaceEthnic? Is there a statistically significant
# interaction (p <= 0.05) in Score03Months by Gender and
# RaceEthnicity (e.g., Gender:RaceEthnic )?
#
# Attempt this type of analysis for all possible combinations
# (e.g., ScreeningScore1 to ScoreExit ~ Gender * RaceEthnic).

agricolae::HSD.test(TwowayGR3Months, "Gender", group=TRUE,
console=TRUE)

```

```

# Use Tukey's HSD (Honestly Significant Difference) Mean
# Comparison Test to determine which breakouts are in common
# groups, and which breakout groups are not.
#
# Attempt this type of analysis for all possible combinations
# (e.g., ScreeningScore1 to ScoreExit and Gender).

agricolae::HSD.test(TwowayGR3Months, "RaceEthnic", group=TRUE,
  console=TRUE)
# Use Tukey's HSD (Honestly Significant Difference) Mean
# Comparison Test to determine which breakouts are in common
# groups, and which breakout groups are not.
#
# Attempt this type of analysis for all possible combinations
# (e.g., ScreeningScore1 to ScoreExit and RaceEthnic).

savecexaxis <- par(cex.axis=0.65) # Axis labels
savelwd <- par(lwd=4)           # Heavy line
par(ask=TRUE)      # Pause
par(mfrow=c(1,2)) # 2 figures - 1 row by 2 column grid
plot(agricolae::HSD.test(TwowayGR3Months, "Gender", alpha=0.05,
  group=TRUE),
  main="Common Subgroups of Score03Months
by Gender")
plot(agricolae::HSD.test(TwowayGR3Months, "RaceEthnic",
  alpha=0.05, group=TRUE),
  main="Common Subgroups of Score03Months
by RaceEthnic")
par(savelwd); par(savecexaxis)
# Plot Tukey's HSD (Honestly Significant Difference) to
# determine which breakouts are in common groups, and which
# are not.
# Notice the X axis and how breakout groups are ordered, not
# by alpha ordering but instead by mean.
#
# Prepare this type of figure for all possible combinations
# (e.g., ScreeningScore1 to ScoreExit and Gender-RaceEthnic).

```

Guided Direction: Going beyond these previously used graphical and statistical functions, use the CGPfunctions::Plot2WayANOVA() function, which produces an excellent summary of Twoway ANOVA statistics as well as an easy-to-understand interaction plot. The CGPfunctions::Plot2WayANOVA() function is quite rich and it should be explored in detail, using a variety of arguments, with the many other Twoway ANOVA tests supported by the data.

R Input

```
install.packages("CGPfunctions", dependencies=TRUE)
library(CGPfunctions)          # Load the CGPfunctions package.
help(package=CGPfunctions)    # Show the information page.
sessionInfo()                 # Confirm all attached packages.

CGPfunctions::Plot2WayANOVA(Score03Months ~
  RaceEthnic * Gender, Rehabilitation.df, plottype="line",
  interact.line.size=1, offset.style="wide",
  overlay.type="violin", mean.label=TRUE, mean.ci=FALSE,
  show.dots=TRUE, posthoc.method="scheffe")
# The default Scheffe mean comparison test for pairwise
# comparisons was selected, to offer an alternate from
# Tukey's HSD (Honestly Significant Difference) test.
#
# Attempt this type of analysis for all possible
# combinations (e.g., ScreeningScore1 to ScoreExit and
# RaceEthnic * Gender).
```

Guided Direction: As an alternate to the CGPfunctions::Plot2WayANOVA() function, attempt the same type of analysis using the interactions::cat_plot() function. Decide which figure meets immediate needs (reporting) and long term needs (presentation).

R Input

```
install.packages("interactions", dependencies=TRUE)
library(interactions)          # Load the interactions package.
help(package=interactions)    # Show the information page.
sessionInfo()                 # Confirm all attached packages.

TwowayGR3Monthslm <-
  lm(Score03Months ~ Gender * RaceEthnic,
  data=Rehabilitation.df)
# Twoway ANOVA model using lm (not aov) for
# G(ender) and R(aceEthnic) -- TwowayGR3Monthslm

summary(TwowayGR3Monthslm)
# Output is regression-focused, using a lm approach.
#
# Attempt this type of analysis for all possible
# combinations (e.g., ScreeningScore1 to ScoreExit and
# RaceEthnic * Gender).
```

```

interactions::cat_plot(TwowayGR3Monthslm, pred=RaceEthnic,
 modx=Gender, interval=TRUE, plot.points=TRUE, geom="point",
  int.type="prediction", line.thickness=0.75, point.size=1.50,
  main.title="Interaction Plot of Score03Months
by RaceEthnic and Gender")
# Prepare this type of figure for all possible combinations
# (e.g., ScreeningScore1 to ScoreExit and Gender-RaceEthnic).

```

7.10.2 Data That Do Not Exhibit Normal Distribution Patterns

This lesson is focused on correlation, not Student's t-Test for either Independent Samples or Matched Pairs, Oneway ANOVA, or Twoway ANOVA. Saying that, go back and look at the shapiro.test() function results from Rehabilitation.df and then compare the calculated p-values to the visualizations in the QQ plots. Is there a disconnect between the figures and Shapiro-based results on normality? How does this concern fit into discussion on the impact of a large N and practical aspects of normality? Then, consider if there is any way to link the concept of normal distribution to the binary object variables, such as the Fail v Pass assessments (e.g., Event12MonthsFailPass, EventExitFailPass)?

Guided Direction: Seeing that there is a degree of concern about normality, conduct a correlation analysis of the relevant variables using both a parametric approach and a nonparametric approach. To start, edit the column names to make screen prints easier to read, knowing that current column names are informative but verbose.

R Input

```

str(Rehabilitation.df)

colnames(Rehabilitation.df) <-
  c("Year", "Gender", "Race", "Screen1", "Screen2",
  "Month03", "Month06", "Month12", "Month18",
  "Exit", "FP12Months", "FPExit", "Months")
# Columns names are shortened to accommodate the way
# they will show in later graphical output.

attach(Rehabilitation.df)

str(Rehabilitation.df)

round(cor(Rehabilitation.df[ ,4:10],

```

```

use="complete.obs",           # Need to account for NA
method=c("pearson")),2)    # Nonparametric
# Generate Pearson's r correlation coefficients
# (parametric) for multiple variables and wrap
# the round() function around the cor() function
# so that output displays the correlation
# coefficients at two places to the right of the
# decimal point.
#
# The syntax [ ,4:10] refers to columns 4 to 10,
# which are the columns with numeric scores.

```

R Output

	Screen1	Screen2	Month03	Month06	Month12	Month18	Exit
Screen1	1.00	0.00	0.32	0.26	0.32	0.32	0.31
Screen2	0.00	1.00	0.25	0.24	0.27	0.29	0.30
Month03	0.32	0.25	1.00	0.61	0.87	0.80	0.77
Month06	0.26	0.24	0.61	1.00	0.90	0.85	0.83
Month12	0.32	0.27	0.87	0.90	1.00	0.93	0.90
Month18	0.32	0.29	0.80	0.85	0.93	1.00	0.97
Exit	0.31	0.30	0.77	0.83	0.90	0.97	1.00

R Input

```

round(cor(Rehabilitation.df[ ,4:10],
  use="complete.obs",           # Need to account for NA
  method=c("spearman")),2) # Nonparametric
# Spearman's rho (nonparametric)

```

R Output

	Screen1	Screen2	Month03	Month06	Month12	Month18	Exit
Screen1	1.00	-0.01	0.29	0.22	0.28	0.28	0.27
Screen2	-0.01	1.00	0.24	0.22	0.25	0.27	0.29
Month03	0.29	0.24	1.00	0.60	0.87	0.79	0.76
Month06	0.22	0.22	0.60	1.00	0.90	0.84	0.82
Month12	0.28	0.25	0.87	0.90	1.00	0.92	0.89
Month18	0.28	0.27	0.79	0.84	0.92	1.00	0.97
Exit	0.27	0.29	0.76	0.82	0.89	0.97	1.00

R Input

```
round(cor(Rehabilitation.df[,4:10],  
use="complete.obs",      # Need to account for NA  
method=c("kendall")),2) # Nonparametric  
# Kendall's tau (nonparametric)
```

R Output

	Screen1	Screen2	Month03	Month06	Month12	Month18	Exit
Screen1	1.00	-0.01	0.20	0.16	0.19	0.19	0.19
Screen2	-0.01	1.00	0.16	0.15	0.17	0.19	0.20
Month03	0.20	0.16	1.00	0.43	0.69	0.60	0.57
Month06	0.16	0.15	0.43	1.00	0.73	0.66	0.63
Month12	0.19	0.17	0.69	0.73	1.00	0.76	0.71
Month18	0.19	0.19	0.60	0.66	0.76	1.00	0.86
Exit	0.19	0.20	0.57	0.63	0.71	0.86	1.00

There are clearly differences in the correlation matrix in terms of calculated correlation coefficients when using Pearson's r, Spearman's rho, and Kendall's tau.²⁰ However, are the differences so great that they require different interpretation of results and impact of results, decision-making outcomes, etc.?

Guided Direction: Going beyond the production of a correlation matrix, use regression techniques to use the data for prediction purposes. Ostensibly, it could be said that the score at exit (e.g., Exit, previously ScoreExit) is the most important statistic in that it represents the final assessment of rehabilitation. Use a stepwise regression approach to build the prediction equation and to see which object variables deserve the greatest degree of attention in terms of their usefulness for prediction purposes.

R Input

```
Fit.Model.RehabilitationScores <-  
lm(Exit ~  
Screen1 + Screen2 + Month03 + Month06 + Month12 + Month18,  
data=Rehabilitation.df)  
  
summary(Fit.Model.RehabilitationScores)
```

²⁰Review the literature to see appropriate applications of the possibly more conservative nonparametric Kendall's tau, as opposed to use of the nonparametric Spearman's rho.

R Output

```

Call:
lm(formula = Exit ~ Screen1 + Screen2 + Month03 + Month06 +
Month12 + Month18, data = Rehabilitation.df)

Residuals:
    Min      1Q  Median      3Q     Max 
-0.3961 -0.0578 -0.0035  0.0518  0.5408 

Coefficients:
            Std. Error t value      Pr(>|t|)    
(Intercept) 0.047021   10.15 < 0.0000000000000002 ***  
Screen1      0.000342    0.97      0.3310    
Screen2      0.004713    4.70 0.00000028 ***  
Month03      0.014168    1.17      0.2434    
Month06      0.014354    2.85  0.0043 **  
Month12      0.029528   -3.00  0.0027 **  
Month18      0.012474   73.87 < 0.0000000000000002 ***  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0903 on 2403 degrees of freedom
(850 observations deleted due to missingness)
Multiple R-squared:  0.942,    Adjusted R-squared:  0.942 
F-statistic: 6.47e+03 on 6 and 2403 DF,
p-value: <0.0000000000000002

```

R Input

```
summary.aov(Fit.Model.RehabilitationScores)
```

R Output

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Screen1	1	32.6	32.6	4005 <0.0000000000000002	***
Screen2	1	29.7	29.7	3649 <0.0000000000000002	***
Month03	1	141.6	141.6	17376 <0.0000000000000002	***
Month06	1	64.2	64.2	7885 <0.0000000000000002	***
Month12	1	3.5	3.5	431 <0.0000000000000002	***
Month18	1	44.5	44.5	5457 <0.0000000000000002	***
Residuals	2403	19.6	0.0		

```
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
850 observations deleted due to missingness
```

Having confirmed that the model is in correct format using the `summary()` and `summary.aov()` functions, apply the `step()` function to begin the stepwise regression process. For this example, use a backward stepwise regression. Notice, below, how the model steps (in a backward fashion) until only those object variables of importance to the construction of a prediction equation remain. The coefficients are then shown for those remaining variables, or as will be seen `Screen2`, `Month06`, `Month12`, `Month18` in this example.

R Input

```
step(Fit.Model.RehabilitationScores, direction="backward")
# Take a backward stepwise regression approach for the
# eventual regression equation.
#
# For this addendum, delete the overly-detail listings
# of Df, Sum of Sq, RSS, and AIC and instead focus on
# the ending section that details the final object
# variables to include in the backward stepwise
# regression equation.
```

R Output

```
Call:
lm(formula = Exit ~ Screen2 + Month06 + Month12 +
    Month18, data = Rehabilitation.df)
```

Coefficients:

(Intercept)	Screen2	Month06	Month12	Month18
0.5180	0.0219	0.0263	-0.0567	0.9227

Using this ending part of the backward stepwise regression printout, it is now possible to build a regression equation for multiple predictor variables and in turn solve for Y-hat:²¹

$$\text{Y-hat} = 0.5189 + (\text{Screen2} * 0.0219) + (\text{Month06} * 0.0263) + \\ (\text{Month12} * -0.0567) + (\text{Month18} * 0.9227)$$

To make this illustration of a stepwise regression convenient, assume that an individual subject had a score of 3.25 for `Screen2`, `Month06`, `Month12`, and

²¹Review the prior note that Y-hat is the predicted outcome value (e.g., dependent variable) of a regression equation.

Month18 (scores can range from 0.0 to 5.0). Solve for Y-hat and predict the eventual score for Exit (e.g., originally based on a column name of ScoreExit).²²

$$\text{Y-hat} = 0.5189 + (3.25 * 0.0219) + (3.25 * 0.0263) + \\ (3.25 * -0.0567) + (3.25 * 0.9227)$$

$$\text{Y-hat} = 0.5189 + 0.071175 + 0.085475 + -0.184275 + 2.99877$$

$$\text{Y-hat} = 3.49004$$

Accordingly, an individual with a 3.25 score for Screen2, Month06, Month12, and Month18 can be expected to have a score at exit (e.g., Exit, ScoreExit) of 3.49004. Use the data to see if this prediction is reasonable. Use a brute-force Boolean approach to clearly demonstrate the selections used for this algorithm.

R Input

```
RcmdrMisc::numSummary(Rehabilitation.df$Exit [  
  (Rehabilitation.df$Screen2 >= 3.15 &  
   Rehabilitation.df$Screen2 <= 3.35) &  
  (Rehabilitation.df$Month06 >= 3.15 &  
   Rehabilitation.df$Month06 <= 3.35) & # Allow for + or -  
  (Rehabilitation.df$Month12 >= 3.15 & # 0.10 such that the  
   Rehabilitation.df$Month12 <= 3.35) & # Boolean search for  
  (Rehabilitation.df$Month18 >= 3.15 & # 3.25 goes from 3.15  
   Rehabilitation.df$Month18 <= 3.35)]) # to 3.35.
```

R Output

mean	sd	IQR	0%	25%	50%	75%	100%	n	NA
3.532	0.184581	0.11	3.22	3.53	3.58	3.64	3.69	5	6

Interestingly, the five subjects in Rehabilitation.df who met these Boolean search conditions (an approximate value of 3.25 for Screen2, Month06, Month12, and Month18, + or -0.10) had a Mean Exit score of 3.532 and a Median (the 50th percentile) Exit score of 3.58—a close approximation to the predicted (e.g., Y-hat) value for Exit of 3.49004. This confirmation of efficacy for the prediction equation provides a fair degree of confidence that it is now possible to predict Exit scores based on scores attained prior to exit from the rehabilitation program (e.g., *Past behavior is the best predictor of future behavior*). The actionable (e.g., practical) value of this finding is that there is now a basis for early intervention activities, based on the premise that an increase in early on scores will result in an increase of scores at the end of rehabilitation.

²²It is not a typographical error that the coefficient for Month12 is -0.0567, a negative number. Coefficients in a regression equation can be negative as well as positive.

Guided Direction: As useful as it may be to predict a numerical variable of interest, be sure to recall that many events in biostatistics are viewed from a binary perspective (e.g., Alive v Dead, Fail v Pass, Continue v Stop). Consider the application of Binary Logistic Regression against the data in `Rehabilitation.df`, focusing on the many possible variables that may have an association with `FP12Months`, the binary assessment of Fail or Pass (e.g., FP) at 12 months into the rehabilitation program.²³

R Input

```
str(Rehabilitation.df)

Rehabilitation.df$Year <- as.factor(Rehabilitation.df$Year)
# Transform Year into a factor-type object variable, so
# that it is applied correctly in the binary logistic
# regression algorithm.

attach(Rehabilitation.df)

str(Rehabilitation.df)
```

Build the Binary Logistic Regression logit (e.g., log odds or the logarithm of the odds) model, `FailPass12MonthsBLR`, by using the `glm()` function. If structured correctly, the logit model can include factor-type object variables as well as numeric object variables.

R Input

```
FailPass12MonthsBLR <- glm(FP12Months ~
  Year + Gender + Race + Screen1 + Screen2 + Month03 +
  Month06 + Month12, data = Rehabilitation.df,
  family = "binomial")
# Factor-type object variables and numeric object
# variables are included in the model.
#
# This example purposely ends with scores at 12 months.
#
# BLR, Binary Logistic Regression

summary(FailPass12MonthsBLR)
```

²³Binary Logistic Regression equations can easily accommodate factor-type object variables in addition to numeric object variables.

R Output

```

Call:
glm(formula = FP12Months ~ Year + Gender + Race + Screen1 +
    Screen2 + Month03 + Month06 + Month12,
    family = "binomial", data = Rehabilitation.df)

Deviance Residuals:
    Min      1Q  Median      3Q     Max 
-2.770 -0.722  0.383  0.720  2.522 

Coefficients: >>> Std.Error was deleted to save space. <<<
                         Estimate   z value    Pr(>|z|)    
(Intercept)          -16.8573  -10.35 < 0.0000000000000002 *** 
Year2008              -0.1623   -0.61      0.54086  
Year2009              -0.2766   -1.00      0.31861  
Year2010              -0.9826   -3.84      0.00012 ***  
Year2011              -0.6962   -2.61      0.00893 **  
Year2012              -0.6252   -2.29      0.02177 *  
Year2013              -0.9766   -3.76      0.00017 ***  
Year2014              -1.2803   -4.99      0.0000006139130 *** 
Year2015              -1.1995   -4.70      0.0000026559816 *** 
Year2016              -1.1728   -4.47      0.0000076647537 *** 
Year2017              -0.5604   -2.13      0.03340 *  
GenderMale             0.5130    4.90      0.0000009402235 *** 
RaceBlack              0.4643    1.20      0.22991  
RaceHispanic            0.2413    0.65      0.51466  
RaceIndigenous          -0.0803   -0.15      0.88369  
RaceUnidentified        0.4378    0.97      0.33425  
RaceWhite               0.4077    1.14      0.25534  
Screen1                 0.0728    6.83      0.0000000000083 *** 
Screen2                 0.4409    3.24      0.00119 **  
Month03                 0.4943    1.35      0.17660  
Month06                 0.8580    2.32      0.02042 *  
Month12                 1.2657    1.75      0.07983 .  
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 3241.3 on 2697 degrees of freedom
Residual deviance: 2434.7 on 2676 degrees of freedom
  (562 observations deleted due to missingness)
AIC: 2479

```

Number of Fisher Scoring iterations: 5

Finally, generate the Odds Ratio statistics for variables in the model. Be sure to notice how Odds Ratio statistics are provided not only for the numeric object variables but also for individual breakouts of the factor-type object variables.²⁴

R Input

```
exp(cbind(OddsRatio = coef(FailPass12MonthsBLR)))
  # Produce output that includes odds-ratio statistics
  # for each object variable in the logit model.
```

R Output

	OddsRatio
(Intercept)	0.0000000477471
Year2008	0.8501758399344
Year2009	0.7583239794873
Year2010	0.3743294655392
Year2011	0.4984816405861
Year2012	0.5351426581005
Year2013	0.3765931343047
Year2014	0.2779494290390
Year2015	0.3013346795920
Year2016	0.3094985204902
Year2017	0.5709748479866
GenderMale	1.6703239480027
RaceBlack	1.5908680222087
RaceHispanic	1.2729555959229
RaceIndigenous	0.9228006371354
RaceUnidentified	1.5492574496466
RaceWhite	1.5033263641408
Screen1	1.0755609775539
Screen2	1.5540524379680
Month03	1.6394055157586

²⁴At first it may seem that some Odds Ratio statistics for breakouts of factor-type object variables are missing, but that is not the case. For these breakouts, the first breakout (e.g., EndingYear = 2007, Gender = Female, Race = Asian) is considered the base. Then, the Odds Ratio statistics for the breakouts are given in reference to change from this base. As an example, the Odds Ratio for Gender reflects the Odds Ratio for movement from Female to Male, or relative to the binary outcome for FP12Months, the GenderMale Odds Ratio from Female to Male = 1.6703239480027 as seen in the OddsRatio column.

Month06	2.3585303060322
Month12	3.5455818144034

Guided Direction: Prepare a full set of details on interpretation and meaning of the Odds Ratio statistics, for numeric object variables and factor object variables. To start, a few interpretations may help, differentiating on how to interpret Odds Ratio output for numeric object variables and factor-type object variables:

- Numeric Object Variable: For a one unit change in Month03, the log odds of passing the Fail v Pass FP12Months test increases by 1.6394055157586.
- Factor Object Variable: Subjects who are Male, as opposed to subjects who are Female, change the log odds of attaining a passing score on the Fail v Pass FP12Months test by 1.6703239480027.

Study on the Binary Logistic Regression approach to problem-solving, knowing that many research designs in biostatistics are structured in a binary fashion. Then, consider the practical applications of Odds Ratio statistics. Note the difference in predictable outcomes for Females v Males. Is it practical, legal, ethical, etc., to offer specialized services to one group of subjects and to **not** provide specialized services to other subjects, based only on membership status within a grouping variable, even if there is a predictive justification for specialized services? Of course, the answer to that question is best left to others.

7.11 Prepare to Exit, Save, and Later Retrieve This R Session

R Input

```

getwd()           # Identify the current working directory.
ls()              # List all objects in the working
                  # directory.
ls.str()          # List all objects, with finite detail.
list.files()       # List files at the PC directory.

save.image("R_Lesson_CorrelationRegression.rdata")

getwd()           # Identify the current working directory.
ls()              # List all objects in the working
                  # directory.
ls.str()          # List all objects, with finite detail.
list.files()       # List files at the PC directory.

```

```
alarm()          # Alarm, notice of upcoming action.  
q()              # Quit this session.  
# Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: File -> Load Workspace. Otherwise, use the load() function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use the .R script file (typically saved as a .txt ASCII-type file) and recreate the analyses and graphics, provided the data files remain available.

7.12 External Data and/or Data Resources Used in This Lesson

`EmpWellAge21to79.csv`, `LivestockVigor.csv`, and `Rehabilitation.csv` were the only external files directly imported into this lesson and they are available at the publisher's Web site associated with this text. Use these files to practice and replicate the outcomes used in this lesson.

All other data in this lesson were enumerated (e.g., user-created) directly in the R session, using a variety of R-based functions. The user-created data in this lesson are **not** included at the publisher's Web site associated with this text and they are instead easily created in R by following the syntax presented in this lesson.



Chapter 8

Working with Large and Complex Datasets

Abstract

The purpose of this lesson on working with large and complex datasets is to provide a realistic demonstration of how R is used for challenging analyses, challenging because the dataset is fairly large, challenging because there are many variables requiring attention, challenging because there are missing data, challenging because certain subjects require special accommodation, challenging because selected data in the original dataset need to be put into filtered subsets, etc. Fortunately, R can accommodate these challenges, as demonstrated in this lesson and the accompanying addenda. Give special attention to the way different Boolean-type selection processes are used to address focused inquiries of importance for subjects in selected breakout groups, as opposed to large-scale analyses against the entire dataset.

Keywords: Analysis of variance (ANOVA), Association, Association plot, bagplot or bivariate boxplot, Bar plot or bar chart, Beanplot, Beeswarm plot, Big data, Boolean selection, Boxplot or box-and-whiskers plot, Box-percentile plot,

Electronic Supplementary Material The online version of this chapter (https://doi.org/10.1007/978-3-030-62404-0_8) contains supplementary material, which is available to authorized users.

Coefficient of correlation, Color gradient plot, Correlation, Correlogram, Density plot, Dotplot or dotchart, Engelmann–Hecker (EH) plot, Exploratory data analysis (EDA), Gantt chart, Graphical themes, Hexbin plot, Histogram, Interaction plot, International classification of diseases (ICD), Line chart or line graph, Long format data, Mosaic plot, Nonparametric, Normality, Parametric, Pearson's r, Pie chart, Pirate chart, Probability, Quantile-quantile (Q-Q) plot, Regression, Scatter plot or scatter diagram, Scatterplot matrix (SPLOM), Staircase plot, Stem-and-leaf plot, Stripchart, Sunflower scatterplot, Tidyverse, Trellis graphics, Triangular plot for 3-D representation, Violin plot, Waffle chart or squared pie chart, Wide format data

8.1 Background

This lesson continues a series of demonstrations on how the R environment supports statistical analyses, with a focus on complex analyses of enumerated breakout groups using Boolean approaches for subject selection. Using R as a platform for data organization, data analysis, and data presentation, this lesson reinforces different approaches to biostatistics against a large and fairly complex dataset. Going beyond R-based functions that were demonstrated in earlier lessons, this lesson introduces and then reinforces additional, more complex, R-based tools and processes. As much as R is stressed in this lesson, the important point to remember is the structured approach to analysis of an otherwise large dataset, with this lesson using data that have a biological orientation.

8.1.1 Description of the Data

The prior lessons in this text used either user-created datasets, .csv (comma-separated values) datasets, or spreadsheets that were generally simple in size and complexity at the beginning but grew in complexity as the lessons progressed:

- A small dataset such as `RabitWeightKg.csv` consisted of 40 data points and a header row, with 20 unique subjects (rows) and two object variables (columns).
- A much larger dataset, such as `Rehabilitation.csv`, consisted of 42,393 potential data points and a header row, with 3261 unique subjects (rows) and 13 object variables (columns).

In contrast, the dataset for this lesson (e.g., `Hospital.csv`) consists of 677,180 potential data points in original format and a header row: 33,859 unique cases (e.g., rows) and 20 original object variables (columns). Be sure to notice the term *in original format*. As `Hospital.csv` is used in this lesson, new (e.g., recoded) object variables are created which expands the dataset in final format to more than 1 million data points. Then, to add even more real-world com-

plexity, different Boolean (e.g., selection) strategies are used to focus only on specified segments (e.g., subgroups) of the dataset and the outcomes of these selection strategies are put into new datasets. Again, these approaches are purposely used in this lesson to provide exposure to the complexities of what many describe as *Big Data* and the way R is used in biostatistics to manage these large datasets and in turn extract useful (e.g., actionable) information.

The dataframe `Hospital.df`, which serves as the dataset in whole or in part for all analyses and graphical presentations in this lesson, should be viewed as a teaching dataset. The data in `Hospital.df` are reflective, in part, of the information typically associated with hospital records. The data in `Hospital.df` focus on only a few selected variables from an otherwise vast array of patient information maintained by medical staff and other interested parties. The data in `Hospital.df` are therefore useful for demonstration purposes in that they are representative of the data typically seen in biostatistics: factor-type data as text, factor-type data as numeric codes, nominal data, ordinal data, interval data, and numeric data. For this collection of data, assume that:

- The term hospital has a legal definition, as per state statutes. In this context, a hospital is a licensed medical facility that not only provides a wide range of medical services for diagnosis, treatment, general nursing care, etc., but a hospital also provides beds for use beyond 24 hours in support of patient care. In that context, a hospital should not be confused with other types of medical facilities, such as a(n): ambulatory surgical center, assisted living community, emergency care center, intensive residential treatment center, long-term care facility, mobile surgical facility, rehabilitation facility, urgent care center, walk-in health care clinic, etc.
- Data are from a group of more than 20 affiliated hospitals that have some degree of shared governance, subject to various levels of governmental oversight and licensing (e.g., county, state, and federal). However, by choice not all hospitals in the region are affiliated with the entity that eventually served as the data source for `Hospital.df`.
- The hospitals supplying data eventually found in the dataset `Hospital.df` are located in a large urban region that is organized for reporting purposes into three separate areas, based on geographic location: Central, North, and South. However, to maintain the highest degree of confidentiality, any identifying information that could ever be used to identify an individual patient is masked and totally unavailable to the principal researcher given access to the data as well as all others.
- The population of residents in these three regions is not evenly distributed and most patients, reflective to a degree of the general population, reside in the central part of the region.

- Selection and eventual admission to a specific hospital is based on a complex interplay of emergency needs, patient preference, location near patient residence, willingness to travel for treatment, allowance of clinical and access privileges to individual physicians at selected hospitals, in-network insurance authorization for specific hospitals and physicians, insurance authorization for specific types of expected procedures, etc. It should not be assumed that all patients receive services at the closest hospital, their first choice hospital, or their first choice physician.
- The many affiliated hospitals are able to share information on individual patients, not only for quality-of-care medical purposes but also for billing purposes.
- The more than 30,000 records in `Hospital.df` represent a random sample (selection of 1 of every X cases without replacement) of a much larger set of patient records for the calendar year associated with this teaching dataset.

As the dataset is organized, note how a Ticket number is used as the identifier for individual records (e.g., rows). This method for identification is used to maintain complete patient confidentiality of all medical records. A unique Ticket number is issued for each hospital intake among hospitals included in this dataset. The Ticket number is not an identifier that could be immediately traced back to a specific date during the calendar year, specific hospital, or an individual patient. Accordingly, each Ticket number represents a unique patient (e.g., subject) intake. Knowing that some individuals have multiple hospital admissions during a calendar year, any individual patient could have had two, three, or more hospital intakes and correspondingly have two, three, or more assigned Ticket numbers during a calendar year, with each hospital intake generating a unique Ticket number. Even with this level of attention to patient privacy, as a safeguard on patient confidentiality, only a few individuals, all with senior levels of permission, could ever gain access to the information needed to transcribe a Ticket number and eventually trace the coding scheme to an individual subject by name.

The dataset `Hospital.df`, in original format, is large, complex, and information-rich. The dataset could support a nearly endless number of statistical analyses. Accordingly, there are many potential statistical analyses that can be gained from the data. For demonstration purposes, this lesson will address two unique analyses, one analysis focusing on differences between breakout groups to a specific object variable and the other analysis focusing on the degree of association between two object variables. The two hypotheses should be sufficient to summarize how statistical analyses benefit from breakout investigations of outcomes and not only analyses at the broadest level of comparison.

8.1.2 First Null Hypothesis (Ho): Mean Comparisons by Breakout Groups

Ho: There is no statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure by the seven different racial groups (e.g., American Indian or Alaska Native, Asian, Black or African American, Native Hawaiian or Other Pacific Islander, Other, Unknown, White) among those subjects who range in age from 40 to 49, inclusive.

8.1.3 Second Null Hypothesis (Ho): Test of Association

Ho: There is no statistically significant correlation ($p \leq 0.05$) between Systolic Blood Pressure and Weight among Black or African American Female subjects who range in age from 40 to 49, inclusive.

8.2 Import Data in Comma-Separated Values (.csv) File Format and/or Self-Generate the Data Using R-Based Functions

As demonstrated in all prior lessons, the first task of any analysis involves organization of resources. The Housekeeping syntax (below) takes care of that beginning activity.

R Input

```
#####
# Housekeeping                                         Use for All Analyses #
#####
date()          # Current system time and date.
Sys.time()       # Current system time and date (redundant).
R.version.string # R version and version release date.
options(digits=6) # Confirm default digits.
options(scipen=999) # Suppress scientific notation.
options(width=60) # Confirm output width.
ls(all.names=TRUE) # List all objects in the working
                   # directory, including hidden files.
rm(list=ls())     # CAUTION: Remove all files in the
                   # working directory. If this
                   # action is not desired, use rm()
                   # one-by-one to remove the objects
                   # that are not needed.
ls.str()         # List all objects, with finite detail.
getwd()          # Identify the current working directory.
setwd("F:/R_BiostatisticsIntroduction")
```

```
# Set to a new working directory.  
# Note the single forward slash and double  
# quotes.  
# This new directory should be the directory  
# where the data file is located, otherwise  
# the data file will not be found.  
getwd()          # Confirm the working directory.  
list.files()     # List files at the PC directory.  
.libPaths()      # Library pathname.  
.Library         # Library pathname.  
sessionInfo()    # R version, locale, and packages.  
search()          # Attached packages and objects.  
searchpaths()     # Attached packages and objects.  
#####
```

The purpose of this collection of R functions in the Housekeeping template, with these many functions all in one convenient beginning section, is to be sure that the working environment, selected directory, etc., are as desired. Additionally, because the data are imported into the R session from an external source, the base::set.seed() function was not used at this point in the lesson.

Create an object called `Hospital.df`. The object `Hospital.df` will be a dataframe, as indicated by the enumerated `.df` extension to the object name. This object will represent the output of applying the `read.table()` function against the comma-separated values file called `Hospital.csv`. Note the arguments used with the `read.table()` function, showing that:

- The first row in the file `Hospital.csv` represents a header (`header=TRUE`), not data, with descriptive variable names. With attention to good documentation practices, the names for each variable should be rich and fully descriptive even if verbose. Shorter variable names can be used later, if needed.
- Decimals are indicated by using a period (`dec=".")` and not a comma or some other symbol. Although it is certainly common to expect the `.` character when placing data into decimal format, it is still best to declare the character used for decimals.
- Inherent to comma-separated values (.csv) file structure, the separator between fields is a comma (`sep=","`) and not a tab or white space. R can accommodate tabs and white space as separators between data, but that would not be the case for a .csv file.

R Input

```
Hospital.df <- utils::read.table (file = "Hospital.csv",
  header=TRUE, dec=". ", sep=",")
# Use the utils::read.table() function to import the
# .csv file Hospital.csv and place it into the object
# Hospital.df, which: (1) has a header row, (2) uses a
# period for decimals, and (3) uses a comma to separate
# one field from another.

getwd()                      # Identify the working directory
ls()                          # List objects
attach(Hospital.df)          # Attach the data, for later use
str(Hospital.df)             # Identify structure
nrow(Hospital.df)            # List the number of rows
ncol(Hospital.df)            # List the number of columns
dim(Hospital.df)             # Dimensions of the data frame
names(Hospital.df)           # Identify names
colnames(Hospital.df)         # Show column names
rownames(Hospital.df)         # Show row names
head(Hospital.df)            # Show the head
tail(Hospital.df)             # Show the tail
# Hospital.df                 # Show the entire dataframe
summary(Hospital.df)          # Summary statistics
```

The R comment character was placed in front of `Hospital.df`. With more than 30,000 cases (e.g., rows) the dataset is simply far too large for any meaningful review on the screen, line-by-line. The `#` character is used to temporarily *commented-out*, and not put into effect, this action. By seeing this line of syntax with the `#` comment character there is a reminder of the possibility of placing the action into effect at a later time.

8.3 Organize the Data and Display the Code Book

The Code Book for `Hospital.df`, shown below, provides detailed information on the object variables selected for this lesson. This dataset represents only the smallest part of the many types of data associated with health care, but it still serves as a useful introduction to working with a large and complex dataset specific to biostatistics.

R Input

```
#####
# Code Book for Hospital.df          #
#####
#                                     #
# Ticket                         #
# Factor (e.g., nominal)          #
#                                     #
# A Ticket number is generated for each unique      #
# patient admission. The coding scheme is organized   #
# so that a Ticket number can only be traced back to   #
# an individual patient (e.g., subject) by those few    #
# individuals with senior levels of administrative     #
# privilege.                                         #
#                                     #
# Ticket numbers for the selected year in this       #
# dataset range from SKY12000000 to SKY12100000.      #
#-----#
#                                     #
# Location and LocationCode          #
# Factor (e.g., nominal)          #
#                                     #
# There are three defined locations for the many      #
# hospitals included in this dataset, coded in        #
# alphabetical sequence:                  #
#                                     #
# Central ..... 1                      #
# North ..... 2                        #
# South ..... 3                        #
#-----#
#                                     #
# PrimaryPayer and PrimaryPayerCode      #
# Factor (e.g., nominal)          #
#                                     #
# The Primary Payer is the entity or individual who   #
# paid 51 percent or more of final hospital charges,   #
# physician charges, and all other associated costs   #
# of each hospital admission:           #
#                                     #
# There are four defined Primary Payers as the data   #
# coding scheme is organized in this dataset, coded     #
# in alphabetical sequence:           #
#                                     #
```

```
# Commercial      1          #
# Government     2          #
# Other          3          #
# Self           4          #
#
# Commercial refers to for-profit insurance companies #
# that provide varying degrees of health-care          #
# coverage. In contrast, Government represents total   #
# payout by the many different local, state, and        #
# federal agencies that fund health-care for          #
# different constituent groups. Other refers to        #
# those very few groups that are neither commercial    #
# nor affiliated with a government agency. Self         #
# refers to those few individuals who pay for          #
# services on their own, out-of-pocket.                 #
#-----#
#
# PrincipalDiagnosis3DigitICD9 and                   #
# PrincipalDiagnosisRangeICD9                         #
# Factor (e.g., nominal)                            #
#
# ICD9 (International Classification of Diseases, 9th #
# Revision; ICD-9) three-digit codes were used as the #
# primary diagnosis coding scheme for each hospital   #
# admission:                                         #
#
# ICD9 001-139  Infectious and Parasitic Diseases   #
# ICD9 140-239  Neoplasms                          #
# ICD9 240-279  Endocrine, Nutritional and          #
#                  Metabolic Diseases, and Immunity    #
#                  Disorders                           #
# ICD9 280-289  Diseases of the Blood and Blood-  #
#                  Forming Organs                     #
# ICD9 290-319  Mental Disorders                   #
# ICD9 320-359  Diseases of the Nervous System    #
# ICD9 360-389  Diseases of the Sense Organs       #
# ICD9 390-459  Diseases of the Circulatory System #
# ICD9 460-519  Diseases of the Respiratory System #
# ICD9 520-579  Diseases of the Digestive System   #
# ICD9 580-629  Diseases of the Genitourinary      #
#                  System                           #
# ICD9 630-679  Complications of Pregnancy,        #
#                  Childbirth, and the Puerperium     #
```

```

# ICD9 680-709 Diseases of the Skin and #
# Subcutaneous Tissue #
# ICD9 710-739 Diseases of the Musculoskeletal #
# System and Connective Tissue #
# ICD9 740-759 Congenital Anomalies #
# ICD9 760-779 Conditions Originating in the #
# Perinatal Period #
# ICD9 780-799 Symptoms, Signs, and Ill-Defined #
# Conditions #
# ICD9 800-999 Injury and Poisoning #
# ICD9 E800-E999 Supplemental Classification of #
# External Causes of Injury and #
# Poisoning #
# ICD9 V01-V89 Supplemental Classification of #
# Factors Influencing Health Status #
# and Contact with Health Services #
#
# For those with special interest, there are many #
# online resources that highlight the history of the #
# ICD process, transition from ICD-9 to ICD-10, and #
# ICD-10 to ICD-11.
#
# For this teaching dataset and more specifically for #
# the use of ICD-9 codes instead of later iterations #
# of International Classification of Diseases coding #
# schemes, is important to recall that the data were #
# obtained soon before the Affordable Care Act #
# (Public Law 111-148, March 23, 2010) was put into #
# place. Along with ICD-9 coding practices, the data #
# reflect prior accommodations for financial #
# reimbursement and eventual out-of-pocket payment #
# for health care services.
#-----
#
# AgeLastBirthday #
# Numeric (e.g., ordinal or interval) #
#
# In an effort to further maintain confidentiality of #
# all patients, age is represented by patient age on #
# their last birthday. Ideally, it would be better #
# to have patient date of birth, to calculate the far #
# more granular number of days to a point in time as #
# a measure for age, instead of the far less precise #

```

```
# presentation of age in years as a whole number.      #
# With AgeLastBirthday, a patient who is 61 years      #
# and 1 day is marked the same as someone who is 61      #
# years and 364 days.                                #
#
# Infants from birth to 364 days are marked as 0 for   #
# AgeLastBirthday.                                    #
#-----#
#
# AgeRange                                         #
# Factor (e.g., ordinal)                         #
#
# There are conditions when it is best to present   #
# age-related outcomes within pre-defined ranges.    #
# The age (as years) ranges used in Hospital.df are: #
#
#      0 (Birth) to 9 ... 0   60 to 69 ..... 6      #
#      10 to 20 ..... 1   70 to 79 ..... 7      #
#      21 to 29 ..... 2   80 to 89 ..... 8      #
#      30 to 39 ..... 3   90 to 99 ..... 9      #
#      40 to 49 ..... 4   100 and older ... 10     #
#      50 to 59 ..... 5                               #
#
# Note how 10 to 20 and not 10 to 19 is coded as 1.    #
# The inclusion of 20 in this age range is needed to  #
# accommodate age of majority (18 years in some cases #
# and 21 years in other cases) issues that may be a   #
# concern when using patient data.                  #
#-----#
#
# Race and RaceCode                                #
# Factor (e.g., nominal)                          #
#
# Using standards inspired by the U.S. Office of    #
# Management and Budget and the U.S. Census Bureau,  #
# race and ethnicity are self-identified and        #
# represent different constructs. For this dataset,  #
# note the following identifications for Race and   #
# RaceCode, listed in alphabetical order:           #
#
#      American Indian or Alaska Native ..... 1      #
#      Asian ..... 2                                #
#      Black or African American ..... 3            #
```

```
# Native Hawaiian or Other Pacific Islander .. 4      #
# Other ..... 5      #
# Unknown ..... 6      #
# White ..... 7      #
#-----#
#
# Ethnicity and EthnicityCode      #
# Factor (e.g., nominal)      #
#
# Using standards inspired by the U.S. Office of      #
# Management and Budget and the U.S. Census Bureau,      #
# race and ethnicity are self-identified and      #
# represent different constructs. For this dataset,      #
# note the following identifications for Ethnicity      #
# and EthnicityCode, listed in alphabetical order:      #
#
# Hispanic or Latino ..... 1      #
# Not Hispanic or Latino ... 2      #
# Unknown ..... 3      #
#-----#
#
# Sex and SexCode      #
# Factor (e.g., nominal)      #
#
# Using standards inspired by the World Health      #
# Organization, sex and gender represent different      #
# constructs. The term sex refers to biological      #
# characteristics whereas the term gender refers to      #
# societal attributes and values. This dataset uses      #
# female and male, only, to identify the sex of any      #
# specific individual patient, listed in alphabetical      #
# order:      #
#
# Female ..... 1      #
# Male ..... 2      #
#-----#
#
# OverNightStay      #
# Numeric (e.g., interval)      #
#
# Hospital patients who are admitted and discharged      #
# the same day are marked as zero (0) for the      #
# variable OverNightStay. Otherwise, the number of      #
```

```
# nights required for patient services is an interval #
# datum which is easily counted and to some degree    #
# serves as a proxy for level of care and severity of #
# required services. Recall, however, that in many    #
# cases, patients who need an extended period for    #
# recovery and cannot receive those services at home #
# are often discharged from a hospital and are then    #
# accommodated at rehabilitation-type facilities that #
# meet long-term care needs in a more cost-efficient #
# manner than what would be experienced with more    #
# expensive hospital care.                            #
#-----#
#
# OverNightStayCode                                #
# Numeric (e.g., ordinal)                         #
#
# Approximately 95 percent of all hospital patients #
# included in this dataset had an overnight stay of #
# 14 days or fewer, with most (50 percent or more)   #
# needing only one to three days in the hospital for #
# each admission. Saying that and given the lack of #
# normal distribution for OverNightStay, an ordinal   #
# (e.g., ordered) approach to time in hospital has   #
# been coded as:                                    #
#
#    0 Nights ..... 0    8 Nights ..... 8  #
#    1 Night ..... 1    9 Nights ..... 9  #
#    2 Nights ..... 2   10 Nights ..... 10 #
#    3 Nights ..... 3   11 Nights ..... 11 #
#    4 Nights ..... 4   12 Nights ..... 12 #
#    5 Nights ..... 5   13 Nights ..... 13 #
#    6 Nights ..... 6   14 Nights ..... 14 #
#    7 Nights ..... 7   15 or more Nights .. 15 #
#-----#
#
# SBP and DBP (Systolic and Diastolic Blood Pressure) #
# Numeric (e.g., interval)                           #
#
# SBP and DBP are among the most common measurements #
# obtained by medical staff and for this dataset they #
# reflect the first attempt to obtain these values,   #
# at or soon after patient admission. Both SBP and   #
# DBP measurements are expressed as millimeters of   #
```

```

# mercury (e.g., mmHg or mm Hg). In most cases, a      #
# digital sphygmomanometer was used to obtain SBP and ##
# DBP measurements, with measurements obtained only   #
# by trained health care professionals. For this      #
# dataset, assume that SBP and DBP measurements are    #
# both valid and reliable and that they represent the  #
# first attempt used to obtain these measures, often   #
# at a time when patients are ill and/or anxious.     #
#-----#
#
# Lbs (Pounds, Weight; from the Latin libra)          #
# Numeric (e.g., interval)                            #
#
# Following along with U.S. customary units, weight    #
# was measured using pounds (e.g., Lbs) and not the    #
# metric unit kilogram (e.g., Kg). For those who      #
# prefer metric measures, use the following algorithm #
# to convert Lbs to Kg, with 1 Lb = 0.45359237 Kg:   #
#
# Number of Pounds * 0.45359237 = Number of Kilograms#
#
# Note how in Hospital.df, there are no weights        #
# listed for patients ranging in age from birth to 20  #
# years. A variety of policy and systems-related      #
# reasons are the basis for these purposely missing   #
# data. However, weights are available for patients   #
# 21 years and older.                                 #
#
# Similar to Systolic Blood Pressure (SBP) and       #
# Diastolic Blood Pressure (DBP), assume that Weight   #
# (Lbs) is both valid and reliable and that this       #
# datum represents the first attempt used to obtain   #
# these measures, at or soon after admission.         #
#-----#
#####

```

The dataframe `Hospital.df` is fairly simple to understand although it is somewhat large with nearly 35,000 rows of data (e.g., cases, or Tickets for this specific health-care dataset). Each variable with coded values is presented as both English-like text and a numeric code equivalent, to make comprehension, selection, and use easier than if variable structure depended only on interpretation of numerical codes listed in the Code Book. Of course, this type of data organization, although convenient, is not typical but it was purposely done for the teaching value of this dataset.

For this lesson, the `class()` function, `str()` function, and `table(duplicated())` functions will be sufficient first steps to provide some degree of assurance that data are organized as desired.

R Input

```
class(Hospital.df) # Object-type
```

R Output

```
[1] "data.frame"
```

R Input

```
str(Hospital.df) # Object-structure
```

R Output

```
'data.frame': 33859 obs. of 20 variables:  
 $ Ticket           : Factor w/ 33859 levels "SKY120  
 $ Location         : Factor w/ 3 levels "Central", "  
 $ LocationCode     : int 1 1 2 2 1 3 3 1 1 1 ...  
 $ PrimaryPayer      : Factor w/ 4 levels "Commercial  
 $ PrimaryPayerCode   : int 2 2 2 1 2 2 2 2 2 2 ...  
 $ PrincipalDiagnosis3DigitICD9: Factor w/ 662 levels "11", "111  
 $ PrincipalDiagnosisRangeICD9: Factor w/ 19 levels "ICD900113  
 $ AgeLastBirthday    : int 74 75 24 26 21 67 72 22 7  
 $ AgeRangeCode       : int 7 7 2 2 2 6 7 2 7 7 ...  
 $ Race              : Factor w/ 7 levels "American I  
 $ RaceCode          : int 3 7 7 3 7 7 7 7 7 3 ...  
 $ Ethnicity         : Factor w/ 3 levels "Hispanic o  
 $ EthnicityCode      : int 2 2 2 2 2 2 2 2 1 2 ...  
 $ Sex               : Factor w/ 2 levels "Female", "M  
 $ SexCode           : int 2 1 2 1 1 2 2 1 2 1 ...  
 $ OverNightStay      : int 1 13 3 4 3 1 9 2 1 2 ...  
 $ OverNightStayCode    : int 1 13 3 4 3 1 9 2 1 2 ...  
 $ SBP               : int 172 110 118 170 112 106 1  
 $ DBP               : int 104 74 60 94 56 72 76 66  
 $ Lbs               : int 156 164 164 180 132 162 1
```

R Input

```
table(duplicated(Hospital.df$Ticket)) # Summary of duplicates
```

R Output

```
FALSE  
33859
```

These functions provide evidence that the object Hospital.df seems to be initially correct:

- Hospital.df is a dataframe, as demonstrated by using the class() function.
- The structure for each variable is initially identified by using the str() function.
- There is evidence that there are no duplicates at the individual level, with this assurance gained by using the duplicated() function, with the table() function wrapped around the duplicated() function to provide a simple read-out of the output.

Recall that the Code Book shows data in their desired formats, which often requires some degree of recoding which has not yet occurred.

Once there is agreement that the data were brought into R in correct original format, it is usually necessary to organize the data to some degree, to be sure that all data are in desired format, as identified in the Code Book:

- The data for object variables LocationCode, PrimaryPayerCode, AgeRangeCode, RaceCode, EthnicityCode, SexCode, and OverNightStayCode are currently viewed as integers, but they should be treated as if they were group-focused factor-type objects. A set of simple R-based actions can easily: (1) transform (e.g., recode) the integer-based object variables into new object variables, (2) change the recoded object variable from original integer format to enumerated factor format, and (3) apply English text labels for the otherwise cryptic numeric codes (e.g., 1, 2, etc.).
- In contrast, the groups for the factor object variables Ticket, Location, PrincipalDiagnosis3DigitICD9, PrincipalDiagnosisRangeICD9, Race, Ethnicity, and Sex are presented in the original file as standard English text, using CamelCase (e.g., camelCase, camel caps, Pascal case, Dromedary case, medial capitals) when needed to address compound words.
- Values for SBP (e.g., Systolic Blood Pressure), DBP (e.g., Diastolic Blood

Pressure), and Lbs (e.g., Weight) are currently whole numbers and as such they are first treated in R as integers. A simple recode action will instead be used to put these values into numeric format, which may be desirable if any math-type actions are used with these three object variables.¹

This transformation (typically called a recode action) of selected object variables is needed and the process, using R-based syntax, follows. There may be some unnecessary (perhaps redundant) actions with the following recode activities, but these are purposely demonstrated in this lesson to provide assurance that each variable is in desired format, for both original variables as well as the newly-created (e.g., enumerated, transformed) variables:

R Input

```
str(Hospital.df)      # structure before recoding  
summary(Hospital.df) # summary before recoding
```

R Input

```
Hospital.df$Ticket <- as.factor(Hospital.df$Ticket)  
  
Hospital.df$Location <- as.factor(Hospital.df$Location)  
Hospital.df$LocationCode <- as.factor(Hospital.df$LocationCode)  
Hospital.df$Location.Recode <- factor(Hospital.df$LocationCode,  
  labels=c("Central", "North", "South"))  
# factor(), not as.factor()  
  
Hospital.df$PrimaryPayer <- as.factor(Hospital.df$PrimaryPayer)  
Hospital.df$PrimaryPayerCode <-  
  as.factor(Hospital.df$PrimaryPayerCode)  
Hospital.df$PrimaryPayer.Recode <-  
  factor(Hospital.df$PrimaryPayerCode,  
  labels=c("Commercial", "Government", "Other", "Self"))  
  
Hospital.df$PrincipalDiagnosis3DigitICD9 <-  
  as.factor(Hospital.df$PrincipalDiagnosis3DigitICD9)  
  
Hospital.df$PrincipalDiagnosisRangeICD9 <-  
  as.factor(Hospital.df$PrincipalDiagnosisRangeICD9)
```

¹Although it was mentioned in the Code Book, once again recall that the data gained for Systolic Blood Pressure and Diastolic Blood Pressure are obtained at the time of or soon after admission, a time when patients are likely ill and/or anxious.

```

Hospital.df$AgeLastBirthday <-
  as.numeric(Hospital.df$AgeLastBirthday)

Hospital.df$AgeRangeCode      <-
  as.factor(Hospital.df$AgeRangeCode)
Hospital.df$AgeRange.Recode <-
  factor(Hospital.df$AgeRangeCode,
  labels=c("Birth-09", "10-20", "21-29", "30-39", "40-49",
          "50-59", "60-69", "70-79", "80-89", "90-99",
          "100-More"))

Hospital.df$Race           <- as.factor(Hospital.df$Race)
Hospital.df$RaceCode       <- as.factor(Hospital.df$RaceCode)
Hospital.df$Race.Recode <- factor(Hospital.df$RaceCode,
  labels=c("AmIndian", "Asian", "Black", "Hawaiian",
          "Other", "Unknown", "White"))

Hospital.df$Ethnicity <- as.factor(Hospital.df$Ethnicity)
Hospital.df$EthnicityCode <-
  as.factor(Hospital.df$EthnicityCode)
Hospital.df$Ethnicity.Recode <-
  factor(Hospital.df$EthnicityCode,
  labels=c("Hispanic", "Non-Hispanic", "Unknown"))

Hospital.df$Sex <- as.factor(Hospital.df$Sex)
Hospital.df$SexCode <- as.factor(Hospital.df$SexCode)
Hospital.df$Sex.Recode <- factor(Hospital.df$SexCode,
  labels=c("Female", "Male"))

Hospital.df$OverNightStay <-
  as.numeric(Hospital.df$OverNightStay)
Hospital.df$OverNightStayCode <-
  as.factor(Hospital.df$OverNightStayCode)
Hospital.df$OverNightStay.Recode <-
  factor(Hospital.df$OverNightStayCode,
  labels=c("0", "1", "2", "3", "4", "5", "6",
          "7", "8", "9", "10", "11", "12", "13",
          "14", "15-More"))

Hospital.df$SBP <- as.numeric(Hospital.df$SBP)

Hospital.df$DBP <- as.numeric(Hospital.df$DBP)

```

```
Hospital.df$Lbs <- as.numeric(Hospital.df$Lbs)
```

Now that the recoding has been completed, including the recoding that is redundant but was done to emphasize the desire to have all variables in correct format, it is best to attach the data (which may also be redundant, but still useful) and to then use a series of functions to be sure once again that the data are correct.

R Input

```
attach(Hospital.df)          # Attach the data, for later use
getwd()                      # Identify the working directory
ls()                          # List objects
nrow(Hospital.df)           # List the number of rows
ncol(Hospital.df)           # List the number of columns
dim(Hospital.df)            # Dimensions of the data frame
names(Hospital.df)          # Identify names
colnames(Hospital.df)       # Show column names
rownames(Hospital.df)       # Show row names
head(Hospital.df)           # Show the head
tail(Hospital.df)           # Show the tail
summary(Hospital.df)        # Summary after recoding
str(Hospital.df)            # Structure after recoding
```

R Output

```
'data.frame': 33859 obs. of 27 variables:
 $ Ticket                  : Factor w/ 33859 levels "SKY120"
 $ Location                 : Factor w/ 3 levels "Central",""
 $ LocationCode              : Factor w/ 3 levels "1","2","3"
 $ PrimaryPayer              : Factor w/ 4 levels "Commercial"
 $ PrimaryPayerCode          : Factor w/ 4 levels "1","2","3"
 $ PrincipalDiagnosis3DigitICD9: Factor w/ 662 levels "11","111"
 $ PrincipalDiagnosisRangeICD9: Factor w/ 19 levels "ICD900113"
 $ AgeLastBirthday           : num 74 75 24 26 21 67 72 22 7
 $ AgeRangeCode               : Factor w/ 11 levels "0","1","2"
 $ Race                      : Factor w/ 7 levels "American I"
 $ RaceCode                  : Factor w/ 7 levels "1","2","3"
 $ Ethnicity                 : Factor w/ 3 levels "Hispanic o"
 $ EthnicityCode              : Factor w/ 3 levels "1","2","3"
 $ Sex                       : Factor w/ 2 levels "Female","M"
 $ SexCode                   : Factor w/ 2 levels "1","2": 2
 $ OverNightStay              : num 1 13 3 4 3 1 9 2 1 2 ...
 $ OverNightStayCode          : Factor w/ 16 levels "0","1","2"
```

```
$ SBP : num 172 110 118 170 112 106 1
$ DBP : num 104 74 60 94 56 72 76 66
$ Lbs : num 156 164 164 180 132 162 1
$ Location.Recode : Factor w/ 3 levels "Central",""
$ PrimaryPayer.Recode : Factor w/ 4 levels "Commercial"
$ AgeRange.Recode : Factor w/ 11 levels "Birth-09"
$ Race.Recode : Factor w/ 7 levels "AmIndian",
$ Ethnicity.Recode : Factor w/ 3 levels "Hispanic",
$ Sex.Recode : Factor w/ 2 levels "Female","M"
$ OverNightStay.Recode : Factor w/ 16 levels "0","1","2"
```

The recoded object variables, consistently named as `DataFrame$Object.Recode` or a similar naming scheme, have been attached to the dataframe `Hospital.df`. For this specific dataset it may have been unnecessary to create all of these `DataFrame$Object.Recode` variables, given how most factor-type variables in `Hospital.df` (and not only the Code Book) show as English-like text and then a corresponding numeric code. There will be many (typically—most) datasets in the future where that is not the case and only numeric codes are provided, with the accompanying Code Book needed to make sense of the many numeric codes. `Hospital.df` is a teaching dataset and some naming schemes and numbering schemes have been used to simplify use of this dataset.

In an attempt to provide an initial Quality Assurance review of the dataset and additional assurance that the data are attached properly and in good form, give attention to the policy decision to exclude from the dataset the object variable `Lbs` for all patients ranging in age from birth to 20 years. Use this decision to exclude weight for selected patients as a way of reviewing data quality. The `is.na()` function and `complete.cases()` function will help in this effort. Wrap the `table()` function around these two functions to put output into an easy to read format.

R Input

```
table(is.na(Hospital.df))
# Find the number of missing data (e.g., NA)
# for all patients.
```

R Output

FALSE	TRUE
910588	3605

R Input

```
table(complete.cases(Hospital.df))
# Find the number of complete cases of all
# patients.
```

R Output

```
FALSE  TRUE
3605 30254
```

R Input

```
table(is.na(subset(Hospital.df,
  Hospital.df$AgeLastBirthday >= 21)))
# Find the number of missing data (e.g., NA)
# for patients >= 21 years old. Recall the
# policy decision to exclude weight (Lbs) for
# all patients from birth to 20 years of age.
```

R Output

```
FALSE
816858
```

R Input

```
table(complete.cases(subset(Hospital.df,
  Hospital.df$AgeLastBirthday >= 21)))
# Find the number of complete cases for
# patients >= 21 years old. Recall the
# policy decision to exclude weight (Lbs) for
# all patients from birth to 20 years of age.
```

R Output

```
TRUE
30254
```

The data seem to be acceptable in terms of presentation and completeness. Many more actions are needed before the two Null Hypotheses can be addressed, but it seems to be appropriate to continue with planned actions.

8.4 Conduct a Visual Data Check Using Graphics (e.g., Figures)

It is only too common for inexperienced researchers to rush into data analysis and immediately attempt numeric descriptive statistics and measures of central tendency. However, it is far better to first generate graphics, to actually see how data are organized. Graphics provide an essential complement to our understanding of the data and graphics also provide a useful check for quality assurance.² For initial purposes the graphical functions of primary interest are figures that demonstrate frequency distribution of factor-type data and figures that demonstrate descriptive statistics and measures of central tendency. Later, figures are presented that demonstrate relationships (e.g., differences, similarities, associations) between and among the data. Among the many R-based tools available for use when producing pre-analysis graphics that provide guidance on the nature of the data, consider how:

- There are packages that purposely support Exploratory Data Analysis (EDA) for examination of the data. These packages include functions that easily generate multiple graphics which offer a sense of general trends of both factor-type object variables and numeric object variables. The initial graphics, which are often quite basic, can be embellished to produce graphics that are more attractive and promote attention by outside readers. The DataExplorer package and the funModeling package each contain functions that deserve attention for convenient graphically-focused EDA.
- The initial functions typically used with factor-type data are `graphics::plot()` and `graphics::barplot()`.³
- The initial functions typically used with numeric-type data are `graphics::hist()`, `graphics::boxplot()`, `stats::density()`, and `stats::qqnorm()` (Figs. 8.1–8.13).

²As demonstrated throughout this series of lessons, the terms QA, Quality Assurance, and quality assurance are used interchangeably, similar to how these terms are seen in the literature.

³Although somewhat verbose, in this lesson R-based functions are often presented in long-format, such as `graphics::plot(Hospital.df$Ticket)` and not `plot(Hospital.df$Ticket)`. To reinforce what has been presented throughout this text, in this summary lesson the package name is often placed before the function name to emphasize that the function is from a collection of functions found in a package and to then identify the package name. Although this is an excellent practice in that it avoids any possible confusion about the origin of the selected function, it is common to see this practice used only for functions found in external packages and not for functions found in packages gained from when R is first downloaded (e.g., core R), such as the packages base, graphics, stats, utils, etc. Even so, this long-format naming practice is used throughout this lesson and is only avoided sparingly.

Functions Found in Exploratory Data Analysis (EDA) Packages

R Input

```
install.packages("DataExplorer", dependencies=TRUE)
library(DataExplorer)          # Load the DataExplorer package.
help(package=DataExplorer)    # Show the information page.
sessionInfo()                 # Confirm all attached packages.

par(ask=TRUE); DataExplorer::plot_intro(Hospital.df,
                                         title="Exploratory Data Analysis (EDA) of Hospital.df:
                                         Rows, Columns, and Data (e.g., Observations)")
                                         # Generate a single graphic that provides a complete sense
                                         # of the dataset: (1) discrete columns (factor-type data),
                                         # (2) continuous columns (numeric data), (3) missing
                                         # columns, (4) complete rows, and (5) missing observations.
```

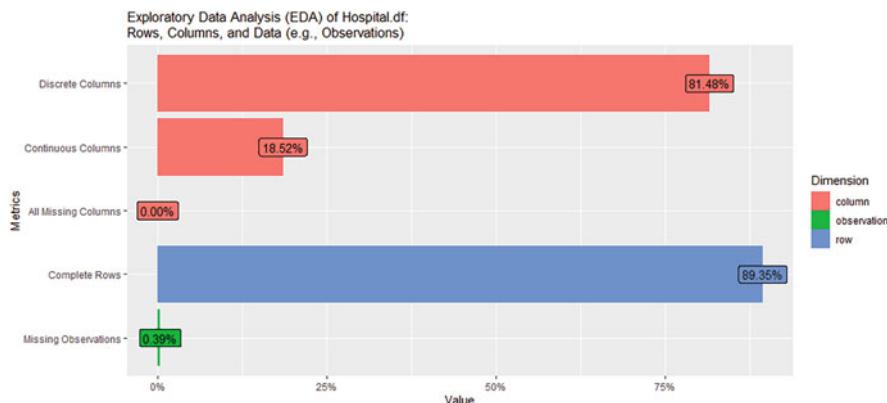


Figure 8.1: Quality assurance exploratory data analysis—1

R Input

```
par(ask=TRUE); DataExplorer::plot_missing(Hospital.df,
                                         title="Exploratory Data Analysis (EDA) of Missing Data
                                         for All Variables in Hospital.df",
                                         missing_only=FALSE)
                                         # Generate a single graphic that provides a complete sense
                                         # of the columns containing missing data.
```

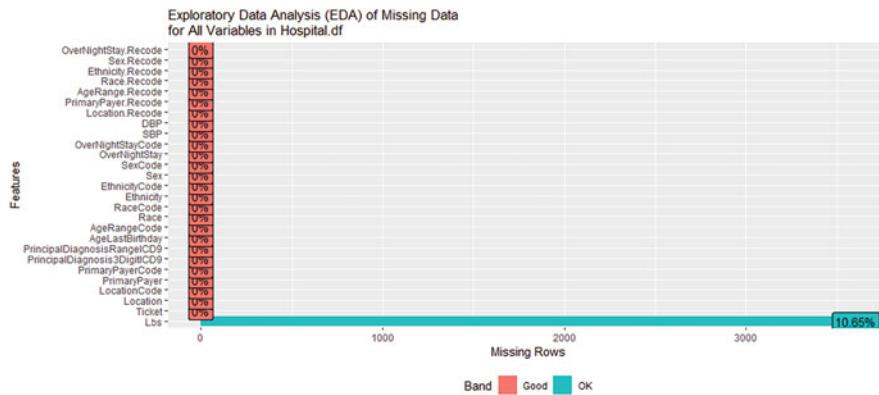


Figure 8.2: Quality assurance exploratory data analysis—2

R Input

```
par(ask=TRUE); DataExplorer::plot_bar(Hospital.df [, 21:27],  
  title="Exploratory Data Analysis (EDA) of Selected  
  Factor-Type Object Variables")  
# Generate a single graphic that provides barplots of the  
# selected recoded factor-type object variables, which are  
# placed in columns 21 to 27.
```

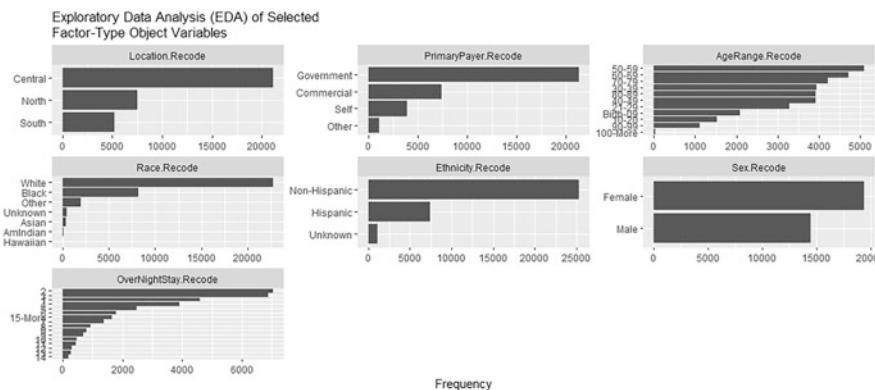


Figure 8.3: Quality assurance exploratory data analysis—3

R Input

```
par(ask=TRUE); DataExplorer::plot_density(Hospital.df  
  [, 18:20],  
  title="Exploratory Data Analysis (EDA) of Selected  
  Numeric Object Variables")  
# Generate a single graphic that provides density plots of  
# selected numeric object variables SBP, DBP, and Lbs,
```

```
# which are placed in columns 18 to 20.
```

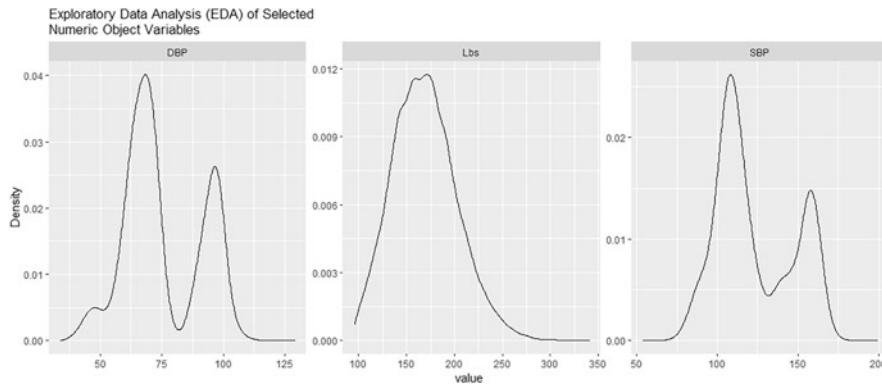


Figure 8.4: Quality assurance exploratory data analysis—4

R Input

```
par(ask=TRUE); DataExplorer::plot_qq(Hospital.df [, 18:20],  
  title="Normal Distribution Pattern of Selected  
  Numeric Object Variables")  
# Generate a single graphic that provides QQ plots of  
# selected numeric object variables SBP, DBP, and Lbs,  
# which are placed in columns 18 to 20.
```

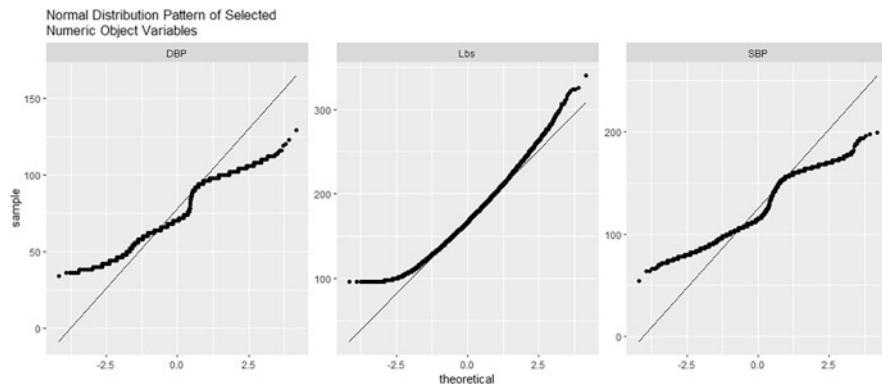


Figure 8.5: Quality assurance exploratory data analysis—5

R Input

```
par(ask=TRUE); DataExplorer::plot_correlation(Hospital.df  
  [, 18:20], title="Correlation Heat Map of Selected  
  Numeric Object Variables")
```

```
# Generate a single graphic that provides a correlation heat
# map of selected numeric object variables SBP, DBP, and Lbs,
# which are placed in columns 18 to 20.
```

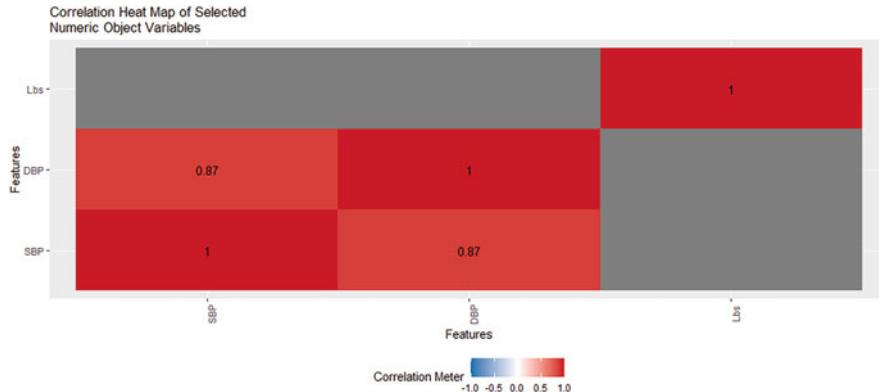


Figure 8.6: Quality assurance exploratory data analysis—6

R Input

```
install.packages("funModeling", dependencies=TRUE)
library(funModeling)           # Load the funModeling package.
help(package=funModeling)      # Show the information page.
sessionInfo()                  # Confirm all attached packages.

par(ask=TRUE); funModeling::freq(Hospital.df [, 21],
plot=TRUE, na.rm=TRUE)
# Generate multiple graphics that provide barplots of the
# selected recoded factor-type object variables, which are
# placed in columns 21 to 27. Observe how (1) the bars are
# in declining frequency order, not alphabetical order, and
# (2) output includes frequency distributions printed to
# the screen.
```

R Input

```
par(ask=TRUE); funModeling::freq(Hospital.df [, 22],
plot=TRUE, na.rm=TRUE)
```

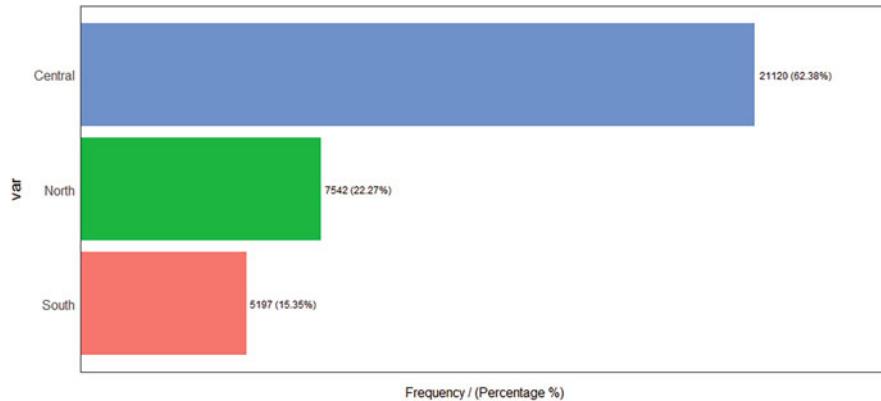


Figure 8.7: Quality assurance exploratory data analysis—7

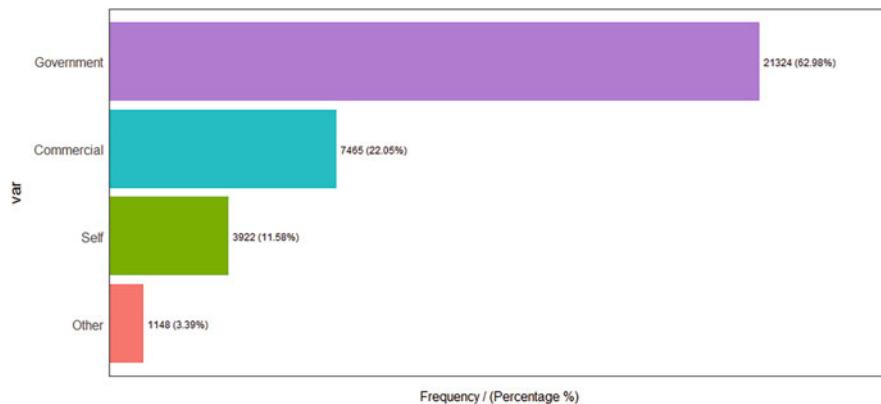


Figure 8.8: Quality assurance exploratory data analysis—8

R Input

```
par(ask=TRUE); funModeling::freq(Hospital.df [, 23],  
plot=TRUE, na.rm=TRUE)
```

R Input

```
par(ask=TRUE); funModeling::freq(Hospital.df [, 24],  
plot=TRUE, na.rm=TRUE)
```

R Input

```
par(ask=TRUE); funModeling::freq(Hospital.df [, 25],  
plot=TRUE, na.rm=TRUE)
```

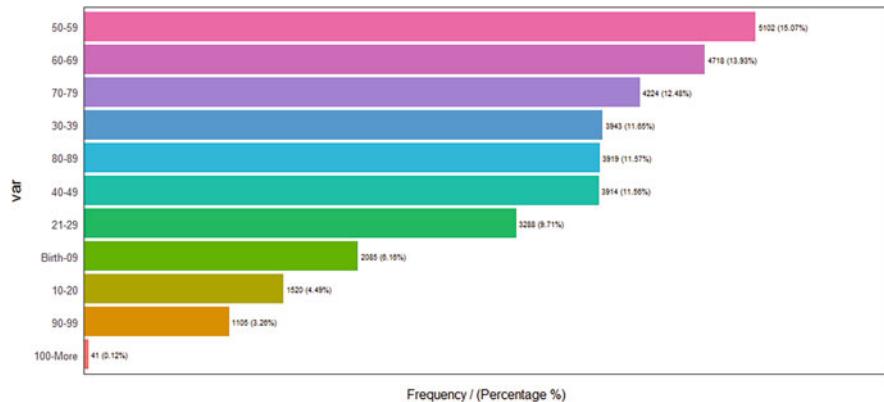


Figure 8.9: Quality assurance exploratory data analysis—9

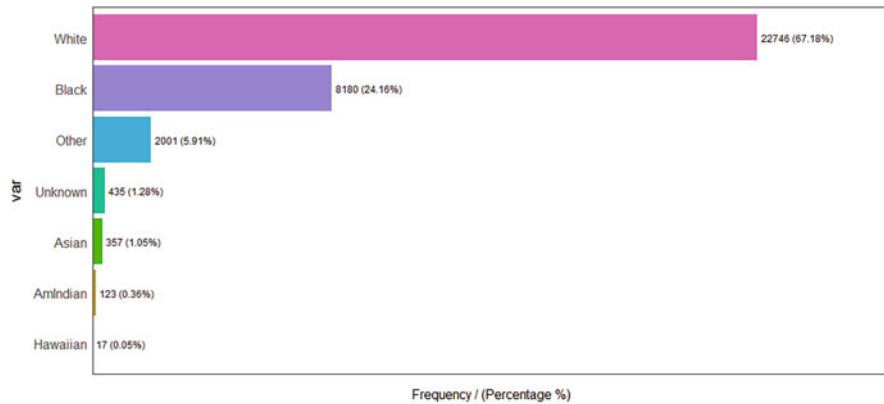


Figure 8.10: Quality assurance exploratory data analysis—10

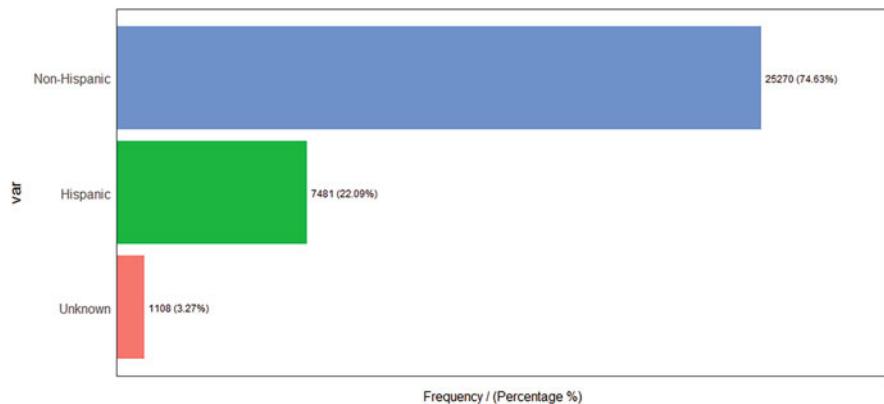


Figure 8.11: Quality assurance exploratory data analysis—11

R Input

```
par(ask=TRUE); funModeling::freq(Hospital.df [, 26],
plot=TRUE, na.rm=TRUE)
```

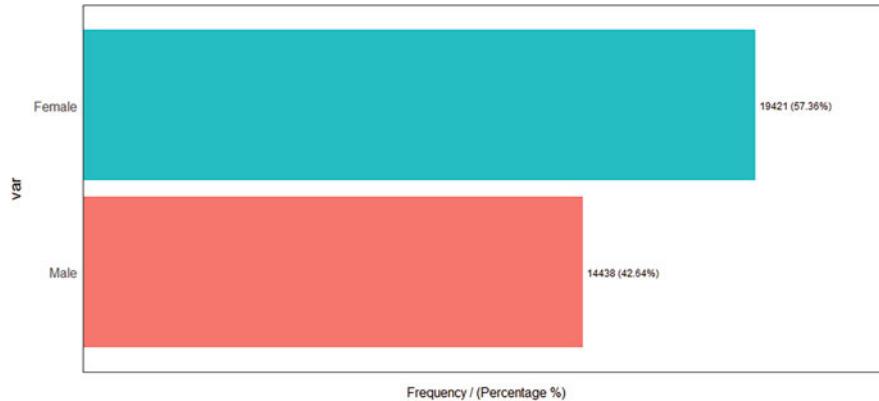


Figure 8.12: Quality assurance exploratory data analysis—12

R Input

```
par(ask=TRUE); funModeling::freq(Hospital.df [, 27],
plot=TRUE, na.rm=TRUE)
```

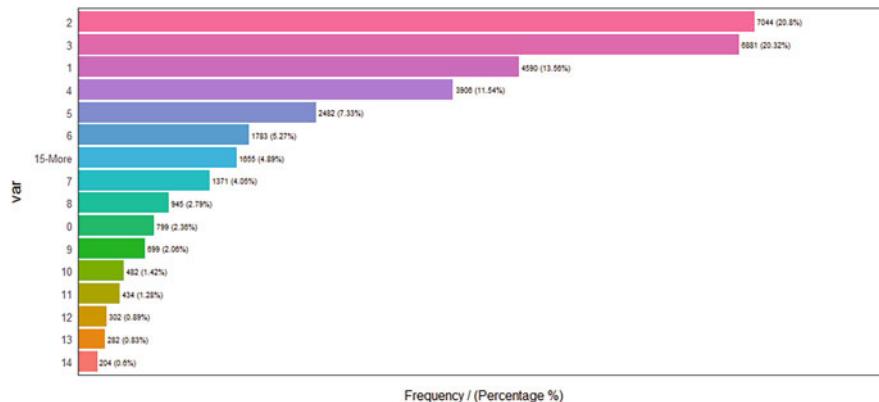


Figure 8.13: Quality assurance exploratory data analysis—13

R Input

```
par(ask=TRUE); funModeling::plot_num(Hospital.df [, 18:20],
bins=50)
# Generate a single graphic that provides histograms of
```

```
# selected numeric object variables SBP, DBP, and Lbs,
# which are placed in columns 18 to 20.
```

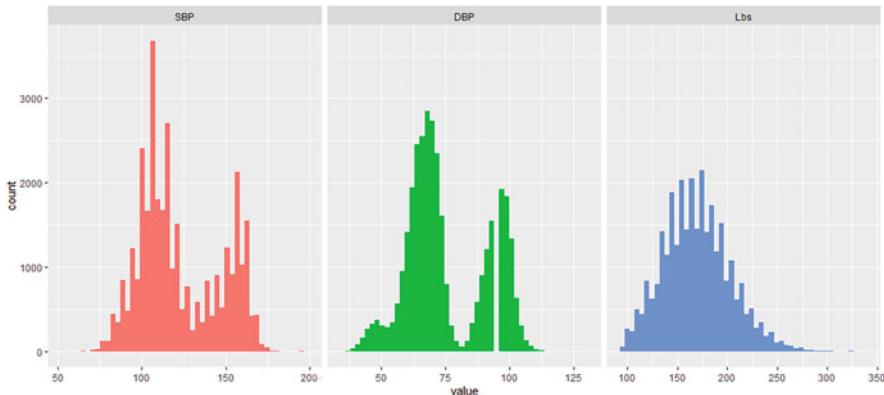


Figure 8.14: Quality assurance exploratory data analysis—14

Going beyond use of these convenient functions found in Exploratory Data Analysis (EDA) packages, there are many other graphical functions available to the R community from among the thousands of packages that supplement what is originally available from when R is first downloaded. However, for a continuing desire for simplicity at the beginning of this lesson, more complex graphically-oriented functions will be demonstrated later in this lesson (Figs. 8.14–8.25).⁴⁵

Common Graphical Functions for Factor-Type Data

R Input

```
par(ask=TRUE)
par(mfrow=c(1,3)) # 3 figures into a 1 row by 3 column grid
graphics::plot(Hospital.df$Location, main="Location")
graphics::plot(Hospital.df$LocationCode, main="Location Codes")
graphics::plot(Hospital.df$Location.Recode,
               main="Location Recoded")
```

⁴⁵When viewing the syntax for these graphically-oriented functions, the `par(ask=TRUE)` function and argument are used to freeze the presentation on the screen, one figure at a time. The top line of the figure, under File—Save as, provides a variety of graphical formats to save each figure: Metafile, Postscript, PDF, PNG, BMP, TIFF, and JPEG. It is also possible to perform a simple copy and paste against each graphical image. R syntax can also be used to save each graphical image. The figures for this text were all saved in .png (portable network graphics) file format.

⁵It is also possible to put multiple figures into one common figure by using the `par(mfrow=c(NumberOfRows,NumberOfColumns))` function and argument, as demonstrated throughout this lesson. This approach is especially useful to show side-by-side comparisons but be sure to have common scales on the X axis and/or the Y axis whenever possible, to make the comparison meaningful.

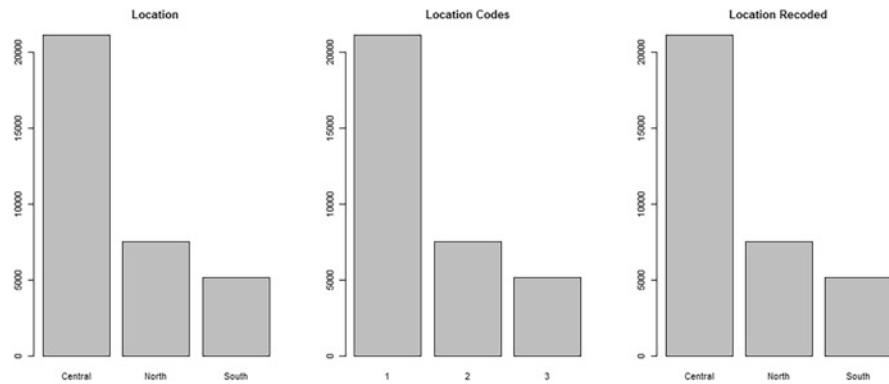


Figure 8.15: Quality assurance exploratory data analysis—15

R Input

```
par.ask=TRUE)
par(mfrow=c(1,3)) # 3 figures into a 1 row by 3 column grid
graphics::plot(Hospital.df$PrimaryPayer, main="Primary Payer")
graphics::plot(Hospital.df$PrimaryPayerCode,
               main="Primary Payer Codes")
graphics::plot(Hospital.df$PrimaryPayer.Recode,
               main="Primary Payer Recoded")
```

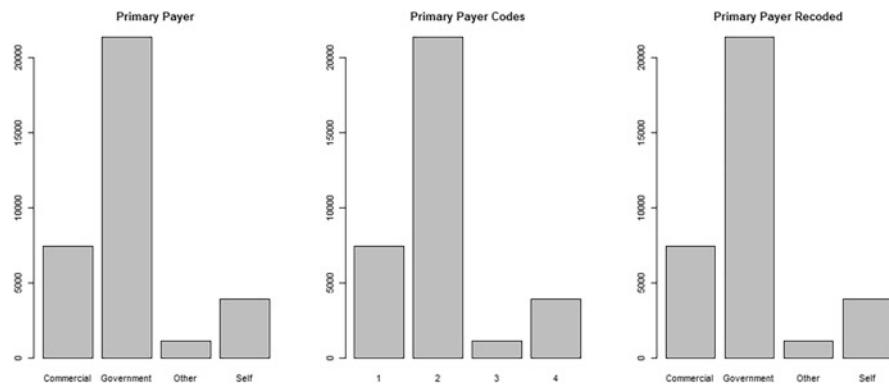


Figure 8.16: Quality assurance exploratory data analysis—16

ssl

R Input

```
par.ask=TRUE)
par(mfrow=c(2,1)) # 2 figures into a 2 row by 1 column grid
```

```
graphics::plot(Hospital.df$PrincipalDiagnosis3DigitICD9,
  main="Primary Diagnosis 3-Digit ICD 9")
# It is simply not possible to have all labels for the many
# separate 3-digit ICD9 codes show in this figure. There are
# just too many codes.
graphics::plot(Hospital.df$PrincipalDiagnosisRangeICD9,
  main="Primary Diagnosis Range of 3-Digit ICD 9")
# Later in this lesson, note how breakouts for the object
# variable Hospital.df$PrincipalDiagnosisRangeICD9 are all
# displayed in a figure by wrapping the table() function and
# the barplot() function.
```

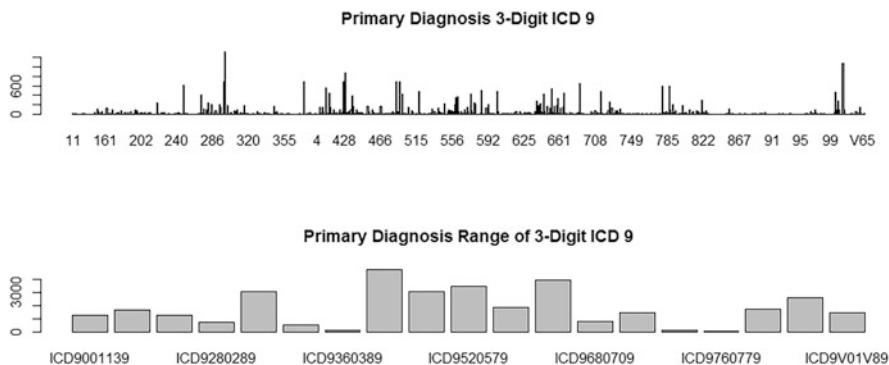


Figure 8.17: Quality assurance exploratory data analysis—17

R Input

```
par.ask=TRUE)
par(mfrow=c(2,1)) # 2 figures into a 2 row by 1 column grid
graphics::plot(Hospital.df$AgeRangeCode,
  main="Age Range Codes")
graphics::plot(Hospital.df$AgeRange.Recode,
  main="Age Range Recoded")
```

R Input

```
par.ask=TRUE)
par(mfrow=c(3,1)) # 3 figures into a 3 row by 1 column grid
graphics::plot(Hospital.df$Race, main="Race")
graphics::plot(Hospital.df$RaceCode, main="Race Codes")
graphics::plot(Hospital.df$Race.Recode, main="Race Recoded")
# Short labels for breakout variables are often best, as is
# evident in this figure.
```

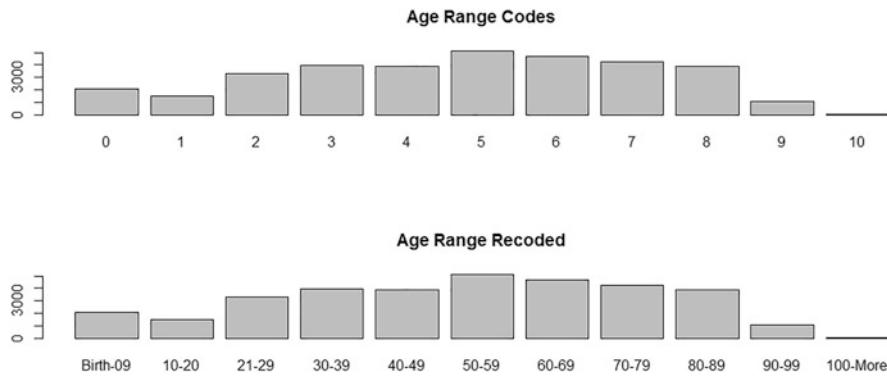


Figure 8.18: Quality assurance exploratory data analysis—18

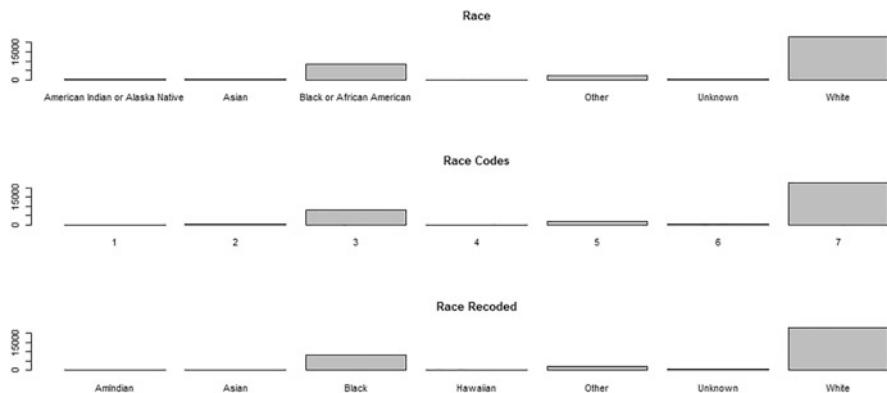


Figure 8.19: Quality assurance exploratory data analysis—19

R Input

```
par(ask=TRUE)
par(mfrow=c(3,1)) # 3 figures into a 3 row by 1 column grid
graphics::plot(Hospital.df$Ethnicity, main="Ethnicity")
graphics::plot(Hospital.df$EthnicityCode,
               main="Ethnicity Codes")
graphics::plot(Hospital.df$Ethnicity.Recode,
               main="Ethnicity Recoded")
```

R Input

```
par(ask=TRUE)
par(mfrow=c(1,3)) # 3 figures into a 1 row by 3 column grid
graphics::plot(Hospital.df$Sex, main="Sex")
graphics::plot(Hospital.df$SexCode, main="Sex Codes")
graphics::plot(Hospital.df$Sex.Recode, main="Sex Recoded")
```

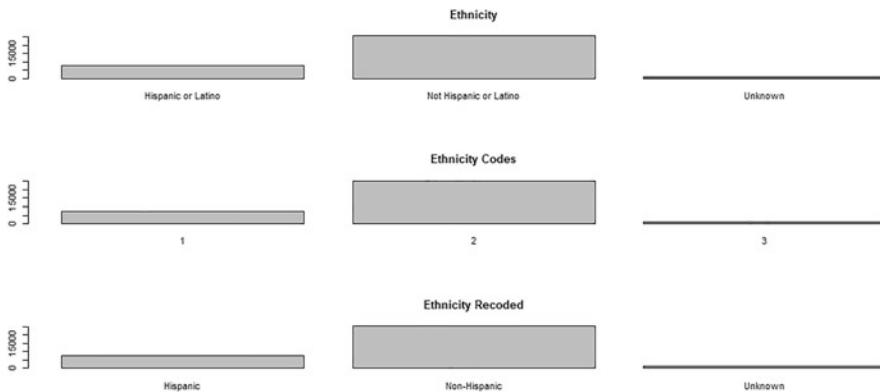


Figure 8.20: Quality assurance exploratory data analysis—20

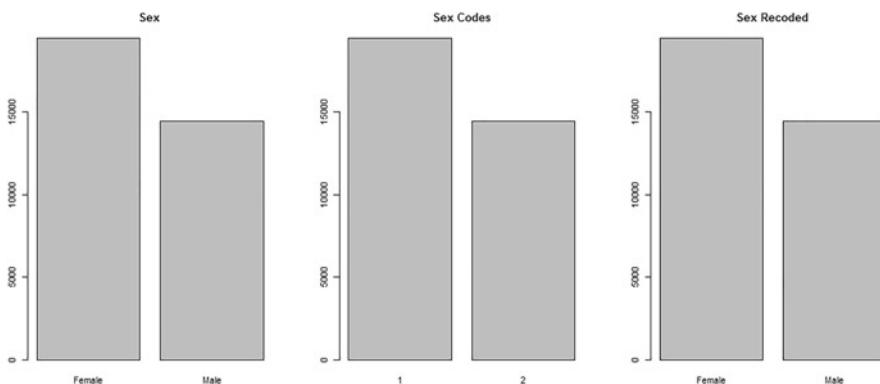


Figure 8.21: Quality assurance exploratory data analysis—21

Common Graphical Functions for Numeric Data

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(Hospital.df$AgeLastBirthday,
  main="Age Last Birthday")
graphics::plot(stats::density(Hospital.df$AgeLastBirthday,
  na.rm=TRUE), main="Age Last Birthday")
# na.rm argument is required
graphics::boxplot(Hospital.df$AgeLastBirthday,
  main="Age Last Birthday")
stats::qqnorm(Hospital.df$AgeLastBirthday,
  main="Age Last Birthday")
```

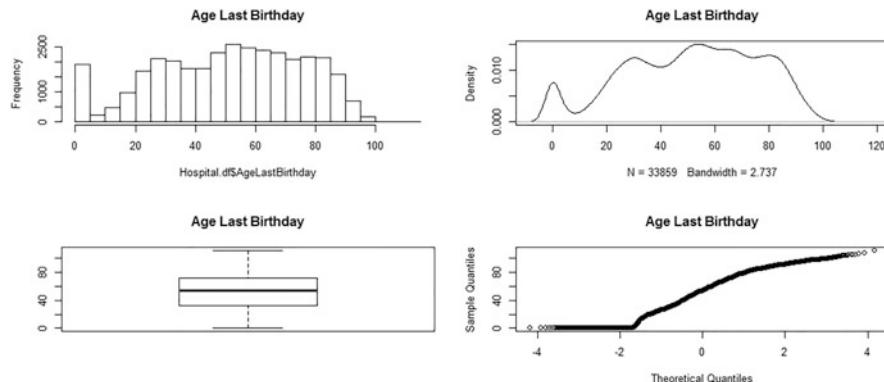


Figure 8.22: Quality assurance exploratory data analysis—22

R Input

```
par(ask=TRUE)
par(mfrow=c(3,2)) # 6 figures into a 3 row by 2 column grid
graphics::hist(Hospital.df$OverNightStay,
  main="Number of Nights for Over Night Stay")
graphics::plot(stats::density(Hospital.df$OverNightStay,
  na.rm=TRUE), main="Number of Nights for Over Night Stay")
# na.rm argument is required
graphics::boxplot(Hospital.df$OverNightStay,
  main="Number of Nights for Over Night Stay")
stats::qqnorm(Hospital.df$OverNightStay,
  main="Number of Nights for Over Night Stay")
graphics::plot(Hospital.df$OverNightStayCode,
  main="Over Night Stay Codes")
graphics::plot(Hospital.df$OverNightStay.Recode,
  main="Over Night Stay Recoded")
```

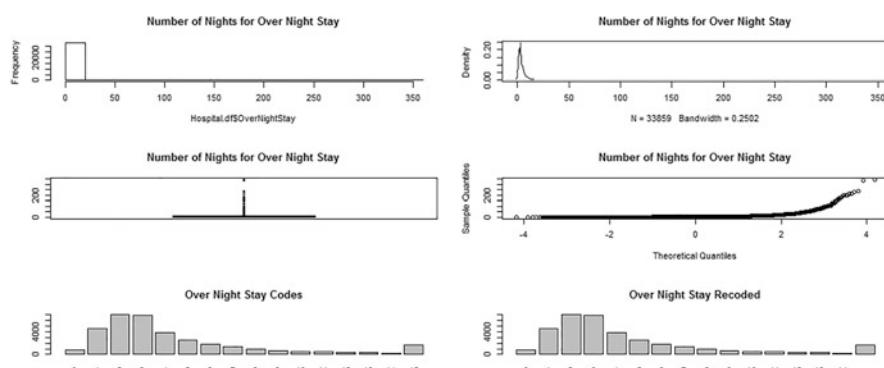


Figure 8.23: Quality assurance exploratory data analysis—23

R Input

```
par.ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(Hospital.df$SBP, main="Systolic Blood Pressure")
graphics::plot(stats::density(Hospital.df$SBP, na.rm=TRUE),
               main="Systolic Blood Pressure") # na.rm argument is required
graphics::boxplot(Hospital.df$SBP,
                   main="Systolic Blood Pressure")
stats::qqnorm(Hospital.df$SBP, main="Systolic Blood Pressure")
```

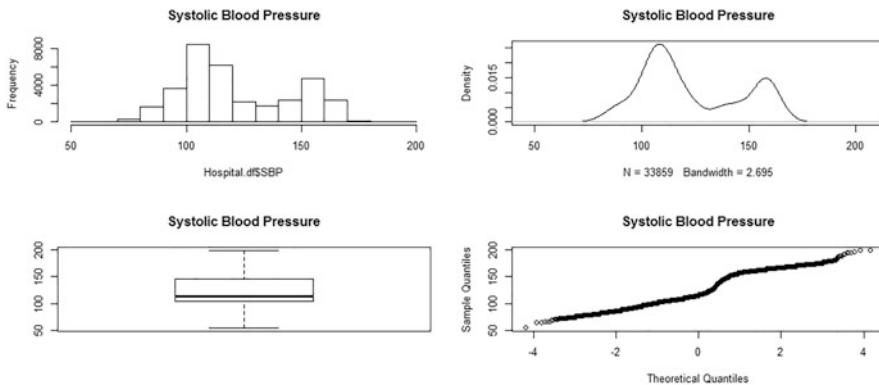


Figure 8.24: Quality assurance exploratory data analysis—24

R Input

```
par.ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(Hospital.df$DBP,
               main="Diastolic Blood Pressure")
graphics::plot(stats::density(Hospital.df$DBP, na.rm=TRUE),
               main="Diastolic Blood Pressure") # na.rm argument is required
graphics::boxplot(Hospital.df$DBP,
                   main="Diastolic Blood Pressure")
stats::qqnorm(Hospital.df$DBP, main="Diastolic Blood Pressure")
```

R Input

```
r.ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(Hospital.df$Lbs, main="Weight - Lbs")
graphics::plot(stats::density(Hospital.df$Lbs, na.rm=TRUE),
```

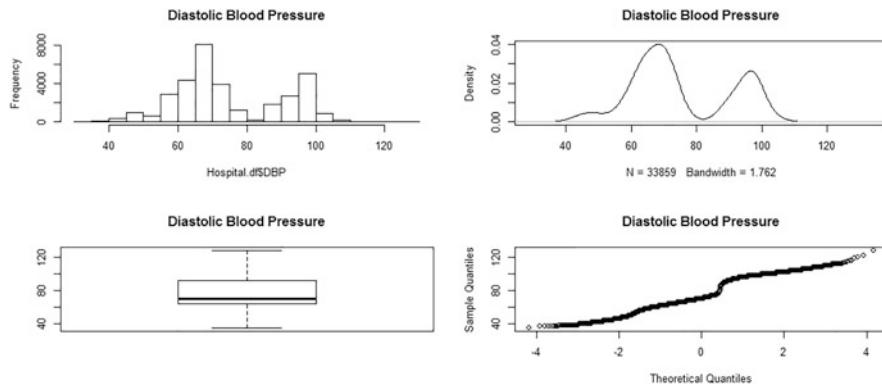


Figure 8.25: Quality assurance exploratory data analysis—25

```
main="Weight - Lbs") # na.rm argument is required
graphics::boxplot(Hospital.df$Lbs, main="Weight - Lbs")
stats::qqnorm(Hospital.df$Lbs, main="Weight - Lbs")
```

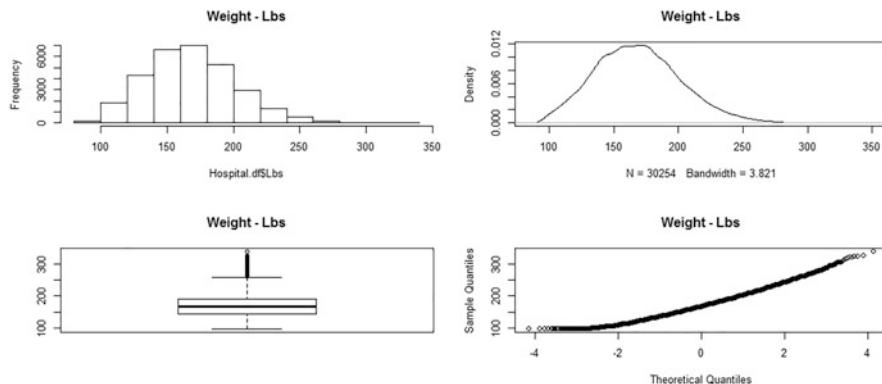


Figure 8.26: Quality assurance exploratory data analysis—26

These initial graphics were purposely presented in a simple format, with few if any embellishments. Learn the basics first and then build on this initial learning with new learning. However, look at the upcoming figures and accompanying syntax for a preview of possibilities on how R can be used to generate graphical images that are of publishable quality (Fig. 8.26).

A sense of how this desire for quality with graphics can be achieved follows, building on the prior `graphics::plot()` function as it was applied against the object variable `PrincipalDiagnosisRangeICD9`. The full set of 3-digit ICD9 codes did not show in the initial figure, due to space limitations. It is possible, however, to have the 3-digit ICD9 codes show when they are presented as a range, based on domain for the medical condition, even though there are 19 separate ranges. Review the syntax and comments (The `#` symbol indicates a

comment, using R.) to see how a space limitation problem with graphics can be addressed (Fig. 8.27).

R Input

```
par("mar") # Confirm default margin (BLTR)
# BLTR - Bottom, Left, Top, Right
```

R Output

```
[1] 5.1 4.1 4.1 2.1
```

R Input

```
par(mar=c(5, 8, 4, 2) + 0.1) # Set a new margin
par(ask=TRUE) # Control the screen
graphics::barplot(base::table(
  Hospital.df$PrincipalDiagnosisRangeICD9),
  main="Primary Diagnosis Range of 3-Digit ICD 9",
  font=2, # Bold
  horiz=TRUE, # Horizontal bars
  las=1, # Horizontal axis labels
  col="red", # Color
  xlab="Number of Cases", # X axis label
  xlim=c(0,5000), # X axis scale
  cex.axis=1.25, # Axis size
  cex.names=0.75, # Label size
  cex.lab=1.25) # Label size
par(mar=c(5.1, 4.1, 4.1, 2.1))# Toggle back to default margin
```

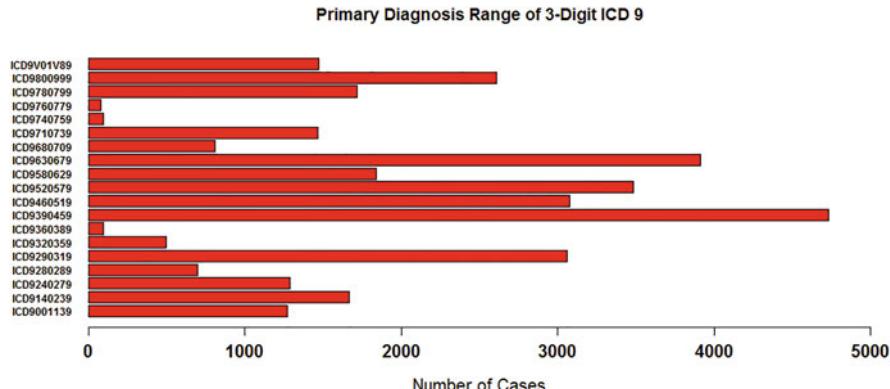


Figure 8.27: Quality assurance exploratory data analysis—27

R Input

```
par("mar") # Confirm default margin (BLTR)
# BLTR - Bottom, Left, Top, Right
```

R Output

```
[1] 5.1 4.1 4.1 2.1
```

The barplot of the Primary Diagnosis Range of 3-Digit ICD 9 values needed some accommodation to have the complete text of 19 separate 3-digit code ranges (e.g., ICD9001139, ICD9140239, ICD9240279, etc.) show in the figure:

- Experimenting with the figure early-on, it was seen that the margins for this figure would need adjusting to accommodate a wide left margin, for the ICD9 3-digit range codes.
- The default margin was confirmed using the `par("mar")` function.
- After some amount of experimentation and trial-and-error, new margin settings were temporarily created to accommodate the anticipated figure since it is presented in horizontal format. The margin settings are in BLTR format: Bottom, Left, Top, Right.
- The `graphics::barplot()` function was then wrapped around (e.g., applied against) the `base::table()` function, or for this specific figure a barplot was created from table values for breakouts for the object variable `Principal-DiagnosisRangeICD9`. The barplot is in horizontal format (e.g., `horiz=TRUE`) and labels are horizontal to the axis (e.g., `las=1`).
- Merely to add appeal and better visualization to the figure, bars were made red and the axis and labels were also adjusted.
- After successful creation of the figure the margins were set back to their default settings and then confirmed that they are correct.

Remember that these graphics are for quality assurance review against the entire collection of data contained in `Hospital.df`. There is no filtering (e.g., selection) of subjects, such as the type of selection processes needed if there were a desire to examine possible differences in Systolic Blood Pressure, Diastolic Blood Pressure, or Weight by age groups. Graphical presentations will be far different when these comparisons are made, such as use of the `graphics::boxplot()` function with embellishments (Fig. 8.28).

R Input

```
par.ask=TRUE)
graphics::boxplot(Hospital.df$SBP ~
  Hospital.df$AgeRange.Recode,
  main="Systolic Blood Pressure by Age Groups",
  xlab="Age Group",                                # X axis label
  ylab="Systolic Blood Pressure",                  # Y axis label
  ylim=c(0,200),                                  # Y axis range
  cex.axis=1.25,                                   # Axis size
  cex.lab=1.25,                                    # Label size
  col.axis="magenta",                            # Axis color
  col.lab ="darkblue",                           # Label color
  col="red",                                       # Box color
  lwd=2,                                         # Line thickness
  font=2)                                         # Bold
```

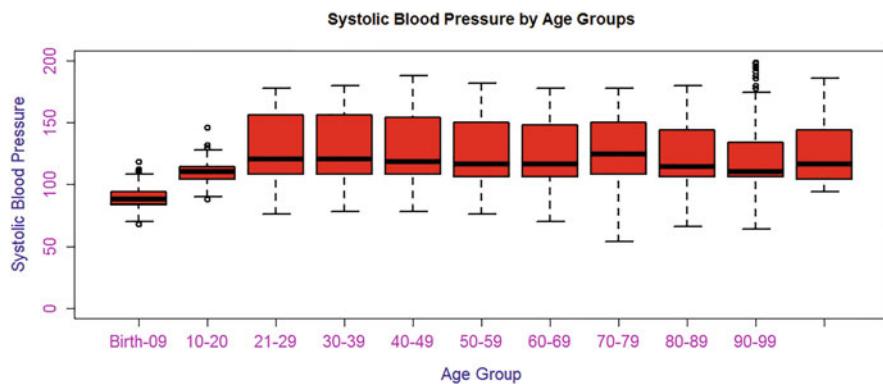


Figure 8.28: Quality assurance exploratory data analysis—28

More than just one or two arguments have now been added to the boxplot, largely to provide an initial demonstration of how a graphical image can be embellished. The use of magenta and darkblue as colors in this figure for the axis and labels are certainly not standard and would never be used in this context for professional publication. Instead, these two colors were purposely selected to demonstrate possibilities for adjustment over default selections and how R supports many different options, whether professionally appealing or not.

To go beyond what has been graphically presented in the first part of this lesson, in separate addenda:

- Additional figures will be shown, demonstrating other ways to graphically show relationships between and among the data.

- The many possibilities of the lattice graphics package will be demonstrated.
- The increasingly popular ggplot2 package will be extensively demonstrated, to show the production of figures that follow along with the notion of *Beautiful Graphics* and the *Grammar of Graphics*.

8.5 Descriptive Statistics for Initial Analysis of the Data

Going back to the prior comments on the use of Exploratory Data Analysis (EDA) packages, consider a few functions from the DataExplorer package and the funModeling package. The summarytools package is also valuable and its use should also be explored when addressing descriptive statistics.

R Input

```
DataExplorer::introduce(Hospital.df)
# Generate a broad overview of the data, with a focus
# on rows, columns, missing data, and total observations.
```

R Output

```
[Output was adjusted to account for original wide format.]
rows    columns discrete_columns continuous_columns
33859        27            22                  5

all_missing_columns
                    0

total_missing_values complete_rows total_observations
                    3605          30254           914193

memory_usage
6838040
```

R Input

```
funModeling::df_status(Hospital.df)
# Generate a by variable summary of quantity and percentage
# of: zeros (q_zeros and p_zeros), missing values (q_na and
# p_na), infinite values (q_inf and p_inf), data type and
# the number of unique values for each variable.
```

R Output

[Output was adjusted to account for original wide format.]					
	variable	q_zeros	p_zeros	q_na	p_na
1	Ticket	0	0.00	0	0.00
2	Location	0	0.00	0	0.00
3	LocationCode	0	0.00	0	0.00
4	PrimaryPayer	0	0.00	0	0.00
5	PrimaryPayerCode	0	0.00	0	0.00
6	PrincipalDiagnosis3DigitICD9	0	0.00	0	0.00
7	PrincipalDiagnosisRangeICD9	0	0.00	0	0.00
8	AgeLastBirthday	1497	4.42	0	0.00
9	AgeRangeCode	2085	6.16	0	0.00
10	Race	0	0.00	0	0.00
11	RaceCode	0	0.00	0	0.00
12	Ethnicity	0	0.00	0	0.00
13	EthnicityCode	0	0.00	0	0.00
14	Sex	0	0.00	0	0.00
15	SexCode	0	0.00	0	0.00
16	OverNightStay	799	2.36	0	0.00
17	OverNightStayCode	799	2.36	0	0.00
18	SBP	0	0.00	0	0.00
19	DBP	0	0.00	0	0.00
20	Lbs	0	0.00	3605	10.65
21	Location.Recode	0	0.00	0	0.00
22	PrimaryPayer.Recode	0	0.00	0	0.00
23	AgeRange.Recode	0	0.00	0	0.00
24	Race.Recode	0	0.00	0	0.00
25	Ethnicity.Recode	0	0.00	0	0.00
26	Sex.Recode	0	0.00	0	0.00
27	OverNightStay.Recode	799	2.36	0	0.00
	variable	q_inf	p_inf	type	unique
1	Ticket	0	0	factor	33859
2	Location	0	0	factor	3
3	LocationCode	0	0	factor	3
4	PrimaryPayer	0	0	factor	4
5	PrimaryPayerCode	0	0	factor	4
6	PrincipalDiagnosis3DigitICD9	0	0	factor	662
7	PrincipalDiagnosisRangeICD9	0	0	factor	19
8	AgeLastBirthday	0	0	numeric	109
9	AgeRangeCode	0	0	factor	11
10	Race	0	0	factor	7

11	RaceCode	0	0	factor	7
12	Ethnicity	0	0	factor	3
13	EthnicityCode	0	0	factor	3
14	Sex	0	0	factor	2
15	SexCode	0	0	factor	2
16	OverNightStay	0	0	numeric	122
17	OverNightStayCode	0	0	factor	16
18	SBP	0	0	numeric	74
19	DBP	0	0	numeric	54
20	Lbs	0	0	numeric	121
21	Location.Recode	0	0	factor	3
22	PrimaryPayer.Recode	0	0	factor	4
23	AgeRange.Recode	0	0	factor	11
24	Race.Recode	0	0	factor	7
25	Ethnicity.Recode	0	0	factor	3
26	Sex.Recode	0	0	factor	2
27	OverNightStay.Recode	0	0	factor	16

R Input

```

install.packages("summarytools", dependencies=TRUE)
library(summarytools)          # Load the summarytools package.
help(package=summarytools)    # Show the information page.
sessionInfo()                 # Confirm all attached packages.

summarytools::freq(Hospital.df$PrincipalDiagnosisRangeICD9,
  round.digits=2, style="simple", totals=TRUE)
# Produce a simple text-based frequency distribution table
# of the selected factor-type object variable. The table
# can be easily copied and pasted into a word-processed
# document if needed. Experiment with this function on
# other variables.
#
# Notice how the funModeling package and the summarytools
# tools each have a function called freq(), thus the reason
# why it is a necessary practice to be verbose and write
# funModeling::freq() and summarytools::freq() and not
# freq() by itself.

```

R Output

Frequencies

Hospital.df\$PrincipalDiagnosisRangeICD9

Type: Factor

	Freq	% Valid	% Valid Cum.	% Total	% Total Cum.
ICD9001139	1270	3.75	3.75	3.75	3.75
ICD9140239	1669	4.93	8.68	4.93	8.68
ICD9240279	1286	3.80	12.48	3.80	12.48
ICD9280289	699	2.06	14.54	2.06	14.54
ICD9290319	3062	9.04	23.59	9.04	23.59
ICD9320359	497	1.47	25.05	1.47	25.05
ICD9360389	94	0.28	25.33	0.28	25.33
ICD9390459	4731	13.97	39.30	13.97	39.30
ICD9460519	3077	9.09	48.39	9.09	48.39
ICD9520579	3483	10.29	58.68	10.29	58.68
ICD9580629	1838	5.43	64.11	5.43	64.11
ICD9630679	3910	11.55	75.65	11.55	75.65
ICD9680709	808	2.39	78.04	2.39	78.04
ICD9710739	1468	4.34	82.38	4.34	82.38
ICD9740759	95	0.28	82.66	0.28	82.66
ICD9760779	80	0.24	82.89	0.24	82.89
ICD9780799	1714	5.06	87.96	5.06	87.96
ICD9800999	2606	7.70	95.65	7.70	95.65
ICD9V01V89	1472	4.35	100.00	4.35	100.00
<NA>	0			0.00	100.00
Total	33859	100.00	100.00	100.00	100.00

R Input

```
summarytools::descr(Hospital.df)
# Produce a simple text-based table of descriptive
# statistics for all numeric object variables. The
# table can be easily copied and pasted into a word-
# processed document if needed.
```

R Output

[Output was adjusted to account for original wide format.]
 Non-numerical variable(s) ignored: Ticket, Location,
 LocationCode, PrimaryPayer, PrimaryPayerCode,

PrincipalDiagnosis3DigitICD9, PrincipalDiagnosisRangeICD9,
 AgeRangeCode, Race, RaceCode, Ethnicity, EthnicityCode, Sex,
 SexCode, OverNightStayCode, Location.Recode,
 PrimaryPayer.Recode, AgeRange.Recode, Race.Recode,
 Ethnicity.Recode, Sex.Recode, OverNightStay.Recode

Descriptive Statistics

Hospital.df

N: 33859

	AgeLastBirthday	DBP	Lbs
Mean	51.82	75.13	169.04
Std.Dev	24.49	15.77	33.42
Min	0.00	34.00	96.00
Q1	33.00	64.00	144.00
Median	54.00	70.00	168.00
Q3	71.00	92.00	190.00
Max	111.00	129.00	340.00
MAD	28.17	11.86	32.62
IQR	38.00	28.00	46.00
CV	0.47	0.21	0.20
Skewness	-0.30	0.28	0.40
SE.Skewness	0.01	0.01	0.01
Kurtosis	-0.71	-0.94	0.17
N.Valid	33859.00	33859.00	30254.00
Pct.Valid	100.00	100.00	89.35

	OverNightStay	SBP
Mean	5.00	122.80
Std.Dev	8.33	24.12
Min	0.00	54.00
Q1	2.00	104.00
Median	3.00	114.00
Q3	5.00	146.00
Max	346.00	199.00
MAD	1.48	20.76
IQR	3.00	42.00
CV	1.67	0.20
Skewness	12.88	0.43
SE.Skewness	0.01	0.01
Kurtosis	308.79	-1.04
N.Valid	33859.00	33859.00

Pct.Valid	100.00	100.00
-----------	--------	--------

R Input

```
view(summarytools::dfSummary(Hospital.df))
# Output of the view(summarytools::dfSummary()) functions
# opens in a browser. The details are extremely helpful
# and provide an excellent summary of the entire dataset.
# As printed to the screen, be sure to give attention to
# the pathway of the generated .html file, if needed for
# later use.
#
# As a separate Web page generated in .html format, the
# output is not included in this lesson. Again, give
# attention to the pathway for the generated .html file.
```

8.5.1 Analyses of Object Variables in Original Format

A set of packages, with each package supporting many different functions and function arguments, is obtained when R is first downloaded. Key the function base::search() at the R prompt to see the many packages and objects (if any) currently attached in the active R session. As a reminder, .GlobalEnv, package:stats, package:graphics, package:grDevices, package:utils, package:datasets, package:methods, Autoloads, and package:base will show when R is first downloaded.

From among these many packages available after the initial R download, the two packages with functions that are most frequently used for generating descriptive statistics and measures of central tendency are the base package and the stats package. Although it may be unnecessary, to reinforce good programming practices the full package name and function name convention is used in most cases throughout this lesson: Package::Function(). In other lessons this naming practice is often used only for functions from external packages (e.g., those packages not included in the initial download of R).

Although this text is focused on how R is used to support biostatistics, a very brief review of the concept of descriptive statistics and measures of central tendency may help with a better understanding of the many R functions presented in this lesson. For this review, which is separate from the main focus of this lesson, consider an example where a feeding study for dogs (*Canis lupus familiaris* or *Canis familiaris*) was undertaken. As part of a brief summary of findings, it was stated that the average weight of dogs in the study was 18.15 kilograms (Kg) (about 40 pounds). This finding may be useful, but by itself the statement, *The average weight of dogs in the study was 18.15 kilograms (Kg)*, is only a start to understanding the implications of later findings from the study:

- When declaring the finding for average weight of dogs in the sample, was there any declaration of the descriptive statistic used to express average: mean, median, mode? These three statistics are measures of centrality (e.g., average), but there can often be large differences between each statistic depending on the nature of data distribution for the measured sample.
- How many dogs were weighed—what was the sample size?
- Were all dogs in the study weighed? Were there any missing data and if so, why were the data missing? Had any dogs been removed for the study and if so, what were the reasons for their removal?
- Considering the wide variance in size and weight between different dog breeds (e.g., A fully grown Chihuahua may weigh only 2.25 Kg, whereas a Newfoundland may weigh 65.75 Kg.), was the sample restricted to dogs of the same breed?
- Considering the wide variance in size and weight between male dogs and female dogs (e.g., It is not unexpected for a healthy adult male dog to weigh 20% or more than a healthy adult female dog of the same age and breed.), was the sample restricted to dogs of the same sex and same approximate age?
- Considering dogs (both sexes) that are not kept for breeding purposes and how neutering may impact weight gain, was the sample restricted to dogs not only of the same sex and same approximate age but also dogs that had been neutered or dogs that had not been neutered?
- On this issue of variance for weights, was a measure of variance, such as standard deviation, provided along with declaration of average weight?
- What protocols did the technicians associated with the study use to weigh the dogs? Was the training for the technicians adequate and were they supervised? How did the technicians accommodate the expected movement of dogs placed on a scale and the way jumping and sudden movements impact weight readings from a scale?
- Even if all protocols were followed correctly, was the scale (or scales) certified as being reliable (e.g., consistent measures) and valid (e.g., accurate measures)? Was this certification of reliability and validity (e.g., true measure) recent?

Although many more methodology-type questions could be asked for any attempt to provide descriptive statistics and measures of central tendency, for now it is best to actually recap how R is used for this purpose. For this demonstration, consider Systolic Blood Pressure (SBP) since this variable and its use in biostatistics (specifically, health care) is known to some degree by most adults.

However, notice in the dataframe `Hospital.df` that SBP is provided for all subjects, with subjects in the dataframe `Hospital.df` ranging in age from soon after birth to over 100 years. Yet, it is reasonable to think that there will be vast differences in SBP between infants and adults. Accordingly, a selection process will be used to create a new object dataframe, with this new object dataframe restricted to adults ranging in age from 21 to 100 years, inclusive, therefore excluding all data from those subjects who on their last birthday were less than 21 years and more than 100 years.

8.5.2 Analyses of Boolean-Based Breakouts of Object Variables

8.5.2.1 Structure the Boolean-Based Data Selection Process

Boolean-based actions will be used to create a new dataset, a dataset that excludes all data from subjects who are less than 21 years and greater than 100 years on their last birthday.⁶ First, confirm that the original dataset is in good form with all data in expected ranges.

R Input

```
length(Hospital.df$Ticket)          # Number of subjects
table(Hospital.df$AgeRange.Recode)    # Frequency of age ranges
summary(Hospital.df$AgeLastBirthday)  # Summary of age
# Outcomes prior to use of the base::subset() function
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0	33.0	54.0	51.8	71.0	111.0

8.5.2.2 Create a New Dataset from an Existing Object Variable

With assurance that the data in the original dataset are in proper form, apply Boolean-based selection strategies to filter-out data that are not needed, resulting in an enumerated dataset that meets new needs. It is far beyond the purpose of this lesson to go into detail on Boolean strategies, but a few key concepts call for use of the following terms: And &, Or |, Not !, Less Than <, Less Than or Equal <=, Greater Than >, Greater Than or Equal >=, Equals ==.⁷

⁶Read a biographical sketch of George Boole (1815–1864) to learn about his contributions to mathematics and eventually how those concepts are applied to data science.

⁷There are many R-based tools that could be used to achieve Boolean-type selections, especially those tools in the tidyverse. The process demonstrated in this lesson was purposely selected to reinforce the heuristics (e.g., the actual process) of how data are selected from a large dataset.

R Input

```
HospitalAge021100.df <- base::subset(Hospital.df,
  Hospital.df$AgeLastBirthday >= 21 &
  Hospital.df$AgeLastBirthday <= 100)
#
# HospitalAge021100.df will consist only of those
# subjects who range in age from 021 to 100
# years, inclusive.
#
# Note use of the base::subset function to create
# a dataframe of only those subjects within the
# specified age ranges, >= 21 years and <= 100
# years.

base::attach(HospitalAge021100.df)      # Attach the data
utils::str(HospitalAge021100.df)         # Structure
base::table(HospitalAge021100.df$AgeRange.Recode)  # Age range
# Give specific attention to the number of subjects in the
# Birth-09 age group and the 10-20 age group, compared to
# the original dataframe Hospital.df.
base::summary(HospitalAge021100.df$AgeLastBirthday) # Summary
# Outcomes after use of the base::subset() function
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
21	40	57	57	74	100

R Input

```
base::length(Hospital.df$Ticket)      # N
```

R Output

```
[1] 33859
```

R Input

```
base::length(HospitalAge021100.df$Ticket) # N
```

R Output

```
[1] 30227
```

R Input

```
base::length(Hospital.df$Ticket) -  
base::length(HospitalAge021100.df$Ticket)  
# Mechanically calculate the difference in the number  
# of subjects from the original dataset (Hospital.df)  
# to the enumerated dataset (HospitalAge021100.df).
```

R Output

```
[1] 3632
```

The four histograms in the 2 row by 2 column grid that follows reinforce the descriptive statistics of Systolic Blood Pressure (SBP) measures for all subjects in Hospital.df and then for subjects between 21 and 100 years, as found in HospitalAge021100.df. As a confirming action, the exclusion of subjects less than 21 and greater than 100 years is also demonstrated by using a 0 years to 120 years scale for the X axis when showing subject distribution by age (Fig. 8.29).

R Input

```
par(ask=TRUE)  
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid  
graphics::hist(Hospital.df$SBP, col="red", ylim=c(0,9500))  
graphics::hist(HospitalAge021100.df$SBP, col="red",  
              ylim=c(0,9500))  
graphics::hist(Hospital.df$AgeLastBirthday, xlim=c(0,120),  
              ylim=c(0,3500), col="red")  
graphics::hist(HospitalAge021100.df$AgeLastBirthday,  
              xlim=c(0,120), ylim=c(0,3500), col="red")  
# Common scales for side-by-side comparative graphics allow  
# meaningful comparisons.
```

Review how HospitalAge021100.df was created by using the base::subset() function, yielding a new dataframe in this R session that includes only those subjects who range in age from 21 to 100, inclusive. These ages were confirmed using the base::length(), base::table(), and base::summary() functions, where base::table(HospitalAge021100.df\$AgeRange.Recode) yielded zero subjects in

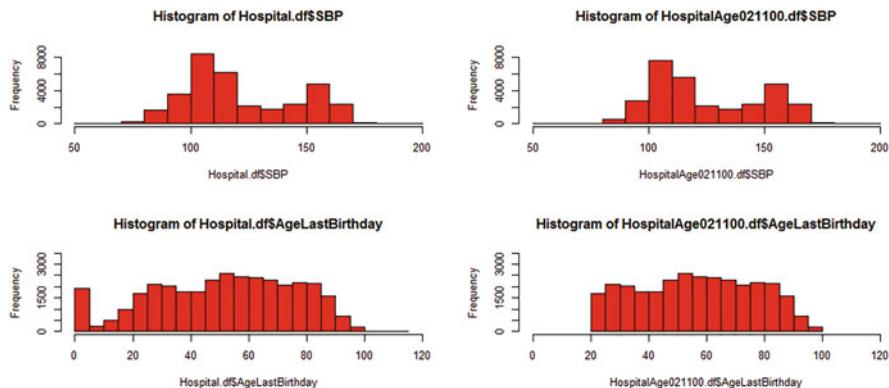


Figure 8.29: Quality assurance review of original dataset and new dataset using systolic blood pressure and age

age group Birth-09 and age group 10–20, the two age groups for those subjects less than 21 years of age. Notice also how the number of subjects 100 years or older was decreased in the `AgeRange.Recode 100-More` age group after the `base::subset()` function was used, now including only those subjects 100 years of age, but not older (e.g., greater than 100 years of age). Because of the subset action and the creation of a new dataframe that includes only those subjects between 21 and 100 years, inclusive, the use and meaning of descriptive statistics and measures of central tendency for Systolic Blood Pressure (SBP) of adults may be more useful than if those less than 21 years or greater than 100 years were also included. Representation is always a concern and it is reasonable to think that vital biometrics of a 6-year-old child and a 56-year-old adult will be sufficiently different.

As a brief reminder on how R is commonly used, recall that the initial download of R includes the base package and the stats package, along with a few other packages. However, there are more than 15,000 R-based external packages posted at worldwide locations and these many packages are available for immediate and free use. Of these many packages, more than a few include functions that address descriptive statistics and measures of central tendency.

A few additional functions from external packages will be demonstrated later in this lesson. Remember that the syntax and output that follows in this part of the lesson is specific to the numeric object variables in `HospitalAge021100.df`. When examining Systolic Blood Pressure, recall that there are no missing datapoints for this object variable. Missing data require special accommodations, which has been demonstrated in early lessons included in this text.

8.5.2.3 Download the New Dataset

The dataset `HospitalAge021100.df` is available for use in the current interactive R session. However, it might be useful to save `HospitalAge021100.df` to

the working directory, for later use. To achieve this aim, use the `write.csv()` function to save the dataset as a .csv (comma-separated values) file.

R Input

```
utils::write.csv(HospitalAge021100.df,
  file="HospitalAge021100.csv")
# The saved file should go to the working directory
# previously declared using the setwd() function, in
# the Housekeeping section of this lesson.
```

For quality assurance purposes, make sure that each row is unique and that there are no duplicate cases. Confirm the number of cases (e.g., rows) in the dataset by using the object variable `HospitalAge021100.df$Ticket`.

R Input

```
base::length(HospitalAge021100.df$Ticket) # N
```

R Output

```
[1] 30227
```

Knowing the N (e.g., `length`) of object variable `HospitalAge021100.df$Ticket` and seeing that the number is in the thousands, wrap the `base::table()` function around the `base::duplicated()` function since it would impossible to keep up with output as it shows on the screen. The `base::duplicated()` function will ideally confirm that there are no duplicated rows by showing all output as FALSE.

R Input

```
base::table(base::duplicated(HospitalAge021100.df$Ticket))
# Use Ticket to determine if there are duplicate admissions
# for hospital services.
```

R Output

```
FALSE
30227
```

The enumerated dataframe `HospitalAge021100.df` seems to be in good form, which can be confirmed using many different R functions. The most immediate choice is use of the `summary()` function against the entire dataframe. How-

ever, going beyond what was previously demonstrated using the Exploratory Data Analysis (EDA) functions, there are a few other functions that provide an overview of an entire dataframe, providing helpful information in a fairly concise and easily understood format. Give attention to use of the `explore::describe()` function and the `summarytools::view(summarytools::dfSummary())` functions, but now for the `HospitalAge021100.df` dataframe.⁸

R Input

```
install.packages("tibble", dependencies=TRUE)
library(tibble) # Load the tibble package.
help(package=tibble) # Show the information page.
sessionInfo() # Confirm all attached packages.

install.packages("explore", dependencies=TRUE)
library(explore) # Load the explore package.
help(package=explode) # Show the information page.
sessionInfo() # Confirm all attached packages.

DescribeHospitalAge021100.tibble <-
  explore::describe(HospitalAge021100.df, out="list")
# Prepare an object that holds output from application of the
# explore::describe() function: type, na, na_pct, unique,
# min, mean, and max.

print(tibble::as_tibble(DescribeHospitalAge021100.tibble),
n=length(HospitalAge021100.df$Ticket))
# Use the length() function to determine the number of rows
# printed to the screen.
```

R Output

```
[Output was adjusted to account for original wide format.]
# A tibble: 27 x 8
  variable      type    na na_pct unique
  <chr>        <chr> <int> <dbl> <int>
1 Ticket        fct     0     0  30227
2 Location      fct     0     0      3
3 LocationCode   fct     0     0      3
4 PrimaryPayer   fct     0     0      4
5 PrimaryPayerCode fct     0     0      4
```

⁸Output from the `explore::describe()` function is generated as a tibble. Wrap the `print()` function around the `tibble::as_tibble()` function to see the entire tibble-based output.

6	PrincipalDiagnosis3DigitICD9	fct	0	0	619
7	PrincipalDiagnosisRangeICD9	fct	0	0	18
8	AgeLastBirthday	dbl	0	0	80
9	AgeRangeCode	fct	0	0	9
10	Race	fct	0	0	7
11	RaceCode	fct	0	0	7
12	Ethnicity	fct	0	0	3
13	EthnicityCode	fct	0	0	3
14	Sex	fct	0	0	2
15	SexCode	fct	0	0	2
16	OverNightStay	dbl	0	0	107
17	OverNightStayCode	fct	0	0	16
18	SBP	dbl	0	0	72
19	DBP	dbl	0	0	48
20	Lbs	dbl	0	0	121
21	Location.Recode	fct	0	0	3
22	PrimaryPayer.Recode	fct	0	0	4
23	AgeRange.Recode	fct	0	0	9
24	Race.Recode	fct	0	0	7
25	Ethnicity.Recode	fct	0	0	3
26	Sex.Recode	fct	0	0	2
27	OverNightStay.Recode	fct	0	0	16

A tibble: 27 x 8

	variable		min	mean	max
	<chr>		<dbl>	<dbl>	<dbl>
1	Ticket		NA	NA	NA
2	Location		NA	NA	NA
3	LocationCode		NA	NA	NA
4	PrimaryPayer		NA	NA	NA
5	PrimaryPayerCode		NA	NA	NA
6	PrincipalDiagnosis3DigitICD9		NA	NA	NA
7	PrincipalDiagnosisRangeICD9		NA	NA	NA
8	AgeLastBirthday		21	57.0	100
9	AgeRangeCode		NA	NA	NA
10	Race		NA	NA	NA
11	RaceCode		NA	NA	NA
12	Ethnicity		NA	NA	NA
13	EthnicityCode		NA	NA	NA
14	Sex		NA	NA	NA
15	SexCode		NA	NA	NA
16	OverNightStay		0	4.86	346
17	OverNightStayCode		NA	NA	NA

18 SBP	54	126.	199
19 DBP	42	77.5	120
20 Lbs	96	169.	340
21 Location.Recode	NA	NA	NA
22 PrimaryPayer.Recode	NA	NA	NA
23 AgeRange.Recode	NA	NA	NA
24 Race.Recode	NA	NA	NA
25 Ethnicity.Recode	NA	NA	NA
26 Sex.Recode	NA	NA	NA
27 OverNightStay.Recode	NA	NA	NA

R Input

```
summarytools::view(summarytools::dfSummary(
  HospitalAge021100.df))
# Give attention to the pathway of the .html output file, if
# needed for later placement into a report.
```

The `explore::describe()` and `summarytools::view(summarytools::dfSummary())` functions generate output that is information-rich and useful for review and diagnostic purposes.⁹ Many tools have initial value for personal use, but as always, use good judgment for what is reported to others.

Overview of Factor Object Variables

The first Null Hypothesis for this lesson involves Diastolic Blood Pressure by the seven different racial groups (e.g., American Indian or Alaska Native, Asian, Black or African American, Native Hawaiian or Other Pacific Islander, Other, Unknown, White) among those subjects who range in age from 40 to 49, inclusive. The factor-type object variables of interest for this Null Hypothesis are `Race.Recode` and `AgeRange.Recode`, as found in the `HospitalAge021100.df` dataframe (Fig. 8.30).

R Input

```
base::table(HospitalAge021100.df$Race.Recode)
```

⁹A Web page in .html format is generated when the `summarytools::view()` function is wrapped around the `summarytools::dfSummary()` function. The file is typically saved to a temporary directory, but the file pathway is printed to the screen, allowing easy retrieval and use, as needed.

R Output

AmIndian	Asian	Black	Hawaiian	Other	Unknown	White
101	299	6994	15	1697	210	20911

R Input

```
base::table(HospitalAge021100.df$AgeRange.Recode)
```

R Output

[Output was adjusted to account for original wide format.]						
Birth-09	10-20	21-29	30-39	40-49	50-59	60-69
0	0	3288	3943	3914	5102	4718
70-79	80-89	90-99	100-More			
4224	3919	1105	14			

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::barplot(base::table(
  HospitalAge021100.df$Race.Recode),
  main="Hospital Patients Age 21 to 100 Years: Race",
  font=2,                                # Bold
  horiz=TRUE,                             # Horizontal bars
  las=1,                                  # Horizontal axis labels
  col=rainbow (7),                         # Color - Number of groups
  xlab="Number of Cases",                  # X axis label
  xlim=c(0,25000),                         # X axis scale
  cex.axis=1.05,                            # Axis size
  cex.names=0.75,                           # Label size
  cex.lab=1.05)                            # Label size
graphics::barplot(base::table(
  HospitalAge021100.df$AgeRange.Recode),
  main="Hospital Patients Age 21 to 100 Years: Age Range",
  font=2,                                # Bold
  horiz=TRUE,                             # Horizontal bars
  las=1,                                  # Horizontal axis labels
  col=rainbow (11),                         # Color - Number of groups
  xlab="Number of Cases",                  # X axis label
```

```

xlim=c(0,6000),          # X axis scale
cex.axis=1.05,            # Axis size
cex.names=0.75,           # Label size
cex.lab=1.05)             # Label size

```

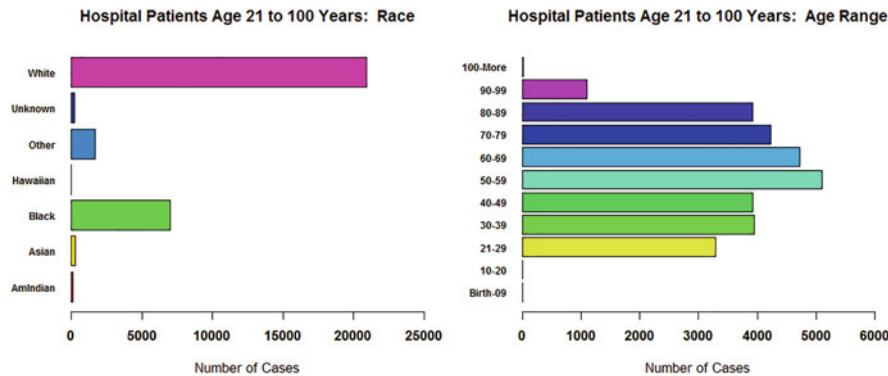


Figure 8.30: Quality assurance review of new dataset using race and age

The second Null Hypothesis for this lesson involves Systolic Blood Pressure and Weight among Black or African American Female subjects who range in age from 40 to 49. Look at the way the graphics::hist() function is used along with Boolean selections that match the selections for this task (Fig. 8.31).

R Input

```
base:::table(HospitalAge021100.df$Race.Recode)
```

R Output

AmIndian	Asian	Black	Hawaiian	Other	Unknown	White
101	299	6994	15	1697	210	20911

R Input

```
base:::table(HospitalAge021100.df$AgeRange.Recode)
```

R Output

[Output was adjusted to account for original wide format.]

Age Range	10-20	21-29	30-39	40-49	50-59	60-69
Birth-09	0	3288	3943	3914	5102	4718

70-79	80-89	90-99	100-More
4224	3919	1105	14

R Input

```
base:::table(HospitalAge021100.df$Sex.Recode)
```

R Output

Female	Male
17512	12715

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::hist(subset(HospitalAge021100.df$SBP,      # Boolean
(HospitalAge021100.df$AgeLastBirthday >= 40 &      # selections
HospitalAge021100.df$AgeLastBirthday <= 49) &      # of Age,
(HospitalAge021100.df$Race.Recode == "Black") &    # Race, and
(HospitalAge021100.df$Sex.Recode == "Female")),    # Sex
main=
"Systolic Blood Pressure (SBP) of Hospital Patients Who
Are Black, Female, and 40 to 49 Years Old",
xlab="Systolic Blood Pressure", ylab="Number of Patients",
breaks=25, col="red", font=2, xlim=c(100,200), ylim=c(0,125))
# By using the Boolean selections, the output will represent
# the SBP of only of those subjects who (1) range in age from
# 40 to 49 on their last birthday, inclusive, (2) are Black,
# and (3) are Female.
#
# Typical to any Boolean-type comparison for equality when
# using R, look at the way == and not = is used for the
# comparison of equivalence.
graphics::hist(subset(HospitalAge021100.df$Lbs,
(HospitalAge021100.df$AgeLastBirthday >= 40 &
HospitalAge021100.df$AgeLastBirthday <= 49) &
(HospitalAge021100.df$Race.Recode == "Black") &
(HospitalAge021100.df$Sex.Recode == "Female")),
main=
"Weight (Lbs) of Hospital Patients Who Are Black,
Female, and 40 to 49 Years Old",
```

```
xlab="Weight (Lbs)", ylab="Number of Patients",
breaks=25, col="red", font=2, xlim=c(0,300), ylim=c(0,50))
```

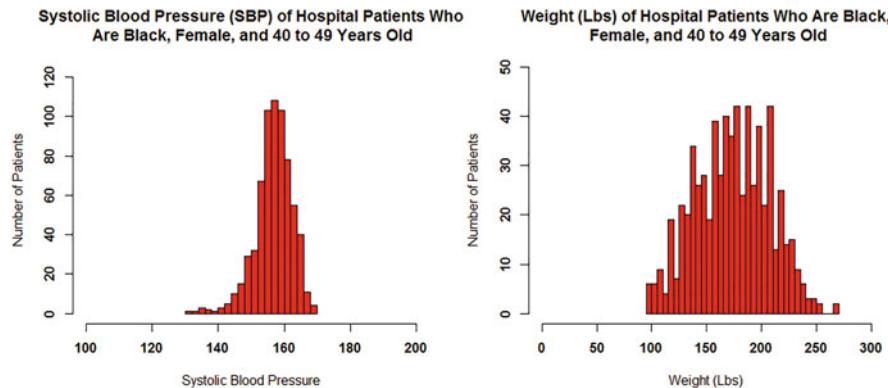


Figure 8.31: Quality assurance review of new dataset using race, sex, and age

The base:::table() function provides a convenient summary of numerical distributions for each breakout group. With this information, adjust the barplot scales as needed, using either the xlim or ylim arguments. The many different figures available with R can be used to provide formative guidance and the figures may also be suitable for presentation to others.

Overview of Numeric Object Variables

The main numeric object variables of interest are Diastolic Blood Pressure for the first Null Hypothesis and Systolic Blood Pressure and Weight for the second Null Hypothesis. There are many functions that can be used to provide a full range of descriptive statistics and measures of central tendency for these three numeric object variables, overall and often by breakout groups of factor-type object variables.

R Input

```
# summarytools:::descr() function and output at the
# univariate level

summarytools:::descr(HospitalAge021100.df,
  stats=c("mean", "sd", "min", "max", "q1", "med", "q3",
  "n.valid", "pct.valid"), transpose=FALSE, headings=TRUE)
# Generate descriptive statistics for all numeric object
# variables, with factor-type object variables ignored.
```

R Output

[Output was adjusted to account for original wide format.]
 Non-numerical variable(s) ignored: Ticket, Location,
 LocationCode, PrimaryPayer, PrimaryPayerCode,
 PrincipalDiagnosis3DigitICD9, PrincipalDiagnosisRangeICD9,
 AgeRangeCode, Race, RaceCode, Ethnicity, EthnicityCode, Sex,
 SexCode, OverNightStayCode, Location.Recode,
 PrimaryPayer.Recode, AgeRange.Recode, Race.Recode,
 Ethnicity.Recode, Sex.Recode, OverNightStay.Recode
Descriptive Statistics
 HospitalAge021100.df
 N: 30227

	AgeLastBirthday	DBP	Lbs
<hr/>			
Mean	57.04	77.54	169.03
Std.Dev	20.03	14.59	33.41
Min	21.00	42.00	96.00
Max	100.00	120.00	340.00
Q1	40.00	66.00	144.00
Median	57.00	72.00	168.00
Q3	74.00	94.00	190.00
N.Valid	30227.00	30227.00	30227.00
Pct.Valid	100.00	100.00	100.00

	OverNightStay	SBP
<hr/>		
Mean	4.86	125.81
Std.Dev	7.44	23.43
Min	0.00	54.00
Max	346.00	199.00
Q1	2.00	106.00
Median	3.00	118.00
Q3	5.00	150.00
N.Valid	30227.00	30227.00
Pct.Valid	100.00	100.00

A similar set of descriptive statistics and measures of central tendency can also be generated using the `s20x::summaryStats()` function, with output at the univariate level and by factor-type object variable breakout levels. The output to the screen will of course be somewhat different from other functions, but the results are still quite good in terms of utility.

R Input

```
install.packages("s20x", dependencies=TRUE)
library(s20x)                      # Load the s20x package.
help(package=s20x)                  # Show the information page.
sessionInfo()                      # Confirm all attached packages.

with(HospitalAge021100.df,
  s20x::summaryStats(SBP))
# Show descriptive statistics, overall.
```

R Output

Minimum value:	54
Maximum value:	199
Mean value:	125.81
Median:	118
Upper quartile:	150
Lower quartile:	106
Variance:	548.73
Standard deviation:	23.43
Midspread (IQR):	44
Skewness:	0.38
Number of data values:	30227

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(SBP ~ Race.Recode))
# Show descriptive statistics, by breakout groups.
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
AmIndian	101	139.624	140	4.65371	6	
Asian	299	110.916	110	6.54999	4	
Black	6994	158.217	158	6.32169	8	
Hawaiian	15	157.600	158	5.71714	5	
Other	1697	147.193	146	10.81213	14	
Unknown	210	141.295	142	13.80515	18	
White	20911	113.198	110	14.59577	16	

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(SBP ~ AgeRange.Recode))
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
21-29		3288	128.634	120	24.4381	48
30-39		3943	128.360	120	24.2314	48
40-49		3914	126.604	118	23.9302	46
50-59		5102	124.053	116	23.3680	44
60-69		4718	124.756	116	23.0944	42
70-79		4224	127.990	124	22.9811	42
80-89		3919	123.151	114	21.8943	38
90-99		1105	119.282	110	20.9003	28
100-More		14	112.286	104	14.6730	11

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(SBP ~ Sex.Recode))
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
Female		17512	126.225	116	23.2578	46
Male		12715	125.230	118	23.6422	42

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(DBP))
# Show descriptive statistics, overall.
```

R Output

Minimum value:	42
Maximum value:	120
Mean value:	77.54
Median:	72

Upper quartile:	94
Lower quartile:	66
Variance:	212.79
Standard deviation:	14.59
Midspread (IQR):	28
Skewness:	0.41
Number of data values:	30227

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(DBP ~ Race.Recode))
# Show descriptive statistics, by breakout groups.
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
AmIndian	101	91.2079	92	4.47284	6	
Asian	299	75.7124	72	9.37855	14	
Black	6994	97.2981	98	4.05982	6	
Hawaiian	15	97.2000	96	6.08511	6	
Other	1697	91.8108	92	6.47793	8	
Unknown	210	89.8952	90	9.44024	12	
White	20911	69.5922	68	9.13093	8	

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(DBP ~ AgeRange.Recode))
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
21-29	3288	77.6405	70	16.05803	32	
30-39	3943	77.6130	70	15.81268	32	
40-49	3914	76.2161	70	15.59877	30	
50-59	5102	74.7428	68	15.36862	28	
60-69	4718	76.4218	70	14.64020	26	
70-79	4224	81.7230	76	12.24232	22	
80-89	3919	79.4422	74	11.59992	22	
90-99	1105	76.6154	72	10.74471	10	
100-More	14	73.2857	70	8.28742	5	

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(DBP ~ Sex.Recode))
```

R Output

	Sample Size	Mean	Median	Std Dev	Midspread
Female	17512	77.7086	72	14.4787	28
Male	12715	77.3025	72	14.7327	28

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(Lbs))
# Show descriptive statistics, overall.
```

R Output

Minimum value:	96
Maximum value:	340
Mean value:	169.03
Median:	168
Upper quartile:	190
Lower quartile:	144
Variance:	1116.3
Standard deviation:	33.41
Midspread (IQR):	46
Skewness:	0.4
Number of data values:	30227

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(Lbs ~ Race.Recode))
# Show descriptive statistics, by breakout groups.
```

R Output

	Sample Size	Mean	Median	Std Dev	Midspread
AmIndian	101	173.248	172	37.1633	46
Asian	299	162.348	162	29.1511	41

Black	6994	174.096	172	36.9170	52
Hawaiian	15	172.667	176	22.2283	26
Other	1697	172.751	172	37.2043	50
Unknown	210	175.524	176	36.7706	52
White	20911	167.048	166	31.5949	44

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(Lbs ~ AgeRange.Recode))
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
21-29		3288	166.716	164	32.7402	44
30-39		3943	163.468	162	31.5694	42
40-49		3914	173.937	172	34.4363	48
50-59		5102	174.591	174	34.6110	48
60-69		4718	171.185	170	34.3020	48
70-79		4224	166.744	166	32.5435	44
80-89		3919	165.009	164	31.7990	42
90-99		1105	166.574	164	30.6788	38
100-More		14	173.714	170	26.4151	40

R Input

```
with(HospitalAge021100.df,
  s20x::summaryStats(Lbs ~ Sex.Recode))
```

R Output

	Sample	Size	Mean	Median	Std Dev	Midspread
Female		17512	164.694	164	30.2280	40
Male		12715	175.013	174	36.5197	50

Many more functions could be demonstrated to generate additional descriptive statistics and measures of central tendency.¹⁰ As a brief warning, the default screen output for some of these functions is quite verbose and the screen output

¹⁰Experiment with: `asbio::Mode()`, `stats::median()`, `base::mean()`, `psych::geometric.mean()`, `psych::harmonic.mean()`, `psych::winsor.mean()`, `stats::sd()`, `stats::var()`, `base::min()`, `base::max()`, `base::range()`, `base::length()`, `base::sum()`, `stats::quantile()`, `psytab::norms()`, `stats::IQR()`, `grDevices::boxplot.stats()`, `stats::fivenum()`, `doBy::descStat()`, `tables::tabular()`, `pastecs::stat.desc()`, `psych::describe()`, and `Deducer::descriptive.table()`.

may show in horizontal (e.g., wide) format instead of vertical (e.g., lengthwise) format, making it difficult to copy and paste into an external word-processed document. Experiment with these functions and their many arguments to develop personal preferences, both for content and later manipulation and presentation.

8.6 Quality Assurance, Data Distribution, and Tests for Normality

8.6.1 Graphics for Normality

When working with data, it is important to consider distribution patterns for the data and perhaps more importantly, to determine if the data for a specific object variable follow a normal distribution pattern (e.g., show what many call the bell-shaped curve, or at least a semblance of the bell-shaped curve) or if the data follow a distribution pattern that does not show normality. Look at the R syntax and accompanying graphical images immediately below to review the difference between a normal distribution pattern (prepared for this hypothetical demonstration using the R-based `rnorm()` function) and a distribution pattern that is not normal (prepared using the R-based `runif()` function). Each enumerated object variable, the theoretical `XNormal` and `XNotNormal` objects, will consist of data from 10,000 subjects (Figs. 8.32, 8.33, and 8.34).

R Input

```
set.seed(8)
# Deploy the random number generator to allow equivalent
# generation of data time and time again.

XNormal      <- stats::rnorm(10000, mean=100, sd=10)
# The distribution pattern is ostensibly normal, due to use
# of the stats::rnorm() function.
#
# N = 10,000 subjects, Mean = 100, and SD = 10, so most
# data should range from 70 to 130, or -3 to +3 standard
# deviations.

XNotNormal <- stats::runif(10000, min=70, max=130)
# The distribution pattern is purposely not normal, due to
```

```
# use of the stats::runif() function.  
#  
# N = 10,000 subjects and most if not all data should range  
# from 70 to 130, but normal distribution is absent.
```

8.6.1.1 Histogram

R Input

```
par.ask=TRUE)  
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid  
graphics::hist(XNormal, main="Distribution is Normal",  
    xlim=c(50,150), col="red", font.lab=2, font.axis=2,  
    breaks=50, ylim=c(0,900))  
graphics::hist(XNotNormal, main="Distribution is Not Normal",  
    xlim=c(50,150), col="red", font.lab=2, font.axis=2,  
    breaks=50, ylim=c(0,900))
```

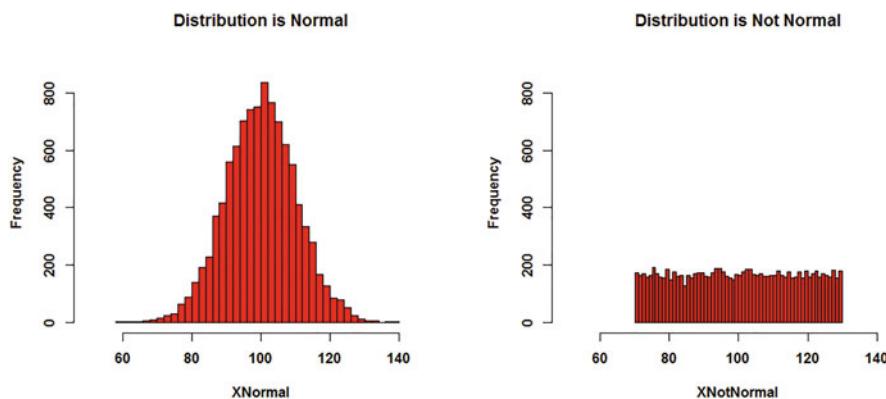


Figure 8.32: Distribution (histogram) of two variables: XNormal and XNotNormal

8.6.1.2 Density Plot

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::plot(stats::density(XNormal, na.rm=TRUE),
  main="Distribution is Normal", col="red", font.lab=2,
  font.axis=2, lwd=4, ylim=c(0.0, 0.04)) # na.rm is required
graphics::plot(stats::density(XNotNormal, na.rm=TRUE),
  main="Distribution is Not Normal", col="red", font.lab=2,
  font.axis=2, lwd=4, ylim=c(0.0, 0.04)) # na.rm is required
```

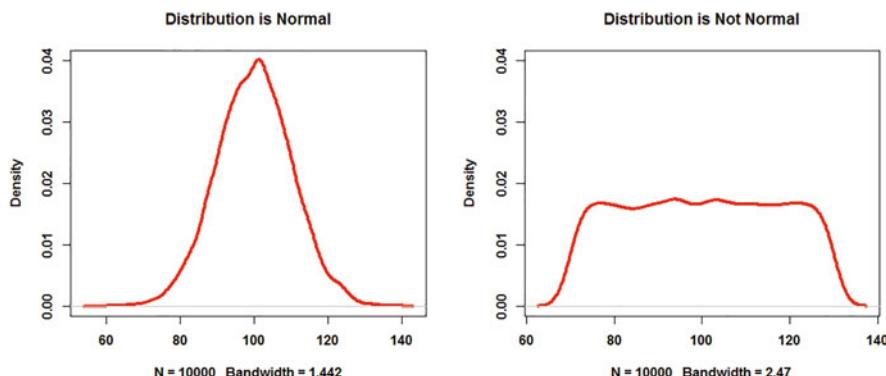


Figure 8.33: Distribution (density plot) of two variables: XNormal and XNotNormal

8.6.1.3 Quantile-Quantile (Q-Q) Plot

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
stats::qqnorm(XNormal, main="Distribution is Normal",
  col="red", font.lab=2, font.axis=2, lwd=4, ylim=c(50,150))
stats::qqnorm(XNotNormal, main="Distribution is Not Normal",
  col="red", font.lab=2, font.axis=2, lwd=4, ylim=c(50,150))
```

Review documentation for the `stats::rnorm()` function and the `stats::runif()` function, to gain a sense of expected output for visual presentation of normal distribution for an object variable (e.g., `XNormal`) as well as visual presentation for an object variable that does not follow normal distribution (e.g., `XNotNormal`). Going beyond visualization, concepts such as central tendency (e.g., Mean, SD, etc.) have vastly different outcomes when data follow normal

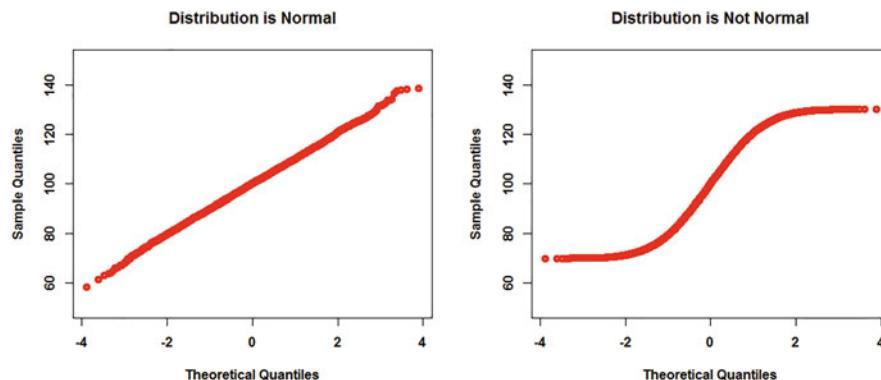


Figure 8.34: Distribution (QQ plot) of two variables: XNormal and XNotNormal

distribution as opposed to when data do not follow normal distribution and that is one of the many reasons why it is important to know if data approximate normal distribution, or not.

With this limited background on normal distribution and an initial attempt to explain why it must be considered when reviewing data, go back to the earlier discussion about a hypothetical research study of feeding for dogs and more specifically the weight of dogs included in the study. At the least, assume that it would be planned for the dogs in this hypothetical research study to be weighed immediately before the treatment was applied and to then be weighed immediately at the end of treatment, although it is likely that if the study were highly detailed weights would be taken more often than only pre-treatment and post-treatment. Other details, for the purpose of this hypothetical discussion, are not necessary and would only cloud the later value of attention to distribution patterns and their importance.

The many questions in this brief review of a hypothetical research study bring to mind concerns about representation of the sample since this issue potentially applies to nearly all empirically-based research and subject representation equally has the potential to impact distribution patterns of the relevant data:

- Were subjects in the sample representative of the population?
- How is representation assessed: by gender, by type, by location, etc.?
- Does representation show for all areas of consideration, for a majority, for only a selected few subjects meeting certain search and select (e.g., Boolean) conditions?

If subjects in a sample were representative of the population, and if the measures from the population show normal distribution, then measures from subjects (if proper measurement protocols are used) should also be expected to follow a normal distribution pattern. Is this assumption confirmed through visual and empirical means?

Of course, expectations are not always met, subjects are not always cooperative, equipment is not always accurate, samples are not always representative, and along with many other possible reasons it is best to use a variety of both graphical tools and normal distribution tests to examine the nature of data distribution. The desire is to determine if data from the sample follow normal distribution patterns and if not, what is the degree of deviation away from normal distribution.

8.6.2 Tests for Normality

As revealing as the statistics and graphics generated by the `summary()`, `qqnorm()`, and `density()` functions graphics may be, it is still best to use statistical tests specifically designed to test for normal distribution. The following R-based packages and functions will be used against the dataframe `HospitalAge021100.df` and from this dataframe the object variable: (1) SBP—all subjects and (2) SBP—subjects between the ages of 21 and 29 on their last birthday, inclusive. A series of normality tests will then be used to make a more informed judgment on the normality of data distribution:

- Anderson–Darling Test for Normality, `nortest::ad.test()`
- Jarque–Bera Test for Normality, `tseries::jarque.bera.test()`
- Lilliefors (Kolmogorov–Smirnov) Test for Normality, `nortest::lillie.test()`
- Shapiro–Wilk Test for Normality, `stats::shapiro.test()`

8.6.2.1 Anderson–Darling Test for Normality

Anderson–Darling Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
install.packages("nortest")
library(nortest)                      # Load the nortest package.
help(package=nortest)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

nortest::ad.test(HospitalAge021100.df$SBP)
```

R Output

```
Outcome: p-value < p-value <0.0000000000000002
```

The p-value associated with this application of the Anderson–Darling Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution.

R Input

```
norntest::ad.test(subset(HospitalAge021100.df$SBP,  
HospitalAge021100.df$AgeLastBirthday >= 21 &  
HospitalAge021100.df$AgeLastBirthday <= 29))
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Anderson–Darling Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution.

8.6.2.2 Jarque–Bera Test for Normality

Jarque–Bera Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
install.packages("tseries")  
library(tseries) # Load the tseries package.  
help(package=tseries) # Show the information page.  
sessionInfo() # Confirm all attached packages.  
  
tseries::jarque.bera.test(HospitalAge021100.df$SBP)
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Jarque–Bera Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution.

R Input

```
tseries::jarque.bera.test(subset(HospitalAge021100.df$SBP,
  HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29))
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Jarque–Bera Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution.

8.6.2.3 Lilliefors (Kolmogorov–Smirnov) Test for Normality Null Hypothesis

Lilliefors (Kolmogorov–Smirnov) Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
nortest::lillie.test(HospitalAge021100.df$SBP)
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Lilliefors (Kolmogorov–Smirnov) Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution.

R Input

```
nortest::lillie.test(subset(HospitalAge021100.df$SBP,
  HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29))
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Lilliefors (Kolmogorov–Smirnov) Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution.

8.6.2.4 Shapiro–Wilk Test for Normality

Shapiro–Wilk Test for Normality Null Hypothesis: The data follow the normal distribution.

It is cautioned that the number of subjects (e.g., non-missing values) supported by the `shapiro.test()` function must be 3 or more but cannot exceed 5000. Due to this limitation, the `base::sample()` function will be applied against the dataframe `Hospital.df` and a new dataframe (`HospitalN4999.df`) will be created, consisting of 4999 subjects. Ostensibly, the enumerated dataframe `HospitalN4999.df` should be representative of the original `Hospital.df` dataframe and the dataset should also have fewer than 5000 subjects, allowing application of the Shapiro–Wilk Test for Normality.

R Input

```
HospitalN4999.df <-  
  Hospital.df[base::sample(1:nrow(Hospital.df),  
  4999, replace=FALSE),]  
  # Use the base::sample() function to create a  
  # new dataset of fewer than 5,000 subjects.  
  
base::length(Hospital.df$Ticket)      # Confirm N
```

R Output

```
[1] 33859
```

R Input

```
base::length(HospitalN4999.df$Ticket) # Confirm N
```

R Output

```
[1] 4999
```

R Input

```
stats::shapiro.test(HospitalN4999.df$SBP)
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Shapiro–Wilk Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution.

R Input

```
base::length(Hospital.df$Ticket) # Confirm N
base::length(HospitalAge021100.df$Ticket) # Confirm N
base::length(subset(HospitalAge021100.df$Ticket, # Confirm N
  HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29))
```

R Output

```
[1] 3288
```

With 3288 subjects between the ages of 21 and 100 on their last birthday, inclusive, the maximum N supported by the stats::shapiro.test() function, which is 5000, is not exceed and use of this function can continue without any adjustments.

R Input

```
stats::shapiro.test(subset(HospitalAge021100.df$SBP,
  HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29))
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Shapiro–Wilk Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data do not follow the normal distribution.

Regarding normal distribution patterns, the different graphics, especially after use of the qqnorm() function, provide initial guidance on normality. Yet, to be more precise the different tests for normality (Anderson–Darling, Jarque–Bera, Lilliefors (Kolmogorov–Smirnov), Shapiro–Wilk) all support the same interpretation:

- The object variable SBP (Systolic Blood Pressure) does not follow normal distribution, overall ($N = 33,859$).
- The object variable SBP (Systolic Blood Pressure) does not follow normal distribution, for subjects between the ages of 21 and 29 on their last birthday, inclusive ($N = 3288$).

However, this dataframe is fairly large not only in terms of N but also in terms of the many different types of variables and their breakouts (Fig. 8.35).¹¹ Are there other variables of interest associated with Hospital.df\$SBP that demand review? As a useful inquiry into this issue, look at the degree of normality as members of the breakout groups become more alike (e.g., at least for this set of data, become more homogeneous).

R Input

```
base::length(Hospital.df$SBP)  
# SBP - All subjects
```

R Output

```
[1] 33859
```

¹¹Search the literature to see how normality is viewed when samples are large, such as the numeric object variables in Hospital.df, where there are thousands of subjects (rows) with multiple object variables (columns). Does a large N mitigate concerns about normality?

R Input

```
base::length(subset(HospitalAge021100.df$SBP,  
  (HospitalAge021100.df$AgeLastBirthday >= 21 &  
   HospitalAge021100.df$AgeLastBirthday <= 29)))  
# SBP - Age = 021 to 029
```

R Output

```
[1] 3288
```

R Input

```
base::length(subset(HospitalAge021100.df$SBP,  
  (HospitalAge021100.df$AgeLastBirthday >= 21 &  
   HospitalAge021100.df$AgeLastBirthday <= 29) &  
   (HospitalAge021100.df$Race.Recode == "White")))  
# SBP - Age = 021 to 029 AND Race = White
```

R Output

```
[1] 1915
```

R Input

```
base::length(subset(HospitalAge021100.df$SBP,  
  (HospitalAge021100.df$AgeLastBirthday >= 21 &  
   HospitalAge021100.df$AgeLastBirthday <= 29) &  
   (HospitalAge021100.df$Race.Recode == "White") &  
   (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic")))  
# SBP - Age = 021 to 029 AND Race = White  
#           AND Ethnic = Non-Hispanic
```

R Output

```
[1] 1189
```

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
stats::qqnorm(Hospital.df$SBP,
  main="SBP - All (N = 33,859)",
  xlim=c(-4,4), ylim=c(0,200), font.axis=2, font.lab=2)
stats::qqline(Hospital.df$SBP,
  col="red", lwd=4, lty=2)
# SBP - All
stats::qqnorm(subset(HospitalAge021100.df$SBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29)),
  main="SBP - Age = 021 to 029 (N = 3,288)",
  xlim=c(-4,4), ylim=c(0,200), font.axis=2, font.lab=2)
stats::qqline(subset(HospitalAge021100.df$SBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29)),
  col="red", lwd=4, lty=2)
# SBP - Age = 021 to 029
stats::qqnorm(subset(HospitalAge021100.df$SBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29) &
  (HospitalAge021100.df$Race.Recode == "White")),
  main="SBP - Age = 021 to 029 AND Race = White (N = 1,915)",
  xlim=c(-4,4), ylim=c(0,200), font.axis=2, font.lab=2)
stats::qqline(subset(HospitalAge021100.df$SBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29) &
  (HospitalAge021100.df$Race.Recode == "White")),
  col="red", lwd=4, lty=2)
# SBP - Age = 021 to 029 AND Race = White
stats::qqnorm(subset(HospitalAge021100.df$SBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29) &
  (HospitalAge021100.df$Race.Recode == "White") &
  (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic")),
  main="SBP - Age = 021 to 029 AND Race = White
  AND Ethnic = Non-Hispanic (N = 1,189)",
  xlim=c(-4,4), ylim=c(0,200), font.axis=2, font.lab=2)
stats::qqline(subset(HospitalAge021100.df$SBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29) &
```

```
(HospitalAge021100.df$Race.Recode == "White") &
(HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic")),
col="red", lwd=4, lty=2)
# SBP - Age = 021 to 029 AND Race = White
#           AND Ethnic = Non-Hispanic
```

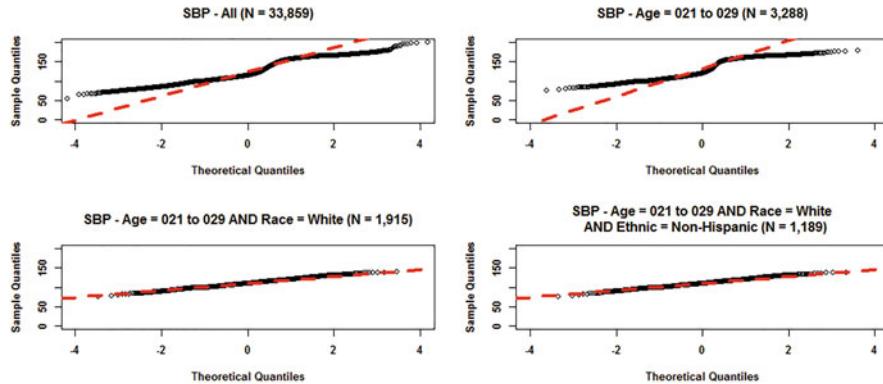


Figure 8.35: Distribution pattern of systolic blood pressure with multiple subsets

To go along with the visual display of normal distribution shown in these graphics, prepared using the `stats::qqnorm()` function, use the `tseries::jarque.bera.test()` function to gain an empirical sense of normal distribution over this increasingly finite set of Boolean-based selections:

R Input

```
tseries::jarque.bera.test(Hospital.df$SBP)
# SBP - All subjects
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Jarque–Bera Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data do not follow the normal distribution.

R Input

```
tseries::jarque.bera.test(subset(HospitalAge021100.df$SBP,  
    (HospitalAge021100.df$AgeLastBirthday >= 21 &  
    HospitalAge021100.df$AgeLastBirthday <= 29)))  
# SBP - Age = 021 to 029
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Jarque–Bera Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data do not follow the normal distribution.

R Input

```
tseries::jarque.bera.test(subset(HospitalAge021100.df$SBP,  
    (HospitalAge021100.df$AgeLastBirthday >= 21 &  
    HospitalAge021100.df$AgeLastBirthday <= 29) &  
    (HospitalAge021100.df$Race.Recode == "White")))  
# SBP - Age = 021 to 029 AND Race = White
```

R Output

```
Outcome: p-value = 0.805
```

The p-value associated with this application of the Jarque–Bera Test for Normality is greater than the p-value of 0.05 and it can be stated that the Null Hypothesis is not rejected (e.g., accepted). The data for this specific breakout group follow the normal distribution.

R Input

```
tseries::jarque.bera.test(subset(HospitalAge021100.df$SBP,  
    (HospitalAge021100.df$AgeLastBirthday >= 21 &  
    HospitalAge021100.df$AgeLastBirthday <= 29) &  
    (HospitalAge021100.df$Race.Recode == "White") &  
    (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic")))  
# SBP - Age = 021 to 029 AND Race = White  
# AND Ethnic = Non-Hispanic
```

R Output

```
Outcome: p-value = 0.863
```

The p-value associated with this application of the Jarque–Bera Test for Normality is greater than the p-value of 0.05 and it can be stated that the Null Hypothesis is not rejected (e.g., accepted). The data for this specific breakout group follow the normal distribution.

As an aid to inquiries of normal distribution by increasingly complex selection process, consider the `RVAideMemoire::byf.shapiro()` function. This function is easily applied and provides useful information on normality by selected breakout groups of factor-type object variables. The Shapiro normality test will be used in the following example. Because the Shapiro test has a limitation of 5000 subjects, the previously created `HospitalN4999.df` dataset will be used. Recall that `HospitalN4999.df` was created from `Hospital.df` and therefore purposely addresses all data, prior to the creation of `HospitalAge021100.df`.

R Input

```
install.packages("RVAideMemoire", dependencies=TRUE)
library(RVAideMemoire)          # Load the RVAideMemoire package.
help(package=RVAideMemoire)    # Show the information page.
sessionInfo()                  # Confirm all attached packages.

RVAideMemoire::byf.shapiro(SBP ~ Race.Recode,
                           data=HospitalN4999.df)
```

R Output

```
Shapiro-Wilk normality tests

data: SBP by Race.Recode

      W           p-value
AmIndian 0.7625       0.001262 ***
Asian     0.8810   0.000018882677510688 ***
Black     0.6937 < 0.0000000000000022 ***
Hawaiian 0.9202       0.538084
Other     0.8686   0.00000000000003161 ***
Unknown   0.8905       0.000118 ***
White     0.9519 < 0.0000000000000022 ***

---
```

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R Input

```
RVAideMemoire::byf.shapiro(SBP ~ AgeRange.Recode,
  data=HospitalN4999.df)
```

R Output

Shapiro-Wilk normality tests

data: SBP by AgeRange.Recode

	W	p-value
Birth-09	0.9893	0.01348 *
10-20	0.9901	0.10404
21-29	0.9080	< 0.00000000000000022 ***
30-39	0.9130	< 0.00000000000000022 ***
40-49	0.9036	< 0.00000000000000022 ***
50-59	0.8964	< 0.00000000000000022 ***
60-69	0.9139	< 0.00000000000000022 ***
70-79	0.9310	< 0.00000000000000022 ***
80-89	0.9118	< 0.00000000000000022 ***
90-99	0.8772	0.0000000001773 ***
100-More	0.9731	0.91288

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R Input

```
RVAideMemoire::byf.shapiro(SBP ~ Sex.Recode,
  data=HospitalN4999.df)
```

R Output

Shapiro-Wilk normality tests

data: SBP by Sex.Recode

	W	p-value
Female	0.9107	< 0.00000000000000022 ***

```

Male    0.9422 < 0.0000000000000022 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Whether functions such as the RVAideMemoire::byf.shapiro() function are used for breakout analyses or direct Boolean-type syntax is used to sequester enumerated subgroups, normality needs to be addressed not only at the broadest level but also by breakouts—often multiple layers down in the selection process. See the prior example of normality where the finding of normality for SBP, overall ($p\text{-value} < 0.000000000000002$), was far different than the finding of normality for SBP of subjects who were 21–29 years old at their last birthday, were White, and were Non-Hispanic ($p\text{-value} = 0.863$).

By knowing normality there will be supporting information on test selection, such as the decision to take a parametric approach to inferential analyses or a nonparametric approach. Often both approaches (parametric v nonparametric) for inferential test selection yield the same results at the broadest level of interpretation, yet it is still vital to know about distribution patterns to have assurance that eventual outcomes are founded on accepted statistical and research processes.

8.7 Statistical Test(s)

From the original dataset Hospital.df and then from HospitalAge021100.df, it may appear that there is a potentially limitless number of comparisons between and among the numeric object variables and the many breakout groups. In turn, these potential comparisons could support a vast number of inferential analyses and inquiries into associations. For the sake of brevity, this summary lesson will focus on two Null Hypotheses:

First Null Hypothesis (Ho): Mean Comparisons by Breakout Groups

Ho: There is no statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure by the seven different racial groups (e.g., American Indian or Alaska Native, Asian, Black or African American, Native Hawaiian or Other Pacific Islander, Other, Unknown, White) among those subjects who range in age from 40 to 49, inclusive.

Second Null Hypothesis (Ho): Test of Association

Ho: There is no statistically significant correlation ($p \leq 0.05$) between Systolic Blood Pressure and Weight among Black or African American Female subjects who range in age from 40 to 49, inclusive.

As a reminder, the dataset Hospital.df serves as a summary teaching dataset for this lesson and provides an opportunity for engagement with a complex

set of data, in terms of working with: (1) a dataset that has a large number of subjects, (2) a dataset that has a large number of object variables, both numeric object variables and factor object variables, and (3) a dataset that has a large number of potential breakout groups (e.g., factors) from among the different factor-type object variables.

8.7.1 Null Hypothesis 1: Analysis of Variance

8.7.1.1 Parametric Oneway ANOVA and Tukey HSD Approach

Given that there are two Null Hypotheses for this lesson, focus initially on the Null Hypothesis that calls for a comparison of a measure of blood pressure by different racial groups—for a specific age group:

Ho: There is no statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure by the seven different racial groups (e.g., American Indian or Alaska Native, Asian, Black or African American, Native Hawaiian or Other Pacific Islander, Other, Unknown, White) among those subjects who range in age from 40 to 49, inclusive.

Notice how the Null Hypothesis address only subjects who ranged in age from 40 years to 49 years on their last birthday, inclusive. By focusing only on subjects with similar ages it is assumed (whether this assumption is correct or not) that this age-based restriction will provide a focused view of differences in blood pressure measures between and among the different racial groups. That is to say, subjects as young as 21 will not be compared to subjects as old as 100, allowing for potentially more homogeneity in formation of the subjects selected for this analysis.

Create a New Dataset from an Existing Dataset

Hospital.df was used to create the dataset HospitalAge021100.df. Prepare a new dataset (HospitalAge040049.df) that consists of only those subjects who were between 40 years to 49 years on their last birthday, inclusive.

R Input

```
base::length(Hospital.df$Ticket)          # N
base::table(Hospital.df$AgeRangeCode)       # Age range
base::summary(Hospital.df$AgeLastBirthday)  # Summary

HospitalAge040049.df <- base::subset(Hospital.df,
  Hospital.df$AgeLastBirthday >= 40 &
  Hospital.df$AgeLastBirthday <= 49)
# HospitalAge040049.df will consist only of those
# subjects who range in age from 40 to 49 on
```

```
# their last birthday, inclusive.

base::attach(HospitalAge040049.df)          # Attach the data
utils::str(HospitalAge040049.df)             # Structure
base::length(HospitalAge040049.df$Ticket)    # N
```

R Output

```
[1] 3914
```

R Input

```
base::table(HospitalAge040049.df$AgeRangeCode)      # Age range
```

R Output

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	3914	0	0	0	0	0	0

R Input

```
base::summary(HospitalAge040049.df$AgeLastBirthday) # Summary
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
40.0	42.0	45.0	44.8	47.0	49.0

The six histograms that follow in the 2 row by 3 column grid reinforce the descriptive statistics of Diastolic Blood Pressure (DBP) measures for: (1) all subjects, (2) subjects between 21 and 100 years, inclusive, (3) subjects between 40 and 49 years, inclusive. To promote uniform presentation, look at the use of the xlim() argument and how it is consistent for DBP (0–140) and for AgeLastBirthday (0–120), to reinforce consistency in visual outcomes for all three datasets (Fig. 8.36).

R Input

```
par(ask=TRUE)
par(mfrow=c(2,3)) # 6 figures into a 2 row by 3 column grid
graphics::hist(Hospital.df$DBP, xlim=c(0,140), col="red")
```

```
graphics::hist(HospitalAge021100.df$DBP, xlim=c(0,140),
  col="red")
graphics::hist(HospitalAge040049.df$DBP, xlim=c(0,140),
  col="red")
graphics::hist(Hospital.df$AgeLastBirthday,
  xlim=c(0,120), col="red")
graphics::hist(HospitalAge021100.df$AgeLastBirthday,
  xlim=c(0,120), col="red")
graphics::hist(HospitalAge040049.df$AgeLastBirthday,
  xlim=c(0,120), col="red")
```

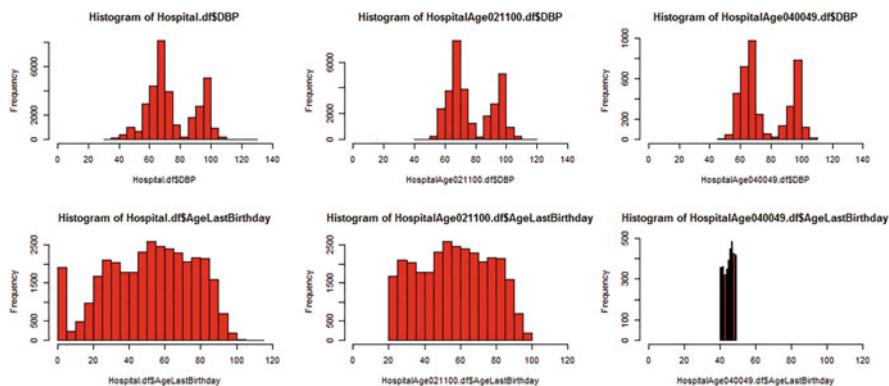


Figure 8.36: Distribution pattern of diastolic blood pressure and age with multiple subsets

A dataset of only those subjects with an age on their last birthday of 40–49, inclusive, provides a potentially homogeneous (e.g., similar) subset, at least in terms of age for this age-specific Null Hypothesis. By restricting AgeLastBirthday in HospitalAge040049.df to this select group, it is proposed, but unknown, that the subjects will be more alike and yield more useful (e.g., actionable) results. If all subjects in the dataset Hospital.df were included, the age-expected potential variance in DBP as well as SBP and Lbs would possibly cloud results for any inquiries into these three object variables. Assumptions on how age is associated with DBP, SBP, and Lbs must be tested using appropriate inferential analyses before these assumptions are accepted. It is not enough to merely speculate that subjects of similar age will have similar blood pressure (DBP and SBP) and weight (Lbs).

As a summary of how the process of enumeration is put into use, carefully review how the base::subset() function was used to create HospitalAge040049.df, a new dataframe in this R session that includes only those subjects who range in age from 40 to 49, inclusive. This outcome is confirmed using the base::length(), base::table(), and base::summary() functions.

Notice how `base::table(HospitalAge040049.df$AgeRangeCode)` yielded zero subjects in all but one `AgeRangeCode` classification, the age range for those subjects ranging in age from 40 to 49, inclusive. These quality assurance measures may at first seem redundant and perhaps even excessive, but they promote eventual correct outcomes and that is ultimately the purpose of these many statistical analyses.

Download the New File, HospitalAge040049.df

The dataset `HospitalAge040049.df` is currently available for use in the interactive R session. However, it might be useful to save `HospitalAge040049.df` to the working directory, for later use. To achieve this aim, use the `utils::write.csv()` function to save the dataset as a .csv (comma-separated values) file.

R Input

```
utils::write.csv(HospitalAge040049.df,  
file="HospitalAge040049.csv")
```

Make sure that each row is unique and that there are no duplicate cases (e.g., `Ticket`). It might be helpful to first confirm the number of cases (e.g., rows) for the object variable `HospitalAge040049.df$Ticket`.

R Input

```
base::length(HospitalAge040049.df$Ticket) # N
```

R Output

```
[1] 3914
```

Knowing the number of datapoints for `HospitalAge040049.df$Ticket` and seeing that the number is in the thousands ($N = 3914$), wrap the `base::table()` function around the `base::duplicated()` function since it is impossible to keep up with output as it shows on the screen. The `base::duplicated()` function will ideally confirm that there are no duplicated rows by showing all output as `FALSE`.

R Input

```
base::table(base::duplicated(HospitalAge040049.df$Ticket))  
# This set of actions should confirm that there are no  
# duplicate cases in the dataset HospitalAge040049.df.
```

R Output

```
FALSE  
3914
```

Before any attempt is made to use an inferential or associative test to address the Null Hypothesis, it is necessary to learn more about Diastolic Blood Pressure (DBP) for those subjects who were between 40 and 49 on their last birthday, inclusive. A limited series of selected graphical images and descriptive statistics will be used immediately below, but consult the many examples at the beginning of this lesson and in other lessons found in this text for a full set of options on these inquiries.

Graphical Images for HospitalAge040049.df\$DBP

As has been presented throughout these lessons, descriptive statistics alone are not enough to gain a complete understanding of data and the relationships between and among the data. Graphical images of many types complement descriptive statistics and should always be prepared, even if there is no desire to eventually share the figures with others. Look below at the different figures for HospitalAge040049.df\$DBP, overall, and later DBP by different racial breakout groups (HospitalAge040049.df\$Race.Recode). Notice how the Violin Plot, explained in more detail in a later part of this lesson, is used along with the more frequently used Box Plot (e.g., Box-and-Whisker Plot, Box-and-Whisker Diagram) (Figs. 8.37 and 8.38).

R Input

```
install.packages("vioplot")
library(vioplot)                      # Load the vioplot package.
help(package=vioplot)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

# Graphics of Diastolic Blood Pressure (DBP)
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(HospitalAge040049.df$DBP,
               main="Histogram of DBP, Age 040 to 049", col="red")
graphics::plot(stats::density(HospitalAge040049.df$DBP,
                               na.rm=TRUE), main="Density Curve of DBP, Age 040 to 049",
               col="red", lwd=3) # na.rm argument is required
vioplot::vioplot(HospitalAge040049.df$DBP,
                 names=c("DBP"), col="red")
title("Violin Plot of DBP, Age 040 to 049")
```

```
stats::qqnorm(HospitalAge040049.df$SBP,
  main="Q-Q Plot of DBP, Age 040 to 049", col="red")
```

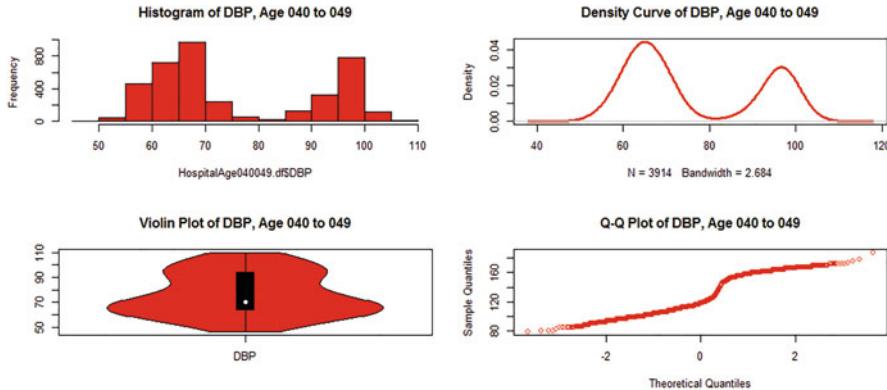


Figure 8.37: Distribution of diastolic blood pressure (age 040–049)—1

R Input

```
# Graphics of Diastolic Blood Pressure (DBP) by Race
par(ask=TRUE)
graphics::boxplot(HospitalAge040049.df$DBP ~
  HospitalAge040049.df$Race.Recode,
  main="Boxplot of Diastolic Blood Pressure by Race",
  xlab="Race",                                     # X axis label
  ylab="Diastolic Blood Pressure - DBP",          # Y axis label
  cex.axis=1.15,                                    # Axis size
  cex.lab=1.15,                                     # Label size
  col="red",                                         # Box color
  lwd=2,                                            # Line thickness
  font.lab=2,                                       # Bold labels
  font=2)                                           # Bold
```

R Input

```
install.packages("UsingR")
library(UsingR)                                     # Load the UsingR package.
help(package=UsingR)                                # Show the information page.
sessionInfo()                                       # Confirm all attached packages.

UsingR::simple.violinplot(HospitalAge040049.df$DBP ~
  HospitalAge040049.df$Race.Recode,
  font=2, lwd=2, col="red", ylim=c(0,120))
```

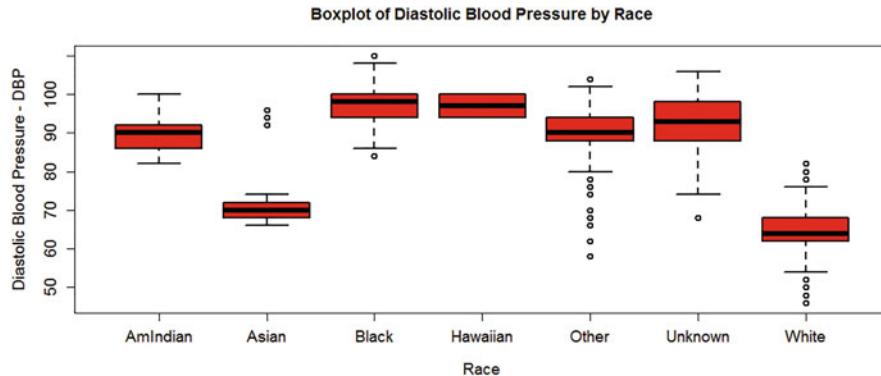


Figure 8.38: Distribution of diastolic blood pressure (age 040–049)—2

```
title("Violin Plot of Diastolic Blood Pressure by Race",
      xlab="Race", ylab="Diastolic Blood Pressure - DBP")
```

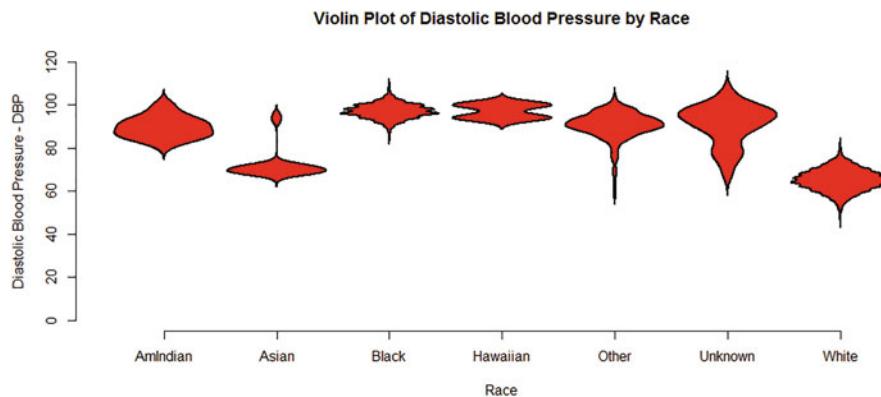


Figure 8.39: Distribution of diastolic blood pressure (age 040–049)—3

The breakout figures of Diastolic Blood Pressure by Race, using functions associated with the `graphics::boxplot()` function and the `UsingR::simple.violinplot()` function, should be viewed along with the descriptive statistics that will be generated in the next section when the `tables::tabular()` function is used. Using these breakout graphics only as a guide, there seems to be a major degree of difference in Diastolic Blood Pressure (DBP) for subjects who are Black and their counterparts who are White, but as always—an inferential test must be used to confirm if: (1) the difference is statistically significant at a reasonable degree of probability (typically $p \leq 0.05$), or (2) the difference is due only to chance (Fig. 8.39).

Descriptive Statistics for HospitalAge040049.df\$DBP

Graphical images provide a glimpse of outcomes and the relationship between and among object variables and their breakout groups. Even so, exact statistics

are needed to fully describe the data in its many forms, overall and by breakout groups. A variety of descriptive statistics about Diastolic Blood Pressure (DBP) follow, from among a much larger set of potential functions associated with descriptive statistics.

R Input

```
base::mean(HospitalAge040049.df$DBP, na.rm=TRUE)
# Mean and argument to account for missing data
```

R Output

```
[1] 76.2161
```

R Input

```
stats::sd(HospitalAge040049.df$DBP)
# Standard deviation
```

R Output

```
[1] 15.5988
```

R Input

```
stats::quantile(HospitalAge040049.df$DBP)
# Quantile scores, 0% 25% 50% 75% 100%
```

R Output

0%	25%	50%	75%	100%
46	64	70	94	110

R Input

```
stats::quantile(HospitalAge040049.df$DBP,
prob = seq(0, 1, length = 11), type = 5)
# Use of quantile() function to produce deciles,
# 0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
```

R Output

0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
46	60	62	64	66	70	72	92	96	98	110

The functions `base::mean()`, `stats::sd()`, etc., provide an understanding of the object variable `HospitalAge040049.df$DBP`, but only at the overall level of understanding for this object variable. A problem is that there is no granularity of `HospitalAge040049.df$DBP` descriptive statistics for the different breakouts of Race, or more specifically `HospitalAge040049.df$Race.Recode`. To achieve that level of understanding, the `tables::tabular()` function can be used to provide detail at the breakout level for factor-type object variables, with presentation in a neatly organized table format that can be easily copied and pasted later into a word-processed document.

R Input

```
install.packages("tables")
library(tables)                                # Load the tables package.
help(package=tables)                            # Show the information page.
sessionInfo()                                   # Confirm all attached packages.

tables::tabular((Race.Recode + 1) ~ (n=1) + Format(digits=2)*
  (DBP)*(min + max + median + mean + sd),
  data=HospitalAge040049.df)
# Race.Recode (row) by DBP (columns)
```

R Output

		DBP				
Race.Recode	n	min	max	median	mean	sd
AmIndian	13	82.0	100.0	90.0	89.7	5.1
Asian	34	66.0	96.0	70.0	71.9	7.2
Black	1143	84.0	110.0	98.0	97.1	3.4
Hawaiian	2	94.0	100.0	97.0	97.0	4.2
Other	234	58.0	104.0	90.0	90.3	6.6
Unknown	26	68.0	106.0	93.0	90.8	9.3
White	2462	46.0	82.0	64.0	65.0	5.0
All	3914	46.0	110.0	70.0	76.2	15.6

The breakout descriptive statistics gained from use of the `tables::tabular()` function provide extremely valuable information about general trends for Diastolic Blood Pressure (DBP) by the different racial groups. As an example, observe

how Mean DBP ranges from 97.1 for Black subjects ($N = 1143$) to 65.0 for White subjects ($N = 2462$). An appropriate inferential test is necessary, however, to determine if there is a statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure (DBP) for these two breakout groups, as well as statistically significant differences in Diastolic Blood Pressure (DBP) between and among the seven racial groups. Until the appropriate inferential tests are attempted, these breakout descriptive statistics certainly offer a glimpse, but only a glimpse, of general trends and expectations.

It is cautioned, however, to always look carefully at the number of subjects for each breakout group. For this part of the lesson, notice how there are fairly low numbers for American Indian ($N = 13$), Asian ($N = 34$), Hawaiian ($N = 2$), and Unknown ($N = 26$), compared to overall ($N \text{ All} = 3914$). This finding of how the dataset is populated is generally reflective of the community represented by the original dataset (`Hospital.df`). Although it is beyond the immediate purpose of this lesson, these low N s, compared to the overall N , provide a suggestion that a nonparametric approach may be warranted when selecting the inferential test that will be used to determine statistical significance by Race for Diastolic Blood Pressure.

Determination of Normal Distribution

The Anderson–Darling Test is commonly used to look into normal distribution and from that finding to determine if either a parametric or nonparametric inferential test should be used to determine statistical significance.

R Input

```
nortest::ad.test(HospitalAge040049.df$SBP)
# Anderson-Darling Test for Normality Null Hypothesis:
# The data follow the normal distribution.
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

The p-value associated with this application of the Anderson–Darling Test for Normality is less than the p-value of 0.05 and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable do not follow the normal distribution. Look at the figure associated with the `stats::qqnorm()` function and the `stats::qqline()` function, resulting in visual reinforcement that DBP values deviate away from normal distribution (Fig. 8.40).

R Input

```
par(ask=TRUE)
stats::qqnorm(HospitalAge040049.df$DBP,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Diastolic
Blood Pressure - Age 040 to 049, Inclusive",
  col="blue", xlim=c(-4,4), ylim=c(0,120), font.axis=2,
  font.lab=2)
stats::qqline(HospitalAge040049.df$DBP,
  col="red", lwd=4, lty=2)
```

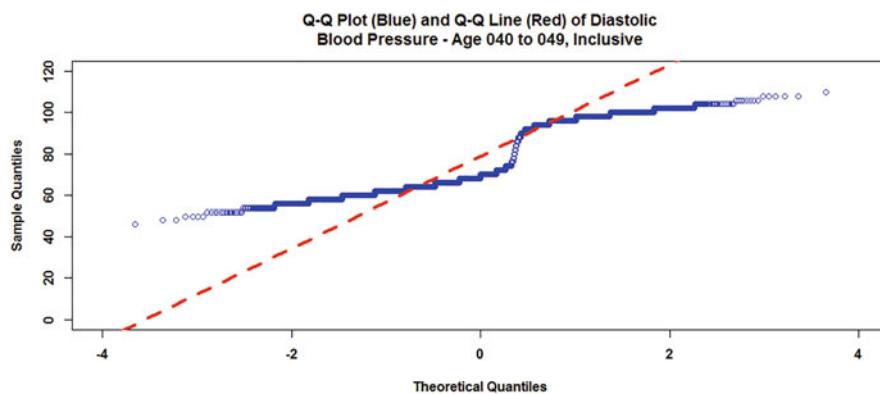


Figure 8.40: Distribution of diastolic blood pressure (age 040–049)—4

Observe the extreme deviation of the Q-Q line in comparison to the Q-Q plot. This figure reinforces the Anderson–Darling Test for Normality finding that the data associated with HospitalAge040049.df\$DBP do not exhibit normal distribution. This finding provides another degree of support for a nonparametric approach as an appropriate test for investigation into this Null Hypothesis.

However, what is the degree of normal distribution for Diastolic Blood Pressure by the seven Race breakout groups? Continuing along with use of the Anderson–Darling Test for Normality, notice the calculated normality statistics for DBP by Race.Recode. Then, compare these statistics to a graphical representation of normality using the QQ plot capabilities of the ggplot2::ggplot() function. As a reminder, remember to consider the issue of sample size, which for DBP is perhaps most easily displayed simply by using the base::summary() function.

R Input

```
nortest::ad.test(subset(HospitalAge040049.df$DBP,
  HospitalAge040049.df$Race.Recode == "AmIndian"))
```

R Output

```
Outcome: p-value = 0.811
```

R Input

```
nortest::ad.test(subset(HospitalAge040049.df$DBP,  
HospitalAge040049.df$Race.Recode == "Asian"))
```

R Output

```
Outcome: p-value = 0.00000000000000176
```

R Input

```
nortest::ad.test(subset(HospitalAge040049.df$DBP,  
HospitalAge040049.df$Race.Recode == "Black"))
```

R Output

```
Outcome: p-value <0.0000000000000002
```

R Input

```
nortest::ad.test(subset(HospitalAge040049.df$DBP,  
HospitalAge040049.df$Race.Recode == "Hawaiian"))
```

R Output

```
Error: sample size must be greater than 7
```

R Input

```
nortest::ad.test(subset(HospitalAge040049.df$DBP,  
HospitalAge040049.df$Race.Recode == "Other"))
```

R Output

```
Outcome: p-value = 0.00000000000000562
```

R Input

```
nortest::ad.test(subset(HospitalAge040049.df$DBP,
HospitalAge040049.df$Race.Recode == "Unknown"))
```

R Output

```
Outcome: p-value = 0.0296
```

R Input

```
nortest::ad.test(subset(HospitalAge040049.df$DBP,
HospitalAge040049.df$Race.Recode == "White"))
```

R Output

```
Outcome: p-value <0.0000000000000002
```

R Input

```
base::summary(HospitalAge040049.df$Race.Recode)
```

R Output

AmIndian	Asian	Black	Hawaiian	Other	Unknown	White
13	34	1143	2	234	26	2462

The ggplot2 package may provide the best way to visually address this question. Install the ggplot2 package and the many associated packages that work in concert with the ggplot2 package.¹²

R Input

```
install.packages("ggplot2", dependencies=TRUE)
library(ggplot2) # Load the ggplot2 package.
help(package=ggplot2) # Show the information page.
sessionInfo() # Confirm all attached packages.

install.packages("ggthemes", dependencies=TRUE)
```

¹²The ggplot2 package is part of tidyverse and it is also obtained when the tidyverse collection of R packages is installed.

```

library(ggthemes)           # Load the ggthemes package.
help(package=ggthemes)      # Show the information page.
sessionInfo()               # Confirm all attached packages.

install.packages("ggmosaic", dependencies=TRUE)
library(ggmosaic)           # Load the ggmosaic package.
help(package=ggmosaic)      # Show the information page.
sessionInfo()               # Confirm all attached packages.

install.packages("gridExtra", dependencies=TRUE)
library(gridExtra)           # Load the gridExtra package.
help(package=gridExtra)      # Show the information page.
sessionInfo()               # Confirm all attached packages.

install.packages("grid", dependencies=TRUE)
library(grid)                 # Load the grid package.
help(package=grid)           # Show the information page.
sessionInfo()               # Confirm all attached packages.

install.packages("scales", dependencies=TRUE)
library(scales)               # Load the scales package.
help(package=scales)         # Show the information page.
sessionInfo()               # Confirm all attached packages.

```

R Input

```

ggplot2::ggplot(HospitalAge040049.df,
aes(sample=DBP)) +
  stat_qq(color="red") +
  stat_qq_line(color="blue", size=1.75) +
  facet_grid(. ~ Race.Recode) +
  ggtitle(
  "QQ Plot of DBP by Race.Recode Breakouts for Subjects
Who Ranged in Age from 40 to 49 Years, Inclusive") +
  theme_bw()

```

Reviewing the QQ plots and the Anderson–Darling statistic, it seems that overall and for some breakout groups, there is a concern about normal distribution of DBP among subjects who are 40–49 years old, inclusive by the different racial groups. With this concern, it may be best to not only attempt a parametric approach to the Null Hypothesis but to then attempt a confirming nonparametric approach to the Null Hypothesis (Fig. 8.41).

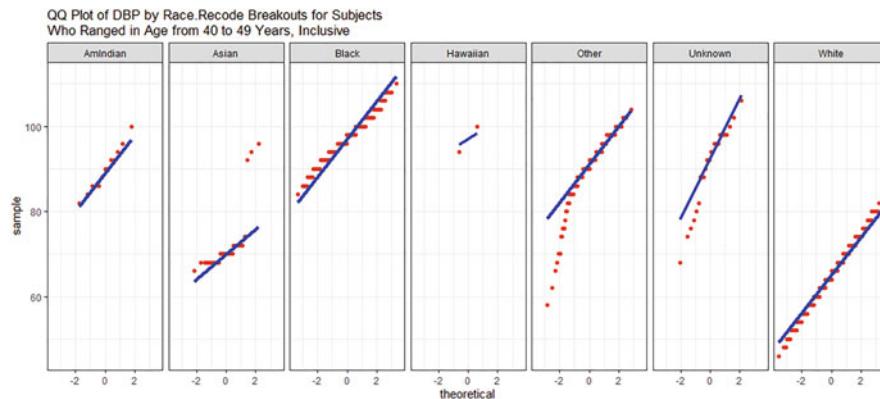


Figure 8.41: Distribution of diastolic blood pressure (age 040–049) and race—1

8.7.1.2 Parametric Oneway ANOVA and Tukey HSD Approach

A first attempt at Oneway ANOVA often begins with the `stats::oneway.test()` function. Consider the `stats::oneway.test()` function an exploratory tool for Oneway ANOVA since the output is fairly limited in terms of broad use if a statistically significant difference ($p \leq 0.05$) is determined. The `stats::oneway.test()` function will identify if there is a statistically significant difference ($p \leq 0.05$) in DBP means by Race.Recode breakout groups, but little else is provided.

R Input

```
stats::oneway.test(DBP ~ Race.Recode,
  data = HospitalAge040049.df,
  var.equal=FALSE) # Assume variance is not equal
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

R Input

```
stats::oneway.test(DBP ~ Race.Recode,
  data = HospitalAge040049.df,
  var.equal=TRUE) # Assume variance is equal
```

R Output

```
p-value < 0.0000000000000002
```

Both applications of the `stats:::oneway.test()` function (e.g., unequal variance and equal variance) provide evidence that there is a statistically significant difference ($p \leq 0.05$) in DBP by Race.Recode breakout groups. That is to say, the calculated p-value is less than the criterion p-value of 0.05.

Even with this finding, the `stats:::oneway.test()` function leaves unanswered questions—which comparisons of DBP by Race.Recode represent common means, if any, and which comparisons of DBP by Race.Recode represent statistically significant different means, if any ($p \leq 0.05$)? From among many possible choices, the `agricolae::HSD.test()` function is quite efficient and fairly easy to use in an attempt to examine Oneway ANOVA outcomes and obtain actionable outcomes. Similar to a Oneway ANOVA approach for all strategies, regardless of the selected functions, the focus is on determining: (1) DBP means for each of the multiple Race.Recode breakout groups, (2) which DBP by Race.Recode breakout group means is in common and, (3) which DBP by Race.Recode breakout group means is different.

R Input

```
install.packages("agricolae", dependencies=TRUE)
library(agricolae)           # Load the agricolae package.
help(package=agricolae)      # Show the information page.
sessionInfo()                # Confirm all attached packages.

agricolae::tapply.stat(HospitalAge040049.df$DBP,
                      HospitalAge040049.df$Race.Recode,
                      function(x) mean(x, na.rm=TRUE))
# Confirm means of DBP by the Race.Recode breakout groups.
# Accommodate missing values, if needed.
```

R Output

	HospitalAge040049.df\$Race.Recode	HospitalAge040049.df\$DBP
1	AmIndian	89.6923
2	Asian	71.9412
3	Black	97.0866
4	Hawaiian	97.0000
5	Other	90.2735
6	Unknown	90.7692
7	White	65.0081

Use Tukey's HSD (Honestly Significant Difference) test to identify DBP means of Race.Recode that are significantly different from each other and in turn those DBP means of Race.Recode that are in common. To achieve this aim, use the agricolae::HSD.test() function, wrapped around the aov() function).

R Input

```
agricolae::HSD.test(
  aov(DBP ~ Race.Recode, data=HospitalAge040049.df),# Model
  trt="Race.Recode",                                     # Treatment
  group=TRUE, console=TRUE, alpha=0.05,                 # Arguments
  main="Diastolic Blood Pressure by Race Breakout Groups")
# Wrap the agricolae::HSD.test() function around the
# Oneway ANOVA model obtained by using the aov()
# function. Select desired arguments, such as group,
# console, and alpha (e.g., p-value).
```

R Output

Study: Diastolic Blood Pressure by Race Breakout Groups

HSD Test for DBP

Mean Square Error: 22.8779

Race.Recode, means

	DBP	std	r	Min	Max
AmIndian	89.6923	5.08895	13	82	100
Asian	71.9412	7.19403	34	66	96
Black	97.0866	3.40372	1143	84	110
Hawaiian	97.0000	4.24264	2	94	100
Other	90.2735	6.56382	234	58	104
Unknown	90.7692	9.34797	26	68	106
White	65.0081	5.01496	2462	46	82

Alpha: 0.05 ; DF Error: 3907

Critical Value of Studentized Range: 4.17173

Groups according to probability of means differences and
alpha level(0.05)

Treatments with the same letter are not significantly different.

	DBP groups
Black	97.0866

Hawaiian	97.0000	ab
Unknown	90.7692	b
Other	90.2735	b
AmIndian	89.6923	b
Asian	71.9412	c
White	65.0081	d

The agricolae::HSD.test() function provides a very convenient summary of the information needed to make an informed interpretation of Oneway ANOVA outcomes:

- Descriptive statistics for each breakout group are provided, including the mean, standard deviation, number, minimum, and maximum.
- Lowercase letters are used to identify common groups and by extension groups that are not in common. In this example, the Diastolic Blood Pressure (DBP) means that are in common are:¹³
 - Group a: Black (Mean DBP = 97.0866) and Hawaiian (Mean DBP = 97.0000)
 - Group b: Hawaiian (Mean DBP = 97.0000), Unknown (Mean = 90.7692), Other (Mean = 90.2735), and AmIndian (Mean = 89.6923)
 - Group c: Asian (Mean DBP = 71.9412)
 - Group d: White (Mean DBP = 65.0081)

The agricolae::HSD.test() function provides the mean comparison information needed to determine where there are statistically significant ($p \leq 0.05$) mean differences in Diastolic Blood Pressure by breakout groups for Race. Again, this comparison is made from a parametric perspective even with the known challenges (e.g., low Ns and lack of normal distribution) for these data, if not overall at least for some breakout groups of Race.

Prepare a violin plot to reinforce these findings of Race.Recode breakout group means of Diastolic Blood Pressure. Because there were only two Hawaiian subjects in this analysis it is likely that their representation in the violin plot will show poorly if at all.

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge040049.df,
aes(x=Race.Recode, y=DBP, fill=Race.Recode)) +
geom_violin() +
coord_flip()
```

¹³Observe the overlap in group commonalities, specifically for Group a and Group b.

```

# Flip the X and Y axis, which is often useful for when there
# are many boxplots in the final output -- 7 in this figure.
stat_summary(fun=mean, geom="point", shape=1, size=4,
  col="black") +
# Add a circle to represent the mean.
scale_fill_manual(values=c("aliceblue", "khaki", "lightblue",
  "orange", "cyan", "pink", "bisque")) +
# Manually declare desired colors to a lighter shade so that
# the mean, as a circle, is easily seen.
ggttitle(
"Subject (Age >= 40 and Age <= 49) Diastolic Blood Pressure by
Race: Violin Plot With Superimposed Mean as a Circle") +
xlab("Race\n") +
ylab("\nDiastolic Blood Pressure") +
scale_y_continuous(labels=scales::comma, limits=c(0,120),
  breaks=scales::pretty_breaks(n = 10)) +
# The scale for DBP is purposely set to start at 0, to show
# the full potential range.
theme_economist(base_size=12) +
theme(legend.position="none")

```

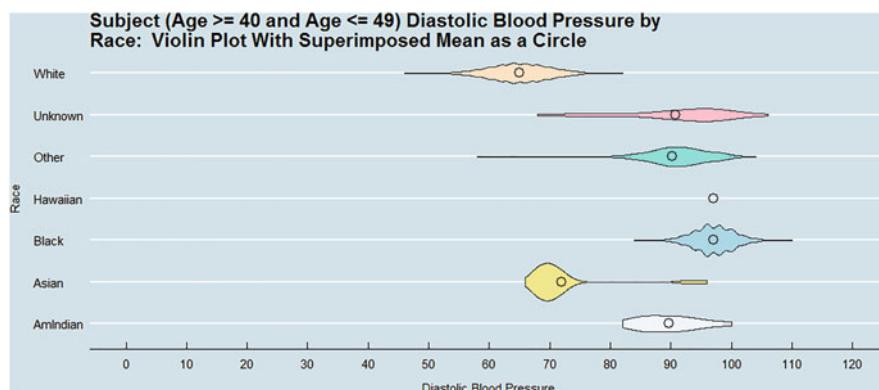


Figure 8.42: Distribution of diastolic blood pressure (age 040–049) and race—2

As presented throughout this lesson and other lessons, the graphical output that can be achieved with R should be viewed as a complement to statistical output only. Graphics help reinforce outcomes for researchers and are essential for many (Fig. 8.42).

8.7.1.3 Nonparametric Kruskal–Wallis Approach

Brief Background on Nonparametric Statistics

Scalable precision, expected distribution patterns, and uniform representation may be desired among those who work in the biological sciences, but these de-

sires are rarely achieved to full satisfaction at all levels of a biologically-focused research endeavor. To address these limitations, especially when the impact of these limitations cannot be ignored, experienced researchers use nonparametric approaches to statistical analysis.

Nonparametric statistics are a subset of inferential analyses that is used when: (1) the precision of an interval scale is not met, (2) extreme deviation from normal distribution cannot be ignored, and (3) the difference in the number of subjects for breakout groups is considerable. Along with the term(s) nonparametric analyses and nonparametric statistics, terms such as ranking tests and distribution-free tests are also used when using a nonparametric approach for statistical analysis.

With the breakout statistics (especially Median) gained from use of the tables::tabular() function as a guide, it is a fairly easy transition to use the kruskal.test() function to gain a more complete sense of the Kruskal–Wallis Test as it is applied against Diastolic Blood Pressure and the different breakouts for Race.

R Input

```
tables::tabular((Race.Recode + 1) ~ (n=1) + Format(digits=2)*
  (DBP)*(median), data=HospitalAge040049.df)      # Median only
# Race.Recode (row) by DBP (columns)
```

R Output

		DBP
Race.Recode	n	median
AmIndian	13	90
Asian	34	70
Black	1143	98
Hawaiian	2	97
Other	234	90
Unknown	26	93
White	2462	64
All	3914	70

R Input

```
stats::kruskal.test(DBP ~ Race.Recode,
  data=HospitalAge040049.df)
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

As evident from application of the `kruskal.test()` function, the p-value is less than the criterion $p \leq 0.05$. From this one test, at the overall level of comparison, it is evident that there are statistically significant differences in DBP values between the different breakouts for Race—based on the Kruskal–Wallis (Rank Sum) Test. But, where is the difference? It would be reasonable to suggest that there is a statistically significant difference ($p \leq 0.05$) between Black (Median = 98, the Median with the greatest value) subjects and White (Median = 64, the Median with the smallest value) subjects, but it would only be unverified conjecture to speculate about comparisons for other breakout groups for Race. Quite simply, the `stats::kruskal.test()` function does not support that type of detailed comparison, but this challenge can be met by the use of other R-based functions.

Although the `stats::kruskal.test()` provides evidence that there are statistically significant differences in Diastolic Blood Pressure (DBP) by Race, as an added quality assurance measure, it is a good idea to use a different R-based function specific to the Kruskal–Wallis Test. To use a popular expression, *Trust, but verify.*, to see if results are consistent. This type of action may not be necessary, but attention to these seemingly redundant quality assurance measures provides confirmation that outcomes are consistent and ostensibly correct. For this additional action, use the `agricolae::kruskal()` function to prepare a comparative test of the many breakout comparisons of Diastolic Blood Pressure (DBP) by the breakouts for Race.¹⁴

R Input

```
agricolae::kruskal(HospitalAge040049.df$DBP,
  HospitalAge040049.df$Race.Recode,
  alpha=0.05, group=FALSE, p.adj="holm",
  main="Kruskal-Wallis Using agricolae::kruskal() Function",
  console=TRUE)
# Use holm for pairwise comparisons. Another choice could
# have been to use bonferroni for pairwise comparisons.
```

¹⁴The `muStat::mu.kruskal.test()` function and the `pgirmess::kruskalmc()` function have many useful features and their use for a Kruskal–Wallis test should also be explored.

R Output

Study: Kruskal-Wallis Using agricolae::kruskal() Function

Kruskal-Wallis test's

Ties or no Ties

Critical Value: 2762.32

Degrees of freedom: 6

Pvalue Chisq : 0

HospitalAge040049.df\$Race.Recode, means of the ranks

	HospitalAge040049.df.DBP	r
AmIndian	2747.27	13
Asian	2102.12	34
Black	3296.87	1143
Hawaiian	3289.25	2
Other	2774.72	234
Unknown	2935.08	26
White	1240.45	2462

Post Hoc Analysis

P value adjustment method: holm

Comparison between treatments mean of the ranks.

	Difference	pvalue	Signif.
AmIndian - Asian	645.15158	0.0135	*
AmIndian - Black	-549.59735	0.0135	*
AmIndian - Hawaiian	-541.98077	1.0000	
AmIndian - Other	-27.44658	1.0000	
AmIndian - Unknown	-187.80769	1.0000	
AmIndian - White	1506.82406	0.0000	***
Asian - Black	-1194.74893	0.0000	***
Asian - Hawaiian	-1187.13235	0.0615	.
Asian - Other	-672.59816	0.0000	***
Asian - Unknown	-832.95928	0.0000	***
Asian - White	861.67248	0.0000	***
Black - Hawaiian	7.61658	1.0000	
Black - Other	522.15077	0.0000	***
Black - Unknown	361.78966	0.0259	*
Black - White	2056.42141	0.0000	***
Hawaiian - Other	514.53419	1.0000	

Hawaiian - Unknown	354.17308	1.0000	
Hawaiian - White	2048.80483	0.0000	***
Other - Unknown	-160.36111	1.0000	
Other - White	1534.27065	0.0000	***
Unknown - White	1694.63176	0.0000	***

An interesting outcome of analyses associated with this Null Hypothesis is the exceptional degree of parity in groupwise comparison outcomes whether the data were analyzed using either a nonparametric perspective or a parametric perspective. From the 21 groupwise comparisons, there is only one group comparison (e.g., Asian–Hawaiian) where the nonparametric determination of significance at $p \leq 0.05$ was different from the parametric determination of significance at $p \leq 0.05$. What is interesting about this specific groupwise comparison is that: (1) it is suspected that low Ns may have an impact on group-specific outcomes, and (2) the calculated nonparametric p was 0.0615, which begins to approximate the criterion $p \leq 0.05$. Otherwise, there was no effective difference in how group comparison outcomes should be interpreted whether using a nonparametric approach or a parametric approach to this Null Hypothesis.

Nonparametric and Parametric Summary of Groupwise Comparisons to the Null Hypothesis

	Nonparametric Kruskal-Wallis agricolae::kruskal()	Parametric Oneway ANOVA stats::lm, stats::anova(), and stats:: TukeyHSD()
		p
		$p \leq .05$
AmIndian - Asian	0.0135	Yes
AmIndian - Black	0.0135	Yes
AmIndian - Hawaiian	1.0000	No
AmIndian - Other	1.0000	No
AmIndian - Unknown	1.0000	No
AmIndian - White	0.0000	Yes
Asian - Black	0.0000	Yes
Asian - Hawaiian	0.0615	No
Asian - Other	0.0000	Yes
Asian - Unknown	0.0000	Yes
Asian - White	0.0000	Yes
Black - Hawaiian	1.0000	No
Black - Other	0.0000	Yes
Black - Unknown	0.0259	Yes
Black - White	0.0000	Yes
Hawaiian - Other	1.0000	No
		$p \leq 0.05$
		Asian-AmIndian
		Black-AmIndian
		Hawaiian-AmIndian
		Other-AmIndian
		Unknown-AmIndian
		White-AmIndian
		Black-Asian
		Hawaiian-Asian
		Other-Asian
		Unknown-Asian
		White-Asian
		Hawaiian-Black
		Other-Black
		Unknown-Black
		White-Black
		Other-Hawaiian

Hawaiian - Unknown	1.0000	No	Unknown-Hawaiian	0.5649837	No
Hawaiian - White	0.0000	Yes	White-Hawaiian	0.0000000	Yes
Other - Unknown	1.0000	No	Unknown-Other	0.9988441	No
Other - White	0.0000	Yes	White-Other	0.0000000	Yes
Unknown - White	0.0000	Yes	White-Unknown	0.0000000	Yes

Summary to the Analysis of Variance Null Hypothesis

In this lesson, the graphics and statistics provided a great deal of information about the many health-related object variables associated with the original dataset `Hospital.df`. Of immediate importance, however, focus on the Null Hypothesis statement and the calculated p-value in comparison to the criterion p-value:

First Null Hypothesis (H_0): Mean Comparisons by Breakout Groups

H_0 : There is no statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure by the seven different racial groups (e.g., American Indian or Alaska Native, Asian, Black or African American, Native Hawaiian or Other Pacific Islander, Other, Unknown, White) among those subjects who range in age from 40 to 49, inclusive.

The Null Hypothesis is rejected (e.g., not accepted) and instead it was determined that there is a statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure by Race. The most noticeable difference is the difference in Diastolic Blood Pressure (DBP) between White subjects (Median White DBP = 64 and Mean White DBP = 65.0) and Black subjects (Median Black DBP = 98 and Mean Black DBP = 97.1).

Going beyond this one comparison of the greatest magnitude of difference, it is equally noticed that there was a statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure (DBP) between White subjects and subjects from all other Race-specific breakout groups. In addition, selected pairwise comparisons for other breakout groups also exhibited statistically significant ($p \leq 0.05$) difference and these differences are clearly detailed in this lesson, whether using a nonparametric approach (i.e., focus on medians) or parametric approach (e.g., focus on means) to data analysis.

8.7.2 Null Hypothesis 2: Correlation—Association

The first Null Hypothesis for this lesson focused on Diastolic Blood Pressure by Race and to achieve some degree of homogeneity among those selected for analysis, subjects were restricted by age—including only those subjects who ranged in age on the last birthday from 40 to 49 years, inclusive. The second Null Hypothesis looks again at blood pressure, but now focusing on the correlation (e.g., association) between Systolic Blood Pressure and Weight. Along with restricting the subjects to a similar age range, even greater homogeneity

is desired and to achieve that aim subjects will include only those subjects who (1) meet the 40–49 age restriction, (2) are Black or African American, (3) and are female. These restrictions should ostensibly allow guided focus on Systolic Blood Pressure while mitigating possible disparate outcomes due to age, race, and sex.

Ho: There is no statistically significant correlation ($p \leq 0.05$) between Systolic Blood Pressure and Weight among Black or African American Female subjects who range in age from 40 to 49, inclusive.

The Null Hypothesis addresses only subjects who: (1) ranged in age from 40 years to 49 years on their last birthday, inclusive, (2) are Black or African American, and (3) are female. By focusing only on subjects with so many similarities it is assumed (whether this assumption is correct or not) that this homogeneous group will provide a more useful view of the association between blood pressure and weight, or more specifically SBP (Systolic Blood Pressure) and Lbs (Weight). It is further assumed that any results from inquiry into this Null Hypothesis can be extrapolated to the general population and that usefulness of the results are not restricted to only those subjects who meet the age, race, and sex criteria associated with the upcoming selection process. But of course, that assumption must be tested and this Null Hypothesis is an ideal starting point (but only a starting point) for that task.

Create a New Dataset from an Existing Dataset

Similar to what was seen earlier when the dataset Hospital.df was used to create the datasets HospitalAge021100.df and HospitalAge040049.df, prepare a new dataset that consists of only those subjects who were between 40 years to 49 years on their last birthday, inclusive, are Black or African American, and are female.¹⁵

R Input

```
base::length(Hospital.df$Ticket)           # N
base::table(Hospital.df$AgeRangeCode)        # Age range
base::summary(Hospital.df$AgeLastBirthday)   # Summary

HospitalAge040049RaceBlackSexFemale.df <-
  subset(Hospital.df,
  (Hospital.df$AgeLastBirthday >= 40 &
  Hospital.df$AgeLastBirthday <= 49) &
  (Hospital.df$Race.Recode == "Black") &
```

¹⁵The dataframe HospitalAge040049RaceBlackSexFemale.df exhibits a degree of self-documentation. There is no mistake that the data involve subjects who range in age from 40 to 49, are Black, are female, and that there is a hospital component to the data. The dataframe name may be verbose, but it is also easy to track in terms of constituent members.

```
(Hospital.df$Sex.Recode == "Female"))
# HospitalAge040049RaceBlackSexFemale.df will consist only of
# those subjects who (1) range in age from 40 to 49 on their
# last birthday, inclusive, (2) are Black, and (3) are
# Female. Typical to any Boolean-type comparison for
# equality when using R, look at the way == and not = is used
# for comparison.

base:::attach(HospitalAge040049RaceBlackSexFemale.df)    # Attach
utils:::str(HospitalAge040049RaceBlackSexFemale.df)
# Attach the data and then display the structure

base:::length(
  HospitalAge040049RaceBlackSexFemale.df$Ticket)           # N
```

R Output

```
[1] 671
```

R Input

```
base:::table(
  HospitalAge040049RaceBlackSexFemale.df$AgeRangeCode) # Age(s)
```

R Output

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	671	0	0	0	0	0	0

R Input

```
base:::table(
  HospitalAge040049RaceBlackSexFemale.df$Race.Recode) # Race
```

R Output

AmIndian	Asian	Black	Hawaiian	Other	Unknown	White
0	0	671	0	0	0	0

R Input

```
base::table(
  HospitalAge040049RaceBlackSexFemale.df$Sex.Recode) # Sex
```

R Output

Female	Male
671	0

R Input

```
base::summary(
  HospitalAge040049RaceBlackSexFemale.df) # All
# Use these quality assurance actions to be sure that only
# those subjects who meet the selection criteria are included
# in this new enumerated dataset.
```

R Output

[Selected output is not shown, to save space.]

AgeRange.Recode	Race.Recode	Sex.Recode	SBP
40-49 : 671	AmIndian: 0	Female: 671	Min. : 130
Birth-09: 0	Asian : 0	Male : 0	1st Qu.: 154
10-20 : 0	Black : 671		Median : 158
21-29 : 0	Hawaiian: 0		Mean : 158
30-39 : 0	Other : 0		3rd Qu.: 162
50-59 : 0	Unknown : 0		Max. : 170
(Other) : 0	White : 0		

As a further QA check, review the original dataset (`Hospital.df`) associated with this lesson and the three datasets that are now subsets of the original dataset: `HospitalAge021100.df` is the original dataset, `HospitalAge040049.df` was then created from the original dataset, and the eventual highly-filtered dataset `HospitalAge040049RaceBlackSexFemale.df`. Focus on N, using the `base::length()` function.

R Input

```
base::length(Hospital.df$Ticket) # N
```

R Output

```
[1] 33859
```

R Input

```
base::length(HospitalAge021100.df$Ticket) # N
```

R Output

```
[1] 30227
```

R Input

```
base::length(HospitalAge040049.df$Ticket) # N
```

R Output

```
[1] 3914
```

R Input

```
base::length(HospitalAge040049RaceBlackSexFemale.df$Ticket) # N
```

R Output

```
[1] 671
```

The four histograms showing in the 2 row by 2 column grid that follows provide a comparative advance organizer for later descriptive statistics of Systolic Blood Pressure (SBP) measures for: (1) all subjects, (2) for subjects between 21 and 100 years, inclusive, (3) for subjects between 40 and 49 years, inclusive, (4) and for subjects between 40 and 49 years, inclusive, who are Black or African American, and who are Female. To promote uniform presentation, look at use of the `xlim` argument and how it is consistent for SBP (0–200), to reinforce consistency in visual outcomes for all datasets and corresponding figures.¹⁶

¹⁶The `ylim` argument was not used, due to the wide disparity in frequency counts as the dataframe Ns (e.g., `length`) rapidly decline as a result of the increasingly restrictive selection criteria.

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(Hospital.df$SBP, xlim=c(0,200), col="red")
graphics::hist(HospitalAge021100.df$SBP, xlim=c(0,200),
               col="red")
graphics::hist(HospitalAge040049.df$SBP, xlim=c(0,200),
               col="red")
graphics::hist(HospitalAge040049RaceBlackSexFemale.df$SBP,
               xlim=c(0,200), col="red")
```

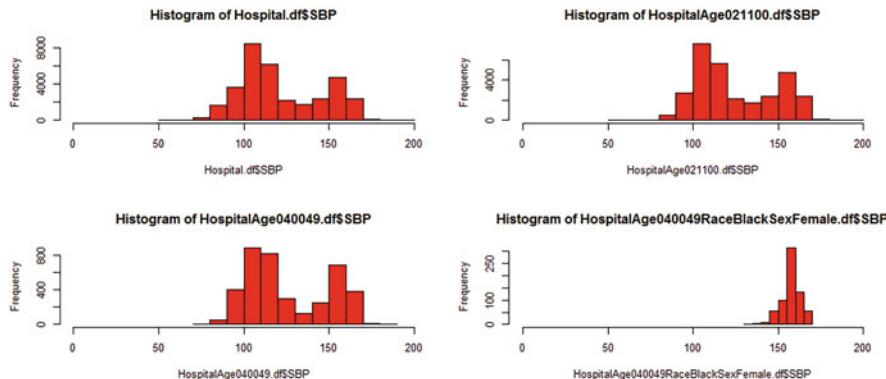


Figure 8.43: Distribution of systolic blood pressure by increasingly restrictive subsets—1

An additional 2 row by 2 column grid is also shown, focusing on the object variable Lbs (Weight)—the other object variable of interest in this inquiry of association between blood pressure and weight, if any. Similar to what was seen before, note the progression of weights as the different datasets are addressed and graphically presented (Figs. 8.43 and 8.44).

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(Hospital.df$Lbs, xlim=c(0,300), col="red")
graphics::hist(HospitalAge021100.df$Lbs, xlim=c(0,300),
               col="red")
graphics::hist(HospitalAge040049.df$Lbs, xlim=c(0,300),
               col="red")
graphics::hist(HospitalAge040049RaceBlackSexFemale.df$Lbs,
               xlim=c(0,300), col="red")
# Give attention to the Y axis and the declining N.
```

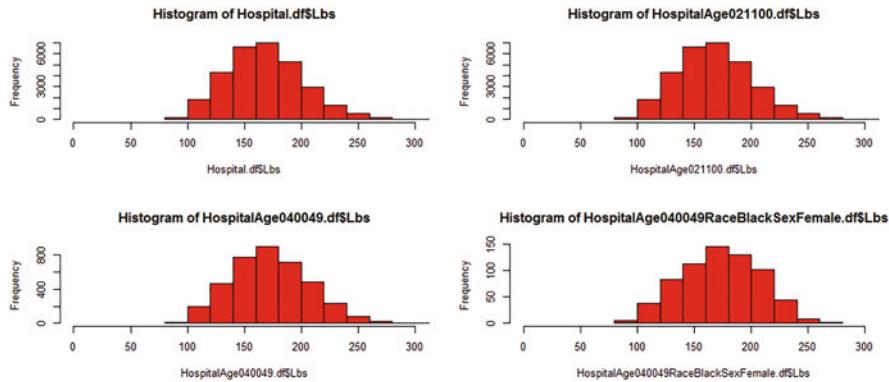


Figure 8.44: Distribution of weight by increasingly restrictive subsets—1

This highly-restricted dataset (e.g., restricted by age, race, and sex) serves as an excellent example of how existing data can and should be configured in multiple ways to learn as much as is practical about subjects, overall and by focused breakouts. Rarely do researchers inquire about subjects only at the broadest level. Broad level inquiries are important and certainly should be attempted, but *Big Data* are typically examined from multiple perspectives (e.g., look into the slang expression *sliced-and-diced* in reference to data) to achieve practical uses from the data. A treatment of some type, whether it is an administered drug, change in diet, temperature change, change in bedding material, etc., may not show significance at the overall level of examination, but breakout groups may show different results and in turn respond differently than large group norms—often with practical (e.g., actionable results with either clinical or financial implications) treatments justified due to statistical examination of these subgroups.

As a brief reminder of how the process of enumeration and creation of a new dataset from an existing dataset is put into use, carefully review how the `base::subset()` function was used to create `HospitalAge040049RaceBlackSexFemale.df`, a new dataframe in this R session that includes only those subjects who share a similar age range (40–49 on their last birthday, inclusive) race (Black or African American), and sex (Female). These selections were confirmed using the `base::length()`, `base::table()`, and `base::summary()` functions. As an additional quality assurance check, give special notice to the way a 2 by 2 grid was created to display four comparative histograms of SBP (Systolic Blood Pressure) and four comparative histograms of Lbs (Weight). These initial quality assurance measures may at first seem redundant, but they promote eventual correct outcomes and that is ultimately the purpose of these many statistical analyses.

Download the New File, HospitalAge040049RaceBlackSexFemale.df

Although the enumerated dataset is available for use in the current interactive R session, it might be useful to save `HospitalAge040049RaceBlackSexFemale.df` to the working directory, for later use. To achieve this aim, use the `utils::write.csv()` function to save the dataset as a .csv (comma-separated values) file.

R Input

```
utils::write.csv(HospitalAge040049RaceBlackSexFemale.df,  
                 file="HospitalAge040049RaceBlackSexFemale.csv")
```

Use syntax against `HospitalAge040049RaceBlackSexFemale.df$Ticket` to be sure that each row is unique and there are no duplicate cases (e.g., `Ticket`). Confirm the number of cases (e.g., `rows`) for the object variable.

R Input

```
base::length(HospitalAge040049RaceBlackSexFemale.df$Ticket) # N
```

R Output

```
[1] 671
```

Wrap the `base::table()` function around the `base::duplicated()` function since it is impossible to keep up with output as it shows on the screen given that there are 671 cases in the newly created dataset. The `base::duplicated()` function will ideally confirm that there are no duplicated rows by showing all output as FALSE.

R Input

```
base::table(base::duplicated(  
    HospitalAge040049RaceBlackSexFemale.df$Ticket))
```

R Output

```
FALSE  
671
```

There are no duplicate cases in `HospitalAge040049RaceBlackSexFemale.df`, as confirmed by these actions.

Before any attempt is made to use an inferential or associative test to address the Null Hypothesis, it is necessary to learn more about Systolic Blood Pressure

(SBP) and Weight (Lbs) for those subjects who were between 40 and 49 on their last birthday, inclusive, are Black or African American, and are Female. A limited series of selected graphical images and descriptive statistics follow, but consult the many examples at the beginning of this lesson and other lessons found in this text for a full set of options on these inquiries.

Graphical Images for HospitalAge040049RaceBlackSexFemale.df: SBP and Lbs

In an attempt to not only provide graphical images of interest to this Null Hypothesis but to also address quality assurance issues when working with an enumerated dataset created from a larger dataset, review the figures that follow, using the `vioplot::vioplot()` function, the `stats::density()` function, and the `stats::qqnorm()` and `stats::qqline()` functions. Do not see these comparative figures as being redundant but instead see them as an essential tool in the desire to demonstrate that all pertinent data are in range and are logical—an example of how graphics are used for both visualization as well as quality assurance, to provide additional evidence that the data associated with this current dataset have been enumerated correctly (Figs. 8.45, 8.46, 8.47, 8.48, 8.49).

Comparative Graphics for Systolic Blood Pressure

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
vioplot::vioplot(Hospital.df$SBP,
  names=c("Systolic Blood Pressure"), col="red", ylim=c(0,200))
title("Violin Plot of SBP: All Subjects")
vioplot::vioplot(HospitalAge021100.df$SBP,
  names=c("Systolic Blood Pressure"), col="red", ylim=c(0,200))
title("Violin Plot of SBP: Age 021 to 100")
vioplot::vioplot(HospitalAge040049.df$SBP,
  names=c("Systolic Blood Pressure"), col="red", ylim=c(0,200))
title("Violin Plot of SBP: Age 040 to 049")
vioplot::vioplot(HospitalAge040049RaceBlackSexFemale.df$SBP,
  names=c("Systolic Blood Pressure"), col="red", ylim=c(0,200))
title("Violin Plot of SBP: Age - 040 to 049, Race - Black
or African American, Sex - Female")
```

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::plot(stats::density(Hospital.df$SBP, na.rm=TRUE),
```

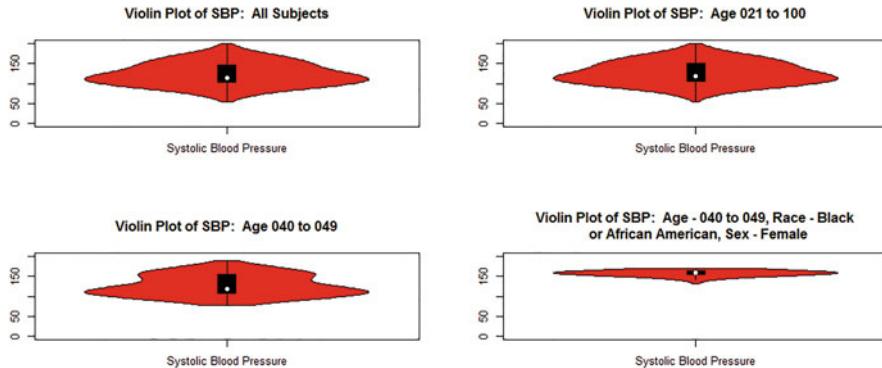


Figure 8.45: Distribution of systolic blood pressure by increasingly restrictive subsets—2

```

main="Density Curve of SBP, All Subjects",
col="red", lwd=3, xlim=c(0,200), ylim=c(0,0.08))
graphics::plot(stats::density(HospitalAge021100.df$SBP,
na.rm=TRUE),
main="Density Curve of SBP, Age 021 to 100",
col="red", lwd=3, xlim=c(0,200), ylim=c(0,0.08))
graphics::plot(stats::density(HospitalAge040049.df$SBP,
na.rm=TRUE),
main="Density Curve of SBP, Age 040 to 049",
col="red", lwd=3, xlim=c(0,200), ylim=c(0,0.08))
graphics::plot(stats::density(
HospitalAge040049RaceBlackSexFemale.df$SBP, na.rm=TRUE),
main="Density Curve of SBP: Age - 040 to 049, Race - Black
or African American, Sex - Female",
col="red", lwd=3, xlim=c(0,200), ylim=c(0,0.08))
# The na.rm argument is required.
# To promote consistency, note how the X axis scale is the
# same for all four figures. The Y axis scale is also
# consistent for all four figures.

```

R Input

```

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
stats::qqnorm(Hospital.df$SBP,
main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP:
All Subjects",
col="blue", xlim=c(-4,4), ylim=c(0,200), font.axis=2,
font.lab=2)

```

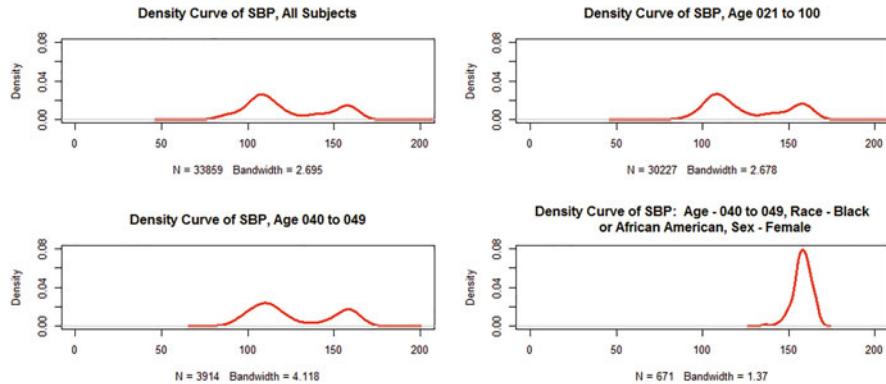


Figure 8.46: Distribution of systolic blood pressure by increasingly restrictive subsets—3

```
stats::qqline(Hospital.df$SBP, col="red", lwd=4, lty=2)
stats::qnorm(HospitalAge021100.df$SBP,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP:
  Age 021 to 100, Inclusive",
  col="blue", xlim=c(-4,4), ylim=c(0,200), font.axis=2,
  font.lab=2)
stats::qqline(HospitalAge021100.df$SBP,
  col="red", lwd=4, lty=2)
stats::qnorm(HospitalAge040049.df$SBP,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP:
  Age 040 to 049, Inclusive",
  col="blue", xlim=c(-4,4), ylim=c(0,200), font.axis=2,
  font.lab=2)
stats::qqline(HospitalAge040049.df$SBP,
  col="red", lwd=4, lty=2)
stats::qnorm(HospitalAge040049RaceBlackSexFemale.df$SBP,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP:  Age -
  040 to 049, Race - Black or African American,
  Sex - Female",
  col="blue", xlim=c(-4,4), ylim=c(0,200), font.axis=2,
  font.lab=2)
stats::qqline(HospitalAge040049RaceBlackSexFemale.df$SBP,
  col="red", lwd=4, lty=2)
```

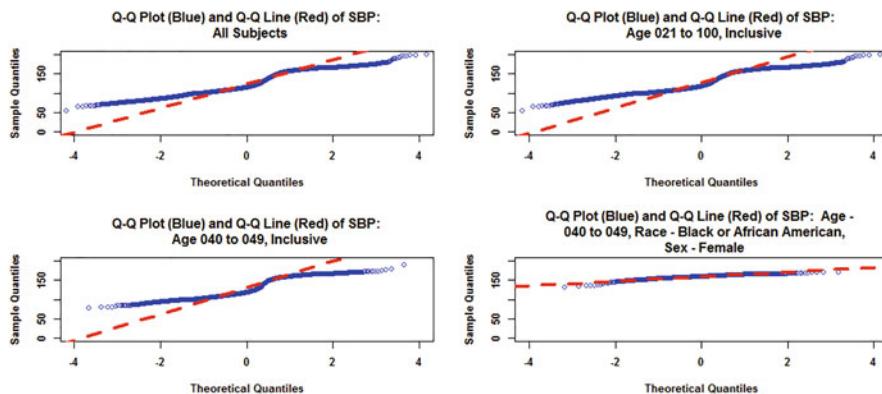


Figure 8.47: Distribution of systolic blood pressure by increasingly restrictive subsets—4

Comparative Graphics for Weight

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
vioplot::vioplot(na.omit(Hospital.df$Lbs),
  names=c("Weight"), col="red", ylim=c(0,325))
title("Violin Plot of Lbs: All Subjects")
vioplot::vioplot(HospitalAge021100.df$Lbs,
  names=c("Weight"), col="red", ylim=c(0,325))
title("Violin Plot of Lbs: Age 021 to 100")
vioplot::vioplot(HospitalAge040049.df$Lbs,
  names=c("Weight"), col="red", ylim=c(0,325))
title("Violin Plot of Lbs: Age 040 to 049")
vioplot::vioplot(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  names=c("Weight"), col="red", ylim=c(0,325))
title("Violin Plot of Lbs: Age - 040 to 049, Race - Black
or African American, Sex - Female")
# Go back to the Code Book and observe how "... there are no
# weights listed for patients ranging in age from birth to
# 20 years." Because these data are missing from the dataset
# it was necessary to wrap the na.omit() function around the
# object variable Hospital.df$Lbs.
```

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
```

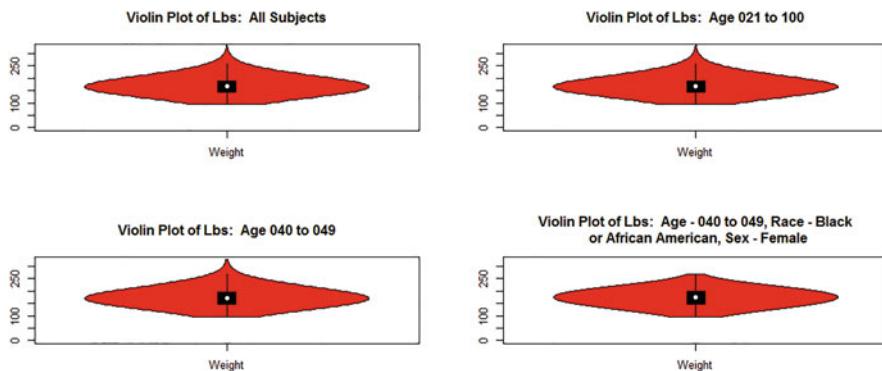


Figure 8.48: Distribution of weight by increasingly restrictive subsets—2

```

graphics::plot(stats::density(Hospital.df$Lbs, na.rm=TRUE),
  main="Density Curve of Lbs, All Subjects",
  col="red", lwd=3, xlim=c(0,325), ylim=c(0,0.015))
graphics::plot(stats::density(HospitalAge021100.df$Lbs,
  na.rm=TRUE),
  main="Density Curve of Lbs, Age 021 to 100",
  col="red", lwd=3, xlim=c(0,325), ylim=c(0,0.015))
graphics::plot(stats::density(HospitalAge040049.df$Lbs,
  na.rm=TRUE),
  main="Density Curve of Lbs, Age 040 to 049",
  col="red", lwd=3, xlim=c(0,325), ylim=c(0,0.015))
graphics::plot(stats::density(
  HospitalAge040049RaceBlackSexFemale.df$Lbs, na.rm=TRUE),
  main="Density Curve of Lbs: Age - 040 to 049, Race - Black
or African American, Sex - Female",
  col="red", lwd=3, xlim=c(0,325), ylim=c(0,0.015))
# The na.rm argument is required.
# To promote consistency, the X axis scale is the same for
# all four figures. The Y axis scale is also consistent
# for all four figures.

```

R Input

```

par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
stats::qqnorm(Hospital.df$Lbs,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Lbs:
All Subjects",
  col="blue", xlim=c(-4,4), ylim=c(0,325), font.axis=2,

```

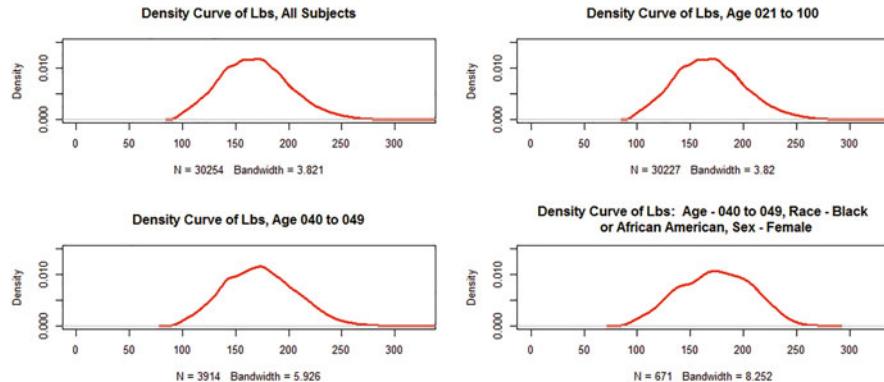


Figure 8.49: Distribution of weight by increasingly restrictive subsets—3

```

font.lab=2)
stats::qqline(Hospital.df$Lbs, col="red", lwd=4, lty=2)
stats::qnorm(HospitalAge021100.df$Lbs,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Lbs:
  Age 021 to 100, Inclusive",
  col="blue", xlim=c(-4,4), ylim=c(0,325), font.axis=2,
  font.lab=2)
stats::qqline(HospitalAge021100.df$Lbs,
  col="red", lwd=4, lty=2)
stats::qnorm(HospitalAge040049.df$Lbs,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Lbs:
  Age 040 to 049, Inclusive",
  col="blue", xlim=c(-4,4), ylim=c(0,325), font.axis=2,
  font.lab=2)
stats::qqline(HospitalAge040049.df$Lbs,
  col="red", lwd=4, lty=2)
stats::qnorm(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Lbs:  Age -
  040 to 049, Race - Black or African American,
  Sex - Female",
  col="blue", xlim=c(-4,4), ylim=c(0,325), font.axis=2,
  font.lab=2)
stats::qqline(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  col="red", lwd=4, lty=2)

```

Absent inferential testing of any type (which will come later in this lesson) and depending only on a few comparative figures, there is some degree of assurance that subjects from the dataset HospitalAge040049RaceBlackSexFemale.df are potentially homogeneous (e.g., similar). If that assumption is true, then it is further assumed (again without current testing and depending only on a few

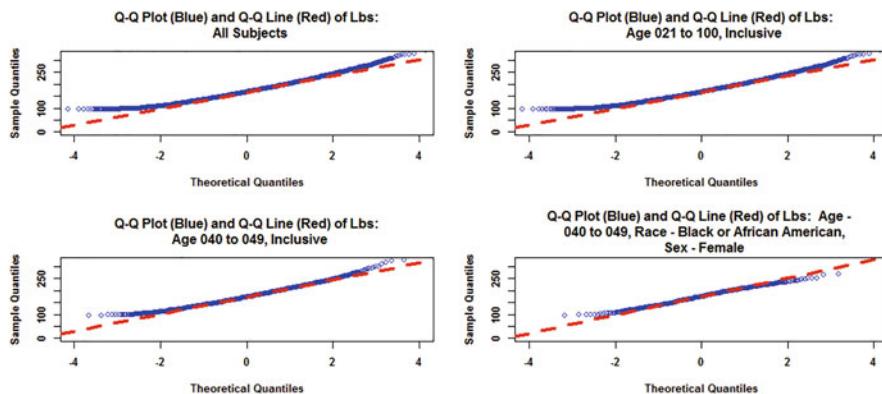


Figure 8.50: Distribution of weight by increasingly restrictive subsets—4

comparative figures) that the data for object variables such as Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), and Weight (Lbs) follow (or at least approach) normal distribution. These initial assumptions are based only on the demographic similarities of subjects in the dataset: age range, race, and sex (Fig. 8.50).

As has been presented throughout this text, descriptive statistics alone are not enough to gain a complete understanding of data and the relationships between and among the data. Graphical images of many types complement descriptive statistics and should always be prepared, even if there were no desire to eventually share the figures with others.

Look below at the different figures for object variables SBP and Lbs. To both save space and to also provide an opportunity for side-by-side comparisons, six different R-based graphically-focused functions have been used to prepare one common figure. The placement of a Violin Plot, Density Plot, Q-Q Plot with accompanying Q-Q Line, Box Plot, Beeswarm Plot, and Bean Plot, all in one common figure, makes it fairly simple to gain a good sense of the data for selected object variables: data distribution, measures of central tendency, data extremes, etc., (Fig. 8.51).

Singular Graphics of HospitalAge040049RaceBlackSexFemale.df\$SBP

R Input

```
base::range(HospitalAge040049RaceBlackSexFemale.df$SBP)
# Use the base::range() function to determine the best
# selection for axis values of SBP.
```

R Output

```
[1] 130 170
```

R Input

```
install.packages("beanplot")
library(beanplot)                      # Load the beanplot package.
help(package=beanplot)                  # Show the information page.
sessionInfo()                          # Confirm all attached packages.

install.packages("beeswarm")
library(beeswarm)                      # Load the beeswarm package.
help(package=beeswarm)                  # Show the information page.
sessionInfo()                          # Confirm all attached packages.

par.ask=TRUE)
par(mfrow=c(2,3)) # 6 figures into a 2 row by 3 column grid
vioplot::vioplot(HospitalAge040049RaceBlackSexFemale.df$SBP,
  names=c("Systolic Blood Pressure"), ylim=c(100,200),
  col="red")
title("Violin Plot of SBP: Age - 040 to 049, Race - Black
or African American, Sex - Female", ylab="SBP")
graphics::plot(stats::density(
  HospitalAge040049RaceBlackSexFemale.df$SBP, na.rm=TRUE),
  main="Density Curve of SBP: Age - 040 to 049, Race - Black
or African American, Sex - Female", xlim=c(100,190),
  ylim=c(0,0.08), col="red", lwd=4) # na.rm is required
stats::qqnorm(HospitalAge040049RaceBlackSexFemale.df$SBP,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP: Age -
040 to 049, Race - Black or African American,
Sex - Female", xlim=c(-4,4), ylim=c(100,200), col="blue")
stats::qqline(HospitalAge040049RaceBlackSexFemale.df$SBP,
  col="red", lwd=4, lty=2)
graphics::boxplot(HospitalAge040049RaceBlackSexFemale.df$SBP,
  main="Box Plot of SBP: Age - 040 to 049, Race - Black
or African American, Sex - Female", ylim=c(100,200),
  ylab="SBP", col="red", font=2)
beeswarm::beeswarm(HospitalAge040049RaceBlackSexFemale.df$SBP,
  main="Beeswarm Plot of SBP: Age - 040 to 049, Race - Black
or African American, Sex - Female)", ylim=c(100,200), pch=16,
  col=c("red"), method="swarm", xlab="", ylab="SBP")
```

```
beanplot::beanplot(HospitalAge040049RaceBlackSexFemale.df$SBP,
  main="Beanplot of SBP: Age - 040 to 049, Race - Black
or African American, Sex - Female", ylim=c(100,200),
  ylab="SBP", font=2, col="red", beanlines="mean",
  beanlinewd=6)
```

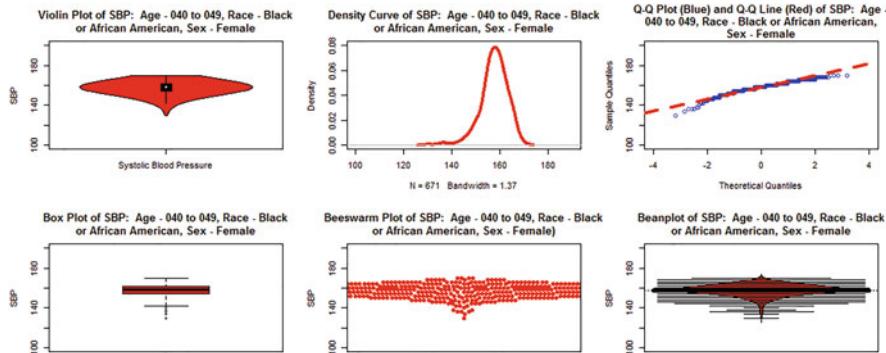


Figure 8.51: Distribution of systolic blood pressure from multiple perspectives—1

The HospitalAge040049RaceBlackSexFemale.df six in one figure of different ways data for object variable SBP are viewed is quite valuable and provides a strong sense of what to expect with future use for this object variable. A final graphic using the descr::histkdnc() function will provide a fairly definitive view of the object variable and adherence to (or deviation away) normal distribution.

Observe how global settings for par() function arguments lwd, font, cex.lab, and cex.axis are adjusted before use of the descr::histkdnc() function and then how these arguments are set back (e.g., toggled) to original values immediately after use of this function. This type of action is used when a function, by itself, does not support desired graphical settings that are alternatively accommodated by use of the par() function and accompanying arguments (Fig. 8.52).

R Input

```
install.packages("descr")
library(descr)                                # Load the descr package.
help(package=descr)                            # Show the information page.
sessionInfo()                                 # Confirm all attached packages.

savelwd      <- par(lwd=4)                      # Heavy line
savefont     <- par(font=2)                      # Bold
savecex.lab  <- par(cex.lab=1.25)               # Label
savecex.axis <- par(cex.axis=1.25)               # Axis
```

```

par(ask=TRUE)
descr::histkdnc(HospitalAge040049RaceBlackSexFemale.df$SBP,
  main="Histogram of SBP with Superimposed Normal Curve (Blue)
  and Density Curve (Red): Age - 040 to 049, Race - Black
  or African American, Sex - Female",
  xlim=c(125,175), ylim=c(0,0.1),
  xlab="Systolic Blood Pressure", ylab="Density",
  col=grey(0.95)) # Use grey to allow contrast with lines
par(savelwd); par(savefont); par(savecex.lab);
par(savecex.axis) # Use ; to move to next line and save space
# For this figure, set the X axis scale at SBP = 125 to 175,
# to emphasize the values and distribution of SBP.

```

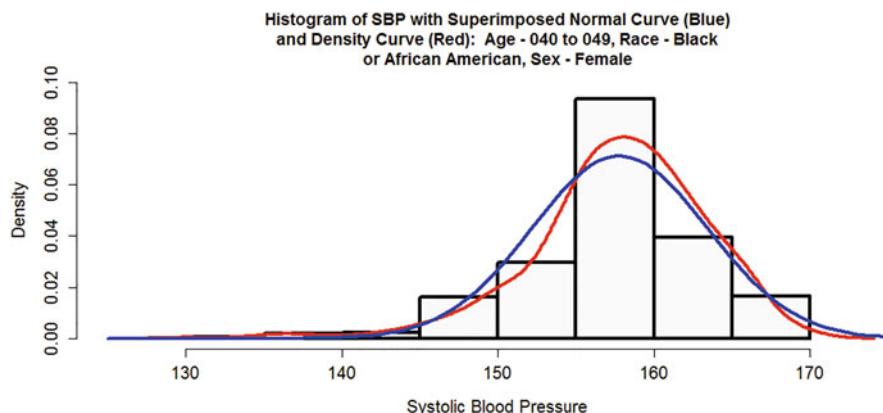


Figure 8.52: Normality of systolic blood pressure for selected subjects—1

Singular Graphics of HospitalAge040049RaceBlackSexFemale.df\$Lbs

R Input

```

range(HospitalAge040049RaceBlackSexFemale.df$Lbs)
# Use the range() function to determine the best
# selection for axis values for Lbs.

```

R Output

```
[1] 96 268
```

R Input

```

par(ask=TRUE)
par(mfrow=c(2,3)) # 6 figures into a 2 row by 3 column grid
vioplot::vioplot(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  names=c("Weight"), ylim=c(90,290), col="red")
title("Violin Plot of Lbs: Age - 040 to 049, Race - Black
or African American, Sex - Female", ylab="Lbs")
graphics::plot(stats::density(
  HospitalAge040049RaceBlackSexFemale.df$Lbs, na.rm=TRUE),
  main="Density Curve of Lbs: Age - 040 to 049, Race - Black
or African American, Sex - Female", xlim=c(75,300),
  ylim=c(0,0.0125), col="red", lwd=4) # na.rm is required
stats::qqnorm(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Lbs: Age -
040 to 049, Race - Black or African American,
Sex - Female", xlim=c(-4,4), ylim=c(90,290), col="blue")
stats::qqline(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  col="red", lwd=4, lty=2)
graphics::boxplot(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  main="Box Plot of Lbs: Age - 040 to 049, Race - Black
or African American, Sex - Female", ylim=c(90,290),
  ylab="Lbs", col="red", font=2)
beeswarm::beeswarm(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  main="Beeswarm Plot of Lbs: Age - 040 to 049, Race - Black
or African American, Sex - Female", ylim=c(90,290), pch=16,
  col=c("red"), method="swarm", xlab="", ylab="Lbs")
beanplot::beanplot(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  main="Beanplot of Lbs: Age - 040 to 049, Race - Black
or African American, Sex - Female", ylim=c(90,290),
  ylab="Lbs", font=2, col="red", beanlines="mean",
  beanlinewd=6)

```

This six in one figure provides different ways of viewing the data for object variable `HospitalAge040049RaceBlackSexFemale.df$Lbs`. Compare these six figures to the final graphic generated by using the `descr::histkdnc()` function. Together, there should now be a strong sense of Weight (Lbs) for this select group of subjects: 40–49-year-old Black or African American females. Specifically, notice how sequestering the data to this finite demographic group provides an enumerated dataset where there is a suggestion (and only a suggestion, absent some type of inferential testing) of normal distribution for the numeric variables of interest to this Null Hypothesis—Systolic Blood Pressure and Weight (Figs. 8.53 and 8.54).

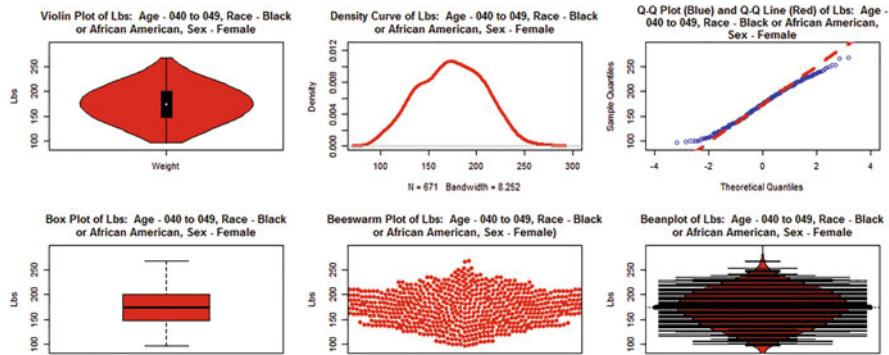


Figure 8.53: Distribution of weight from multiple perspectives—1

R Input

```
savelwd      <- par(lwd=4)           # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab <- par(cex.lab=1.25)    # Label
savecex.axis <- par(cex.axis=1.25)   # Axis
par(ask=TRUE)
descr::histkdnc(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  main="Histogram of Lbs with Superimposed Normal Curve (Blue)
  and Density Curve (Red): Age - 040 to 049, Race - Black
  or African American, Sex - Female",
  xlim=c(75,300), ylim=c(0,0.0125),
  xlab="Weight", ylab="Density", col=grey(0.95))
par(savelwd); par(savefont); par(savecex.lab);
par(savecex.axis) # Use ; to move to next line and save space
# For this figure, set the X axis scale at Lbs = 75 to 300,
# to emphasize the values and distribution of Lbs.
```

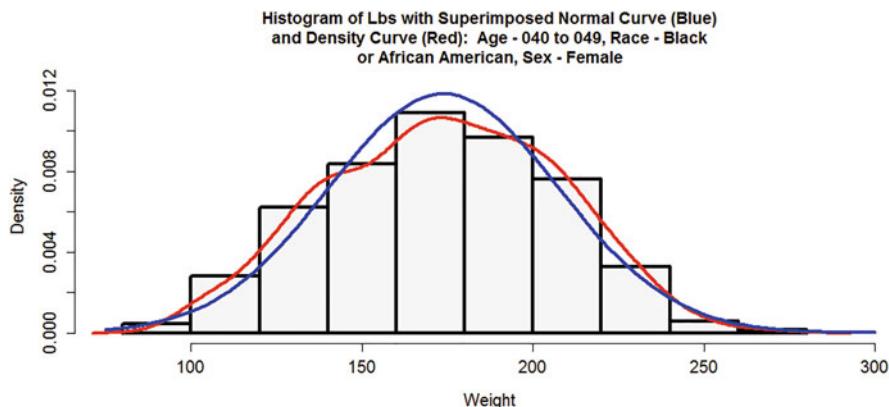


Figure 8.54: Normality of weight for selected subjects—1

Descriptive Statistics and Measures of Central Tendency of Systolic Blood Pressure (SBP) and Weight (Lbs)

The graphics associated with this breakout analysis of 40–49-year-old Black or African American females provides a good sense of direction about the two object variables of immediate concern, Systolic Blood Pressure (SBP) and Weight (Lbs). Yet, descriptive statistics as well as measures of central tendency are equally needed to know as much as possible about the data—all in anticipation of later inferential testing.

R Input

```
tables::tabular(SBP*(length + mean + sd + median + min + max) ~
  1, data=HospitalAge040049RaceBlackSexFemale.df)
```

R Output

```
          All
SBP  length 671.000
      mean    157.797
      sd      5.594
      median 158.000
      min     130.000
      max     170.000
```

R Input

```
tables::tabular(Lbs*(length + mean + sd + median + min + max) ~
  1, data=HospitalAge040049RaceBlackSexFemale.df)
```

R Output

```
          All
Lbs  length 671.0
      mean    173.9
      sd      33.7
      median 174.0
      min     96.0
      max     268.0
```

It may also be best to give a broader view of the data, providing descriptive statistics and measures of central tendency for all numeric variables. The `summarytools::descr()` is quite good for this challenge.

R Input

```
summarytools::descr(HospitalAge040049RaceBlackSexFemale.df)
# Produce a simple text-based table of descriptive
# statistics for all numeric object variables. The
# table can be easily copied as pasted into a word-
# processed document if needed.
```

R Output

Non-numerical variable(s) ignored: Ticket, Location, LocationCode, PrimaryPayer, PrimaryPayerCode, PrincipalDiagnosis3DigitICD9, PrincipalDiagnosisRangeICD9, AgeRangeCode, Race, RaceCode, Ethnicity, EthnicityCode, Sex, SexCode, OverNightStayCode, Location.Recode, PrimaryPayer.Recode, AgeRange.Recode, Race.Recode, Ethnicity.Recode, Sex.Recode, OverNightStay.Recode

Descriptive Statistics

HospitalAge040049RaceBlackSexFemale.df

N: 671

	AgeLastBirthday	DBP	Lbs	OverNightStay	SBP
Mean	44.23	96.87	173.86	4.66	157.80
Std.Dev	2.89	3.17	33.70	7.57	5.59
Min	40.00	84.00	96.00	0.00	130.00
Q1	42.00	94.00	148.00	2.00	154.00
Median	44.00	96.00	174.00	3.00	158.00
Q3	47.00	100.00	200.00	5.00	162.00
Max	49.00	106.00	268.00	108.00	170.00
MAD	2.97	2.97	38.55	1.48	5.93
IQR	5.00	6.00	52.00	3.00	8.00
CV	0.07	0.03	0.19	1.62	0.04
Skewness	0.16	-0.22	-0.03	7.92	-0.90
SE.Skewness	0.09	0.09	0.09	0.09	0.09
Kurtosis	-1.23	0.42	-0.59	87.04	2.11
N.Valid	671.00	671.00	671.00	671.00	671.00
Pct.Valid	100.00	100.00	100.00	100.00	100.00

Tests for Normal Distribution

In an attempt to examine consistency in outcomes between different tests that serve the same general purpose, the following tests for normal distribution will be applied against the two object variables associated with the dataset Hospi-

talAge040049RaceBlackSexFemale.df, both SBP (Systolic Blood Pressure) and Lbs (Weight):

- Anderson–Darling Test for Normality, nortest::ad.test()
- Jarque–Bera Test for Normality, tseries::jarque.bera.test()
- Lilliefors (Kolmogorov–Smirnov) Test for Normality, nortest::lillie.test()
- Shapiro–Wilk Test for Normality, stats::shapiro.test()

These tests for normality will be used to make judgment on data distribution patterns and normality. Based on the results of these tests, there will be a decision to use either a nonparametric approach or a parametric approach (or both) as the Null Hypothesis is addressed. Even with the production of these statistics and accompanying p-values, it is always useful to first prepare appropriate graphics, which for this inquiry into normality calls for a Q-Q Plot with an accompanying Q-Q Line. Because any test of association requires attention to two separate entities, one for the X axis and one for the Y axis, the two variables in question (Lbs and SBP) will be presented graphically side-by-side, to allow for meaningful review and comparison. Remember, of course, that there are different scales for the Y axis since Lbs ranges from less than 100 pounds to nearly 300 pounds and SBP ranges from approximately 125–175 mmHg (Fig. 8.55).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
stats::qqnorm(HospitalAge040049RaceBlackSexFemale.df$SBP,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of SBP: Age - 040 to 049, Race - Black or African American, Sex - Female", xlim=c(-4,4), ylim=c(125,175), col="blue")
stats::qqline(HospitalAge040049RaceBlackSexFemale.df$SBP,
  col="red", lwd=4, lty=2)
stats::qqnorm(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  main="Q-Q Plot (Blue) and Q-Q Line (Red) of Lbs: Age - 040 to 049, Race - Black or African American, Sex - Female", xlim=c(-4,4), ylim=c(90,280), col="blue")
stats::qqline(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  col="red", lwd=4, lty=2)
```

It is certainly possible that normal distribution is absent in both figures (SBP and Lbs), based only on a visual review. For both figures (SBP and Lbs), observe the deviation of the Q-Q line in comparison to the Q-Q plot, especially the high degree of noise at each end of the Q-Q Plot. Although this visual review

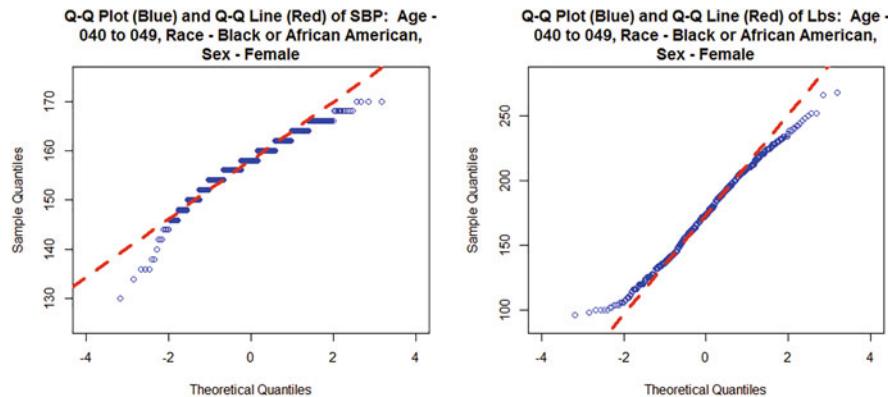


Figure 8.55: Side-by-side Q-Q plot of systolic blood pressure and weight

provides initial direction, the four tests used to address normal distribution will provide the final degree of information needed to make a more certain judgment.

Normal Distribution of HospitalAge040049RaceBlackSexFemale.df\$SBP, Systolic Blood Pressure

SBP and Anderson–Darling Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
nortest::ad.test(HospitalAge040049RaceBlackSexFemale.df$SBP)
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

SBP and Jarque–Bera Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
tseries::jarque.bera.test(
  HospitalAge040049RaceBlackSexFemale.df$SBP)
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

SBP and Lilliefors (Kolmogorov–Smirnov) Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
northeast::lillie.test(HospitalAge040049RaceBlackSexFemale.df$SBP)
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

SBP and Shapiro–Wilk Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
stats::shapiro.test(HospitalAge040049RaceBlackSexFemale.df$SBP)
```

R Output

```
Outcme: p-value = 0.000000000000164
```

For all four tests that address normal distribution of SBP (Systolic Blood Pressure) from among subjects who range in age from 40 to 49 (inclusive), are Black or African American, and are female, the calculated p-value (ranging from $p\text{-value} \leq 0.000000000000164$ to $p\text{-value} \leq 0.0000000000000002$) is less than the criterion $p \leq 0.05$ and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable (SBP) do not follow the normal distribution.

Normal Distribution of HospitalAge040049RaceBlackSexFemale.df\$Lbs, Weight

Lbs and Anderson–Darling Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
northeast::ad.test(HospitalAge040049RaceBlackSexFemale.df$Lbs)
```

R Output

```
Outcome: p-value = 0.0046
```

Lbs and Jarque–Bera Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
tseries::jarque.bera.test(  
  HospitalAge040049RaceBlackSexFemale.df$Lbs)
```

R Output

```
Outcme: p-value = 0.00812
```

Lbs and Lilliefors (Kolmogorov–Smirnov) Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
nortest::lillie.test(HospitalAge040049RaceBlackSexFemale.df$Lbs)
```

R Output

```
Outcome: p-value = 0.0118
```

Lbs and Shapiro–Wilk Test for Normality Null Hypothesis: The data follow the normal distribution.

R Input

```
stats::shapiro.test(HospitalAge040049RaceBlackSexFemale.df$Lbs)
```

R Output

```
Outcome: p-value = 0.00176
```

For all four tests that address normal distribution of Lbs (Weight) from among subjects who range in age from 40 to 49 (inclusive), are Black or African American, and are female, the calculated p-value (ranging from `p-value <= 0.00176` to `p-value <= 0.0118`) is less than the criterion `p <= 0.05` and it can be stated that the Null Hypothesis is rejected (e.g., not accepted). The data for this specific object variable (Lbs) do not follow the normal distribution.

Regarding normal distribution patterns, the different graphics, especially after use of the `qqnorm()` function, provide initial guidance on normality. Yet, to be

more precise the different tests for normality (Anderson–Darling, Jarque–Bera, Lilliefors (Kolmogorov–Smirnov), Shapiro–Wilk) all reach the same conclusion regarding the enumerated dataset HospitalAge040049RaceBlackSexFemale.df:

- The object variable SBP (Systolic Blood Pressure) does not follow normal distribution.
- The object variable Lbs (Weight) does not follow normal distribution.

Given these outcomes on normal distribution of the data, it is suggested that a nonparametric approach to statistical analysis will be entirely appropriate, either on its own or as a complement to a prior parametric approach.

Conduct the Statistical Analysis

The second Null Hypothesis addresses the degree of association (e.g., correlation) between Systolic Blood Pressure and Weight. To mitigate the possibility of influence from other variables such as different age ranges, race, and sex, the dataset for this specific analysis was restricted to those subjects who ranged in age from 40 to 49 (inclusive), are Black or African American, and are female.

Second Null Hypothesis (Ho): Test of Association

Ho: There is no statistically significant correlation ($p \leq 0.05$) between Systolic Blood Pressure and Weight among Black or African American Female subjects who range in age from 40 to 49, inclusive.

This Null Hypothesis addresses a select group of 671 subjects from an original sample of 33,859 subjects. That is to say, from the original dataset of 33,859 subjects found in Hospital.df, the highly-filtered enumerated dataset with the verbose name HospitalAge040049RaceBlackSexFemale.df was created, consisting only of the 671 subjects who, again, ranged in age from 40 to 49 (inclusive), are Black or African American, and are female

Review the prior discussion in this lesson, other lessons, and external resources on how nonparametric statistics are a subset of inferential analyses that are used when: (1) the precision of an interval scale is not met, (2) extreme deviation from normal distribution cannot be ignored, and (3) the difference in the number of subjects for breakout groups is considerable. The visual presentations previously prepared as well as outcomes from the tests of normal distribution justify concerns about normal distribution patterns for SBP and Lbs from the dataset HospitalAge040049RaceBlackSexFemale.df.

The stats::cor() function and the stats::cor.test() function are both used to obtain correlation coefficients. From these two functions, appropriate argument selections are made to obtain the correlation coefficient, Pearson's r for a parametric view toward the data and Spearman's rho for a nonparametric view toward the data. Given the previously mentioned concern about normal distribution or more correctly, deviation away from normal distribution, the stats::cor

function and the stats::cor.test function will be used along with the appropriate argument to distinguish between parametric (e.g., method="pearson") and nonparametric (e.g., method="spearman") estimates of correlation. With correlation studies it is the method argument that differentiates between a parametric approach and a nonparametric approach.

Parametric Pearson's r

R Input

```
stats::cor(HospitalAge040049RaceBlackSexFemale.df$Lbs,  
          HospitalAge040049RaceBlackSexFemale.df$SBP,  
          method="pearson")
```

R Output

Outcome: Pearson's r = -0.00587947

R Input

```
stats::cor.test(HospitalAge040049RaceBlackSexFemale.df$Lbs,  
                HospitalAge040049RaceBlackSexFemale.df$SBP,  
                method="pearson")
```

R Output

Outcome: p-value = 0.879 and Pearson's r = -0.00587947

Nonparametric Spearman's rho

R Input

```
stats::cor(HospitalAge040049RaceBlackSexFemale.df$Lbs,  
          HospitalAge040049RaceBlackSexFemale.df$SBP,  
          method="spearman")
```

R Output

Outcome: Spearman's rho = 0.00237561

R Input

```
stats::cor.test(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  HospitalAge040049RaceBlackSexFemale.df$SBP,
  method="spearman")
```

R Output

Outcome: p-value = 0.951 and Spearman's rho = 0.00237561

Recall that correlation coefficients range from -1.0 (Strong Negative Correlation) to +1.0 (Strong Positive Correlation). Equally, recall that correlation is not synonymous with causation. Variable X and Variable Y may show a degree of correlation (e.g., association), but that does not immediately suggest that Variable X causes Variable Y.

For this inquiry into the degree of correlation between Weight (Lbs) and Systolic Blood Pressure (SBP), both Pearson's r and Spearman's rho provide evidence that the correlation coefficient approximates zero, Pearson's $r = -0.00587947$ and Spearman's $\rho = 0.00237561$. For this one time only analysis of 40–49-year-old Black or African American females, there is simply no correlation (e.g., association) between Weight and Systolic Blood Pressure, as determined by applying both a parametric approach and a nonparametric approach toward the data.

Regardless of whether the test of association is viewed from a parametric (e.g., Pearson's r) approach or a nonparametric (e.g., Spearman's rho) approach, it is helpful to better understand the meaning of correlation by graphically presenting the relationship between the two object variables. For this task, assume that Lbs (Weight) is placed on the X axis and that Systolic Blood Pressure (SBP) is placed on the Y axis. Then, as the figure is embellished add a legend to fully communicate the numeric outcomes along with the visual presentation (Fig. 8.56).

R Input

```
graphics::plot(HospitalAge040049RaceBlackSexFemale.df$Lbs,
  HospitalAge040049RaceBlackSexFemale.df$SBP)
# This graphic has no embellishments and only default
# settings and scales.
```

There are more than a few ways that R can be used to enhance the presentation of how correlation is presented. Regard how the `graphics::plot()` function has been embellished. Look also at the way the `car::regLine()` function was used

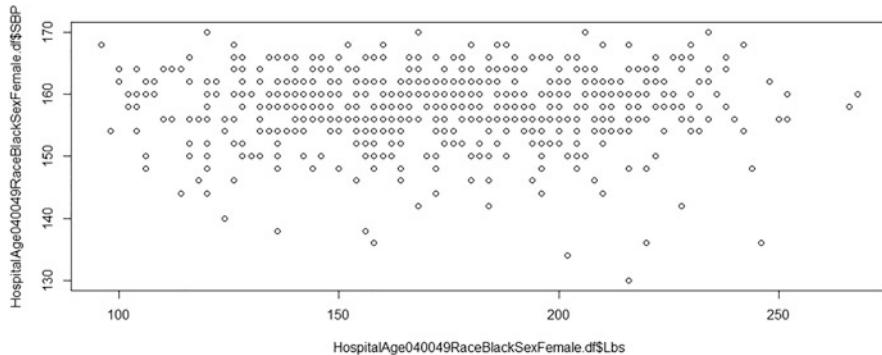


Figure 8.56: Correlation of systolic blood pressure and weight—1

along with the `graphics::plot()` function, to insert a regression line. Finally, notice how the `graphics::legend()` function was added to the figure to present numeric information about Spearman's rho and Pearson's r, to be seen along with the visual presentation (Fig. 8.57).

R Input

```

install.packages("car")
library(car)                                # Load the car package.
help(package=car)                            # Show the information page.
sessionInfo()                               # Confirm all attached packages.

savelwd      <- par(lwd=4)                  # Heavy line
savefont     <- par(font=2)                  # Bold
savecex.lab  <- par(cex.lab=1.25)    # Label
savecex.axis <- par(cex.axis=1.25)   # Axis
# Change global settings
par(ask=TRUE)

graphics::plot(SBP ~ Lbs,                      # Y axis ~ X axis
  data=HospitalAge040049RaceBlackSexFemale.df,
  xlab="Weight (Lbs)", ylab="Systolic Blood Pressure (SBP)",
  main="Scatter Plot of Weight by Systolic Blood Pressure With
  Regression Line: Age = 40 to 49 Years, Race = Black
  or African American, Sex = Female",
  xlim=c(090,290), ylim=c(115,180), pch=c(16))
car::regLine(lm(SBP ~ Lbs,
  data=HospitalAge040049RaceBlackSexFemale.df),
  lwd=4, lty=1, col="red")
graphics::legend("topleft",
  xjust=1, bty="y", box.lwd=1, box.col="black",
  text.col="royalblue", text.font=2,

```

```

"Spearman's rho = 0.00237561 and p-value <= 0.951")
graphics::legend("bottomright",
  xjust=1, bty="y", box.lwd=1, box.col="black",
  text.col="royalblue", text.font=2,
  "Pearson's r = -0.00587947 and p-value <= 0.879")
par(savelwd); par(savefont); par(savecex.lab);
par(savecex.axis)
# Go back to original settings

```

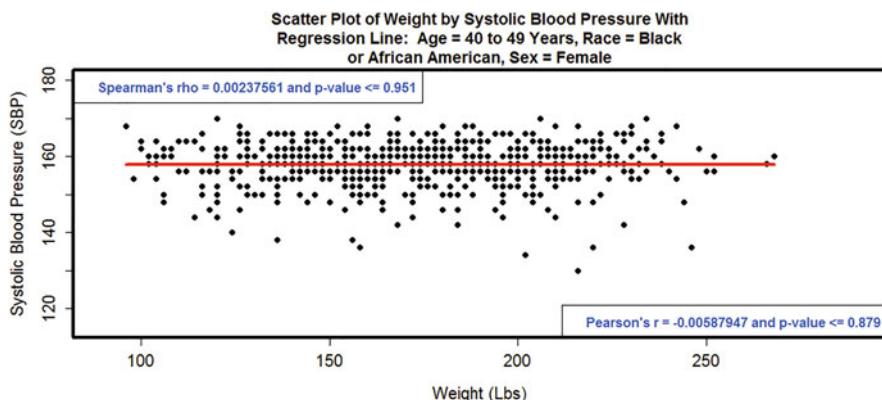


Figure 8.57: Correlation of systolic blood pressure and weight—2

If more embellished scatter plots of X:Y are needed, the car::scatterplot() function and the aplpack::bagplot() function should both be considered. Each function has merit and provides more detail than can be easily obtained using the graphics::plot() function (Fig. 8.58).

R Input

```

savelwd      <- par(lwd=4)          # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab  <- par(cex.lab=1.5)    # Label
savecex.axis <- par(cex.axis=1.5)   # Axis
# Change global settings
par(ask=TRUE)
car::scatterplot(SBP ~ Lbs,    # Y ~ X
  data=HospitalAge040049RaceBlackSexFemale.df,
  xlab="Weight (Lbs)", ylab="Systolic Blood Pressure (SBP)",
  main="Scatter Plot of Weight by Systolic Blood Pressure With
Regression Line and Ellipse: Age = 40 to 49 Years,
Race = Black or African American, Sex = Female",
  xlim=c(090,290), ylim=c(115,180),

```

```

reg.line=TRUE, boxplots="xy", grid=FALSE, pch=16,
font.lab=2, font.axis=2, lwd=2, lty=1, cex.main=1.25,
ellipse=TRUE, robust=TRUE, reset.par=FALSE)
# Data-concentration ellipse and center of ellipse
# Note also how boxplots show along each axis
legend("topleft", xjust=0, bty="n", text.col="darkblue",
text.font=2,
"Spearman's rho = 0.00237561 and p-value <= 0.951")
legend("topright", xjust=1, bty="n", text.col="darkblue",
text.font=2,
"Pearson's r = -0.00587947 and p-value <= 0.879")
par(savelwd); par(savefont); par(savecex.lab)
par(savecex.axis)
# Go back to original settings

```

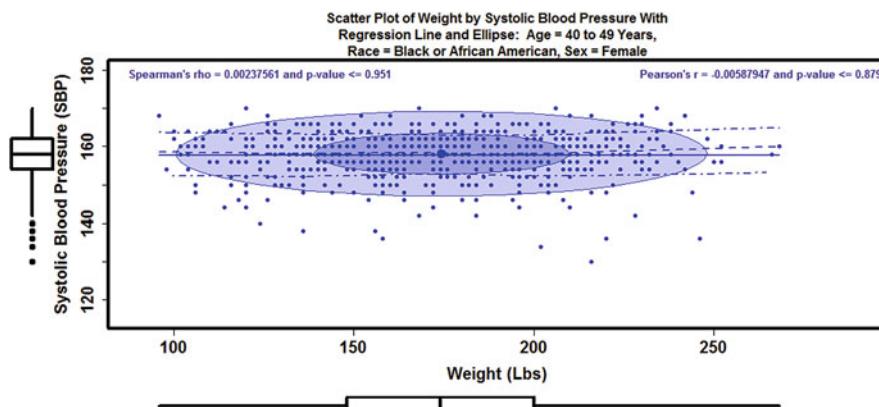


Figure 8.58: Correlation of systolic blood pressure and weight—3

The X:Y scatter plot is well-established and there are many ways to prepare and present a scatter plot. A fairly new approach, however, is to use the bagplot to show the degree of association between two separate variables, Variable X v Variable Y. The bagplot is a bivariate boxplot, where 50% of all points are held in the central bag. A fence surrounds the bag and from this fence the remaining points show outward, showing distribution points and extreme values, if there are any. Compare the visualization of the association between Weight (X axis, Lbs) and Systolic Blood Pressure (Y axis, SBP) in a bagplot (below) as compared to the more traditional X v Y scatter plot of these two object variables (Fig. 8.59).

R Input

```

install.packages("aplpack")
library(aplpack)                      # Load the aplpack package.
help(package=aplpack)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

savelwd      <- par(lwd=2)             # Heavy line
savefont     <- par(font=2)            # Bold
savecex.lab <- par(cex.lab=1.25)       # Label
savecex.axis <- par(cex.axis=1.25)      # Axis
par(ask=TRUE)

aplpack::bagplot(HospitalAge040049RaceBlackSexFemale.df$Lbs,
                 HospitalAge040049RaceBlackSexFemale.df$SBP,
                 main="Bag Plot of Weight by Systolic Blood Pressure:
Age = 40 to 49 Years, Race = Black or African
American, Sex = Female",
                 xlab="Weight (Lbs)", ylab="Systolic Blood Pressure (SBP)",
                 na.rm=TRUE,                      # Accommodate missing data
                 show.outlier=TRUE,              # At the R prompt, key )
                 show.whiskers=TRUE,            # help(bagplot) to see details for
                 show.looppoints=TRUE,          # each argument, show.outlier,
                 show.bagpoints=TRUE,           # show.whiskers, etc. Then, decide
                 show.loophull=TRUE,            # which arguments meet individual
                 show.baghull=TRUE,             # needs.
                 pch=c(22))                   # Filled square red symbol
par(savelwd); par(savefont); par(savecex.lab)
par(savecex.axis)

```

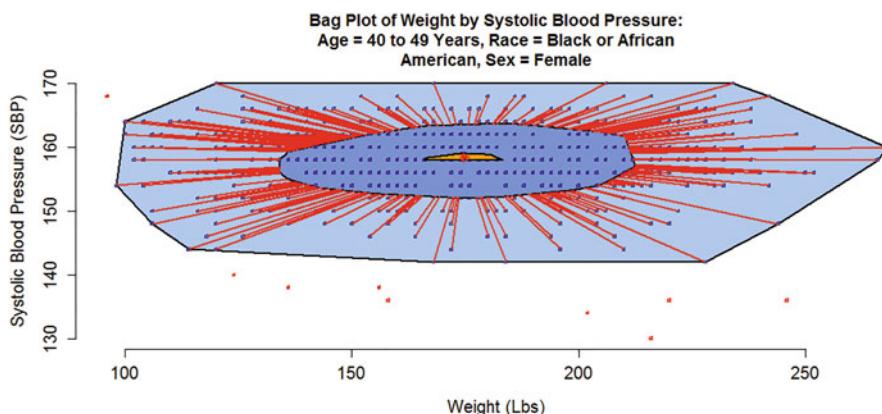


Figure 8.59: Correlation of systolic blood pressure and weight—4

Notice in the bagplot how the X v Y (e.g., Lbs v SBP) values seem to radiate throughout, without any clear indication of a specific direction, indicating that there is no correlation between the two object variables. That is to say, the points are truly scattered, with no discernible direction evident, either in a positive or negative direction.

Summary to the Correlation (e.g., Association) Null Hypothesis

This part of the lesson, using data gained from the original dataset Hospital1.df, focused on the association, if any, between Systolic Blood Pressure and Weight for the 671 subjects who ranged in age from 40 to 49, inclusive, are Black or African American, and are female. It was judged that this degree of selection and the desire to prepare a fairly homogeneous subgroup would mitigate any possible influence, overall, from age ranges, race, and sex.

Ho: There is no statistically significant correlation ($p \leq 0.05$) between Systolic Blood Pressure and Weight among Black or African American Female subjects who range in age from 40 to 49, inclusive.

There were observed concerns about normal distribution for the two variables in question (SBP and Lbs from HospitalAge040049RaceBlackSexFemale.df). Because of these concerns the statistical test of association (e.g., correlation) was conducted not only from a parametric (e.g., Pearson's) perspective but also from a nonparametric (e.g., Spearman's rho) perspective.

Whether the data were examined from a parametric or nonparametric perspective, the overall outcomes were consistent. It was observed that there was no statistically significant correlation ($p \leq 0.05$) between Systolic Blood Pressure and Weight and that the coefficient of correlation approximated 0 (Pearson's $r = -0.00587947$ and Spearman's $\rho = 0.00237561$):

Parametric

Pearson's $r = -0.00587947$
 $p\text{-value} \leq 0.879$

Nonparametric

Spearman's $\rho = 0.00237561$
 $p\text{-value} \leq 0.951$

These statistics are also visually reinforced in the different figures presented in this part of the lesson. Even with a regression line added to the figures, there is no discernable pattern of correlation, either positive or negative. Instead, the plotted points seem to be truly scattered.

8.8 Summary of Outcomes

The dataset `Hospital.csv` was imported into R and organized as a dataframe, `Hospital.df`. From these first actions and from among the many possibilities for how the data could have been examined, two specific Null Hypotheses were declared, one Null Hypothesis focusing on a test of difference and the other Null Hypothesis focusing on a test of association.

8.8.1 Outcomes for First Null Hypothesis (H_0): Mean Comparisons by Breakout Groups

H_0 : There is no statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure by the seven different racial groups (e.g., American Indian or Alaska Native, Asian, Black or African American, Native Hawaiian or Other Pacific Islander, Other, Unknown, White) among those subjects who range in age from 40 to 49, inclusive.

Using Oneway ANOVA and Tukey's HSD (Honestly Significant Difference) mean comparison test, it was determined that there was a statistically significant difference ($p \leq 0.05$) in Diastolic Blood Pressure (DBP) by the seven different racial groups (e.g., American Indian or Alaska Native, Asian, Black or African American, Native Hawaiian or Other Pacific Islander, Other, Unknown, White) among those subjects who range in age from 40 to 49, inclusive. The following output identifies Mean DBP for each of the race breakout groups along with the Tukey's HSD mean comparison outcome listing those breakout groups that share a common Mean DBP and those that do not share a common Mean DBP:

	DBP	groups
Black	97.0866	a
Hawaiian	97.0000	ab
Unknown	90.7692	b
Other	90.2735	b
AmIndian	89.6923	b
Asian	71.9412	c
White	65.0081	d

Obviously, the greatest difference in Mean DBP was between Black subjects (Mean DBP = 97.0866) and White subjects (Mean DBP = 65.0081). The outcomes were generally consistent whether the data were examined using a parametric inferential test or a nonparametric inferential test.

8.8.2 Outcomes for Second Null Hypothesis (H_0): Test of Association

H_0 : There is no statistically significant correlation ($p \leq 0.05$) between Systolic Blood Pressure and Weight among Black or African American Female subjects who range in age from 40 to 49, inclusive.

Using both a Pearson's r for a parametric view of the data or a Spearman's rho for a nonparametric view of the data, it was determined that there was no meaningful correlation (e.g., association) between Systolic Blood Pressure (SBP) and Weight (Lbs) among Black or African American Female subjects who range in age from 40 to 49, inclusive. The estimates of correlation for both views indicated that the selected statistic approximated 0:

Parametric, Pearson's r = -0.00587947

Nonparametric, Spearman's rho = 0.00237561

The two estimates of association were visually confirmed by preparing a scatter plot of the data, with Weight (Lbs) placed on the X axis and Systolic Blood Pressure (SBP) placed on the Y axis. As expected with a scatter plot with an estimated correlation that approximated 0, there was simply no discernable pattern to the data other than to say that the data were truly scattered throughout the figure.

8.9 Addendum 1: Additional Graphics, to Show Relationships Between and Among Data

The use of many different types of graphics has been stressed throughout this text and is reinforced in this addendum. Numerical statistics such as mean, standard deviation, median, and mode are certainly important and their use can never be minimized. The presentation of separate numerical statistics in table format is also vital to biostatistics, as a long-standing way to communicate numerical findings to others. However, to gain immediate attention and understanding, especially by those who are not well-experienced with biostatistics, graphics and selection of the most appropriate type of graphic should always be among first considerations when presenting outcomes from analysis of a dataset. People generally respond to what they see and graphics support this aim.

The R-based graphical functions that have been used the most, up to this point of the lesson, have been based on:

- Examples of functions used for factor-type data are the `graphics::plot()` function and the `graphics::barplot()` function, often with the `base::table()` function serving as a wrapper.
- Examples of functions used for numeric-type data are the `graphics::boxplot()`, `graphics::hist()`, `graphics::plot()`, and `stats::qqnorm()` functions, and often the `graphics::plot()` function wrapped around the `stats::density()` function.

The graphics produced with these functions in the front part of this lesson have been purposely simple, with only a few embellishments (e.g., title and axis labels, text size, color selection, etc.) used to enhance the images. Yet, one of the main reasons why R has gained such popularity among those engaged in biostatistics is that R supports a large collection of functions and accompanying arguments used to generate selected graphics. Many of these graphically-oriented functions are made available when R is first downloaded. Otherwise, consider the use of functions found in the more than 15,000 separate packages available to the R community—functions that provide opportunities for the presentation of *Beautiful Graphics*, a term used in the R community for figures that are without compare and of the highest quality.

The purpose of this addendum is to provide a brief review of the many graphically-oriented functions associated with R. To provide some degree of consistency in presentation, only a few selected object variables will be emphasized from among the many object variables found in the `HospitalAge021100.df` dataframe. As needed, a few other object variables will be included in these demonstrations, but the emphasis is on data from the enumerated dataset `HospitalAge021100.df`:¹⁷

- Factor-type object variable in `HospitalAge021100.df`: `Location.Recode` and `PrimaryPayer.Recode`
- Numeric-type object variables in `HospitalAge021100.df`: `SBP`, `DBP`, and `Lbs`

8.9.1 Graphical Presentation of Grouped (e.g., Factor-Type) Data

8.9.1.1 Association Plot

The association plot is used to show the degree of difference (if any) in observed and expected frequencies for a two-dimensional contingency table (Fig. 8.60).

R Input

```
savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab <- par(cex.lab=1.15) # Label
```

¹⁷`Hospital.df` is a dataset that has been selected and organized for teaching purposes. As a subset of `Hospital.df`, `HospitalAge021100.df` should be viewed as a dataset that follows along with this aim. In the many graphical images that follow, the name for the type of figure (e.g., barplot, density plot, histogram, etc.) shows in the figure title. That action is fine for this lesson, to reinforce the name of the specific figure. However, it is generally best to avoid this practice when producing figures for presentation to others and especially the professional community. Figure titles should be descriptive, but it is redundant to include the name for the figure type in the title, other than for teaching purposes which is the case in this addendum.

```

savecex.axis <- par(cex.axis=1.15)# Axis
par(ask=TRUE)
graphics::assocplot(base::table(
  HospitalAge021100.df$PrimaryPayer.Recode,           # Row
  HospitalAge021100.df$Location.Recode),                 # Column
  main="Association Plot of Primary Payer by Location",
  xlab="Primary Payer", ylab="Location")
par(savelwd); par(savefont); par(savecex.lab);
par(savecex.axis)
mtext("
The box is black and above the baseline when observed frequency
is greater than expected frequency. The box is red and below
the baseline when observed frequency is less than expected
frequency.",
line=0, side=1, cex=0.65, font=2)
# Be sure to notice how global embellishments (e.g., line
# width, font, label size, etc.) were used before R syntax
# was used, by using the par() function. Global
# embellishments were then set back to the original values
# by again using the par() function. This action is used
# selectively throughout this lesson, to make graphical
# images bold and easy to view when presented in either a
# paper or as part of a large group visual presentation.
#
# Key help(par) at the R prompt to learn more about these
# graphical parameters and how to they can be used to best
# effect.

```

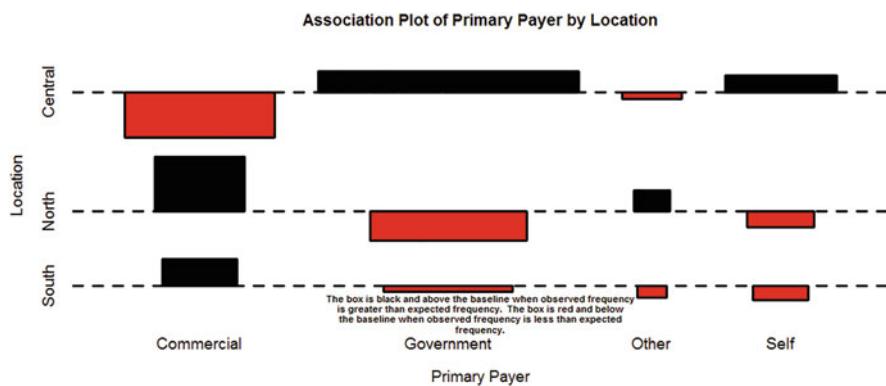


Figure 8.60: Association plot

8.9.1.2 Bar Plot

The bar plot (e.g., bar chart, bar graph, barplot, column bar chart) is used to show rectangular bars (e.g., columns) that represent grouped data, with the length of each bar in proportion to the frequency (e.g., N, percentage) of each breakout group. The rectangular bars can be plotted either in a vertical manner or a horizontal manner by adjusting the horiz argument. Grouped frequency can be plotted either in a stacked manner or a side-by-side manner by adjusting the beside argument.

R Input

```
PayerLocation <- base::table(
  HospitalAge021100.df$PrimaryPayer.Recode,           # Row
  HospitalAge021100.df$Location.Recode)                # Column
# Use the base::table() function to create an enumerated
# object variable (e.g., PayerLocation) that represents a
# table of summed frequencies of rows (e.g., Primary Payer)
# by columns (e.g., Location).
#
# The syntax for PayerLocation could have been part of the
# overall syntax for the figure, but if it is used multiple
# times then it is efficient to create a separate object of
# the Row by Column relationship.
```

```
PayerLocation
```

R Output

	Central	North	South
Commercial	3131	2015	1204
Government	12682	3787	2853
Other	579	315	112
Self	2440	666	443

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
par(ask=TRUE)
graphics::barplot(PayerLocation,
  beside=FALSE,                                     # Stacked barplot
  main="Stacked Barplot of Primary Payer by Location",
```

```

xlab="Primary Payer and Location",
ylab="Number of Subjects",
ylim=c(0,20000),                                     # Adjust scale Y axis
col=c("red", "cyan", "blue", "black"),
font.axis=2, font.lab=2,                                # Bold
cex.axis=1.25, cex.names=1.25, cex.lab=1.25,          # Size
legend=rownames(PayerLocation))

graphics::barplot(PayerLocation,
beside=TRUE,                                         # Grouped barplot
main="Side-by-Side Barplot of Primary Payer by Location",
xlab="Primary Payer and Location",
ylab="Number of Subjects",
ylim=c(0,20000),                                     # Adjust scale Y axis
col=c("red", "cyan", "blue", "black"),
font.axis=2, font.lab=2,                                # Bold
cex.axis=1.25, cex.names=1.25, cex.lab=1.25,          # Size
legend=rownames(PayerLocation))

# Note the difference in the way outcomes show in a stacked
# barplot v a grouped barplot.

```

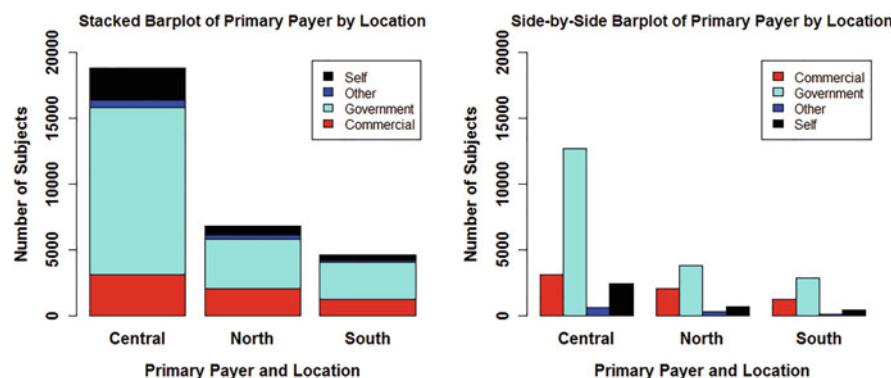


Figure 8.61: Stacked bar plot and side-by-Sie bar plot

8.9.1.3 Mosaic Plot

The mosaic plot is used to show proportional representation of expected frequencies for data that might otherwise be presented in a contingency table. The size of each tile (e.g., cell) in a mosaic plot is in proportion to the contribution of each breakout grouping compared to total counts (Figs. 8.61 and 8.62).

R Input

```

par(ask=TRUE)
savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab  <- par(cex.lab=1.25) # Label
savecex.axis <- par(cex.axis=1.25) # Axis
graphics::mosaicplot(~Location.Recode + PrimaryPayer.Recode,
  data=HospitalAge021100.df,
  main="Mosaic Plot of Primary Payer by Location",
  col=c("red", "cyan", "blue", "black"),
  xlab="Location", ylab="Primary Payer",
  type="pearson")
par(savelwd); par(savefont); par(savecex.lab)
par(savecex.axis)

```

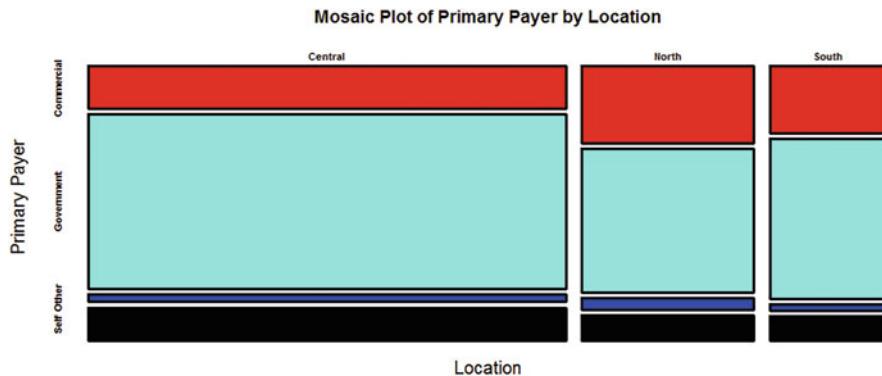


Figure 8.62: Mosaic plot—1

R Input

```

install.packages("vcd")
library(vcd)                      # Load the vcd package.
help(package=vcd)                  # Show the information page.
sessionInfo()                     # Confirm all attached packages.

plot.new()                         # Prepare a new plot
par(ask=TRUE)
vcd::mosaic(~Location.Recode + PrimaryPayer.Recode,
  data=HospitalAge021100.df,
  main="Mosaic Plot of Primary Payer by Location",
  shade=TRUE,
  spacing=spacing_increase,

```

```

spacing_args = list(start = 1.0, rate = 1.5))
mtext("Similar in use to an association plot, blue indicates that
the observed frequency for an individual cell is greater
than the expected frequency and red indicates that
the observed frequency for an individual cell
is less than the expected frequency.",
line=-5, side=4, cex=0.95, font=2)
# Note how the mtext() function was organized, to allow
# desired placement of text along with the figure:
# line=-5 ..... Move text over to the right
# side=4 ..... Place text in the right margin
# cex=0.95 ..... Size (font)
# font=2 ..... Bold (font)

```

Mosaic Plot of Primary Payer by Location

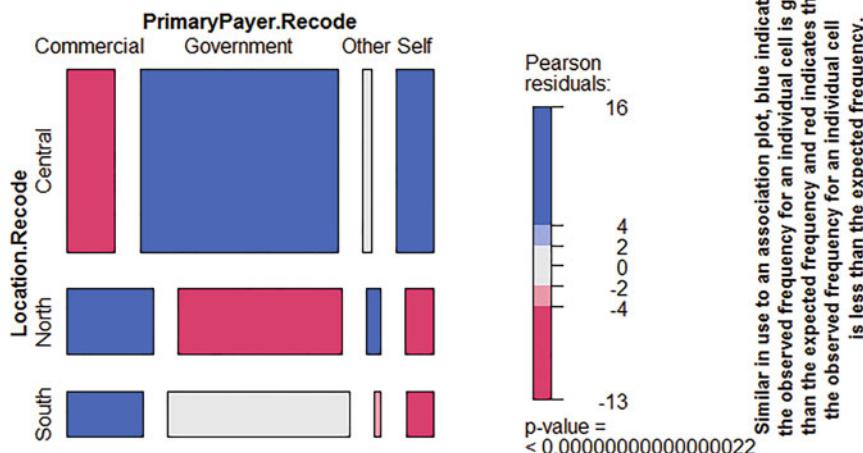


Figure 8.63: Mosaic plot—2

8.9.1.4 Pie Chart

The pie chart is used to represent proportions of breakout groups in view of total counts. For a pie chart, each breakout representation is presented as a wedge-shaped slice of a round pie, hence the name pie chart (Fig. 8.63).

Pie charts are generally discouraged due to the difficulty of comparing the relationships between the pie slices (e.g., wedges). Even so, pie charts are often found in the mass media and it may be necessary to create a pie chart, for public (but not professional) use.

R Input

```
base:::table(HospitalAge021100.df$PrimaryPayer.Recode)
# N for each breakout group
```

R Output

Commercial	Government	Other	Self
6350	19322	1006	3549

R Input

```
base:::prop.table(base:::table(
  HospitalAge021100.df$PrimaryPayer.Recode))
# % for each breakout group
```

R Output

Commercial	Government	Other	Self
0.2100771	0.6392298	0.0332815	0.1174116

Look at the N and percentage for each PrimaryPayer.Recode breakout group and then look at the simple pie chart. Is it possible to do anything but guess about the N and percentage representation for each slice of the pie? What is the scale? Is any degree of accuracy possible with a pie chart? (Fig. 8.64)

R Input

```
par(ask=TRUE)
graphics:::pie(base:::table(
  HospitalAge021100.df$PrimaryPayer.Recode),
  main="Pie Chart of Primary Payer",
  font=2, col=c("red", "cyan", "blue", "black"))
# The number of colors corresponds to the number of
# pie wedges, or breakout groups.
```

R Input

```
base:::table(HospitalAge021100.df$Location.Recode)
# N for each breakout group
```

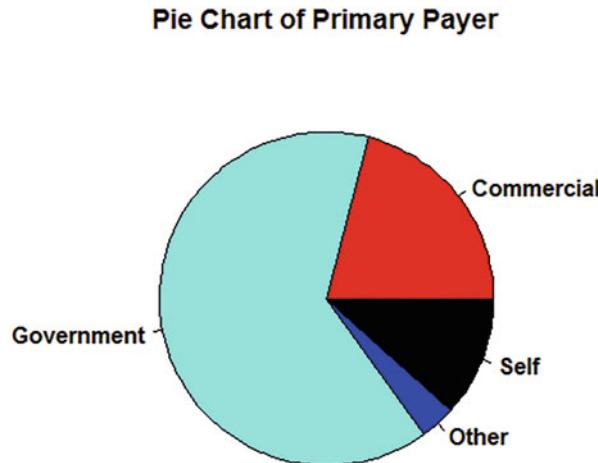


Figure 8.64: Pie chart—1

R Output

```
Central    North    South
 18832     6783     4612
```

R Input

```
base::prop.table(base::table(
  HospitalAge021100.df$Location.Recode))
# % for each breakout group
```

R Output

```
Central    North    South
 0.623019  0.224402  0.152579
```

Look at the N and percentage for each Location.Recode breakout group and then look at the simple pie chart. Is it possible to do anything but guess about the N and percentage representation for each slice of the pie? What is the scale? Is any degree of accuracy possible with a pie chart? (Fig. 8.65)

R Input

```
par(ask=TRUE)
graphics::pie(base::table(
  HospitalAge021100.df$Location.Recode),
  main="Pie Chart of Location",
```

```
font=2, col=c("red", "cyan", "blue"))
# The number of colors corresponds to the number of
# pie wedges, or breakout groups.
```

Pie Chart of Location

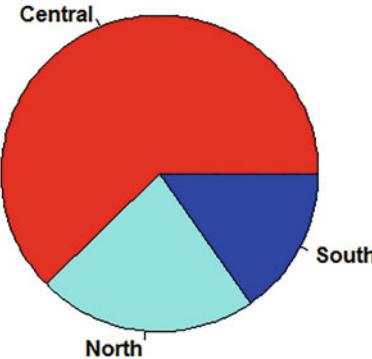


Figure 8.65: Pie chart—2

It is also possible to make a more complex 3D (e.g., three-dimensional) pie chart by using the `plotrix::pie3D()` function. It is interesting to view this type of specialized graphic, but does it improve comprehension of the representation (e.g., either N or percentage) of breakout groups (e.g., slices or wedges of the pie)? (Figs. 8.66 and 8.67)

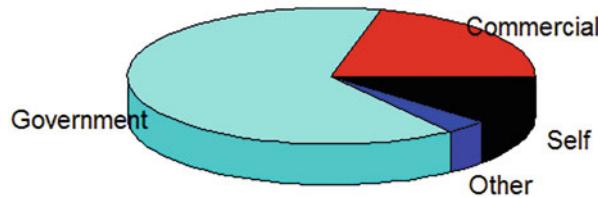
R Input

```
install.packages("plotrix")
library(plotrix)                      # Load the plotrix package.
help(package=plotrix)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

par(ask=TRUE)
plotrix:::pie3D(base::table(
  HospitalAge021100.df$PrimaryPayer.Recode),
  main="3D Pie Chart of Primary Payer",
  labels=c("Commercial", "Government", "Other", "Self"),
  labelcex=1.25, font=2,
  col=c("red", "cyan", "blue", "black"))
mtext("Pie charts are generally avoided since they do not support
      accurate comparisons between different factors (e.g.,
```

```
breakout groups) .",
side=1, cex=0.95, font=2)
```

3D Pie Chart of Primary Payer



Pie charts are generally avoided since they do not support accurate comparisons between different factors (e.g., breakout groups).

Figure 8.66: Pie chart—3

R Input

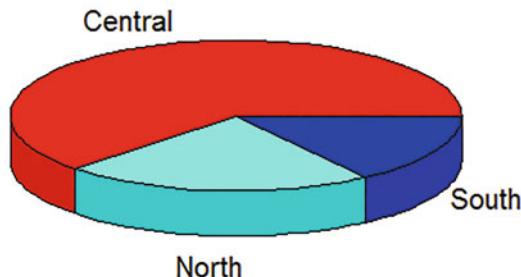
```
par(aspect=TRUE)
plotrix:::pie3D(base:::table(
  HospitalAge021100.df$Location.Recode),
  main="3D Pie Chart of Location",
  labels=c("Central", "North", "South"),
  labelcex=1.25, font=2,
  col=c("red", "cyan", "blue"))
mtext("Pie charts are generally avoided since they do not support
      accurate comparisons between different factors (e.g.,
      breakout groups).",
      side=1, cex=0.95, font=2)
```

8.9.1.5 Waffle Chart (e.g., Squared Pie Chart)

R Input

```
PayerLocation <- base:::table(
  HospitalAge021100.df$PrimaryPayer.Recode,           # Row
  HospitalAge021100.df$Location.Recode)                # Column

PayerLocation
```

3D Pie Chart of Location

Pie charts are generally avoided since they do not support accurate comparisons between different factors (e.g., breakout groups).

Figure 8.67: Pie chart—4

R Output

	Central	North	South
Commercial	3131	2015	1204
Government	12682	3787	2853
Other	579	315	112
Self	2440	666	443

The waffle chart is used to show the contribution of breakout groups compared to total counts. The waffles of a waffle chart show as squares, which many consider easier to understand than the wedges of a pie chart.

Observe output from use of the base::margin.table() function for Primary Payer and Location, recalling how these values are part of the previously enumerated object PayerLocation. Use this output to create enumerated object variables that can be used with a waffle chart:

PriPay - Primary Payer

Locat - Location

R Input

```
base::margin.table(PayerLocation)      # Total count
```

R Output

```
[1] 30227
```

R Input

```
base::margin.table(PayerLocation, 1) # 1 represents rows
```

R Output

Commercial	Government	Other	Self
6350	19322	1006	3549

R Input

```
base::margin.table(PayerLocation, 2) # 2 represents columns
```

R Output

Central	North	South
18832	6783	4612

Using HospitalAge021100.df, create the object PriPay, which represents an object of the names and values for PrimaryPayer.Recode (Fig. 8.68).

R Input

```
PriPay <- c('Commercial (6,350)' =6350,
           'Government (19,322)' =19322,
           'Other (1,006)' =1006,
           'Self (3,549)' =3549)
# Note the single quote character used for this
# syntax. It is the single quote character that
# on most keyboards shows immediately below the
# tilde (e.g., ~) character.
```

```
PriPay
```

R Output

Commercial (6,350)	Government (19,322)
6350	19322
Other (1,006)	Self (3,549)
1006	3549

R Input

```

install.packages("waffle")
library(waffle)                      # Load the waffle package.
help(package=waffle)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

par(ask=TRUE)
waffle::waffle(PriPay/500,
  rows=6, size=0.1, col=c("red", "cyan", "blue", "black"),
  title="Waffle Chart of Primary Payer",
  xlab="With 30,227 Subjects Each Square =~ 500 Subjects")
# Experiment, as needed, to determine the best number of
# subjects for each cell in the waffle chart, or 500 for
# this example.

```

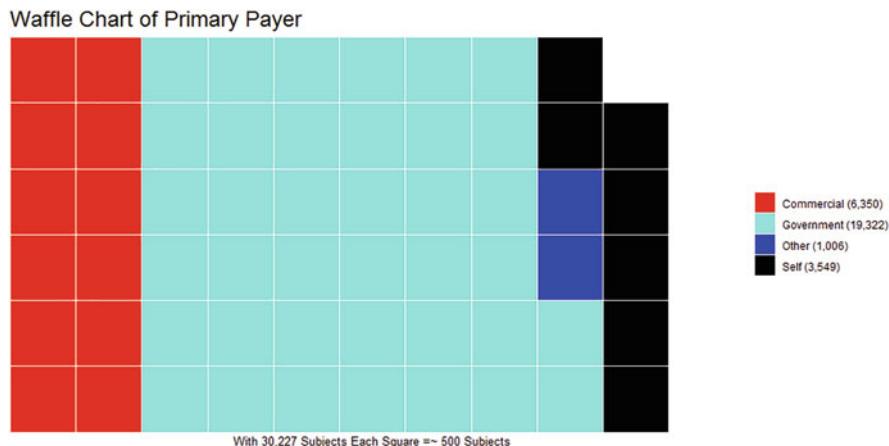


Figure 8.68: Waffle chart—1

Create the object Locat, which represents a numeric object of the names and values for HospitalAge021100.df\$Location.Recode (Fig. 8.69).

R Input

```
Locat <- c('Central (18,832)' =18832,
          'North (6,783)' =6783,
          'South (4,612)' =4612)
```

Locat

R Output

Central (18,832)	North (6,783)	South (4,612)
18832	6783	4612

R Input

```
par(ask=TRUE)
waffle::waffle(Locat/500,
  rows=6, size=0.1, col=c("red", "cyan", "blue"),
  title="Waffle Chart of Location",
  xlab="With 30,227 Subjects Each Square =~ 500 Subjects")
# Experiment, as needed, to determine the best number of
# subjects for each cell in the waffle chart, or 500 for
# this example.
```

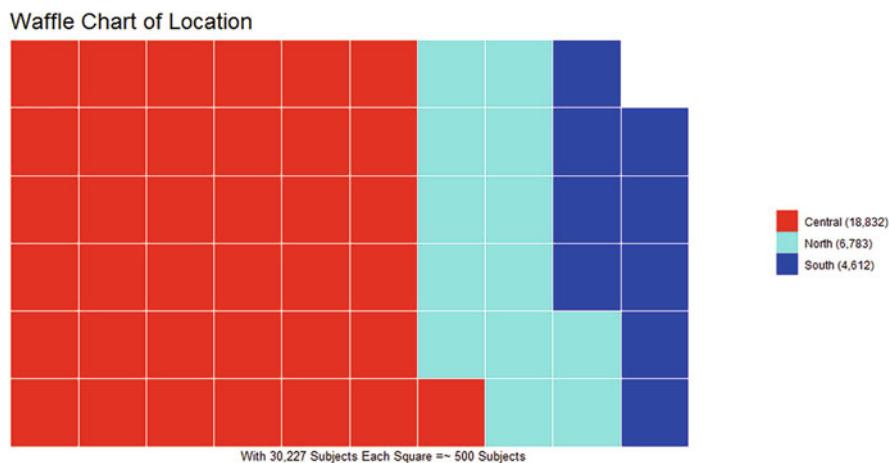


Figure 8.69: Waffle chart—2

Go back to the prior statement that the Pie Chart is not the best choice when showing the representation of breakout groups for factor-type object variables. With this caution, compare the pie chart to the waffle chart and determine which figure, of the two, is easier to understand in terms of knowing with any degree of assurance the N and percentage representation of each breakout group.

8.9.2 Graphical Presentation of Interval and Other (e.g., Measured) Numeric Data

8.9.2.1 Bagplot (e.g., Bivariate Boxplot)

The bagplot is a bivariate boxplot, where 50% of all points are held in the central bag. A fence surrounds the bag and from this fence the remaining points show outward, showing distribution points and extreme values, if there are any.

Immediately below, look at the way the relationship between Diastolic Blood Pressure and Weight is presented using the Bagplot. Does this presentation give a different view than would be seen using other, more traditional, types of figures? Be sure to note how the base::subset() function is used to apply the aplpack::bagplot() function against White Non-Hispanic subjects who ranged in age from 21 to 29 on their last birthday, inclusive (Fig. 8.70).

R Input

```
par(ask=TRUE)
aplyack::bagplot(
  (base::subset(HospitalAge021100.df$DBP,
    (HospitalAge021100.df$AgeLastBirthday >= 21 &
      HospitalAge021100.df$AgeLastBirthday <= 29) &
      (HospitalAge021100.df$Race.Recode == "White") &
      (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic"))),
  (base::subset(HospitalAge021100.df$Lbs,
    (HospitalAge021100.df$AgeLastBirthday >= 21 &
      HospitalAge021100.df$AgeLastBirthday <= 29) &
      (HospitalAge021100.df$Race.Recode == "White") &
      (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic"))),
  main=
  "Bagplot of Diastolic Blood Pressure (DBP) by Weight (Lbs):  

  Age Last Birthday From 21 and 29 Years, Race = White,  

  and Ethnicity = Non-Hispanic",
  xlab="Diastolic Blood Pressure", ylab="Weight (Lbs)",
  na.rm=TRUE, # Accommodate missing data
  show.outlier=TRUE,
  show.whiskers=TRUE,
  show.looppoints=TRUE,
  show.bagpoints=TRUE,
  show.loophull=TRUE,
  show.baghull=TRUE,
  pch=21, # Filled circle plotting symbol
  cex.axis=1.25, # Axis size
```

```
cex.lab=1.25) # Label size
# At the R prompt, key help(bagplot) to see descriptions
# for each argument: show.outlier, show.whiskers, etc. Then,
# decide which arguments meet individual needs.
```

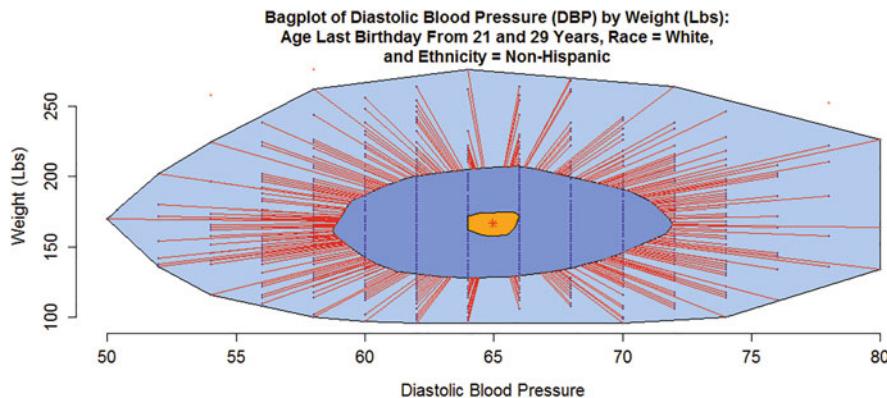


Figure 8.70: Bagplot

8.9.2.2 Beanplot

The beanplot is used to display the distribution pattern of continuous numeric data by borrowing from features found in a density plot and a box plot. The beanplot is especially useful as a graphical tool for when data do not follow expectations of normal distribution, such as bimodal distributions or other data distribution patterns that are unusual and unexpected (Fig. 8.71).

R Input

```
par.ask=TRUE)
par(mfrow=c(1,3)) # 3 figures into a 1 row by 3 column grid
beanplot::beanplot(HospitalAge021100.df$DBP,
  main="Beanplot of Diastolic Blood Pressure (DBP): Mean",
  ylab="Diastolic", font=2, col="red",
  beanlines="mean", beanlinewd=6,
  cex.axis=1.25, cex.lab=1.25)
beanplot::beanplot(HospitalAge021100.df$DBP,
  main="Beanplot of Diastolic Blood Pressure (DBP): Median",
  ylab="Diastolic", font=2, col="red",
  beanlines="median", beanlinewd=6,
  cex.axis=1.25, cex.lab=1.25)
beanplot::beanplot(HospitalAge021100.df$DBP,
  main="Beanplot of Diastolic Blood Pressure (DBP):
```

```
Quantiles", ylab="Diastolic", font=2, col="red",
beanlines="quantiles", beanlinewd=6,
cex.axis=1.25, cex.lab=1.25)
# Look at use of the beanlines argument, with three
# examples in this common figure.
```

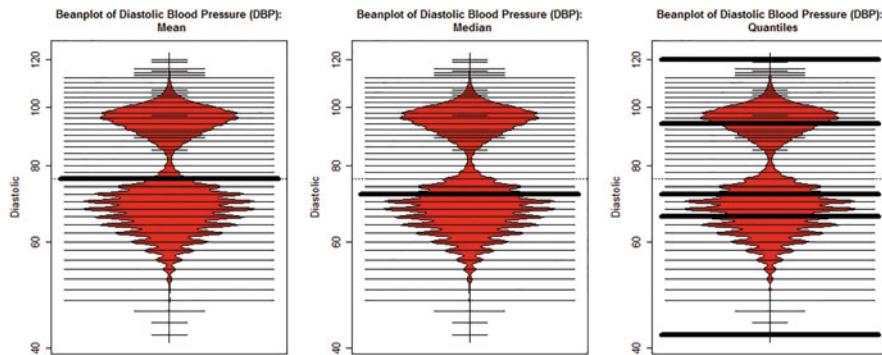


Figure 8.71: Beanplot—1

R Input

```
par(ask=TRUE)
par(mfrow=c(1,3)) # 3 figures into a 1 row by 3 column grid
beanplot::beanplot(HospitalAge021100.df$Lbs,
  main="Beanplot of Weight (Lbs): Mean",
  ylab="Weight (Lbs)", font=2, col="red",
  beanlines="mean", beanlinewd=6,
  cex.axis=1.25, cex.lab=1.25)
beanplot::beanplot(HospitalAge021100.df$Lbs,
  main="Beanplot of Weight (Lbs): Median",
  ylab="Weight (Lbs)", font=2, col="red",
  beanlines="median", beanlinewd=6,
  cex.axis=1.25, cex.lab=1.25)
beanplot::beanplot(HospitalAge021100.df$Lbs,
  main="Beanplot of Weight (Lbs): Quantiles",
  ylab="Diastolic", font=2, col="red",
  beanlines="quantiles", beanlinewd=6,
  cex.axis=1.25, cex.lab=1.25)
```

As with all functions, or for the `beanplot::beanplot()` function in this example, be sure to look at the help page (e.g., key `help(beanplot)` at the R prompt) to learn about the many arguments and their potential use. Arguments such as `main`, `xlab`, `ylab`, `font`, `col`, `cex.axis`, and `cex.lab`, etc., are fairly common and used with many different functions. However, arguments such as `beanlines` and

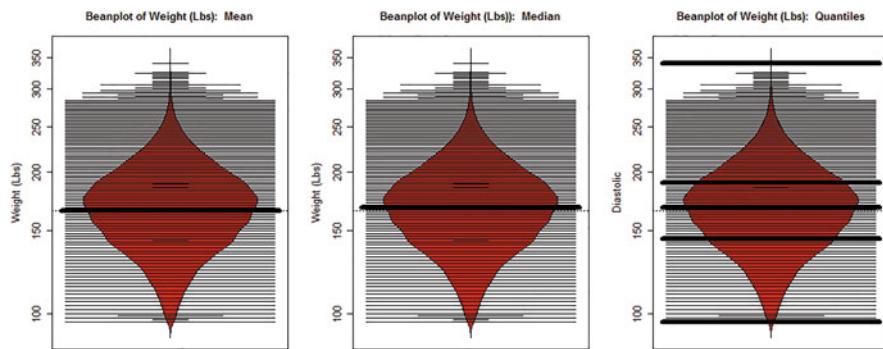


Figure 8.72: Beanplot—2

beanlinewd are unique and specific to this function. Learn about the potential of these arguments to use this type of graphic to full advantage (Fig. 8.72).

8.9.2.3 Beeswarm Plot

The beeswarm plot is used to demonstrate a one-dimensional scatter plot. The beeswarm plot is similar to a stripchart but with improvement on how points are presented.

The beeswarm plot is an excellent graphical alternate to the stripchart, having features that many find useful. A long delay in processing time, however, is a known issue with the way the beeswarm() function is deployed in R when using data from a fairly large dataframe, such as Hospital.df or HospitalAge021100.df. A solution to this problem is to take a representative sample of perhaps 5% of all data from the large dataframe and in turn create and use the new dataframe to generate the Beeswarm figure.

Look at the way the base::sample() function is used to create HospitalAge021100PCT05.df, a new dataframe representing 5% of the data from HospitalAge021100.df:

R Input

```
HospitalAge021100PCT05.df <- HospitalAge021100.df [
  base::sample(nrow(HospitalAge021100.df),
  replace=FALSE,
  size=0.05*nrow(HospitalAge021100.df)),]
# Create the new dataframe, HospitalAge021100PCT05.df.

# Compare descriptive statistics from the new dataframe
# (HospitalAge021100PCT05.df) to the original dataframe
# (HospitalAge021100.df).
```

R Input

```
base::length(HospitalAge021100PCT05.df$Ticket)
```

R Output

```
[1] 1511
```

R Input

```
base::mean(HospitalAge021100PCT05.df$DBP)
```

R Output

```
[1] 77.6486
```

R Input

```
base::mean(HospitalAge021100PCT05.df$Lbs)
```

R Output

```
[1] 167.948
```

R Input

```
base::length(HospitalAge021100.df$Ticket)
```

R Output

```
[1] 30227
```

R Input

```
base::mean(HospitalAge021100.df$DBP)
```

R Output

```
[1] 77.5378
```

R Input

```
base::mean(HospitalAge021100.df$Lbs)
```

R Output

```
[1] 169.035
```

It appears that HospitalAge021100PCT05.df ($N = 1500$ subjects) is in general parity with HospitalAge021100.df ($N = 30,227$). Apply the beeswarm::beeswarm() function against data in HospitalAge021100PCT05.df (Fig. 8.73).

R Input

```
par.ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
beeswarm::beeswarm(DBP ~ Location.Recode,
  data=HospitalAge021100PCT05.df,
  main="Beeswarm Plot of Diastolic Blood Pressure (DBP) by
Location",
  pch=16, col=c("red", "cyan", "blue"),
  method="swarm",
  xlab="", ylab="Diastolic Blood Pressure (DBP)")
par.ask=TRUE)
beeswarm::beeswarm(DBP ~ PrimaryPayer.Recode,
  data=HospitalAge021100PCT05.df,
  main="Beeswarm Plot of Diastolic Blood Pressure (DBP) by
Primary Payer",
  pch=16, col=c("red", "cyan", "blue", "black"),
  method="swarm",
  xlab="", ylab="Diastolic Blood Pressure (DBP)")
```

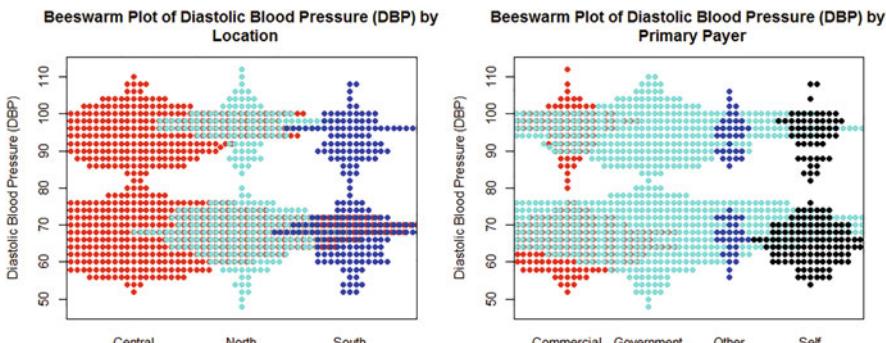


Figure 8.73: Beeswarm plot—1

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
beeswarm::beeswarm(Lbs ~ Location.Rocode,
  data=HospitalAge021100PCT05.df,
  main="Beeswarm Plot of Weight (Lbs) by Location",
  pch=16, col=c("red", "cyan", "blue"),
  method="swarm",
  xlab="", ylab="Weight (Lbs)")
beeswarm::beeswarm(Lbs ~ PrimaryPayer.Rocode,
  data=HospitalAge021100PCT05.df,
  main="Beeswarm Plot of Weight (Lbs) by Primary Payer",
  pch=16, col=c("red", "cyan", "blue", "black"),
  method="swarm",
  xlab="", ylab="Weight (Lbs)")
# Consider the corral="gutter" argument if it is a
# problem to see runaway points that show outside of
# the designated region, which may be the case with
# extreme values that show in breakout groups that
# are larger than others.

```

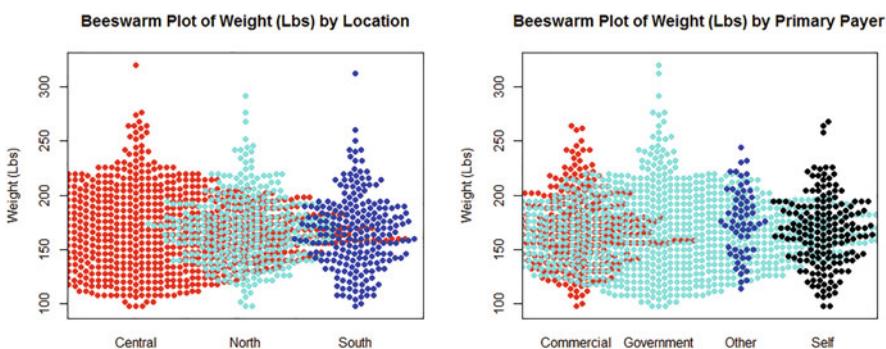


Figure 8.74: Beeswarm plot—2

Notice how the Beeswarm Plot gives a sense of the number of subjects in each breakout group, the general degree of data distribution for each breakout group, and extreme values at both ends of the continuum (e.g., minimum and maximum) for each breakout group—all in one overall figure. The Beeswarm graphic may not be overly-popular, currently, but it deserves more attention and consideration for possible use—at least for formative data review purposes. And, by using a sample of the overall dataset the problem of slow generation of the beeswarm figure is fairly easy to accommodate (Fig. 8.74).

8.9.2.4 Boxplot (e.g., Box-Plot, Box-and-Whiskers Diagram, Box-and-Whiskers Plot)

The boxplot is used to display the distribution pattern of continuous numeric data. The boxplot is a well-established and frequently used graphical tool, based on the presentation of a box or rectangle, that offers a numerical summary of: minimum, first quartile, median, third quartile, and maximum (Figs. 8.75, 8.76, and 8.77).

R Input

```
stats::fivenum(HospitalAge021100.df$DBP, na.rm=TRUE)
# Tukey's 5: minimum, lower-hinge, median,
# upper-hinge, maximum
```

R Output

```
[1] 42 66 72 94 120
```

R Input

```
grDevices::boxplot.stats(HospitalAge021100.df$DBP)
# Boxplot statistics: (1) lower-whisker, lower-hinge,
# median, upper-hinge, upper-whisker; (2) N (e.g., non-NAs);
# (3) extreme notch values; and (4) outliers.
```

R Output

```
$stats
[1] 42 66 72 94 120

$n
[1] 30227

$conf
[1] 71.74554 72.25446

$out
numeric(0)
```

R Input

```

par(ask=TRUE)
graphics::boxplot(HospitalAge021100.df$DBP,
  main="Boxplot of Diastolic Blood Pressure (DBP)",
  col="red", lwd=2, ylim=c(0,150),
  cex.axis=1.25, ylab="Diastolic", cex.lab=1.25)
# Legend #####
savefamily <- par(family="mono") # Courier font
savefont <- par(font=2)          # Bold
legend("topleft",
  legend = c(
    "fivemum() Output"           ,
    "=====",
    "Minimum ..... 042" ,
    "Lower-Hinge ..... 066" ,
    "Median ..... 072" ,
    "Upper-Hinge ..... 094" ,
    "Maximum ..... 120"),
  ncol=1, locator(1), xjust=1, text.col="darkblue",
  cex=0.90, inset=0.01, bty="n")
par(savefamily); par(savefont)
#####
# Legend #####
savefamily <- par(family="mono") # Courier font
savefont <- par(font=2)          # Bold
legend("bottomright",
  legend = c(
    "boxplot.stats() Output" ,
    "=====",
    "Lower-Whisker ..... 042" ,
    "Lower-Hinge ..... 066" ,
    "Median ..... 072" ,
    "Upper-Hinge ..... 094" ,
    "Upper-Whisker ..... 120"),
  ncol=1, locator(1), xjust=0.5, text.col="darkblue",
  cex=0.90, inset=0.01, bty="n")
par(savefamily); par(savefont)
mtext(
  "If any, outliers show as small bubbles beyond the whiskers.", 
  side=1, cex=0.90, font=2)
#####

```

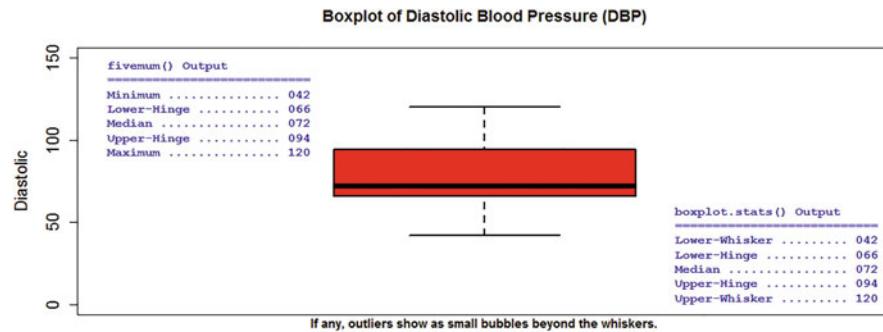


Figure 8.75: Boxplot—1

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::boxplot(HospitalAge021100.df$DBP ~
  HospitalAge021100.df$PrimaryPayer.Recode,
  main="Boxplot of Diastolic Blood Pressure (DBP) by
  Primary Payer", ylab="Diastolic",
  col=c("red", "cyan", "blue", "violet"),
  lwd=2, ylim=c(0,150), cex.axis=0.95, , cex.lab=1.15)
graphics::boxplot(HospitalAge021100.df$DBP ~
  HospitalAge021100.df$Location.Recode,
  main="Boxplot of Diastolic Blood Pressure (DBP) by
  Location", ylab="Diastolic",
  col=c("red", "cyan", "blue"),
  lwd=2, ylim=c(0,150), cex.axis=0.95, cex.lab=1.15)
```

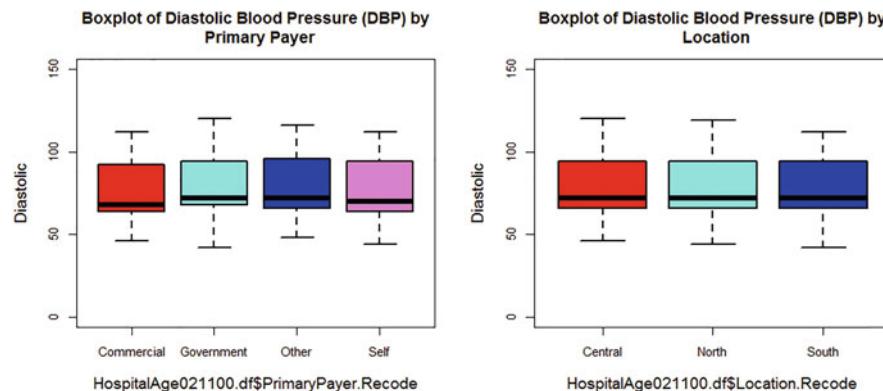


Figure 8.76: Boxplot—2

R Input

```
fivenum(HospitalAge021100.df$Lbs, na.rm=TRUE)
# Tukey's 5: minimum, lower-hinge, median,
# upper-hinge, maximum
```

R Output

```
[1] 96 144 168 190 340
```

R Input

```
boxplot.stats(HospitalAge021100.df$Lbs)
# Boxplot statistics: (1) lower-whisker, lower-hinge,
# median, upper-hinge, upper-whisker; (2) N (e.g., non-NAs);
# (3) extreme notch values; and (4) outliers.
```

R Output

```
$stats
[1] 96 144 168 190 258

$n
[1] 30227

$conf
[1] 167.582 168.418

$out
# A list of outliers was
# deleted to save space.
```

R Input

```
par(ask=TRUE)
graphics::boxplot(HospitalAge021100.df$Lbs,
  main="Boxplot of Weight (Lbs)",
  col="red", lwd=2, ylim=c(0,360),
  cex.axis=1.25, ylab="Weight", cex.lab=1.25)
# Legend #####
savefamily <- par(family="mono") # Courier font
```

```

savefont <- par(font=2) # Bold
legend("topleft",
  legend = c(
    "fivemum() Output", ",",
    "=====",
    "Minimum ..... 096",
    "Lower-Hinge ..... 144",
    "Median ..... 168",
    "Upper-Hinge ..... 190",
    "Maximum ..... 340"),
  ncol=1, locator(1), xjust=1, text.col="darkblue",
  cex=0.90, inset=0.01, bty="n")
par(savefamily); par(savefont)
#####
# Legend #####
savefamily <- par(family="mono") # Courier font
savefont <- par(font=2) # Bold
legend("bottomright",
  legend = c(
    "boxplot.stats() Output", ",",
    "=====",
    "Lower-Whisker ..... 096",
    "Lower-Hinge ..... 144",
    "Median ..... 168",
    "Upper-Hinge ..... 190",
    "Upper-Whisker ..... 258"),
  ncol=1, locator(1), xjust=0.5, text.col="darkblue",
  cex=0.90, inset=0.01, bty="n")
par(savefamily); par(savefont)
mtext(
  "If any, outliers show as small bubbles beyond the whiskers.", side=1, cex=0.90, font=2)
#####

```

R Input

```

par.ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::boxplot(HospitalAge021100.df$Lbs ~
  HospitalAge021100.df$PrimaryPayer.Recode,
  main="Boxplot of Weight (Lbs) by Primary Payer",
  ylab="Weight", col=c("red", "cyan", "blue", "violet"),
  lwd=2, ylim=c(0,360), cex.axis=0.95, , cex.lab=1.15)

```

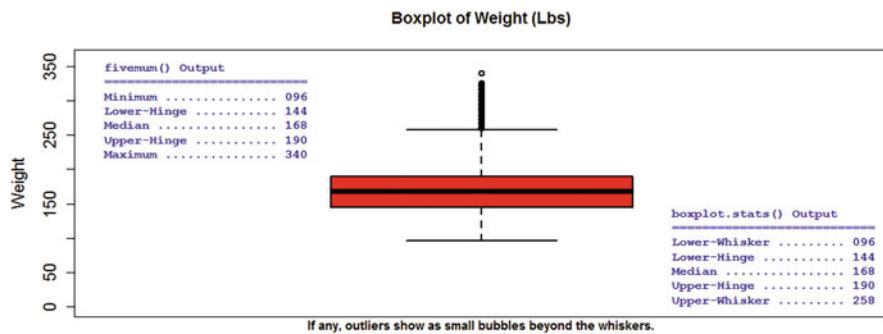


Figure 8.77: Boxplot—3

```

legend("bottom",
       legend = c("Outliers show as small bubbles."),
       ncol=1, locator(1), xjust=1, text.col="darkblue",
       cex=0.90, bty="n")
graphics::boxplot(HospitalAge021100.df$Lbs ~
  HospitalAge021100.df$Location.Recode,
  main="Boxplot of Weight (Lbs) by Location",
  ylab="Weight", col=c("red", "cyan", "blue"),
  lwd=2, ylim=c(0,360), cex.axis=0.95, cex.lab=1.15)
legend("bottom",
       legend = c("Outliers show as small bubbles."),
       ncol=1, locator(1), xjust=1, text.col="darkblue",
       cex=0.90, bty="n")

```

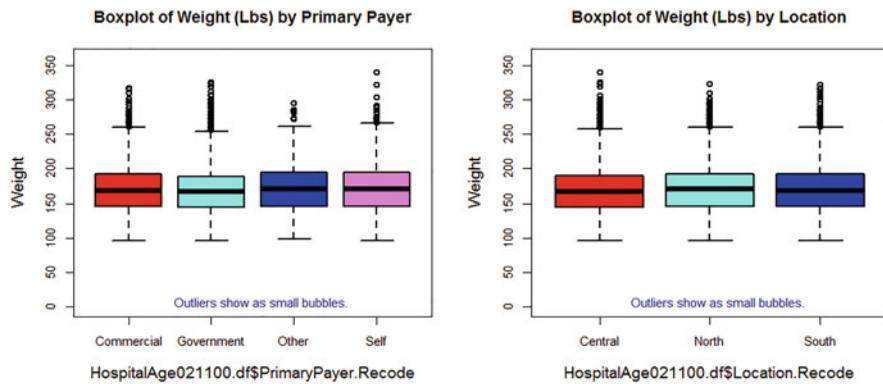


Figure 8.78: Boxplot—4

Compare output for Diastolic Blood Pressure and Weight. Use the `beanplot::beanplot()` function and the `graphics::boxplot()` function (Fig. 8.78):

- For an object variable where there may be concerns about normal distribution, such as `HospitalAge021100.df$DBP`, the `beanplot::beanplot()`

function provides a sense of what may seemingly be a near-bivariate distribution pattern. The `graphics::boxplot()` function, in contrast, with only a box-shaped appearance, does not give a clear sense of this distribution pattern.

- However, for an object where there are a considerable number of outliers, such as `HospitalAge021100.df$Lbs`, the `graphics::boxplot` function provides a sense of the number of outliers and their location along the continuum of data distribution (e.g., whether the outliers predominate near the maximum value or whether the outliers predominate near the minimum value, etc.). The `beanplot::beanplot()` function, in contrast, may provide a sense of the presence of outliers but not with the immediate degree of recognition gained by use of the `graphics::boxplot()` function.

Thus, whether the graphics are ever published or reviewed by an external audience, it is always a good idea to prepare multiple types of figures, to gain perspective that may not always be present if one and only one graphical tool was used (Fig. 8.79).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
beanplot::beanplot(HospitalAge021100.df$DBP,
  main="Beanplot of Diastolic Blood Pressure (DBP): Median",
  ylab="Diastolic", font=2, col="red",
  beanlines="median", beanlinewd=6,
  cex.axis=1.15, cex.lab=1.15)
legend("bottom",
  legend = c(
    Is the data distribution pattern evident?"),
  ncol=1, locator(1), xjust=1, text.col="darkblue",
  cex=0.90, bty="n")
graphics::boxplot(HospitalAge021100.df$DBP,
  main="Boxplot of Diastolic Blood Pressure (DBP)",
  col="red", lwd=2,
  cex.axis=1.15, ylab="Diastolic", cex.lab=1.15)
legend("bottom",
  legend = c(
    Is the data distribution pattern evident?"),
  ncol=1, locator(1), xjust=1, text.col="darkblue",
  cex=0.90, bty="n")
```

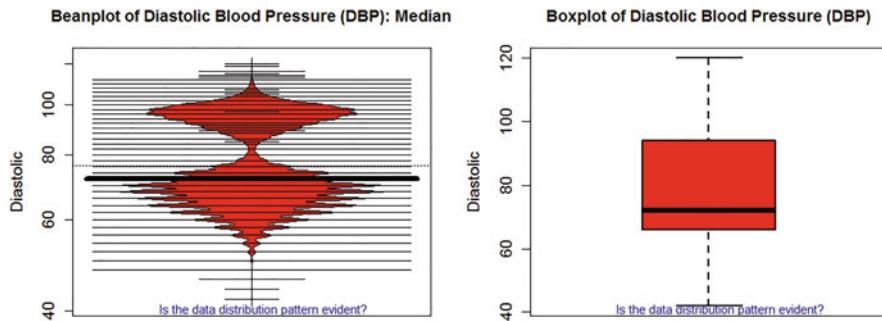


Figure 8.79: Beanplot and boxplot—1

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
beanplot::beanplot(HospitalAge021100.df$Lbs,
  main="Beanplot of Weight (Lbs): Median",
  ylab="Weight", font=2, col="red",
  beanlines="median", beanlinewd=6,
  cex.axis=1.25, cex.lab=1.25)
legend("bottom",
  legend = c(
    How do outliers show?"),
  ncol=1, locator(1), xjust=1, text.col="darkblue",
  cex=0.90, bty="n")
graphics::boxplot(HospitalAge021100.df$Lbs,
  main="Boxplot of Weight (Lbs)",
  col="red", lwd=2, ylim=c(0,360),
  cex.axis=1.25, ylab="Weight", cex.lab=1.25)
legend("bottom",
  legend = c(
    How do outliers show?"),
  ncol=1, locator(1), xjust=1, text.col="darkblue",
  cex=0.90, bty="n")

```

It may be interesting to also consider the notch=TRUE argument when using the graphics::boxplot() function. Generally the notch in a notched boxplot will not be overly visible when working with object variables that show any degree of normal distribution and have a high N, but the notch=TRUE argument is a valuable option when working with object variables that either fail to exhibit normal distribution or have low N. Consider the two demonstration object variables that follow for a brief view of how the notch=TRUE option might be useful (Fig. 8.80).

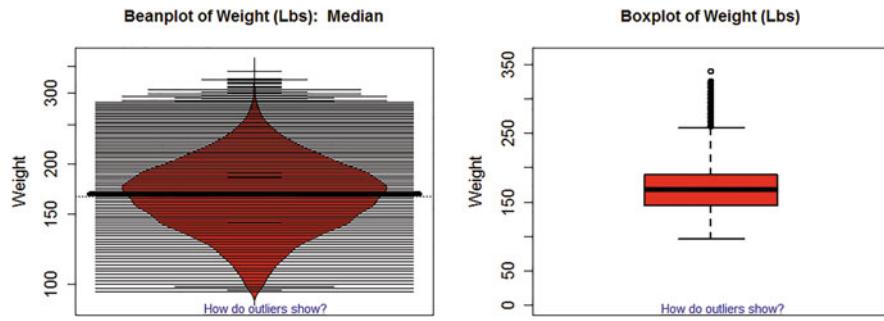


Figure 8.80: Beanplot and boxplot—2

R Input

```
xNormalN100 <- rnorm(100, mean=100, sd=10)
# The set.seed() function was used previously in this R
# session, it is still in effect, and it does not need to
# be repeated.

xNotNormalN100 <- runif(100, min=70, max=130)
```

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::boxplot(xNormalN100, notch=FALSE,
                   main="Normal Distribution N = 100, notch=FALSE")
graphics::boxplot(xNormalN100, notch=TRUE,
                   main="Normal Distribution N = 100, notch=TRUE")
graphics::boxplot(xNotNormalN100, notch=FALSE,
                   main="NotNormal Distribution N = 100, notch=FALSE")
graphics::boxplot(xNotNormalN100, notch=TRUE,
                   main="NotNormal Distribution N = 100, notch=TRUE")
```

When N is fairly low, or $N = 100$ in the above example, note how the notch shows for both the object variable with normal distribution as well as the object variable that does not exhibit normal distribution. However, look below at graphical output when the N is larger, or 1000 in the contrived object variables that follow (Fig. 8.81).

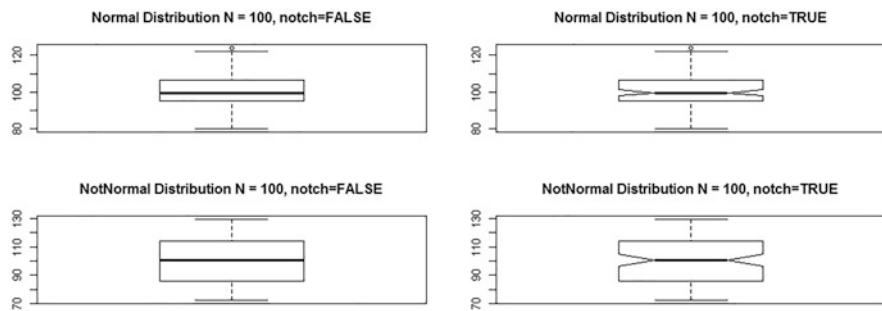


Figure 8.81: Boxplot—5

R Input

```
xNormalN1000 <- rnorm(1000, mean=100, sd=10)
xNotNormalN1000 <- runif(1000, min=70, max=130)
```

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::boxplot(xNormalN1000, notch=FALSE,
  main="Normal Distribution N = 1000, notch=FALSE")
graphics::boxplot(xNormalN1000, notch=TRUE,
  main="Normal Distribution N = 1000, notch=TRUE")
graphics::boxplot(xNotNormalN1000, notch=FALSE,
  main="NotNormal Distribution N = 1000, notch=FALSE")
graphics::boxplot(xNotNormalN1000, notch=TRUE,
  main="NotNormal Distribution N = 1000, notch=TRUE")
```

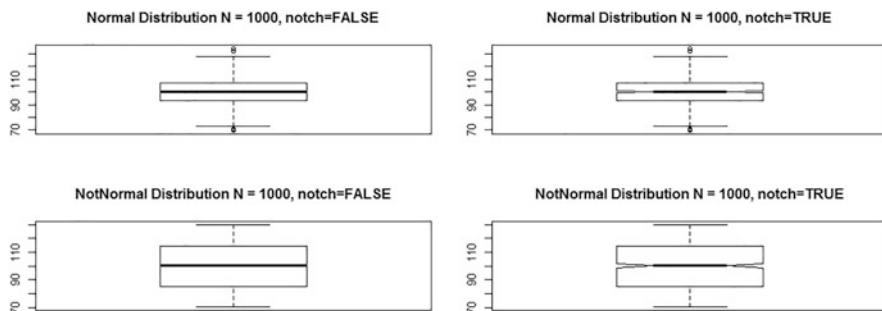


Figure 8.82: Boxplot—6

Is the notch readily evident when $N = 1000$ compared to when $N = 100$? Again, it may be useful to at least consider using the `notch=TRUE` argument when

preparing a boxplot, but as always with R— experiment to see what is best for personal needs and later communication of outcomes with outside readers (Fig. 8.82).

8.9.2.5 Box-Percentile Plot

The box-percentile plot is used to display the distribution pattern of continuous numeric data. The advantage of a box-percentile plot is that the resulting figure closely represents actual data distribution and is not restricted to uniform representation as a box (or rectangle), only. Keeping within the framework of a boxplot, however, the box-percentile plot also highlights the median, 25th, and 75th percentiles, showing lines within the box-percentile plot.

To save space, the Box-Percentile Plot will be shown only for Weight (Lbs) and for breakouts of Weight (Lbs) by Primary Payer and by Location. However, to add some degree of challenge, note how the base::subset() function was applied against the object variable HospitalAge021100.df\$Lbs, to include only those subjects who were between 21 and 29 years old—inclusive (Age on Last Birthday), White (Race), and Non-Hispanic (Ethnicity). This subset, against three levels of selection (Age, Race, and Ethnicity), may show different distribution patterns for the data than what is evident for the far broader dataset that represents all subjects (Figs. 8.83, 8.84, and 8.85).

R Input

```
install.packages("Hmisc")
library(Hmisc)                                # Load the Hmisc package.
help(package=Hmisc)                            # Show the information page.
sessionInfo()                                 # Confirm all attached packages.

savelwd      <- par(lwd=3)                      # Heavy line
savefont     <- par(font=2)                      # Bold
savecex.lab  <- par(cex.lab=1.25) # Label
savecex.axis <- par(cex.axis=1.25) # Axis
par(ask=TRUE)
Hmisc::bpplot(
  base::subset(HospitalAge021100.df$Lbs,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29) &
  (HospitalAge021100.df$Race.Recode == "White") &
  (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic"))),
  main=
  Box-Percentile Plot of Weight (Lbs): Age Last Birthday =
  21 to 29, Race = White, and Ethnicity = Non-Hispanic\n',
```

```

name=FALSE, ylab="Weight (Lbs)")
par(savelwd); par(savefont); par(savecex.lab);
par(savecex.axis)
legend("topleft",
       legend = c(
         "The solid lines mark the 25th percentile,",
         "the median (50th percentile) and the 75th",
         "percentile"),
       ncol=1, locator(1), xjust=1, text.col="darkblue",
       cex=1.25, inset=0.01, bty="n")
# Note how the name argument is set to FALSE
# Note the use of \n to force a line, for spacing

```

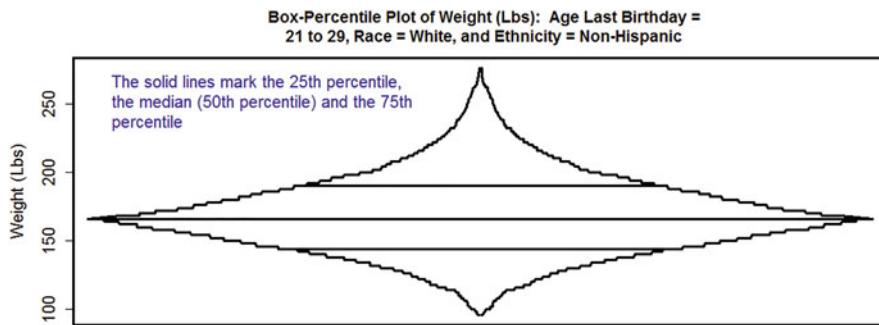


Figure 8.83: Box-percentile plot—1

R Input

```

savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab  <- par(cex.lab=1.25) # Label
savecex.axis <- par(cex.axis=1.25) # Axis
par(ask=TRUE)
Hmisc::bpplot(base::split(
  base::subset(HospitalAge021100.df$Lbs,
    (HospitalAge021100.df$AgeLastBirthday >= 21 &
      HospitalAge021100.df$AgeLastBirthday <= 29) &
      (HospitalAge021100.df$Race.Recode == "White") &
      (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic"))),
  HospitalAge021100.df$PrimaryPayer.Recode),
  main="Box-Percentile Plot of Weight (Lbs) by Primary Payer:
  Age Last Birthday = 21 to 29, Race = White, and
  Ethnicity = Non-Hispanic\n",
  name=TRUE, ylab="Weight (Lbs)")
```

```
par(savelwd); par(savefont); par(savecex.lab);
par(savecex.axis)
# Notice the base::split function, the name argument for
# the Hmisc::bplot() function, and their impact on output.
```

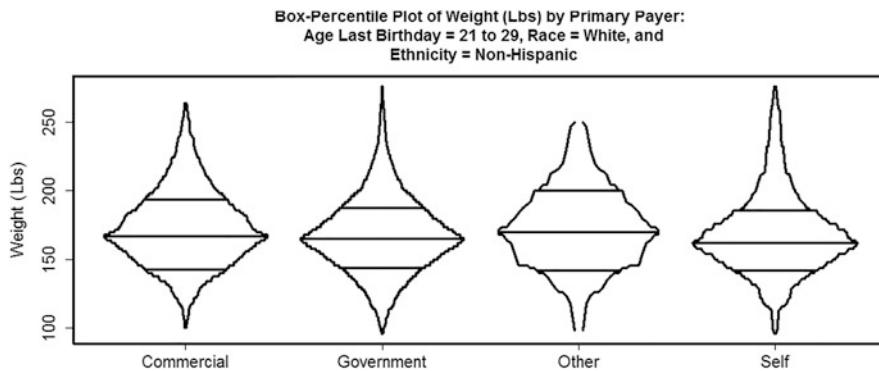


Figure 8.84: Box-percentile plot—2

R Input

```
savelwd      <- par(lwd=3)      # Heavy line
savefont     <- par(font=2)      # Bold
savecex.lab  <- par(cex.lab=1.25) # Label
savecex.axis <- par(cex.axis=1.25) # Axis
par(ask=TRUE)
Hmisc::bpplot(base::split(
  base::subset(HospitalAge021100.df$Lbs,
    (HospitalAge021100.df$AgeLastBirthday >= 21 &
    HospitalAge021100.df$AgeLastBirthday <= 29) &
    (HospitalAge021100.df$Race.Recode == "White") &
    (HospitalAge021100.df$Ethnicity.Recode == "Non-Hispanic"))),
  HospitalAge021100.df$Location.Recode),
  main="Box-Percentile Plot of Weight (Lbs) by Location: Age
  Last Birthday = 21 to 29, Race = White, and
  Ethnicity = Non-Hispanic\n",
  name=TRUE, ylab="Weight (Lbs)")
par(savelwd); par(savefont); par(savecex.lab); par(savecex.axis)
# Notice the base::split function, the name argument for
# the Hmisc::bplot() function, and their impact on output.
```

The purpose of these initial plots is to gain a general sense of the data and to equally look for outliers. In an attempt to look for outliers, the ylim argument has been avoided, so that all data are plotted. Extreme values may or may not be outliers, but they are certainly interesting and demand attention.

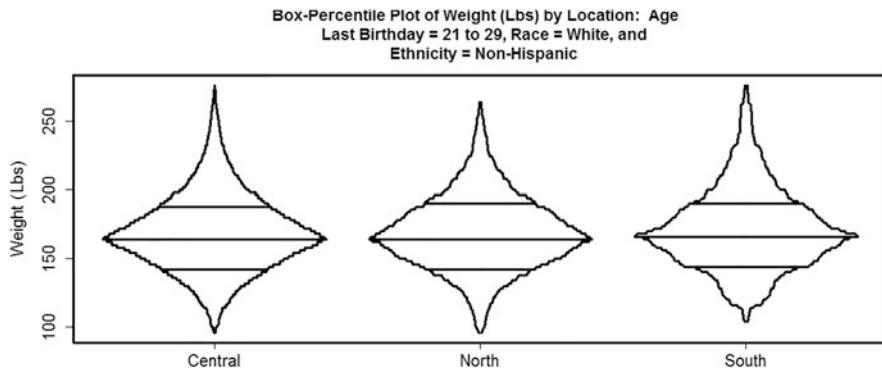


Figure 8.85: Box-percentile plot—3

8.9.2.6 Density Plot

The density plot is used to display the distribution pattern of continuous numeric data. The density plot is a well-established graphical tool, consisting of a line that follows kernel density estimates and in turn a readily evident and easily understood visualization of data distribution (Fig. 8.86).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::plot(stats::density(HospitalAge021100.df$DBP,
  na.rm=TRUE),                                     # na.rm=TRUE is required
  main="Density Plot of Diastolic Blood Pressure (DBP)",
  col="red", lwd=3, cex.axis=1.25)
graphics::plot(stats::density(HospitalAge021100.df$Lbs,
  na.rm=TRUE), main="Density Plot of Weight (Lbs)",
  col="red", lwd=3, cex.axis=1.25)
# Use this graphic to gain a sense of the xlim scale and
# ylim scale.
```

8.9.2.7 Dotplot (e.g., Dotchart)

The dotplot is used to display the distribution pattern of continuous numeric data. The dotplot is very simple and merely presents datapoints as dots, typically as filled circles. The dotplot can be used to represent values for one continuous numeric variable. Frequently, however, the dotplot is also used for comparative purposes, where distributions for breakout groups are presented side-by-side as dotplots.

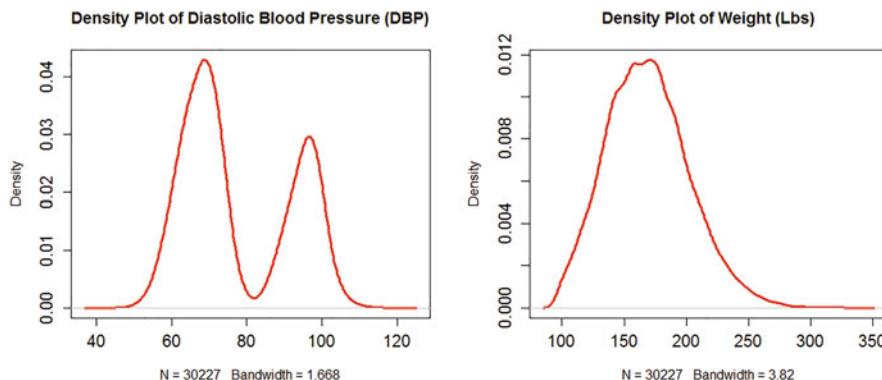


Figure 8.86: Density plot

The graphics::dotchart() function does not work well when N is overly-large. Saying that, for this lesson use the base::subset() function to apply the dotchart only for those subjects who were 21–29 (inclusive) on their last birthday.

The graphics::dotchart() function is generally used only for exploratory purposes and only a few embellishments will be used in this demonstration. Breakouts by Primary Payer and Location will not be displayed since there are other R functions that better serve this purpose (Fig. 8.87).

R Input

```
par.ask=TRUE)
graphics::dotchart(base::subset(HospitalAge021100.df$SBP,
  groups=HospitalAge021100.df$PrimaryPayer,
  HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29),
  main="Dot Plot of Diastolic Blood Pressure for Subjects
  Age 21 to 29 on Their Last Birthday (Inclusive)",
  xlab="Diastolic Blood Pressure (DBP)",
  col="darkred", pch=16, cex.axis=1.25, cex.lab=1.25,
  font=2)
```

R Input

```
par.ask=TRUE)
dotchart(base::subset(HospitalAge021100.df$Lbs,
  groups=HospitalAge021100.df$PrimaryPayer,
  HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29),
  main="Dot Plot of Weight (Lbs) for Subjects Age 21 to 29
  on Their Last Birthday (Inclusive)",
```

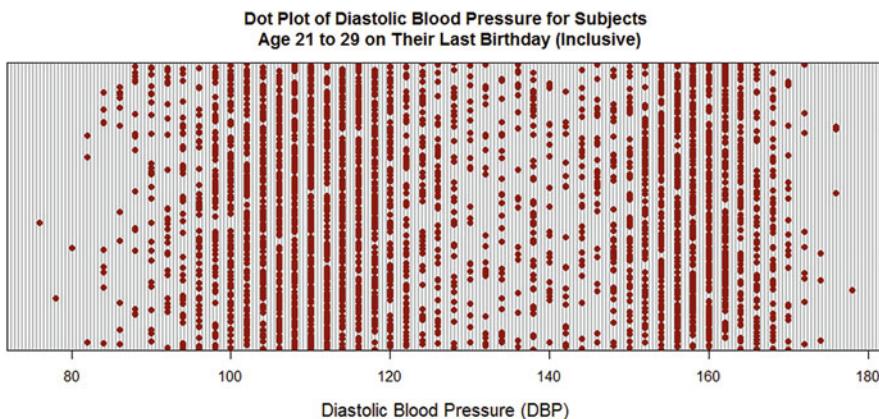


Figure 8.87: Dot plot—1

```
xlab="Weight (Lbs)" ,
col="darkred", pch=16, cex.axis=1.25, cex.lab=1.25,
font=2)
```

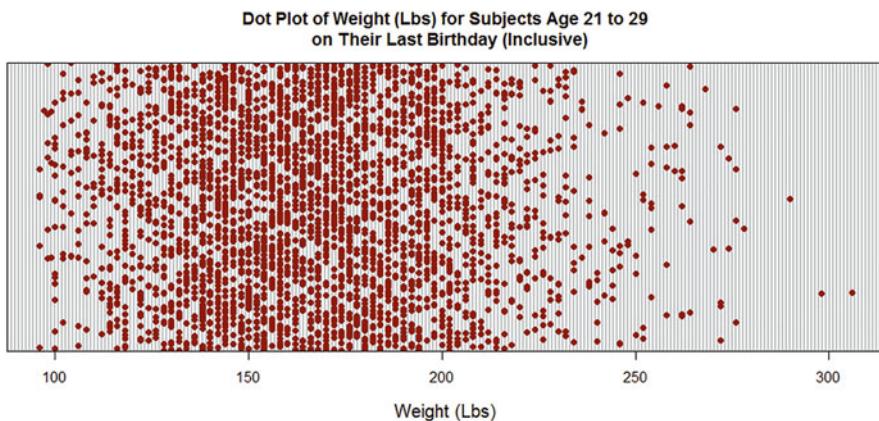


Figure 8.88: Dot plot—2

8.9.2.8 Engelmann–Hecker (EH) Plot

The Engelmann–Hecker (EH) plot is used to display the distribution of numerical data that are grouped, allowing for a fairly easy comparison that in many ways combines the advantages of a dot plot and a box plot. A useful feature of the EH plot is that the data distribution pattern of the breakout groups parallels to a degree what may be expected in a dot plot while presenting the mid-point, or median (Figs. 8.88, 8.89, 8.90, and 8.91).

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
plotrix::ehplot(HospitalAge021100.df$Lbs,
                 HospitalAge021100.df$Location.Recode,
                 main=
                 "Engelmann-Hecker Plot of Weight (Lbs) by
                 Location With Superimposed Boxplot",
                 xlab="Location", ylab="Weight (Lbs)",
                 box=TRUE, boxborder="blue", median=TRUE, pch=20, col="red",
                 offset=0.0005)
                 # Note the box=TRUE argument
graphics::boxplot(HospitalAge021100.df$Lbs ~
                   HospitalAge021100.df$Location.Recode,
                   main="Boxplot of Weight (Lbs) by Location",
                   xlab="Location", ylab="Weight (Lbs)")

```

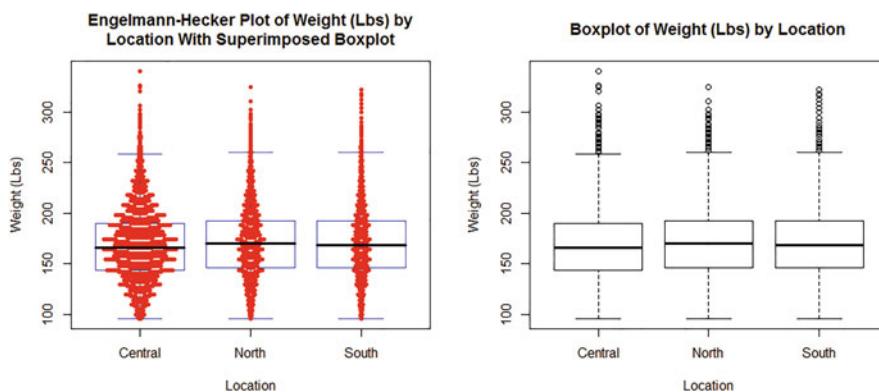


Figure 8.89: Engelmann–Hecker plot and boxplot—1

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
plotrix::ehplot(HospitalAge021100.df$Lbs,
                 HospitalAge021100.df$PrimaryPayer.Recode,
                 main=
                 "Engelmann-Hecker Plot of Weight (Lbs) by Primary
                 Payer Without Superimposed Boxplot",
                 xlab="Primary Payer", ylab="Weight (Lbs)",
                 median=TRUE, pch=20, col="red", offset=0.0005)
graphics::boxplot(HospitalAge021100.df$Lbs ~

```

```
HospitalAge021100.df$PrimaryPayer.Recode,
main="Boxplot of Weight (Lbs) by Primary Payer",
xlab="Primary Payer", ylab="Weight (Lbs)")
```

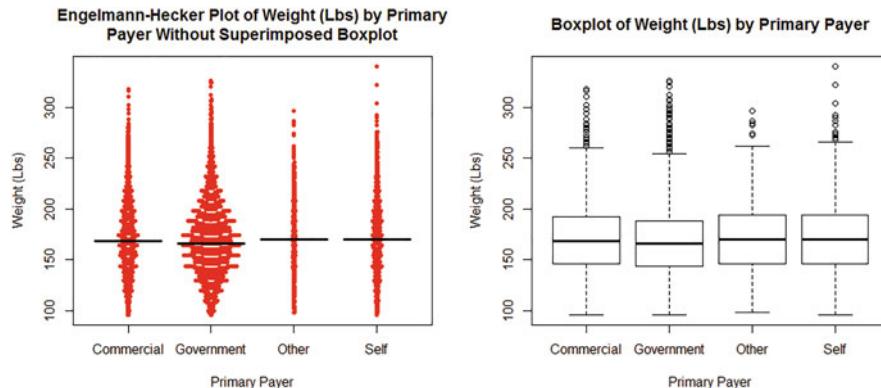


Figure 8.90: Engelmann–Hecker plot and boxplot—2

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
plotrix::ehplot(HospitalAge021100.df$DBP,
  HospitalAge021100.df$PrimaryPayer.Recode,
  main=
  "Engelmann-Hecker Plot of Diastolic Blood Pressure
  (DBP) by Primary Payer With Superimposed Boxplot",
  xlab="Primary Payer", ylab="Diastolic Blood Pressure (DBP)",
  box=TRUE, boxborder="blue", median=TRUE, pch=20, col="red",
  offset=0.0005)
# Note the box=TRUE argument
graphics::boxplot(HospitalAge021100.df$DBP ~
  HospitalAge021100.df$PrimaryPayer.Recode,
  main="Boxplot of Diastolic Blood Pressure (DBP) by
  Primary Payer",
  xlab="Primary Payer", ylab="Diastolic Blood Pressure (DBP)")
```

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
plotrix::ehplot(HospitalAge021100.df$DBP,
  HospitalAge021100.df$Location.Recode,
```

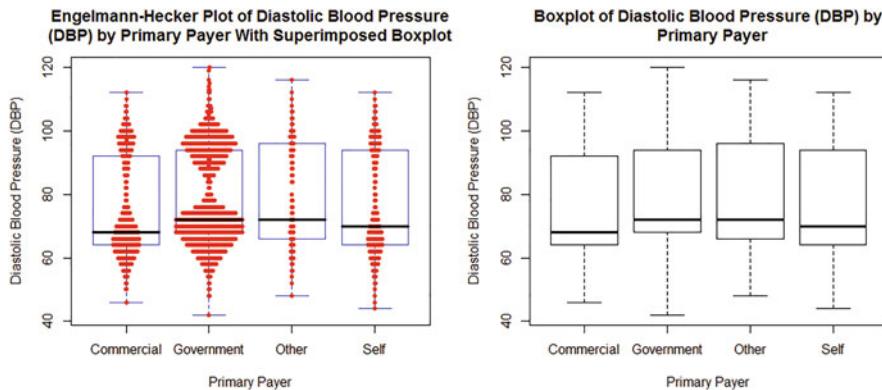


Figure 8.91: Engelmann–Hecker plot and boxplot—3

```
main=""
Engelmann-Hecker Plot of Diastolic Blood Pressure
(DBP) by Location Without Superimposed Boxplot",
xlab="Location", ylab="Diastolic Blood Pressure (DBP)",
median=TRUE, pch=20, col="red", offset=0.0005)
graphics::boxplot(HospitalAge021100.df$DBP ~
HospitalAge021100.df$Location.Recode,
main="Boxplot of Diastolic Blood Pressure
(DBP) by Location",
xlab="Location", ylab="Diastolic Blood Pressure (DBP)")
```

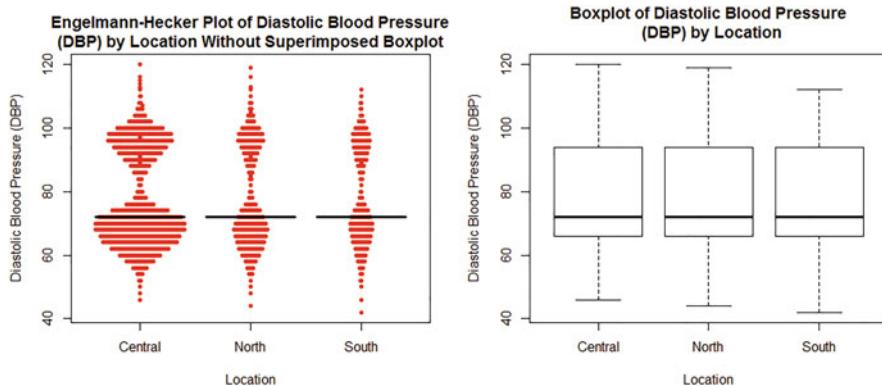


Figure 8.92: Engelmann–Hecker plot and boxplot—4

The Engelmann–Hecker (EH) Plot gives a sense of N as well as distribution and median. Experiment with the value for the offset argument, which is dependent to a large degree on N, to produce the desired figure (Fig. 8.92).

8.9.2.9 Histogram

The histogram is used to display the distribution pattern of continuous numeric data. The histogram is a well-established graphical tool and it is frequently used to give a sense of how the frequency distribution of data shows in selected groupings (e.g., bins) of the continuous variable. As such, a continuous numeric object variable with values that range from 1 to 50 may have frequency counts show in a histogram for groupings of 1–5, 6–10, 11–15, 16–20, etc., until the last bin of frequency counts for all datapoints in the 45–50 bin.

The `graphics::hist()` function can be used to prepare a histogram of either frequencies (use the `freq=TRUE` argument) or density (use the `freq=FALSE` argument). It is less common, however, to see a density-focused histogram. When a density-type graphic is desired, it is more common to use the `graphics::plot()` function wrapped around the `stats::density()` function (Fig. 8.93).

R Input

```
par.ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 1 row by 2 column grid
graphics::hist(HospitalAge021100.df$Lbs, freq=TRUE)
graphics::hist(HospitalAge021100.df$Lbs, freq=FALSE)
graphics::hist(HospitalAge021100.df$DBP, freq=TRUE)
graphics::hist(HospitalAge021100.df$DBP, freq=FALSE)
```

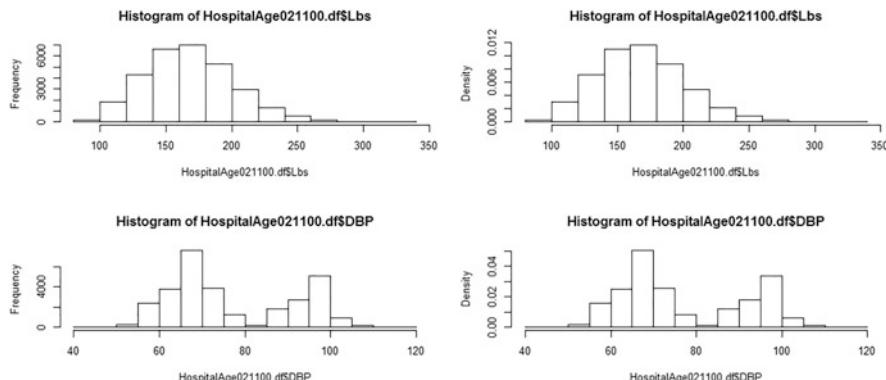


Figure 8.93: Histogram—1

R Input

```
par.ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::hist(HospitalAge021100.df$Lbs,
main="Histogram of Weight (Lbs)", # Title
```

```

freq=TRUE,                                     # Counts, not density
xlab="Weight (Lbs)",                         # X axis label
col="red",                                      # Color for columns
cex.axis=1.25,                                 # Increase size - axis
cex.lab=1.25,                                  # Increase size - label
font=2,                                         # Bold
breaks=25,                                     # Column breakouts
xlim=c(50,350),                                # X axis scale
ylim=c(0,3500))                                 # Y axis scale

graphics::hist(HospitalAge021100.df$DBP,
               main="Histogram of Diastolic Blood Pressure (DBP)",
               freq=TRUE,                                     # Counts, not density
               xlab="Diastolic Blood Pressure (DBP)",       # X axis label
               col="red",                                      # Color for columns
               cex.axis=1.25,                                 # Increase size - axis
               cex.lab=1.25,                                 # Increase size - label
               font=2,                                       # Bold
               breaks=25,                                    # Column breakouts
               xlim=c(30,130),                                # X axis scale
               ylim=c(0,8500))                                # Y axis scale

```

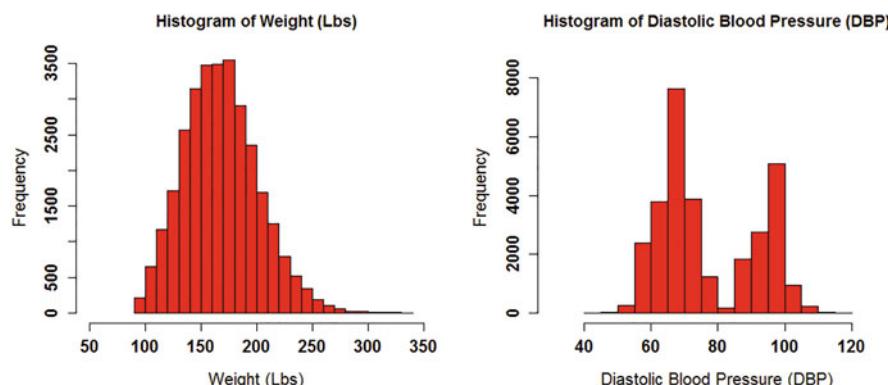


Figure 8.94: Histogram—2

The Hmisc::histbackback() function is used to prepare an exploratory back-to-back histogram of a continuous object variable (e.g., Lbs or DBP) against the binary object variable for gender (e.g., Sex) from the HospitalAge021100.df dataframe (Figs. 8.94 and 8.95).

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid

```

```
Hmisc::histbackback(base::split
(HospitalAge021100.df$Lbs, HospitalAge021100.df$Sex),
probability=FALSE, axes=TRUE,
main = 'Back to Back Histogram of Weight (Lbs) by Sex',
ylab="Weight (Lbs)",
xlim=c(-7500,7500)) # Adjust xlim to needs
# Note the use of ' and not " for the main argument

Hmisc::histbackback(split
(HospitalAge021100.df$DBP, HospitalAge021100.df$Sex),
probability=FALSE, axes=TRUE,
main = 'Back to Back Histogram of Diastolic Blood Pressure
(DBP) by Sex',
ylab="Diastolic Blood Pressure (DBP)",
xlim=c(-7500,7500)) # Adjust xlim to needs
# Note the use of ' and not " for the main argument.
```

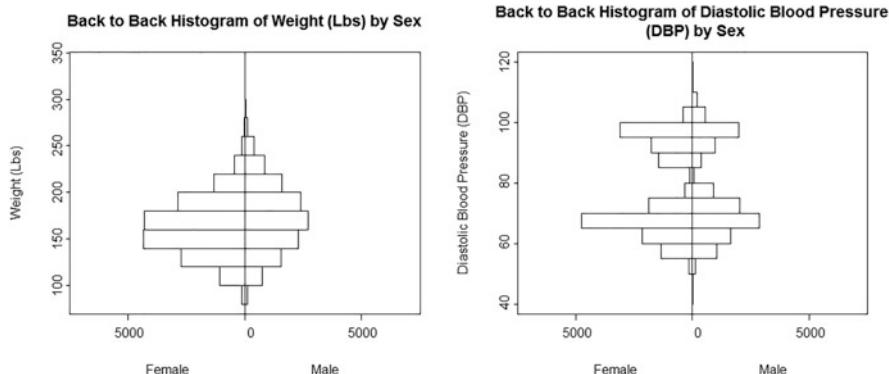


Figure 8.95: Histogram—3

The psych::bi.bars() function is equally useful and produces what many would consider a more visually appealing back-to-back histogram given that the psych::bi.bars() function has arguments that support color (Fig. 8.96).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
psych::bi.bars(HospitalAge021100.df$Lbs,
HospitalAge021100.df$Sex, # Grouping variable
main = "Vertical Back-to-Back Histogram of Weight
(Lbs) by Sex",
ylab="Frequency: Pink = Female and Blue = Male",
horiz=FALSE, # Vertical orientation
```

```

col=c("hotpink", "blue"),           # Female=Pink and Male=Blue
zero=TRUE,                         # Automate the scale
xlab="Weight (Lbs)"                # X axis label
psych::bi.bars(HospitalAge021100.df$DBP,
HospitalAge021100.df$Sex,          # Grouping variable
main = "Vertical Back-to-Back Histogram of Diastolic
Blood Pressure (DBP) by Sex",
ylab="Frequency: Pink = Female and Blue = Male",
horiz=FALSE,                        # Vertical orientation
col=c("hotpink", "blue"),           # Female=Pink and Male=Blue
zero=TRUE,                          # Automate the scale
xlab="Diastolic Blood Pressure (DBP)")

```

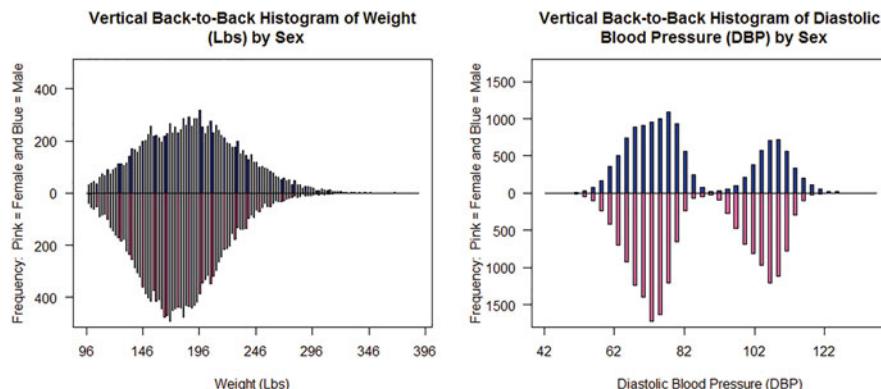


Figure 8.96: Histogram—4

The cumulative histogram is another approach to displaying data, obtaining a summative figure in histogram-format (Fig. 8.97).

R Input

```

HistogramDBP <- graphics::hist(HospitalAge021100.df$DBP)
HistogramDBP$counts <- base::cumsum(HistogramDBP$counts)
# The cumsum() function generates a cumulative sum.

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::hist(HospitalAge021100.df$DBP,
main="Histogram of Diastolic Blood Pressure (DBP)")
graphics::plot(HistogramDBP,
main="Cumulative Histogram of Diastolic Blood
Pressure (DBP)",
xlab="Diastolic Blood Pressure (DBP)", # Label

```

```

col="red",
border="blue",
font=2,
cex.lab=1.25)                                # Bar color
                                                # Bar border color
                                                # Bold
                                                # Axis size

```

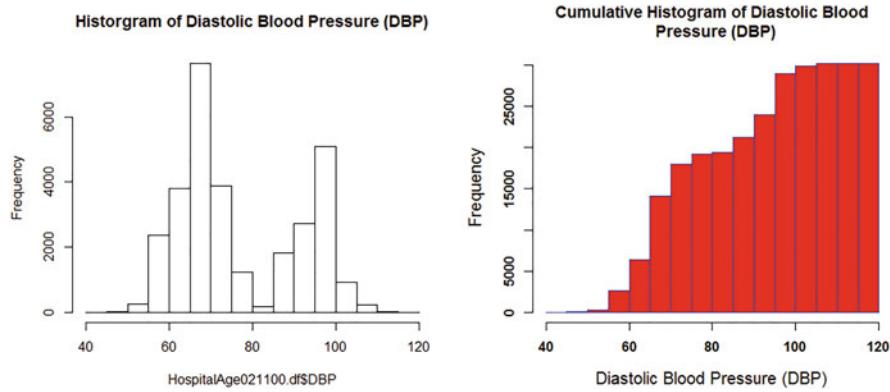


Figure 8.97: Histogram—5

R Input

```

HistogramLbs <- graphics::hist(HospitalAge021100.df$Lbs)
HistogramLbs$counts <- base::cumsum(HistogramLbs$counts)

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::hist(HospitalAge021100.df$Lbs,
               main="Histogram of Weight (Lbs)")
graphics::plot(HistogramLbs,
               main="Cumulative Histogram of Weight (Lbs)",
               xlab="Weight (Lbs)",                                 # Label
               col="red",                                         # Bar color
               border="blue",                                     # Bar border color
               font=2,                                            # Bold
               cex.lab=1.25)                                     # Axis size

```

It is also possible, and extremely helpful, to produce a histogram that has a normal curve and a density curve included in the figure (e.g., a histogram with superimposed normal curve and density curve) (Fig. 8.98).

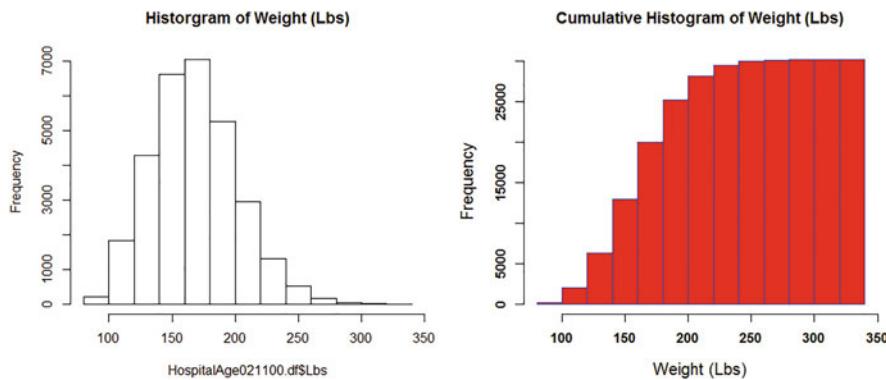


Figure 8.98: Histogram—6

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)         # Bold
savecex.lab  <- par(cex.lab=1.25) # Label
savecex.axis <- par(cex.axis=1.25) # Axis
descr::histkdnc(HospitalAge021100.df$Lbs,
  main="Histogram of Weight (Lbs): Superimposed Normal
  Curve (Blue) and Density Curve (Red)",
  xlab="Weight (Lbs)",
  col="cyan")                      # Color for bars
par(savelwd); par(savefont);
par(savecex.lab); par(savecex.axis)
savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)         # Bold
savecex.lab  <- par(cex.lab=1.25) # Label
savecex.axis <- par(cex.axis=1.25) # Axis
descr::histkdnc(HospitalAge021100.df$DBP,
  main="Histogram of Diastolic Blood Pressure (DBP):
  Superimposed Normal Curve (Blue) and
  Density Curve (Red)",
  xlab="Diastolic Blood Pressure (DBP)",
  col="cyan")                      # Color for bars
par(savelwd); par(savefont);
par(savecex.lab); par(savecex.axis)

```

Even if the figure were never shared with readers or used in a presentation, when preparing a histogram it is best to at least consider adding a normal curve and a density curve to the histogram, to gain a sense of the data and data distribution patterns (Fig. 8.99).

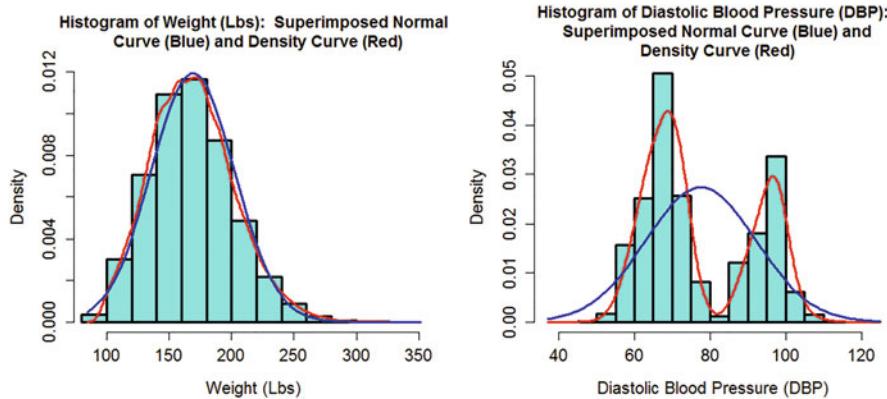


Figure 8.99: Histogram—7

8.9.2.10 Line Chart (e.g., Line Graph)

The line chart is used to display the distribution pattern of continuous numeric data. The visible line of a line chart is prepared as a series of datapoints called markers that are connected by straight line segments. The segments between datapoints can be solid lines, dotted lines, dashed lines, etc. The line chart is a well-established basic tool for graphical presentation of data and it is used in many fields associated with biostatistics along with use in other disciplines.

To produce a simple line chart of an interval (e.g., continuous) object variable such as Diastolic Blood Pressure (DBP) or Weight (Lbs), it is best to put the values for the object variable into a manageable number of breakout groups (e.g., bins, breaks). There is certainly no one and only one correct number of groups to plot on a line chart, but it is both common (and manageable) to have 5–25 groups represented as individual points (e.g., data points) connected by a line in a line chart. Too few markers (e.g., points on the line chart) and the line chart has little value and too many markers (e.g., points on the line chart) and the line chart is simply too busy and it may be difficult to understand.

Although there are a few ways to estimate and finally select an appropriate number of breakout groups (e.g., bins, breaks) to serve as markers for a specific object variable when later put into a line chart, experiment with a few simple histograms. For HospitalAge021100.df prepare a histogram and use breaks=10 and breaks=20, merely to get a feel for the data and guide the most appropriate selection (Fig. 8.100).

R Input

```
par(ask=TRUE)
par(mfrow=c(2,2)) # 4 figures into a 2 row by 2 column grid
graphics::hist(HospitalAge021100.df$Lbs, breaks=10,
```

```

main="Histogram of Lbs:  breaks=10")
graphics::hist(HospitalAge021100.df$Lbs,  breaks=20,
               main="Histogram of Lbs:  breaks=20")
graphics::hist(HospitalAge021100.df$DBP,  breaks=10,
               main="Histogram of DBP:  breaks=10")
graphics::hist(HospitalAge021100.df$DBP,  breaks=20,
               main="Histogram of DBP:  breaks=20")

```

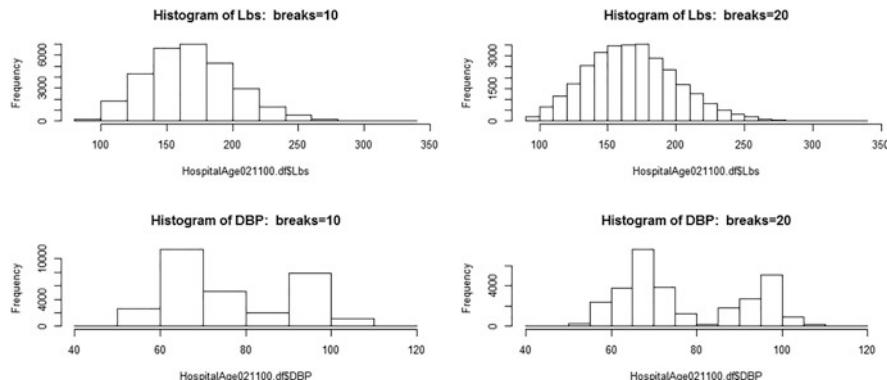


Figure 8.100: Histogram—8

Use the `epiDisplay::pyramid()` function to obtain information on break points and value labels. In this case, ignore the figure associated with the `epiDisplay::pyramid()` function and concentrate instead on the screen output generated by using the `binwidth` argument. After a degree of experimentation it seemed best to use `binwidth=20` for Lbs and `binwidth=5` for DBP but of course recall that these `binwidth` values can be easily changed as needed and would change if other object variables were examined.

R Input

```

par(ask=TRUE)
epiDisplay::pyramid(HospitalAge021100.df$Lbs,
                     HospitalAge021100.df$Sex,
                     main="Lbs - To Gain Line Chart Values",
                     printTable=TRUE, binwidth=20)

```

R Output

HospitalAge021100.df\$Lbs	Female	Male
[80,100]	123	93
(100,120]	1075	743

(120,140]	2727	1549
(140,160]	4344	2273
(160,180]	4331	2708
(180,200]	2891	2373
(200,220]	1345	1590
(220,240]	468	845
(240,260]	154	374
(260,280]	45	120
(280,300]	8	30
(300,320]	1	12
(320,340]	0	5

Use the output gained from the argument `printTable=TRUE` and then merely add the values for Female and Male to obtain the TOTAL for each breakout group, gained from `binwidth=20` in this example.

HospitalAge021100.df\$Lbs	Female	Male	TOTAL (Hand Calculated)
[80,100]	123	93	0216
(100,120]	1075	743	1818
(120,140]	2727	1549	4276
(140,160]	4344	2273	6617
(160,180]	4331	2708	7039
(180,200]	2891	2373	5264
(200,220]	1345	1590	2935
(220,240]	468	845	1313
(240,260]	154	374	0528
(260,280]	45	120	0165
(280,300]	8	30	0038
(300,320]	1	12	0013
(320,340]	0	5	0005

Using this information gained from applying the `epiDisplay::pyramid()` function against `HospitalAge021100.df$Lbs` and then adding the breakout frequency counts for Female and Male into TOTAL, construct a simple object variable (`LbsBreakouts`) and add an axis from the breakouts generated by using the `printTable=TRUE` argument (Fig. 8.101).

R Input

```
LbsBreakouts <- c(
  0216, 1818, 4276, 6617, 7039, 5264, 2935, 1313,
  0528, 0165, 0038, 0013, 0005)
# Create an object variable (LbsBreakouts) using
# output from the epiDisplay::pyramid() function
```

```

par(aspect=TRUE)
graphics::plot(LbsBreakouts,
  main="Line Chart of Weight (Lbs)",
  pch=21,                                # Plotting symbol
  type="b",                                # Display points and lines
  lty=1,                                   # Solid line
  lwd=6,                                   # Thick line
  col=c("red"),                            # Color
  xlab="Breakout Values of Weight (Lbs)",
  ylab="Frequency of All Subjects",
  font=2,                                   # Bold
  cex.lab=1.25,                            # Size - labels
  cex.axis=1.25,                            # Size - axis
  ann=TRUE,                                # Annotation - title, axis label
  xaxt='n')                                # Suppress (y/n) default X/Y axis
# Note below how at=1:13 is used for X axis labels.
axis(
  1,                                     # Axis location - bottom
  at=1:13,                               # Location of tick marks
  las=0,                                   # Labels parallel to axis
  col=c("darkblue"),                      # Line color
  font=2,                                   # Bold
  cex.axis=0.75,                           # Size
  tck=-0.05,                               # Length of tick mark
  labels=c(                                # Label text
    "080-100", "100-120", "120-140", "140-160",
    "160-180", "180-200", "200-220", "220-240",
    "240-260", "260-280", "280-300", "300-320",
    "320-340"))

```

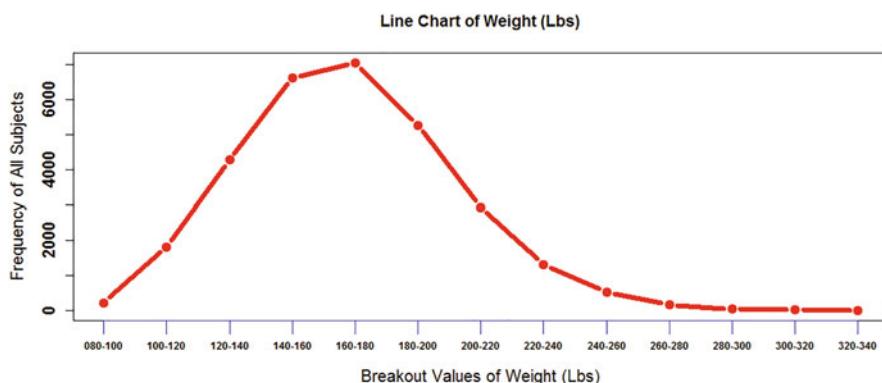


Figure 8.101: Line chart—1

R Input

```
par.ask=TRUE)
epiDisplay::pyramid(HospitalAge021100.df$DBP,
  HospitalAge021100.df$Sex,
  main="DBP - To Gain Line Chart Values",
  printTable=TRUE, binwidth=5)
```

R Output

HospitalAge021100.df\$DBP	Female	Male
[40,45]	0	2
(45,50]	13	11
(50,55]	148	112
(55,60]	1351	1028
(60,65]	2156	1633
(65,70]	4767	2878
(70,75]	1864	2020
(75,80]	346	885
(80,85]	115	64
(85,90]	1446	375
(90,95]	1782	961
(95,100]	3097	1985
(100,105]	396	540
(105,110]	31	196
(110,115]	0	21
(115,120]	0	4

Use the output gained from printTable=TRUE and then merely add the values for Female and Male to obtain the TOTAL for each breakout group, gained from binwidth=5 in this example.

HospitalAge021100.df\$DBP	Female	Male	TOTAL (Hand Calculated)
[40,45]	0	2	0002
(45,50]	13	11	0024
(50,55]	148	112	0260
(55,60]	1351	1028	2379
(60,65]	2156	1633	3789
(65,70]	4767	2878	7645
(70,75]	1864	2020	3884
(75,80]	346	885	1231
(80,85]	115	64	0179
(85,90]	1446	375	1821
(90,95]	1782	961	2743

(95, 100]	3097	1985	5082
(100, 105]	396	540	0936
(105, 110]	31	196	0227
(110, 115]	0	21	0021
(115, 120]	0	4	0004

Using this information gained from applying the `epiDisplay::pyramid()` function against `HospitalAge021100.df$DBP` and then adding the breakout frequency counts for Female and Male into TOTAL, construct a simple object variable (`DBPBreakouts`) and add an axis from the breakouts generated by using the `printTable=TRUE` argument (Fig. 8.102).

R Input

```
labels=c( # Label text
  "040-045", "045-050", "050-055", "055-060",
  "060-065", "065-070", "070-075", "075-080",
  "080-085", "085-090", "090-095", "095-100",
  "100-105", "105-110", "110-115", "115-120"))
```

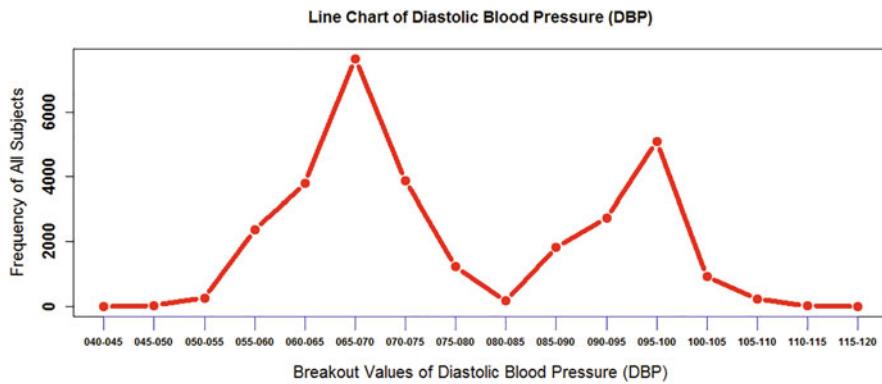


Figure 8.102: Line chart—2

Along with the histogram, the line chart is one of the most frequently seen figures used to communicate findings. Other figure types may be unfamiliar to the general reader, but the histogram and line chart are frequently seen in mass media and there is value in their use.

8.9.2.11 Pirate Plot

The Pirate Plot deserves attention in that it was developed to combine a sense of original data, descriptive statistics, and inferential data. Look below at the fairly detailed graphics that combine in one convenient figure a good view of the data, dispersion of the data, and side-by-side comparisons of one breakout group to another breakout group (Figs. 8.103 and 8.104).

R Input

```
install.packages("yarrr")
library(yarrr) # Load the yarrr package.
help(package=yarrr) # Show the information page.
sessionInfo() # Confirm all attached packages.

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 2 row by 2 column grid
yarrr::pirateplot(formula=Lbs ~ PrimaryPayer.Recode,
  data = HospitalAge021100.df, theme=1, # Background colors
```

```
main ="Pirate Plot of Weight (Lbs) by Primary Payer",
xlab="Primary Payer", ylab="Weight (Lbs)", ylim=c(0,350),
cex.lab=1.25, cex.axis=1.25, cex.names=1.10)
yarrr::pirateplot(formula = Lbs ~ Location.Recode,
data = HospitalAge021100.df, theme=1, # Background colors
main ="Pirate Plot of Weight (Lbs) by Location",
xlab="Location", ylab="Weight (Lbs)", ylim=c(0,350),
cex.lab=1.25, cex.axis=1.25, cex.names=1.10)
```

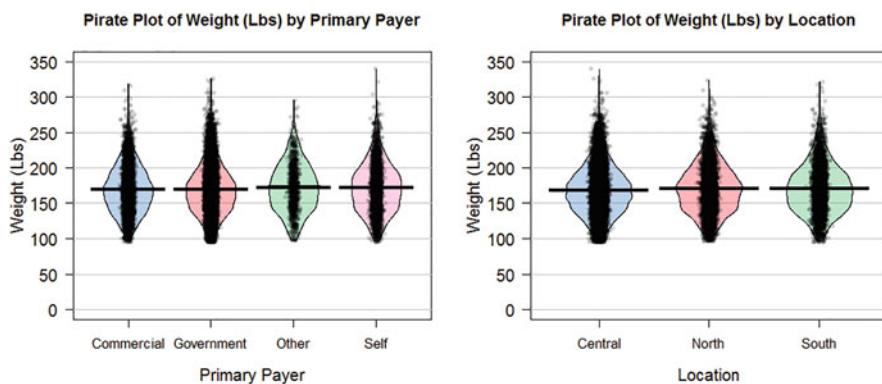


Figure 8.103: Pirate plot—1

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 2 row by 2 column grid
yarrr::pirateplot(formula = DBP ~ PrimaryPayer.Recode,
data = HospitalAge021100.df, theme=1, # Background colors
main ="Pirate Plot of Diastolic Blood Pressure (DBP) by Primary Payer", xlab="Primary Payer",
ylab="Diastolic Blood Pressure (DBP)", ylim=c(0,125),
cex.lab=1.25, cex.axis=1.25, cex.names=1.10)
yarrr::pirateplot(formula = DBP ~ Location.Recode,
data = HospitalAge021100.df, theme=1, # Background colors
main ="Pirate Plot of Diastolic Blood Pressure (DBP) by Location", xlab="Location",
ylab="Diastolic Blood Pressure (DBP)", ylim=c(0,125),
cex.lab=1.25, cex.axis=1.25, cex.names=1.10)
```

8.9.2.12 Quantile-Quantile (Q-Q) Plot

The quantile-quantile (Q-Q) plot is used to plot quantiles in an effort to display distributions. Specific to R, the stats::qqnorm() function is used to prepare a

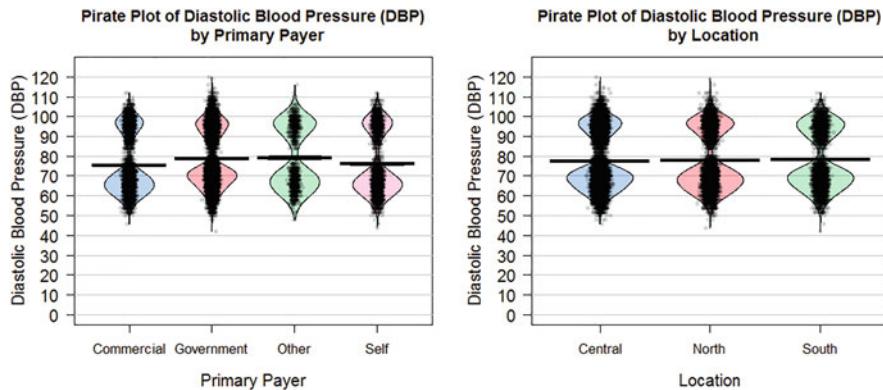


Figure 8.104: Pirate plot—2

Q-Q Plot and by this action provide a sense of data distribution patterns: (1) data either show adherence to a normal distribution pattern or (2) data show deviance away from normal distribution (Fig. 8.105).¹⁸

R Input

```
base::length(base::subset(HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29)))
```

R Output

```
[1] 3288
```

R Input

```
stats::shapiro.test(base::subset(HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29)))
# Shapiro-Wilk Normality Test
```

R Output

```
Outcome: p-value < 0.0000000000000002
```

¹⁸Before preparing a Q-Q Plot, determine N and a normality test statistic. Then, consider the impact of a large N on the practical interpretation of normality test statistics.

R Input

```
base::length(base::subset(HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
   HospitalAge021100.df$AgeLastBirthday <= 29) &
   (HospitalAge021100.df$Race.Recode == "White")))
```

R Output

```
[1] 1915
```

R Input

```
stats::shapiro.test(base::subset(HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
   HospitalAge021100.df$AgeLastBirthday <= 29) &
   (HospitalAge021100.df$Race.Recode == "White")))
# Shapiro-Wilk Normality Test
```

R Output

```
Outcome: p-value = 0.000000000000112
```

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
stats::qqnorm(base::subset(HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
   HospitalAge021100.df$AgeLastBirthday <= 29)),
  main="DBP - Age = 021 to 029 (N = 3,288)",
  xlim=c(-4,4), ylim=c(0,125), font.axis=2, font.lab=2)
stats::qqline(base::subset(HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
   HospitalAge021100.df$AgeLastBirthday <= 29)),
  col="red", lwd=4, lty=2)
# SBP - Age = 021 to 029, only
stats::qqnorm(base::subset(HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
   HospitalAge021100.df$AgeLastBirthday <= 29) &
   (HospitalAge021100.df$Race.Recode == "White")),
```

```

main="DBP - Age = 021 to 029 AND Race = White (N = 1,915)",
xlim=c(-4,4), ylim=c(0,125), font.axis=2, font.lab=2)
stats::qqline(base::subset(HospitalAge021100.df$DBP,
(HospitalAge021100.df$AgeLastBirthday >= 21 &
HospitalAge021100.df$AgeLastBirthday <= 29) &
(HospitalAge021100.df$Race.Recode == "White")),
col="red", lwd=4, lty=2)
# SBP - Age = 021 to 029 AND Race = White

```

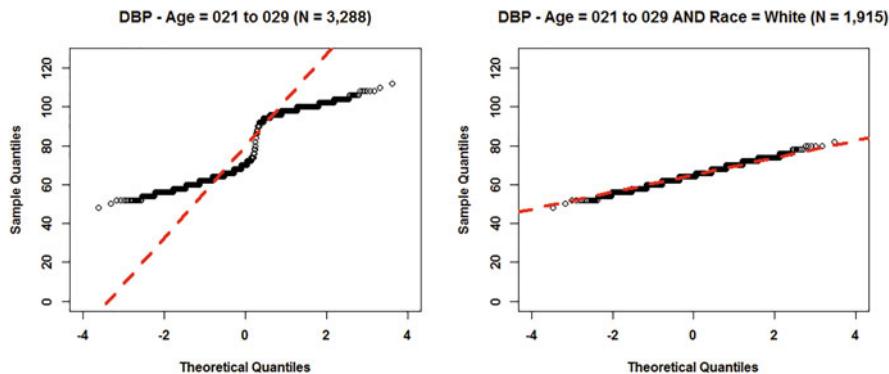


Figure 8.105: Quantile-quantile plot—1

R Input

```

par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
stats::qqnorm(base::subset(HospitalAge021100.df$Lbs,
(HospitalAge021100.df$AgeLastBirthday >= 21 &
HospitalAge021100.df$AgeLastBirthday <= 29)),
main="Lbs - Age = 021 to 029 (N = 3,288)",
xlim=c(-4,4), ylim=c(0,325), font.axis=2, font.lab=2)
stats::qqline(base::subset(HospitalAge021100.df$Lbs,
(HospitalAge021100.df$AgeLastBirthday >= 21 &
HospitalAge021100.df$AgeLastBirthday <= 29)),
col="red", lwd=4, lty=2)
# Lbs - Age = 021 to 029, only
stats::qqnorm(base::subset(HospitalAge021100.df$Lbs,
(HospitalAge021100.df$AgeLastBirthday >= 21 &
HospitalAge021100.df$AgeLastBirthday <= 29) &
(HospitalAge021100.df$Race.Recode == "White")),
main="Lbs - Age = 021 to 029 AND Race = White (N = 1,915)",
xlim=c(-4,4), ylim=c(0,325), font.axis=2, font.lab=2)
stats::qqline(base::subset(HospitalAge021100.df$Lbs,

```

```
(HospitalAge021100.df$AgeLastBirthday >= 21 &
HospitalAge021100.df$AgeLastBirthday <= 29) &
(HospitalAge021100.df$Race.Recode == "White")),
col="red", lwd=4, lty=2)
# Lbs - Age = 021 to 029 AND Race = White
```

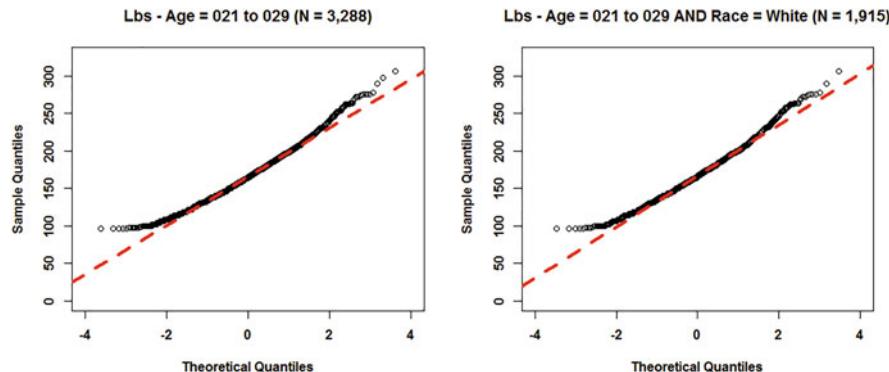


Figure 8.106: Quantile-quantile plot—2

Because Q-Q plots are focused on data distribution, they are of extreme importance as a diagnostic tool. Yet, their nature does not communicate well and are usually only published or presented for the benefit of others who are well-experienced in biostatistics (Fig. 8.106).

8.9.2.13 Scatter Plot (e.g., Scatterplot, Scatter Diagram)

The scatter plot is used to display the correlation (e.g., association), if any, between two separate variables. To achieve this aim, the values for each of the two variables are plotted along an axis. The values for one variable are plotted along the X axis and values for the other variable are plotted along the Y axis. The resulting values, when plotted, resemble the distribution pattern of pellets fired from a shotgun (e.g., scatter gun, hence the name scatter plot) when there is no correlation between the two variables. The plotted values along the X axis and Y axis approximate a straight line when there is a strong correlation between the two variables.

Before a scatter plot is prepared, it is common to first determine the correlation coefficient for each variable, to gain a sense of the statistical association between variables in question.

R Input

```
stats::cor(HospitalAge021100.df$SBP,  
HospitalAge021100.df$DBP)
```

R Output

```
[1] 0.854961
```

R Input

```
stats::cor(HospitalAge021100.df$SBP,  
HospitalAge021100.df$Lbs)
```

R Output

```
[1] 0.0742602
```

R Input

```
stats::cor(HospitalAge021100.df$DBP,  
HospitalAge021100.df$Lbs)
```

R Output

```
[1] 0.0670301
```

R Input

```
par(ask=TRUE)  
par(mfrow=c(2,2)) # 3 figures into a 1 row by 2 column grid  
# with one cell empty  
graphics::plot(HospitalAge021100.df$SBP,  
HospitalAge021100.df$DBP,  
main="Scatter Plot of Systolic Blood Pressure (SBP)  
by Diastolic Blood Pressure (DBP)",  
font=2, col="red", xlab="SBP", ylab="DBP",  
pch=19, cex.axis=1.25, cex.lab=1.25)  
legend("topleft",  
legend = c("Pearson's r = 0.854961"),  
ncol=1, locator(1), xjust=1, text.col="darkblue", cex=1.05,
```

```

inset=0.001, bty="n")
graphics::plot(HospitalAge021100.df$SBP,
HospitalAge021100.df$Lbs,
main="Scatter Plot of Systolic Blood Pressure (SBP)
by Weight (Lbs)",
font=2, col="red", xlab="SBP", ylab="Lbs",
pch=19, cex.axis=1.25, cex.lab=1.25)
legend("topleft",
legend = c("Pearson's r = 0.0742602"),
ncol=1, locator(1), xjust=1, text.col="darkblue", cex=1.05,
inset=0.001, bty="n")
graphics::plot(HospitalAge021100.df$DBP,
HospitalAge021100.df$Lbs,
main="Scatter Plot of Diastolic Blood Pressure (DBP)
by Weight (Lbs)",
font=2, col="red", xlab="DBP", ylab="Lbs",
pch=19, cex.axis=1.25, cex.lab=1.25)
legend("topleft",
legend = c("Pearson's r = 0.0670301"),
ncol=1, locator(1), xjust=1, text.col="darkblue", cex=1.05,
inset=0.001, bty="n")

```

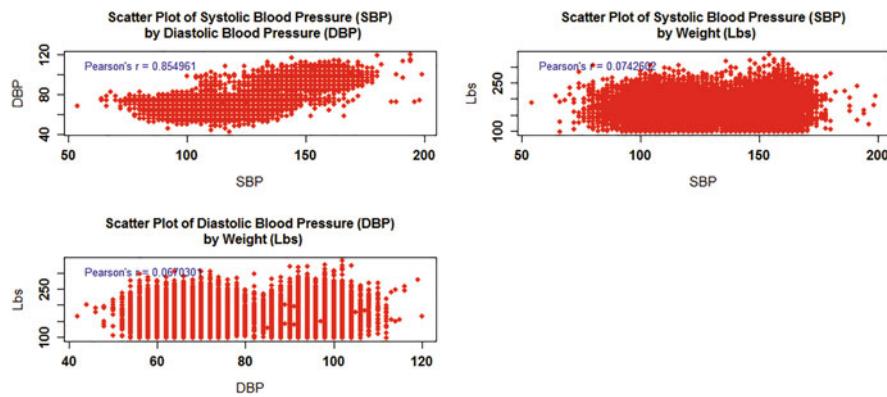


Figure 8.107: Scatter plot—1

It is also possible to create a smoothed scatter plot, which may help with comprehension of outcomes, especially if there is a fair degree of noise at either end of the original X by Y scatter plots (Fig. 8.107).

R Input

```

par.ask=TRUE)
par(mfrow=c(2,2)) # 3 figures into a 1 row by 2 column grid
# with one cell empty
graphics::plot(DBP ~ SBP, data = HospitalAge021100.df,
  main="Smoothed Scatter Plot of Systolic Blood Pressure by
Diastolic Blood Pressure",
  col="red", xlab="Systolic", ylab="Diastolic",
  pch=19, cex.axis=1.25, cex.lab=1.25, font=2)
smooth <- with(HospitalAge021100.df, stats::lowess(DBP ~ SBP,
  f=0.75))
lines(smooth, lwd=6)
graphics::plot(DBP ~ Lbs, data = HospitalAge021100.df,
  main="Smoothed Scatter Plot of Weight by
Diastolic Blood Pressure",
  col="red", xlab="Weight", ylab="Diastolic",
  pch=19, cex.axis=1.25, cex.lab=1.25, font=2)
smooth <- with(HospitalAge021100.df, stats::lowess(DBP ~ Lbs,
  f=0.75))
lines(smooth, lwd=6)
graphics::plot(SBP ~ Lbs, data = HospitalAge021100.df,
  main="Smoothed Scatter Plot of Weight by
Systolic Blood Pressure",
  col="red", xlab="Weight", ylab="Systolic",
  pch=19, cex.axis=1.25, cex.lab=1.25, font=2)
smooth <- with(HospitalAge021100.df, stats::lowess(SBP ~ Lbs,
  f=0.75))
lines(smooth, lwd=6)
# Locally Weighted Scatter Plot Smoothing (lowess) is used
# to create a smooth line through a scatterplot, to better
# understand the relationship between variables.
#
# Above, note the stats::lowess() function argument f=0.75.
# A larger f value provides increased smoothness, which is
# usually desirable. Experiment with different f values
# for this argument to see differences in smoothness in the
# final graphic.

```

When using the scatter plot, be sure to remember the common statement *Correlation does not imply or suggest causation—only association*. when examining the correlation between two variables. That is to say, Variable X may show some degree of correlation (e.g., association) with Variable Y, yet, that does not mean that Variable X causes Variable Y (Fig. 8.108).

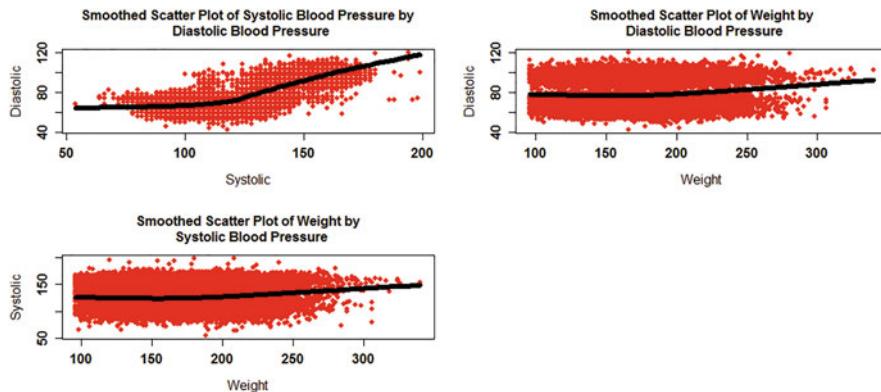


Figure 8.108: Scatter plot—2

It is also possible to prepare a three-dimensional X-Y-Z scatter plot. Admittedly this extra level of detail may make it difficult to interpret the implications and meaning of the outcomes (Fig. 8.109).

R Input

```
install.packages("scatterplot3d")
library(scatterplot3d)          # Load the scatterplot3d package.
help(package=scatterplot3d)    # Show the information page.
sessionInfo()                  # Confirm all attached packages.

scatterplot3d::scatterplot3d(HospitalAge021100.df$SBP,
  HospitalAge021100.df$DBP, HospitalAge021100.df$Lbs,
  # There are three object variables in this listing,
  # SBP - DBP - Lbs
  main="3D X-Y-Z Scatter Plot of SBP-DBP-Lbs",
  highlight.3d=TRUE,           # Points of different colors
  type="p",                   # Points are plotted
  axis=TRUE, tick.marks=TRUE, label.tick.marks=TRUE,
  col.axis="blue", grid=TRUE, col.grid="black", pch = 16,
  xlab="Systolic Blood Pressure (SBP)",
  ylab="Diastolic Blood Pressure (DBP)",
  zlab="Weight (Lbs)",
  cex.axis =1.15, cex.lab=1.15, font.axis=2, font.lab=2)
```

8.9.2.14 Color Gradient Plot

The color gradient plot is another way to view the relationship between the interval object variables, or SBP-DBP-Lbs (e.g., Systolic Blood Pressure by Diastolic Blood Pressure by Weight). Notice how a color gradient is used to

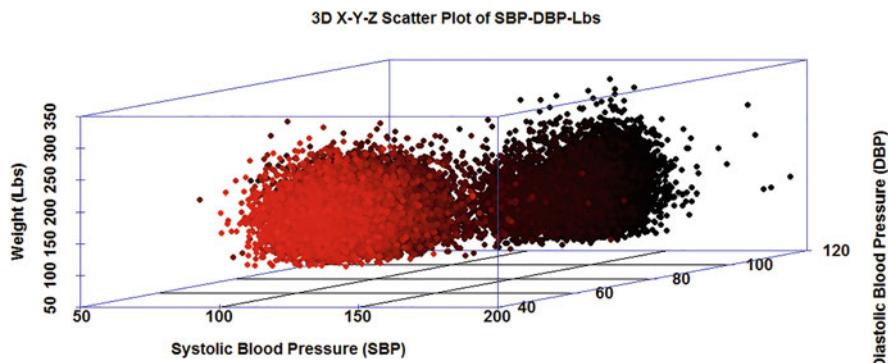


Figure 8.109: Scatter plot—3

increase comprehension while condensing output within the graphic (Figs. 8.110, 8.111, 8.112, and 8.113).

R Input

```
install.packages("openair")
library(openair)                      # Load the openair package.
help(package=openair)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.

par.ask=TRUE)
openair::scatterPlot(HospitalAge021100.df,
  main="Hexbin Scatterplot of Systolic Blood Pressure
  (X Axis) and Diastolic Blood Pressure (Y Axis)",
  x="SBP", y="DBP",
  method="hexbin", smooth=FALSE)
```

R Input

```
par.ask=TRUE)
openair::scatterPlot(HospitalAge021100.df,
  main="Hexbin Scatterplot of Systolic Blood Pressure
  (X Axis) and Weight (Y Axis)",
  x="SBP", y="Lbs",
  method="hexbin", smooth=FALSE)
```

R Input

```
par.ask=TRUE)
openair::scatterPlot(HospitalAge021100.df,
```

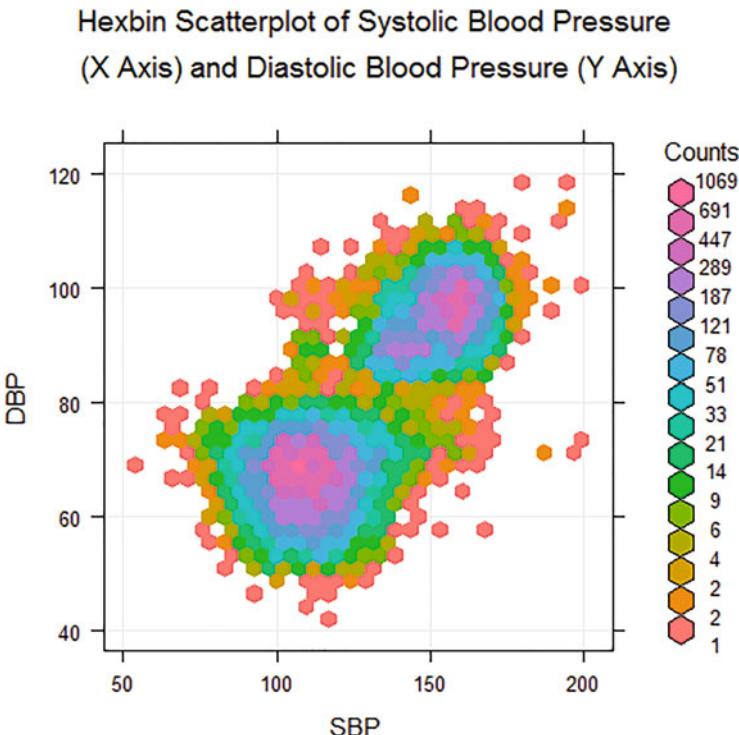


Figure 8.110: Color gradient plot—1

```
main="Hexbin Scatterplot of Diastolic Blood Pressure  
(X Axis) and Weight (Y Axis)",  
x="DBP", y="Lbs",  
method="hexbin", smooth=FALSE)
```

R Input

```
par(ask=TRUE)  
openair::scatterPlot(HospitalAge021100.df,  
  main="Color Gradient Scatterplot of Systolic (X Axis),  
  Diastolic (Y Axis), and Weight (Z Axis)",  
  sub="Scale for Weight (Lbs) is on the Right Margin",  
  x="SBP", y="DBP", z="Lbs",  
  method="scatter", smooth = FALSE)  
# This figure examines relationships for three  
# object variables on a X axis, Y axis, and Z axis.
```

**Hexbin Scatterplot of Systolic Blood Pressure
(X Axis) and Weight (Y Axis)**

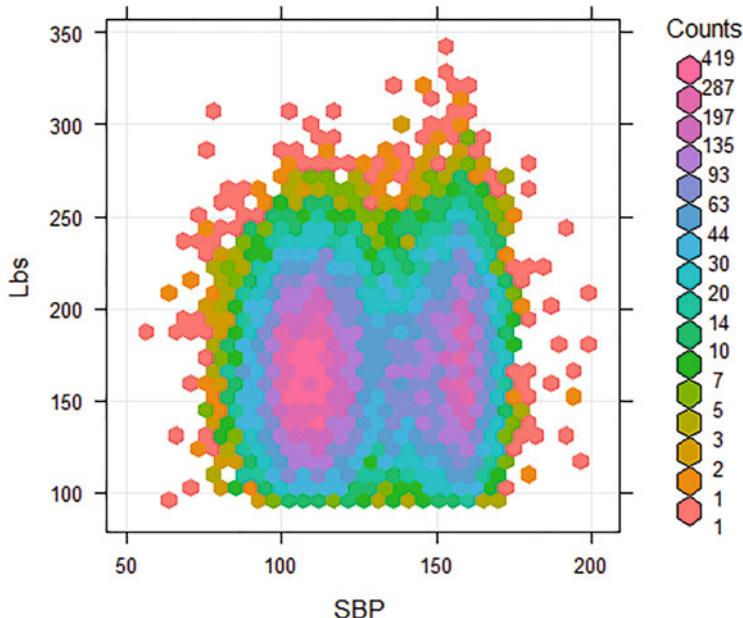


Figure 8.111: Color gradient plot—2

8.9.2.15 Correlogram

The correlogram is used to display the correlation (e.g., association) between variables in a pictogram-type format, possibly using either icons or color gradients to represent the degree of association. Correlograms are typically presented in matrix-format, where many separate correlations are grouped into one common figure, with each cell in the graphical matrix representing a singular correlation outcome.

The `corrgram::corrgram()` function is applied to an intact data frame, thus the need to create a new data frame of data only for the SBP, DBP, and Lbs object variables (Fig. 8.114).

R Input

```
HospitalAge021100Corrgram.df <- data.frame(
  HospitalAge021100.df$SBP,
  HospitalAge021100.df$DBP,
  HospitalAge021100.df$Lbs)
base:::attach(HospitalAge021100Corrgram.df)
base:::summary(HospitalAge021100Corrgram.df)
```

Hexbin Scatterplot of Diastolic Blood Pressure
(X Axis) and Weight (Y Axis)

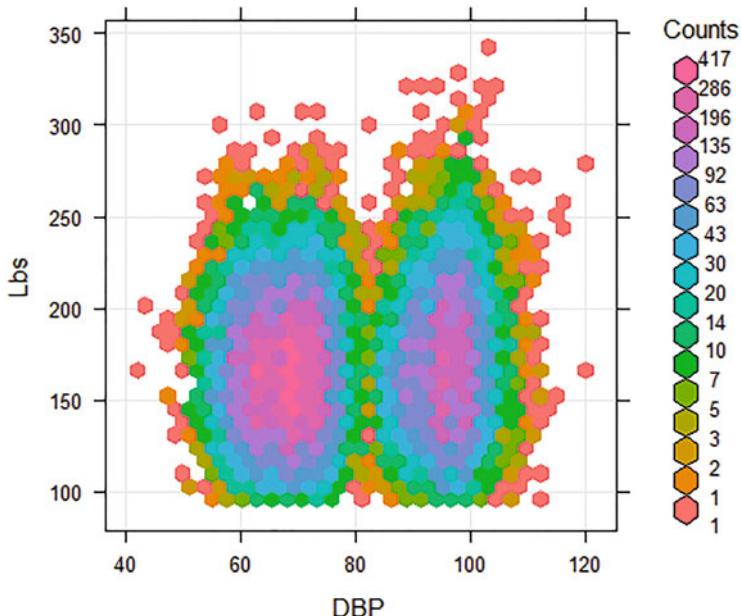


Figure 8.112: Color gradient plot—3

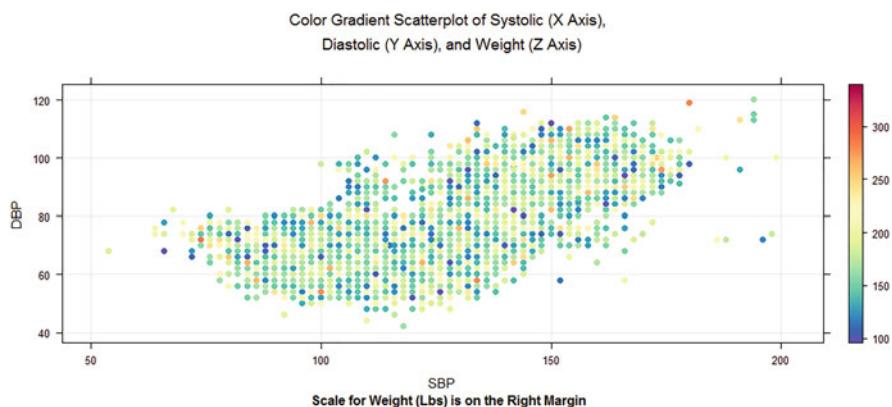


Figure 8.113: Color gradient plot—4

```
# Create a data frame that has data only for object
# variables SBP, DBP, and Lbs. Use this data frame
# with the corrgram::corrgram() function.
```

R Output

```
HospitalAge021100.df.SBP HospitalAge021100.df.DBP
Min. : 54           Min. : 42.0
1st Qu.:106         1st Qu.: 66.0
Median :118         Median : 72.0
Mean   :126         Mean   : 77.5
3rd Qu.:150         3rd Qu.: 94.0
Max.   :199         Max.   :120.0

HospitalAge021100.df.Lbs
Min.   : 96
1st Qu.:144
Median :168
Mean   :169
3rd Qu.:190
Max.   :340
```

R Input

```
install.packages("corrgram")
library(corrgram)          # Load the corrgram package.
help(package=corrgram)      # Show the information page.
sessionInfo()               # Confirm all attached packages.

corrgram::corrgram(HospitalAge021100Corrgram.df,
  main="Correlogram of Systolic Blood Pressure, Diastolic
  Blood Pressure, and Weight",
  type="data",              # Confirm data as opposed to matrix
  dir="right",               # Orientation of output
  upper.panel=panel.pts,    # Correlation output - points
  lower.panel=panel.cor)    # Correlation output - Pearson's r
```

8.9.2.16 Hexbin Plot

It is also possible to prepare a correlation-type graphic where a hexbin (e.g., hexagonal binning routine) is used to consolidate space and increase comprehension of outcomes in a simple graphic (Figs. 8.115 and 8.116).

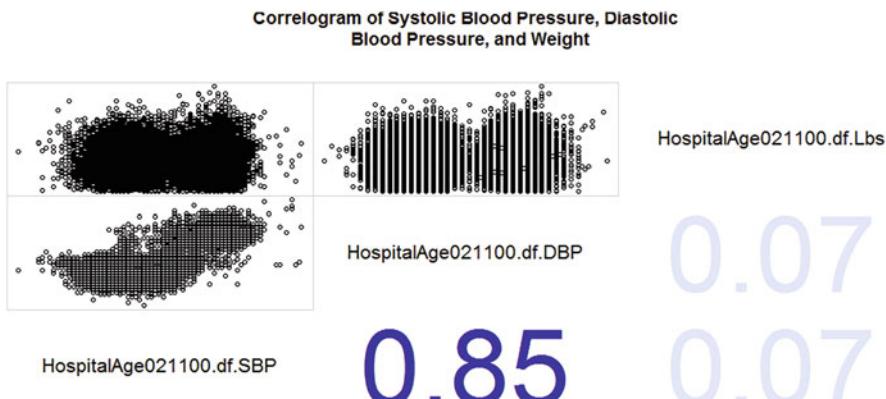


Figure 8.114: Correlogram

R Input

R Input

```

par.ask=TRUE)
hexbin::hexbinplot(DBP ~ SBP | Sex.Recode,
  HospitalAge021100.df,
  main="Hexbin Correlogram (e.g., Scatter Plot) of Systolic
Blood Pressure and Diastolic Blood Pressure by Sex",
  xlab="SBP", ylab="DBP",
  colorkey=TRUE,           # Include a legend
  style="nested.lattice",   # Legend properties
  type="r",                 # Add a regression line
  lwd=6)

```

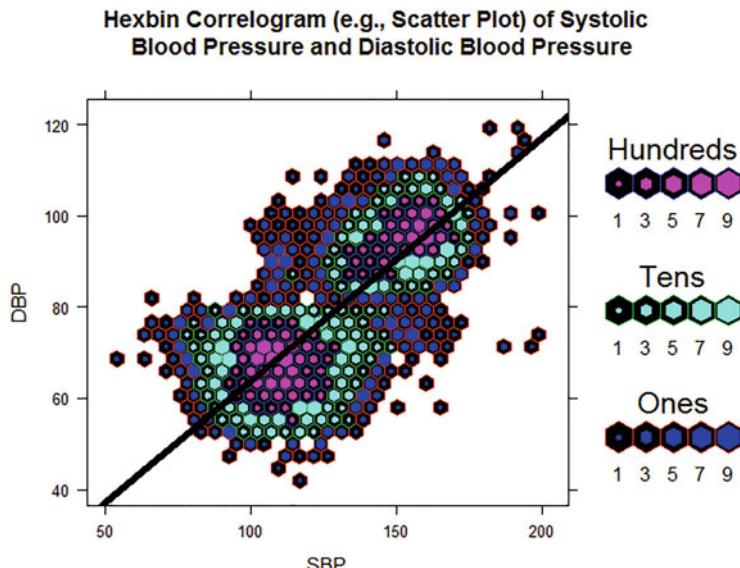


Figure 8.115: Hexbin plot—1

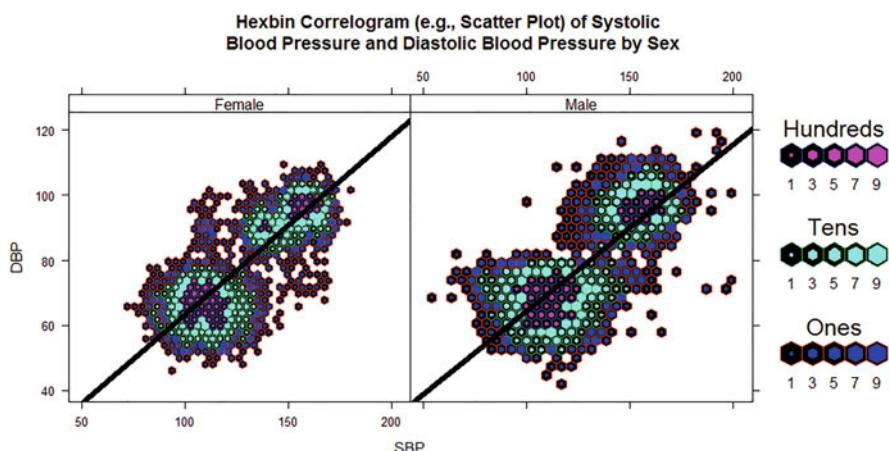


Figure 8.116: Hexbin plot—2

8.9.2.17 Scatter Plot Matrix

The scatter plot matrix is another useful tool, to show the association between multiple object variables, all in a simple and easy-to-view graphic.

R Input

```
par(ask=TRUE)
graphics::pairs(~SBP + DBP + Lbs,
  data=HospitalAge021100.df,
```

```
main="Scatter Plot Matrix of Systolic Blood Pressure,
Diastolic Blood Pressure, and Weight",
col="red", pch=18, cex.axis=1.25, cex.lab=1.25, font=2)
```

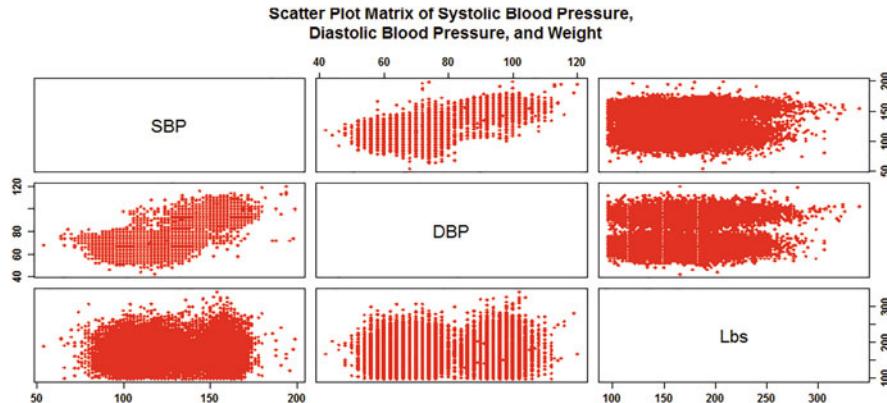


Figure 8.117: Scatter plot matrix—1

The `car::scatterplotMatrix()` function produces a scatter plot matrix of factor object variable breakouts. Explore how this could be useful for diagnostic purposes, before any attempt is made to share the graphics for a research study with others (Figs. 8.117 and 8.118).

R Input

```
install.packages("car")
library(car) # Load the car package.
help(package=car) # Show the information page.
sessionInfo() # Confirm all attached packages.

par(ask=TRUE)
car::scatterplotMatrix(~SBP + DBP + Lbs | Sex.Recode,
  data=HospitalAge021100.df,
  main="Scatter Plot Matrix of Systolic Blood Pressure,
  Diastolic Blood Pressure, and Weight by Sex",
  col=c("red", "blue"), pch=18,
  cex.axis=1.25,
  cex.lab=1.25,
  font=2, lwd=3)
# Note the tilde (e.g., ~) character and also the pipe
# (e.g., |) symbols in this example.
# Allow for a minute or more of processing time if the
# dataset is large, such as the dataset for this example.
```

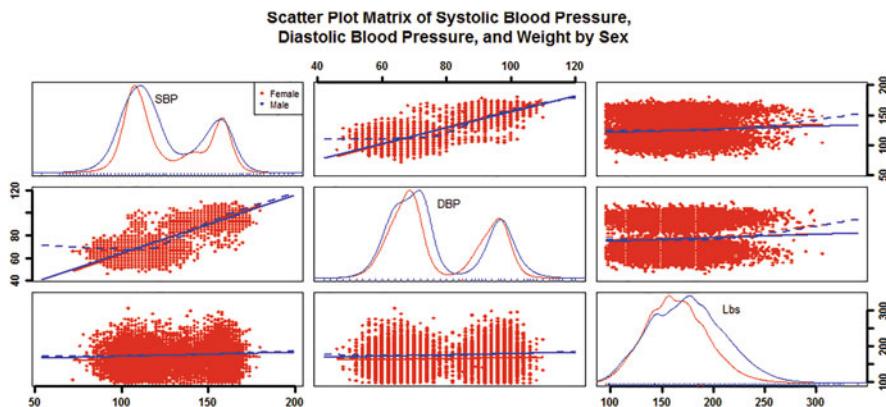


Figure 8.118: Scatter plot matrix—2

8.9.2.18 Sunflower Scatterplot

With large datasets, such as the `Hospital.df` dataset of more than 30,000 subjects, overlap can cause confusion when X-Y points are generated. In a sunflower scatter plot, multiple points are presented using the metaphor of sunflowers with multiple petals. This provides another visualization of how X and Y are related. The sunflower plot is not common and it requires time to learn how to interpret it, but it has value and should be considered for selected users, especially for those who can be expected to have exposure to this type of graphic—researchers in biostatistics (Fig. 8.119).

R Input

```
par.ask=TRUE)
par(mfrow=c(1,3)) # 3 figures into a 1 row by 3 column grid
graphics::sunflowerplot(HospitalAge021100.df$DBP ~
  HospitalAge021100.df$SBP,
  main="Sunflower Plot of Systolic Blood Pressure and
  Diastolic Blood Pressure",
  col="red", xlab="Systolic", ylab="Diastolic",
  cex.axis=1.25, cex.lab=1.25, font=2,
  seg.col="blue", seg.lwd=2)
graphics::sunflowerplot(HospitalAge021100.df$SBP ~
  HospitalAge021100.df$Lbs,
  main="Sunflower Plot of Weight and
  Systolic Blood Pressure",
  col="red", xlab="Weight", ylab="Systolic",
  cex.axis=1.25, cex.lab=1.25, font=2,
  seg.col="blue", seg.lwd=2)
graphics::sunflowerplot(HospitalAge021100.df$DBP ~
```

```
HospitalAge021100.df$Lbs,
main="Sunflower Plot of Weight and
Diastolic Blood Pressure",
col="red", xlab="Weight", ylab="Diastolic",
cex.axis=1.25, cex.lab=1.25, font=2,
seg.col="blue", seg.lwd=2)
# The arguments seg.col and seg.lwd refer to the color and
# line width for sunflower segments in this type of figure.
```

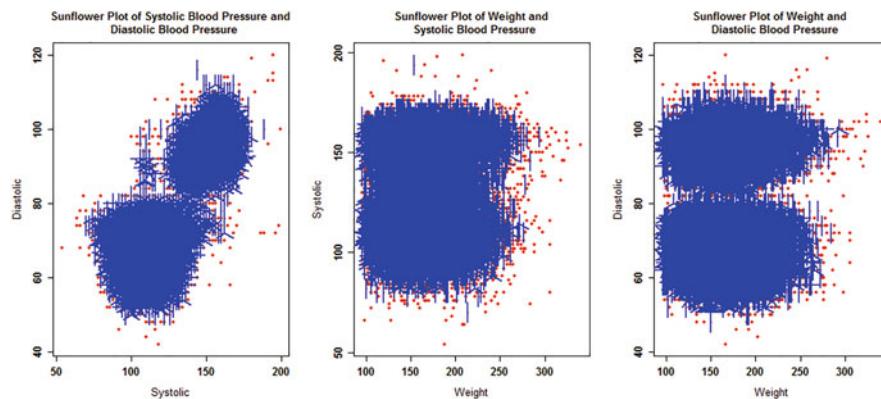


Figure 8.119: Sunflower scatterplot

8.9.2.19 Stem-and-Leaf Plot

The stem-and-leaf plot is somewhat similar to a histogram and it is used to produce a simple display of frequency distributions for data groups (e.g., bins) of continuous numeric data. Output from the `graphics::stem()` function is in text format and is printed directly to the screen—not a separate graphic. This type of output is simple and easy to understand, but it lacks the degree of presentation that would be expected from more contemporary highly-embellished graphical figures.

R Input

```
graphics::stem(HospitalAge021100.df$DBP, width=40)
```

R Output

The decimal point is 1 digit(s) to the right of the |

```
4 | 24
4 | 66888888
```

```

5 | 0000000000000000222222222222+236
5 | 6666666666666666666666666666+1138
6 | 00000000000000000000000000000000+4950
6 | 6666666666666666666666666666+4966
7 | 00000000000000000000000000000000+6483
7 | 6666666666666666666666666666+1066
8 | 00000000000000000000000000000000+263
8 | 5666666666666666666666666666+878
9 | 00000000000000000000000000000000+3607
9 | 6666666666666666666666666666+3707
10 | 00000000000000000000000000000000+2230
10 | 5666666666666666666666666666+162
11 | 0000000000000000000000000000000022+6
11 | 5669
12 | 0

```

R Input

```
graphics::stem(HospitalAge021100.df$Lbs, width=40)
```

R Output

The decimal point is 1 digit(s) to the right of the |

```

9 | 6666666666666666666666666666+85
10 | 00000000000000000000000000000000+552
11 | 00000000000000000000000000000000+994
12 | 00000000000000000000000000000000+1533
13 | 00000000000000000000000000000000+2358
14 | 00000000000000000000000000000000+3073
15 | 00000000000000000000000000000000+3371
16 | 00000000000000000000000000000000+3446
17 | 00000000000000000000000000000000+3491
18 | 00000000000000000000000000000000+3021
19 | 00000000000000000000000000000000+2460
20 | 00000000000000000000000000000000+1767
21 | 00000000000000000000000000000000+1282
22 | 00000000000000000000000000000000+820
23 | 00000000000000000000000000000000+540
24 | 00000000000000000000000000000000+321
25 | 00000000000000000000000000000000+176
26 | 00000000000000000000000000000000+81

```

```
27 | 0000000000000000000222222222+34
28 | 00000022222244444466688888
29 | 0022244444668888
30 | 022466668
31 | 0268
32 | 02446
33 |
34 | 0
```

Because of visual limitations compared to other graphical tools, the stem-and-leaf plot is used for exploratory data analysis at the first stage of engagement and is only rarely presented in publications and presentations.

8.9.2.20 Stripchart

The stripchart is used to produce a one-dimensional scatter plot of numeric data. Output from a stripchart is fairly simple and as such the stripchart is a good selection for exploratory data analysis (Fig. 8.120).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
graphics::stripchart(HospitalAge021100.df$DBP,
  main="Stripchart of Diastolic Blood Pressure",
  xlab="Diastolic Blood Pressure (DBP)", col="red",
  pch=1,                                     # Plotting symbol
  method="jitter",                            # Aid in visualization of
  jitter=0.001,                               # overlapping values
  vertical=FALSE,                            # Horizontal orientation
  cex.axis=1.25, cex.lab=1.25, font=2)
graphics::stripchart(HospitalAge021100.df$Lbs,
  main="Stripchart of Weight",
  xlab="Weight (Lbs)", col="red",
  pch=1,                                     # Plotting symbol
  method="jitter",                            # Aid in visualization of
  jitter=0.001,                               # overlapping values
  vertical=FALSE,                            # Horizontal orientation
  cex.axis=1.25, cex.lab=1.25, font=2)
# Jittering is used to prevent overplotting in a graphic
# where there are many datapoints, aiding in overall
# visualization of the final graphic. Experiment with
# different values for the argument jitter.
```

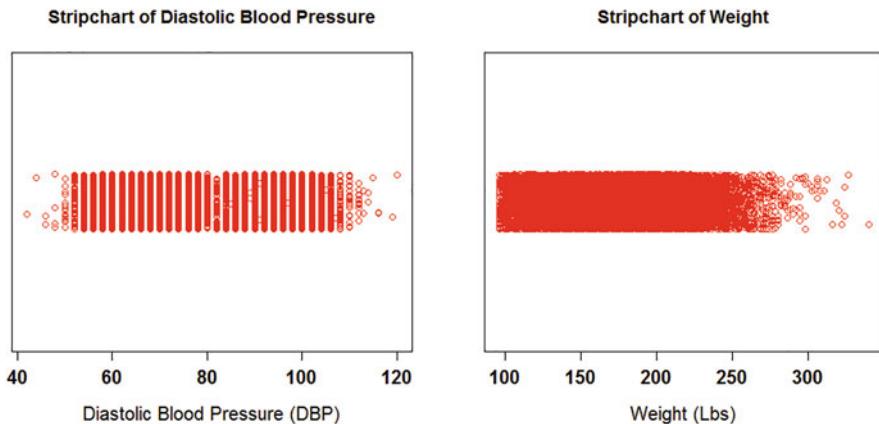


Figure 8.120: Stripchart

Because of the simplicity of output from the `graphics::stripchart()` function, it is rarely used in presentations and publications. However, in a desire to communicate ideas in a simple manner, use of the `graphics::stripchart()` function may deserve more attention.

8.9.2.21 Violin Plot

The violin plot is used to generate a graphic that includes the best features of a boxplot and a density plot. It is an extremely valuable selection from the many tools used for the presentation of continuous numeric data and it should be considered for use whenever general trends about continuous numeric data are presented in graphical format (Figs. 8.121 and 8.122).

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
vioplot::vioplot(HospitalAge021100.df$DBP,
  names=c("DBP"), col="red")
title("Violin Plot of Diastolic Blood Pressure")
vioplot::vioplot(HospitalAge021100.df$Lbs,
  names=c("Lbs"), col="red")
title("Violin Plot of Weight")
```

R Input

```
par(ask=TRUE)
par(mfrow=c(1,2)) # 2 figures into a 1 row by 2 column grid
UsingR::simple.violinplot(HospitalAge021100.df$DBP ~
```

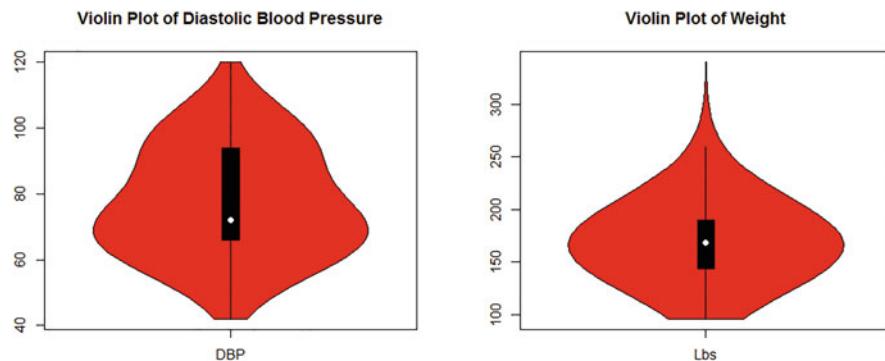


Figure 8.121: Violin plot—1

```
HospitalAge021100.df$Sex.Recode,
lty=1, lwd=6, col="red",           # Line type, width, and color
ylim=c(0,125))                   # Axis scale
title("Violin Plot of Diastolic Blood Pressure by Sex")
UsingR::simple.violinplot(HospitalAge021100.df$Lbs ~
HospitalAge021100.df$Sex.Recode,
lty=1, lwd=6, col="red",           # Line type, width, and color
ylim=c(0,350))                  # Axis scale
title("Violin Plot of Weight by Sex")
```



Figure 8.122: Violin plot—2

8.9.3 Specialized Graphics

8.9.3.1 Density Ridge

Density ridges combine the presentation of a density plot and overlapping line plots. It may have use for exploratory analyses. In R, density ridges are generated using the `ggplot2::ggplot()` function and `theme_ridges()` (Figs. 8.123, 8.124, 8.125, and 8.126).

R Input

```
install.packages("ggridges")
library(ggridges)                      # Load the ggridges package.
help(package=ggridges)                  # Show the information page.
sessionInfo()                          # Confirm all attached packages.

par.ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df, aes(x=Lbs,
y=PrimaryPayer.Recode, fill=PrimaryPayer.Recode)) +
geom_density_ridges() +
theme_ridges()
```

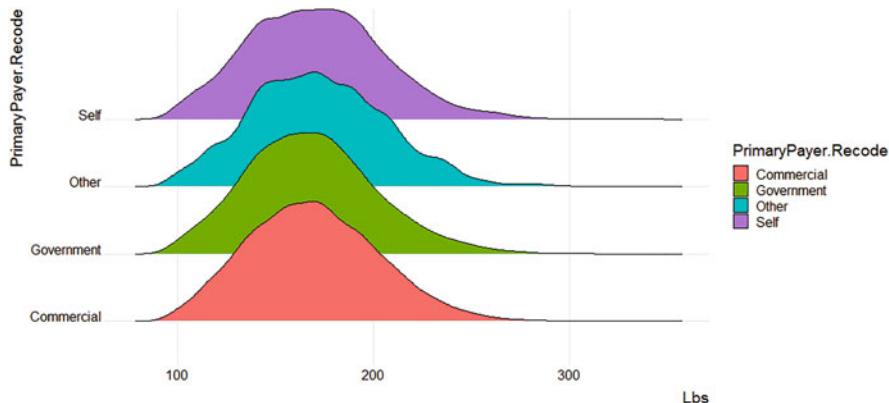


Figure 8.123: Density ridge—1

R Input

```
par.ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df, aes(x=DBP,
y=Location.Recode, fill=Location.Recode)) +
geom_density_ridges(scale=1) +
theme_ridges()
```

R Input

```
par.ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df, aes(x=Lbs,
y=PrimaryPayer.Recode, group=PrimaryPayer.Recode,
fill=PrimaryPayer.Recode)) +
stat_binline(bins=75) +
theme_ridges()
# Histogram using theme_ridges()
```

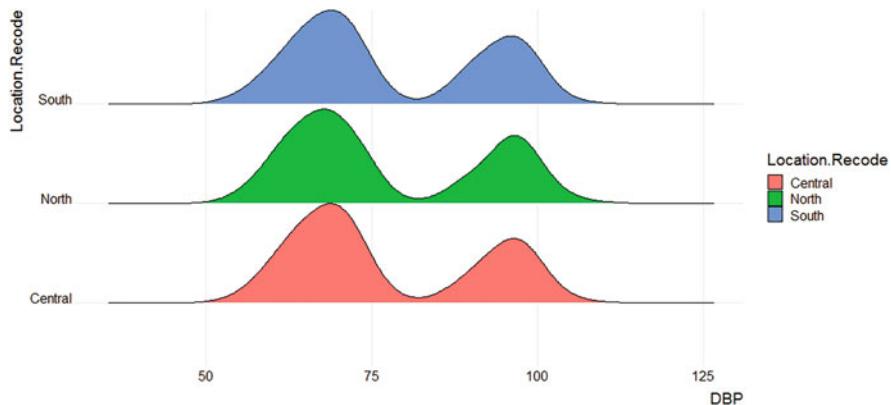


Figure 8.124: Density ridge—2

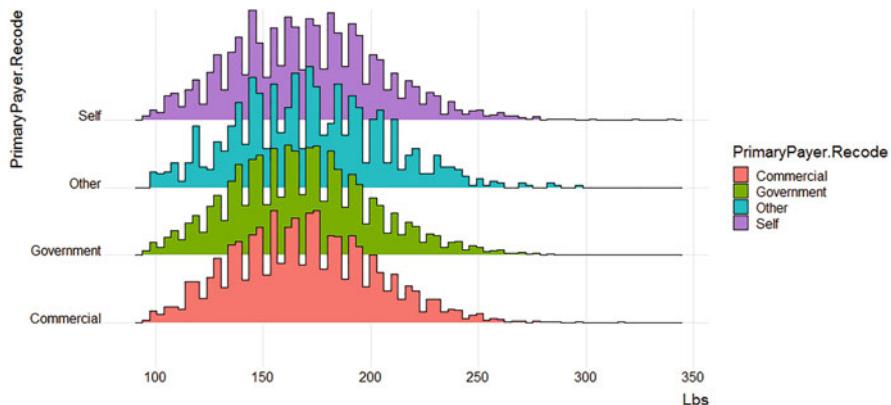


Figure 8.125: Density ridge—3

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df, aes(x=DBP,
y=Location.Recode, group=Location.Recode,
fill=Location.Recode)) +
stat_binline(bins=75) +
theme_ridges()
# Histogram using theme_ridges()
```

8.9.3.2 Gantt Chart

The Gantt chart is used to organize and graphically present scheduling activities, typically for when project deliverables have due dates. The syntax presented below is interesting in the way that it is organized (especially how dates are used in R), but there are other specialized software programs that may be better at project management and production of a Gantt chart (Fig. 8.127).

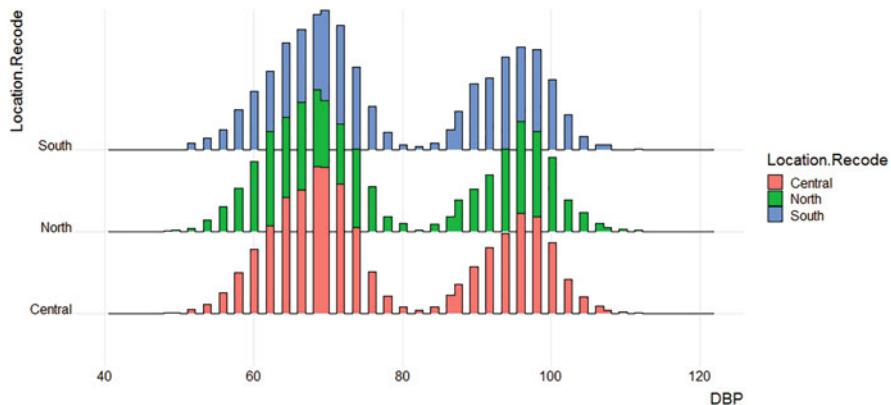


Figure 8.126: Density ridge—4

R Input

```

par(ask=TRUE)
savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)          # Bold
savecex.axis <- par(cex.axis=1.25) # Axis
Ymd.format <-"%Y/%m/%d"           # Date Format
gantt.info <-list(labels=
  c("Select Subjects", "Obtain Biological Data",
    "Distribute Survey", "Send Survey Reminder",
    "Prepare Summary", "Publish"),
  starts=
  as.POSIXct(strptime(      # Start Dates
  c("2020/01/01", "2020/03/01", "2020/03/01",
    "2020/05/01", "2020/06/01", "2020/11/01"),
  format=Ymd.format)),
  ends=
  as.POSIXct(strptime(      # End Dates
  c("2020/03/01", "2020/05/01", "2020/05/01",
    "2020/06/01", "2020/10/01", "2020/12/31"),
  format=Ymd.format)),
  priorities=c(1,1,1,3,2,1)) # Color-coded
vgridpos <-as.POSIXct(strptime(
  c("2020/01/01", "2020/02/01", "2020/03/01",
    "2020/04/01", "2020/05/01", "2020/06/01",
    "2020/07/01", "2020/08/01", "2020/09/01",
    "2020/10/01", "2020/11/01", "2020/12/01"),
  format=Ymd.format))
vgridlab <-

```

```

c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
gantt.chart(gantt.info, main=
  "Gantt Chart of Survey Actions - CY2020\n",
  priority.legend=TRUE,
  vgridpos=vgridpos,
  vgridlab=vgridlab,
  hgrid=TRUE,
  label.cex=1.25)
par(savelwd); par(savefont); par(savecex.axis)

```

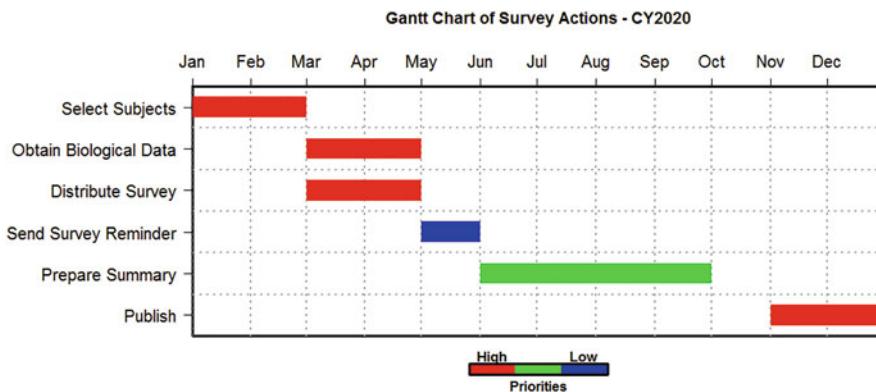


Figure 8.127: Gantt chart

8.9.3.3 Interaction Plot

The interaction plot is used to plot the means (or other selected statistics) of twoway combinations of selected object variables, therefore displaying interaction (if any) between the variables. The interaction plot is a valuable tool for better understanding complex relationships between and among the data (Figs. 8.128 and 8.129).

R Input

```

savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)         # Bold
savecex.lab  <- par(cex.lab=1.25)   # Label
savecex.axis <- par(cex.axis=1.25)  # Axis
par(ask=TRUE)
stats::interaction.plot(HospitalAge021100.df$Sex.Recode,
  HospitalAge021100.df$Location.Recode,
  HospitalAge021100.df$DBP,

```

```

main="Interaction Plot: Sex, Location, and
Diastolic Blood Pressure",
fun=mean,                      # Use mean instead of median
legend=TRUE,                     # Include a legend
trace.label="Location",          # Overall legend
fixed=TRUE,                      # Legend order
col=c("red", "green", "blue"),   # Line color
lwd=4,                           # Line width
lty=c("solid", "dashed", "dotdash"), # Line types
xlab=" ",                         # Blank label to allow for output
ylab="Mean Diastolic",
font.lab=2,
xtick=TRUE)                      # Include tick marks on X axis
par(savelwd)        # Return to original setting.
par(savefont)       # Return to original setting.
par(savecex.lab)    # Return to original setting.
par(savecex.axis)   # Return to original setting.

```

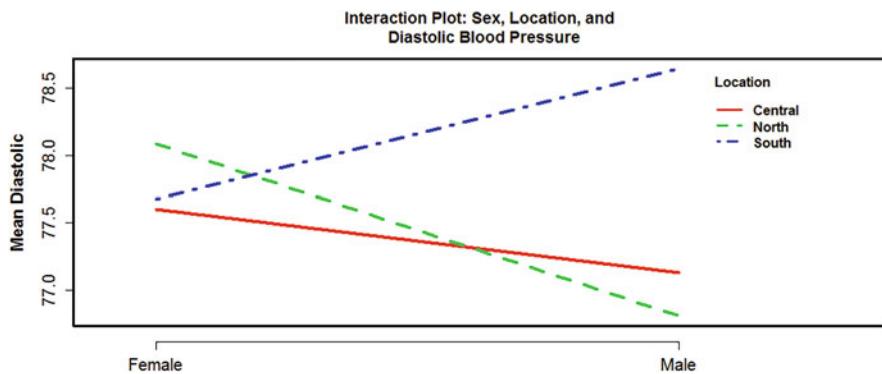


Figure 8.128: Interaction plot—1

R Input

```

savelwd      <- par(lwd=3)          # Heavy line
savefont     <- par(font=2)          # Bold
savecex.lab  <- par(cex.lab=1.25)   # Label
savecex.axis <- par(cex.axis=1.25)   # Axis
par(ask=TRUE)
stats::interaction.plot(HospitalAge021100.df$Sex.Recode,
                        HospitalAge021100.df$Location.Recode,
                        HospitalAge021100.df$Lbs,
                        main="Interaction Plot: Sex, Location, and Weight",
                        fun=mean,                      # Use mean instead of median

```

```

legend=TRUE,                      # Include a legend
trace.label="Location",          # Overall legend
fixed=TRUE,                      # Legend order
col=c("red", "green", "blue"),   # Line color
lwd=4,                            # Line width
lty=c("solid", "dashed", "dotdash"), # Line types
xlab=" ",                         # Blank label to allow for output
ylab="Mean Weight (Lbs)",        # Y-axis label
font.lab=2,                       # Font size for labels
xtick=TRUE)                       # Include tick marks on X axis
par(savelwd)                      # Return to original setting.
par(savefont)                     # Return to original setting.
par(savecex.lab)                  # Return to original setting.
par(savecex.axis)                 # Return to original setting.

```

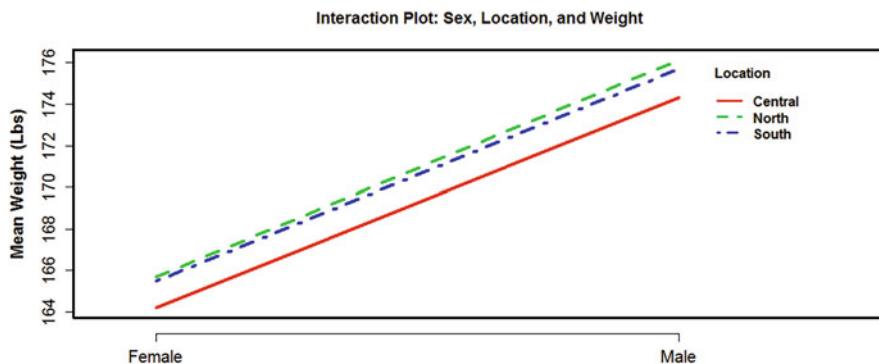


Figure 8.129: Interaction plot—2

8.9.3.4 Staircase Plot

The staircase plot is used to display a series of totals (represented as bars) for breakout events as the totals change over the life of an activity or project.

This example for a staircase plot is based on parameters for a survey on blood pressure screening. In this example for the staircase plot, consider how it is often best to prepare a graphical image of the population, invited sample, and responding sample to gain a better sense of survey participants and equally, those who did not participate. For this teaching example, consider how 2000 individuals were asked to participate in a survey about blood pressure screening. Then look at the many events (and corresponding numbers) for each stage of the survey process (e.g., event) until the final set of respondents completed the survey:

- Population: N = 2000 participants who were asked to participate in the survey

- Opt-Out: N = 150 participants who indicated that they did not want to participate in the survey, either by formally declining participation or ignoring the original request for participation
- Invited Sample: N = 1850 participants who indicated that they would participate in the survey
- No Response: N = 325 invited survey members who did not return a survey, although they had originally indicated agreement to participate
- Responding Sample: N = 1525 invited survey members who returned a completed survey

The `plotrix::staircase.plot()` function is used to present, similar to a descending staircase, the progression from Population to Invited Sample and finally to Responding Sample (Fig. 8.130).

R Input

```
BPStudy.SampleSize <- c(2000, -150, 1850, -325, 1525)
BPStudy.Totals     <- c(TRUE, FALSE, TRUE, FALSE, TRUE)
BPStudy.Labels     <- c("Population", "Opt-Out",
                        "Invited Sample", "No Response",
                        "Responding Sample")

savelwd            <- par(lwd=3)           # Heavy line
savefont           <- par(font=2)          # Bold
savecex.lab        <- par(cex.lab=1.25) # Label
savecex.axis       <- par(cex.axis=1.25) # Axis
par(ask=TRUE)
plotrix::staircase.plot(
  BPStudy.SampleSize, BPStudy.Totals, BPStudy.Labels,
  main="Participation in Blood Pressure Survey (82.43%)",
  total.col="blue",           # Color for bar totals
  inc.col=2:3,                # Color for bar increments
  bg.col="white",             # Color for the background
  direction="s")              # Direction (south) for bars
par(savelwd)         # Go back to default settings.
par(savefont)        # Go back to default settings.
par(savecex.lab)    # Go back to default settings.
par(savecex.axis)   # Go back to default settings.
```

8.9.3.5 Triangular Plot for 3-D Representation

A triangular plot for 3-D representation is used to visualize the ratios of three separate object variables in relation to a constant sum. With R, the `vcd::ternary.plot()` function is used to prepare the specialized equilateral triangular.

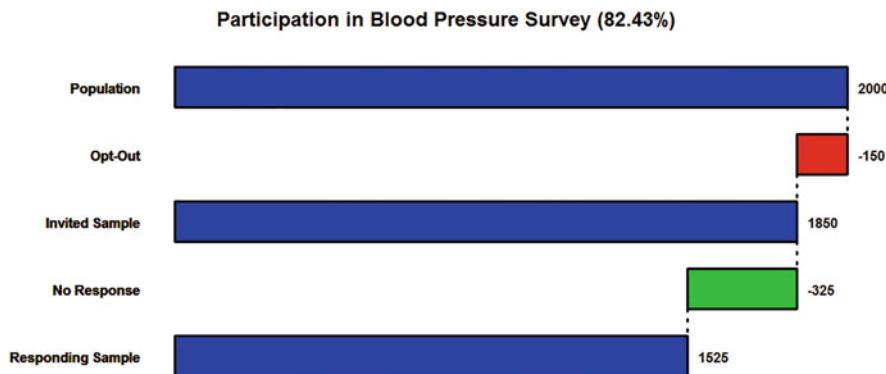


Figure 8.130: Staircase plot

To use the `vcd::ternaryplot()` function it is best to put data for the three selected object variables into a separate dataframe. Note how this action is easily achieved in R, against Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP), and Weight (Lbs) (Fig. 8.131).¹⁹

R Input

```
HospitalAge021100Ternaryplot.df <- data.frame(
  HospitalAge021100.df$SBP,
  HospitalAge021100.df$DBP,
  HospitalAge021100.df$Lbs)
base:::attach(HospitalAge021100Ternaryplot.df)
base:::summary(HospitalAge021100Ternaryplot.df)

par(ask=TRUE)
vcd:::ternaryplot(
  main="Triangular Plot for 3-D Representation of Systolic
  Blood Pressure, Diastolic Blood Pressure, and Weight",
  HospitalAge021100Ternaryplot.df,    # ID the dataframe
  prop_size=TRUE,                      # Proportional plot
  dimnames_position="edge",           # Dimension labels position
  dimnames_color="red")               # Dimension labels color
```

The 3-D Triangular Plot may be useful for diagnostic purposes, but the complexity may be more challenging than many can easily grasp. Yet, it deserves attention among the many tools that help explain how variables are related to each other, overall and at the breakout level.

¹⁹The preparation of a dataframe with only SBP, DBP, and Lbs may have been done before, but look at the unique name for this action, reflecting the upcoming figure.

**Triangular Plot for 3-D Representation of Systolic
Blood Pressure, Diastolic Blood Pressure, and Weight**

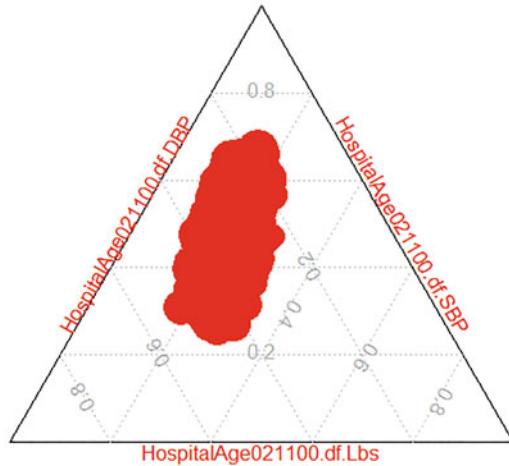


Figure 8.131: Triangular plot

8.10 Addendum 2: Graphics Using the lattice Package

The lattice package provides a long-standing and well-established option for the graphical presentation of various types of data when using R, including factor-type data of counted breakout groups as well as continuous numeric-type data of measured entities. The lattice package and associated functions are typically used against multivariate data and are based on the use of trellis graphics—so named for its grid-like (e.g., think of a garden trellis as a grid) approach to show the relationship between and among data from various object variables.

For both Addendum 2 (lattice package) and Addendum 3 (ggplot2 package), the appropriate syntax for each demonstrated figure type is presented. However, refer to prior discussions on the specific figures, in an attempt to minimize what would otherwise be redundant discussion. Comments about the different lattice-based functions are kept to a minimum and are only offered when an argument or some other feature unique to the lattice package calls for discussion.

R Input

```
install.packages("lattice")
library(lattice)                      # Load the lattice package.
help(package=lattice)                  # Show the information page.
sessionInfo()                         # Confirm all attached packages.
```

The `lattice::barchart()` function is used to produce a very simple bar plot, similar to what is generated with the `graphics::barplot()` function. However, notice

how the lattice::barchart() function does not need the base::table() function serving as a wrapper around the object variable that is used to generate the bar plot.²⁰ Also, compare the output of the lattice::barchart() function with no embellishments to the output of the graphics::barplot() function with no embellishments. The default colors and general presentation of lattice-based graphics certainly have value for visual appeal (Fig. 8.132).

R Input

```
par.ask=TRUE)
lattice::barchart(HospitalAge021100.df$Location,
main="Location: Age 21 to 100", horizontal=FALSE)
```

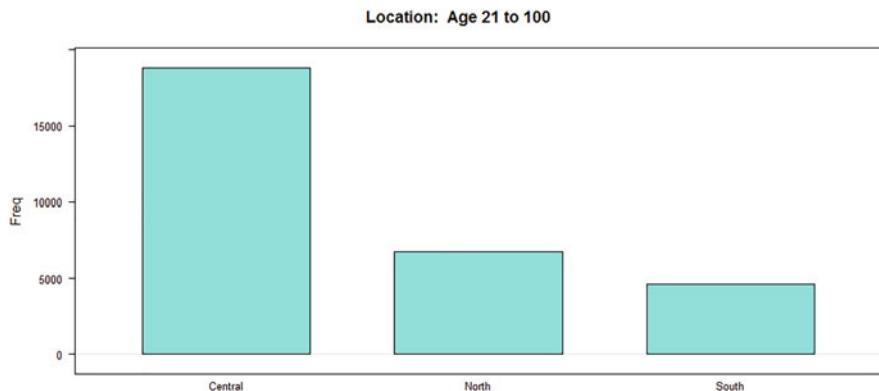


Figure 8.132: lattice package—barchart

Focusing on the graphics most commonly used to visually display the relationship between and among data, a few more lattice-based figures follow. Notice the way multiple lattice-based figures are grouped together into one common figure (Figs. 8.133 and 8.134).

R Input

```
latticebwplot1 <-
lattice::bwplot(DBP ~ Sex.Recode,
data=HospitalAge021100.df,
main="DBP by Sex: Age 21 to 100",
par.settings = simpleTheme(lwd=2),
par.strip.text=list(cex=1.15, font=2),
scales=list(cex=1.15),
ylab="Diastolic Blood Pressure (DBP)",
```

²⁰Go back to prior syntax in this lesson on how the base::table() function is used as a wrapper.

```

ylim=c(0,150), col="red", aspect=0.5,
layout=c(1,1)) # 1 Column by 1 Row
# Box Plot for Breakout Groups

latticebwplot2 <-
lattice::bwplot(DBP ~ Sex.Recode | Ethnicity.Recode,
  data=HospitalAge021100.df,
  main="DBP by Sex and by Ethnicity: Age 21 to 100",
  par.settings = simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  ylab="Diastolic Blood Pressure (DBP)",
  ylim=c(0,150), aspect=0.5, col="red",
  layout=c(3,1)) # 3 Column by 1 Row

latticedensityplot1 <-
lattice::densityplot(~ DBP,
  data=HospitalAge021100.df,
  main="DBP: Age 21 to 100",
  type="count", # Count
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  xlab=list("DBP", cex=1.15, font=2),
  xlim=c(0,150), # X axis range
  ylab=list("Density", cex=1.15, font=2),
  aspect=1,
  col="red", lwd=4, # Color and line width
  layout=c(1,1)) #1 Column by 1 Row

latticedensityplot2 <-
lattice::densityplot(~ DBP | Sex.Recode,
  data=HospitalAge021100.df,
  main="DBP by Sex: Age 21 to 100",
  type="count", # Count
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  xlab=list("DBP", cex=1.15, font=2),
  xlim=c(0,150), # X axis range
  ylab=list("Density", cex=1.15, font=2),
  aspect=1,
  col="red", lwd=4, # Color and line width

```

```

layout=c(2,1)) # 2 Column by 2 Row
# Density Plot for Breakout Groups

print(latticebwplot1, split = c(1, 1, 2, 2),
      more = TRUE)
print(latticebwplot2, split = c(2, 1, 2, 2),
      more = TRUE)
print(latticedensityplot1, split = c(1, 2, 2, 2),
      more = TRUE)
print(latticedensityplot2, split = c(2, 2, 2, 2),
      more = FALSE)
# Put four lattice-based figures into one
# common figure.

```

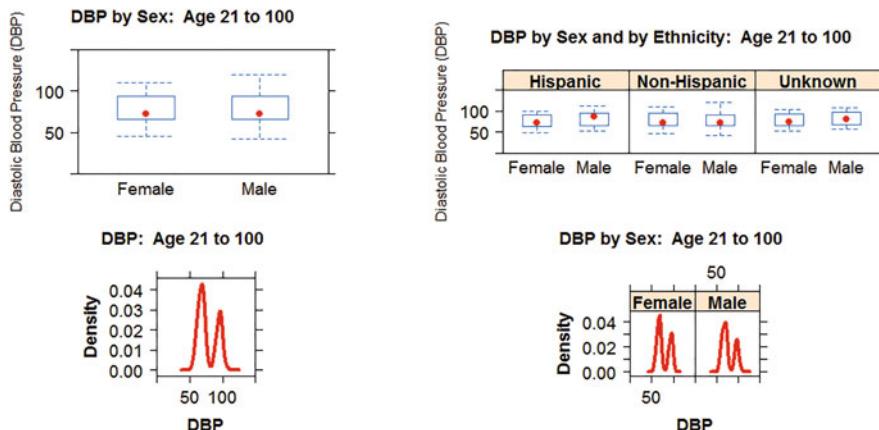


Figure 8.133: lattice package—box plot and density plot

R Input

```

latticehistogram1 <-
lattice::histogram(~ DBP,
  data=HospitalAge021100.df,
  main="DBP: Age 21 to 100",
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  xlab=list("DBP", cex=1.15, font=2),
  col="red", xlim=c(0,150),           # X axis range
  layout=c(1,1))                   # 1 Column by 1 Row

latticehistogram2 <-

```

```

lattice::histogram(~ DBP | Sex.Recode,
  data=HospitalAge021100.df,
  main="DBP by Sex: Age 21 to 100",
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  xlab=list("DBP", cex=1.15, font=2),
  col="red", xlim=c(0,150),           # X axis range
  layout=c(2,1))                  # 5 Column by 2 Row
# Histogram by Breakout Groups

latticeqqmath1 <-
lattice::qqmath(~ DBP,
  data=HospitalAge021100.df,
  main="DBP: Age 21 to 100",
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  xlab=list("Qnorm", cex=1.15, font=2),
  ylab=list("DBP", cex=1.15, font=2),
  col="red", layout=c(1,1))    # 1 Column by 1 Row

latticeqqmath2 <-
lattice::qqmath(base::subset(
  HospitalAge021100.df$DBP,
  (HospitalAge021100.df$AgeLastBirthday >= 21 &
  HospitalAge021100.df$AgeLastBirthday <= 29) &
  (HospitalAge021100.df$Race.Recode == "White")),
  main="DBP: Age 21 to 100 and Race = White",
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  xlab=list("Qnorm", cex=1.15, font=2),
  ylab=list("DBP", cex=1.15, font=2),
  col="red", layout=c(1,1))    # 1 Column by 1 Row
# QQ Plots by Breakout Groups

print(latticehistogram1, split = c(1, 1, 2, 2),
  more = TRUE)
print(latticehistogram2, split = c(2, 1, 2, 2),
  more = TRUE)
print(latticeqqmath1, split = c(1, 2, 2, 2),
  more = TRUE)

```

```
print(latticeqqmath2, split = c(2, 2, 2, 2),
      more = FALSE)
```

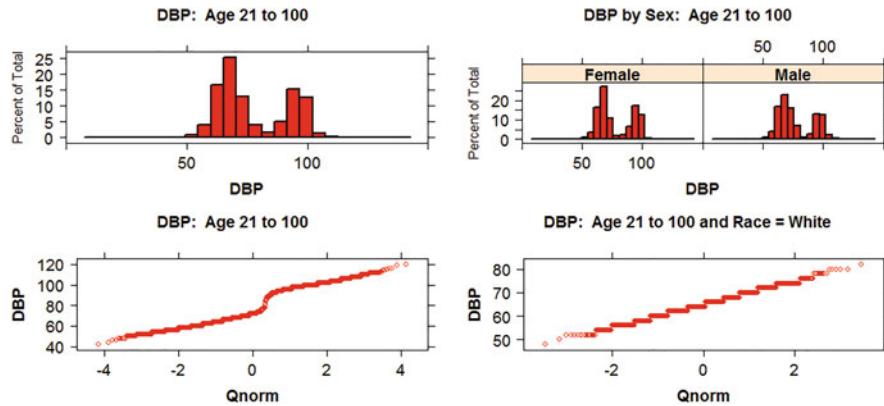


Figure 8.134: lattice package—histogram and QQ plot

R Input

```
latticexyplot1 <-
lattice::xyplot(DBP ~ SBP,
  data=HospitalAge021100.df,
  main="SBP by DBP:  Age 21 to 100",
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(cex=1.15),
  xlab=list("Lbs", cex=1.15, font=2),
  col="red", grid=TRUE,           # Show a grid
  layout = c(1,1))               # 1 Column by 1 Row

latticexyplot2 <-
lattice::xyplot(DBP ~ SBP | Sex.Recode,
  data=HospitalAge021100.df,
  main="SBP by DBP and by Sex:  Age 21 to 100",
  par.settings=simpleTheme(lwd=2),
  par.strip.text=list(cex=1.15, font=2),
  scales=list(x=list(rot=45), cex=1.10),
  xlab=list("Weight (Lbs)", cex=1.15, font=2),
  col="red", grid=TRUE,           # Show a grid
  layout = c(2,1))               # 2 Column by 1 Row
# Look at the 45 degree rotation for X axis labels,
# achieved by using rot=45 as part of the scales
# argument
```

```

latticesplom1 <-
lattice::splom(~HospitalAge021100.df[18:19],
  data=HospitalAge021100.df, groups=Sex.Recode,
  panel=panel.superpose,
  key=list(title="Scatter Plot Matrix",
  text=list(c("SBP", "DBP"))))
# As the dataset is organized and confirmed by using
# the base::names() function, merely count the columns
# to see that object variables SBP and DBP -- columns
# 18 to 19.
# Prepare this SPLOM by itself for columns 18:20 to
# see a much better presentation, when larger and not
# part of a four-in-one figure.
#
# Scatter Plot Matrix - SPLOM

latticecloud1 <-
lattice::cloud(DBP~Lbs*SBP,
  data=HospitalAge021100.df,
  main="Lbs by SBP and by DBP: Age 21 to 100 ")
# This simple lattice-based 3-D Scatter Plot is shown
# with no embellishments, to present a 3-D image in
# simple format.
#
# Many individuals find it difficult to interpret a
# 3-D presentation of correlation. Use the 3-D
# sparingly and only for those who can be reasonably
# expected to understand how to interpret findings
# for this multidimensional graphic.

print(latticexyplot1, split = c(1, 1, 2, 2),
  more = TRUE)
print(latticexyplot2, split = c(2, 1, 2, 2),
  more = TRUE)
print(latticesplom1, split = c(1, 2, 2, 2),
  more = TRUE)
print(latticecloud1, split = c(2, 2, 2, 2),
  more = FALSE)

```

Although the graphics associated with the `ggplot2::ggplot()` function are now quite popular, the `lattice` package still has great value. Prepare the many figures presented in this addendum and then experiment with other variables (Fig. 8.135).

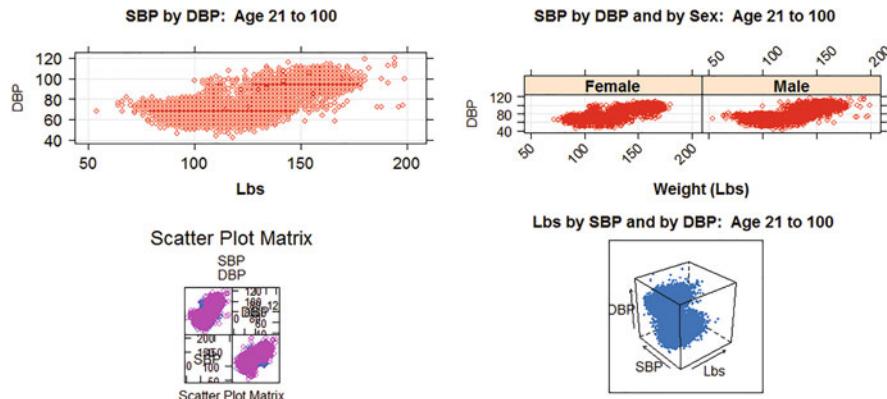


Figure 8.135: lattice package—scatter plot, scatter plot matrix (SPLOM), and 3D scatter plot

8.11 Addendum 3: Graphics Using the ggplot2 Package

The ggplot2 package has gained wide-spread use in the R community, largely because the ggplot2 package supports the production of *Beautiful Graphics*—a term that represents figures that are not only of the highest professional quality but are also quite attractive to view in terms of color and presentation. The ggplot2 package is often ranked among the five most frequently downloaded R packages. In addition, the ggplot2 package is among the collection of R packages included in the tidyverse, furthering its use among those who use R.

This addendum will demonstrate how ggplot2 is used to parallel the many types of figures previously demonstrated (e.g., bar chart, boxplot, density plot, histogram, etc.) but notice how the presentation of these figures is easily changed by using a wide selection of attractive themes associated with the ggthemes package. Further, with only a minimal degree of effort user-created graphical themes can be created for use with ggplot2, which is also demonstrated.

Bar Plot

The ggplot2::ggplot() function can be used to make simple graphics and from that simple graphic additional syntax is added to make exceptionally vivid, colorful, and professional figures. Look at a simple barplot generated using the ggplot2::ggplot() function and then a barplot of the same data, but embellished to a degree by using a ggplot-based theme (Fig. 8.136).

R Input

```
ggplotbarplot1 <-
  ggplot2::ggplot(HospitalAge021100.df, aes(x=Location)) +
    geom_bar()
```

```

# Prepare a simple ggplot2 figure, showing a drab and
# generally colorless bar plot, with no embellishments.
# Even so, look at the basics of this figure: (1) the
# ggplot2::ggplot() function is applied against the
# HospitalAge021100.df dataset, (2) aes() (e.g.,
# aesthetic mapping) was used to layer the figure with
# additional detail(s), which for this figure was the
# declaration that the X axis should focus on the
# Location object variable, and (3) the resulting
# figure should be a bar plot, gained by using the geom
# geom_bar().

ggplotbarplot2 <-
ggplot2::ggplot(HospitalAge021100.df, aes(x=Location)) +
  geom_bar() +  # The + symbol is essential when using ggplot2
  theme_bw()
# Different themes are used with ggplot2 to produce more
# visually appealing figures. In this figure theme_bw() was
# used to make a slight improvement over the prior figure,
# changing the background color from gray to white.

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotbarplot1,
  ggplotbarplot2, ncol=2)

```

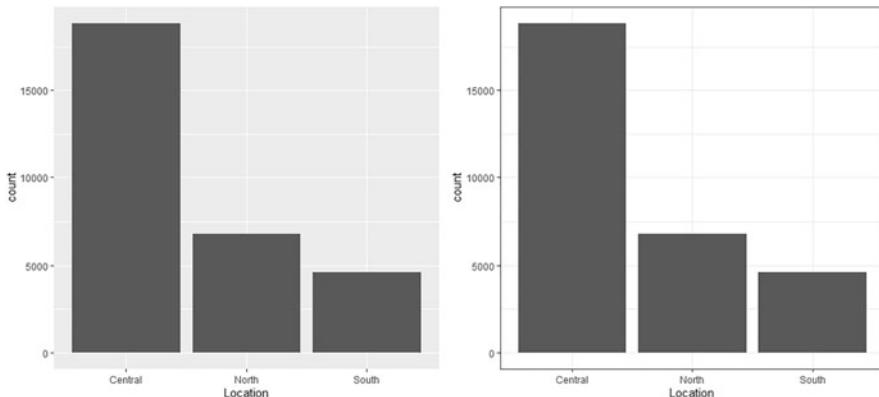


Figure 8.136: ggplot2 package—barchart 1

Although the use of `theme_bw()` may be seen as an improvement, the `ggplot2` environment supports many additional options for graphical presentation, all using different themes. Look below how a wide selection of themes were used to enhance the resulting figure. Notice also how additional information was added to `geom_bar()` and that a title was added using `ggtitle()` (Fig. 8.137).

R Input

```

par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df, aes(x=Location)) +
  geom_bar(stat="count", fill="blue", color="black") +
  ggtitle("Subject (Age >= 21 and Age <= 100) Representation by
Location: Counts") +
  xlab("Location") +
  ylab("Count") +
  scale_y_continuous(labels=scales::comma, limits=c(0,20000)) +
# From this point on in this section of syntax, notice how
# the many options with ggplot2 are used to create a
# figure that is based on bold (e.g., face="bold") and large
# (e.g., size=14) fonts. This selection is purposely used to
# create figures where details are easily seen, especially
# for large-group presentations where the image is projected
# to a screen. Research the literature and experiment with
# different colors, font types, and font sizes to see what
# works best, based on local conditions and needs.
theme(plot.title=element_text(face="bold", size=14)) +
theme(axis.title.x=element_text(face="bold", size=14)) +
theme(axis.text.x=element_text(face="bold", size=14)) +
theme(axis.title.y=element_text(face="bold", size=14)) +
theme(axis.text.y=element_text(face="bold", size=14)) +
# Following along with large bold fonts, notice how the axis
# is also easy to view, with large tick marks that are also
# thick and extend a fair distance beyond regular placement.
theme(axis.ticks.x=element_line(size=2)) +
theme(axis.ticks.y=element_line(size=2)) +
theme(axis.ticks.length=unit(0.5,"cm")) +
theme(panel.background=element_rect(fill="grey90"))

```

The use of additional themes is valuable in an attempt to alter font appearance and size, axis and tick mark presentation, title, appearance and size, etc. However, the use of these additional themes requires many lines of syntax that could easily be placed into a reproducible set of syntax, constructing a user-created theme which is called `theme_MacYates()` in this example. Recall that `theme_MacYates` is a function with specific attributes, as defined immediately below:

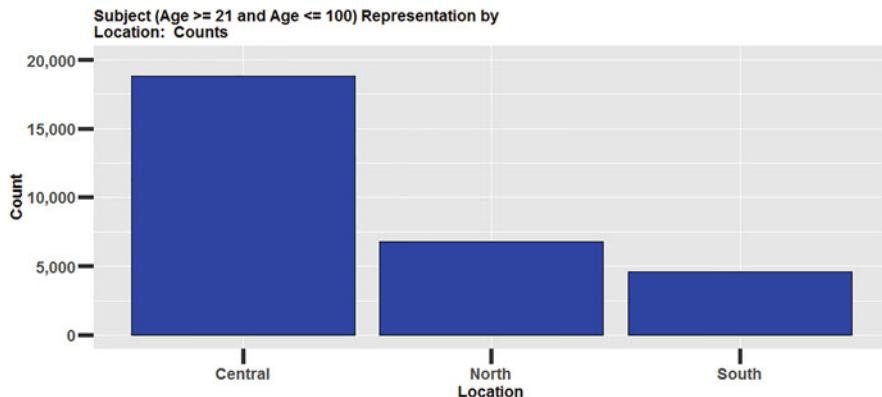


Figure 8.137: ggplot2 package—barchart 2

R Input

```
#####
theme_MacYates <- function(base_size=12, base_family="sans"){
  theme(
    plot.title=element_text(face="bold", size=14, hjust=0),
    plot.caption=element_text(face="bold", size=8, hjust=0),
    axis.title.x=element_text(face="bold", size=14,
      hjust=0.5),
    axis.text.x=element_text(face="bold", size=12, angle=00,
      color="black"),
    axis.title.y=element_text(face="bold", size=12, vjust=1,
      angle=90),
    axis.text.y=element_text(face="bold", size=10, angle=00,
      color="black", hjust=1),
    legend.title=element_text(face="bold", size=12),
    legend.text=element_text(face="bold", size=12),
    axis.ticks.x=element_line(size=1.0),
    axis.ticks.y=element_line(size=1.0),
    axis.ticks.length=unit(0.20,"cm"),
    axis.line=element_line(color="black", size=1.5,
      linetype = "solid"),
    panel.background=element_rect(fill="whitesmoke")
  )
}
# hjust - horizontal justification; 0 = left edge to 1 = right
# edge, with 0.5 the default
# vjust - vertical justification; 0 = bottom edge to 1 = top
# edge, with 0.5 the default
```

```
# angle - rotation; generally 0 to 90 degrees, with 0 the
# default
#####
#####
```

To learn more about the structure of the theme_MacYates() function, key the following at the R prompt:

R Input

```
base::class(theme_MacYates)
utils::str(theme_MacYates)
base::attributes(theme_MacYates)
```

Apply theme_MacYates() against the next figure to see how easy it is to apply the user-created theme to a figure (Fig. 8.138).

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df, aes(x=Location)) +
  geom_bar(stat="count", position="stack", aes(fill=Sex)) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Representation by
Location and by Sex: Stacked Bar Chart of Counts") +
  xlab("Location") +
  ylab("Count") +
  scale_y_continuous(labels=scales::comma, limits=c(0,20000)) +
  theme_MacYates()
# In contrast to the prior figure, where nine lines of syntax
# were needed to produce the desired view of bold fonts and
# large fonts, the use of theme_MacYates() required only one
# line of syntax to generate a desirable user-created theme.
```

Going along with the use of different themes for presentation purposes, review how the ggthemes package was brought into this R session, earlier in the main part of this lesson when all other ggplot2-related packages were downloaded. By using the ggthemes package, many different themes can be presented to make appealing figures, all by selecting one theme or another. The selected themes should ultimately serve the purpose of best conveying intended outcomes, but there are about 15 commonly used themes to select from. Further, these many themes have different options for additional geoms and scales. A few of the leading themes available from the ggthemes package are demonstrated. Contrast the presentation of these themes to theme_MacYates(), which emphasizes visualizations that put text into bold and large fonts (Figs. 8.139 and 8.140).

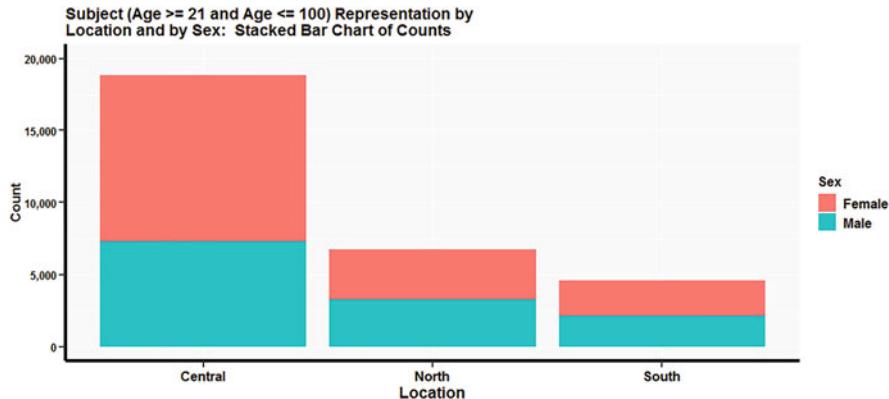


Figure 8.138: ggplot2 package—barchart 3

R Input

```

ggplotbarplotsidebyside <-
  ggplot2::ggplot(HospitalAge021100.df, aes(x=Location)) +
    geom_bar(stat="count", position="dodge", aes(fill=Sex)) +
    ggtitle("Subject (Age >= 21 and Age <= 100) Representation by
    Location and by Sex: Side-by-Side Bar Chart of Counts") +
    xlab("Location") +
    ylab("Count") +
    scale_y_continuous(labels=scales::comma, limits=c(0,12500)) +
    theme_base(base_size=10)
  # Although theme_base() was selected, remember that there are
  # many available themes.

ggplotbarplotstacked <-
  ggplot2::ggplot(HospitalAge021100.df, aes(x=Location)) +
    geom_bar(stat="count", position="fill", aes(fill=Sex)) +
    ggtitle("Subject (Age >= 21 and Age <= 100) Representation by
    Location and by Sex: Stacked Bar Chart of Proportions") +
    xlab("Location") +
    ylab("Proportion (e.g., Proxy for Percentage)") +
    scale_y_continuous(labels=scales::comma, limits=c(0,1),
      breaks=scales::pretty_breaks(n = 10)) +
    theme_calc()

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotbarplotsidebyside,
  ggplotbarplotstacked, ncol=2)

```

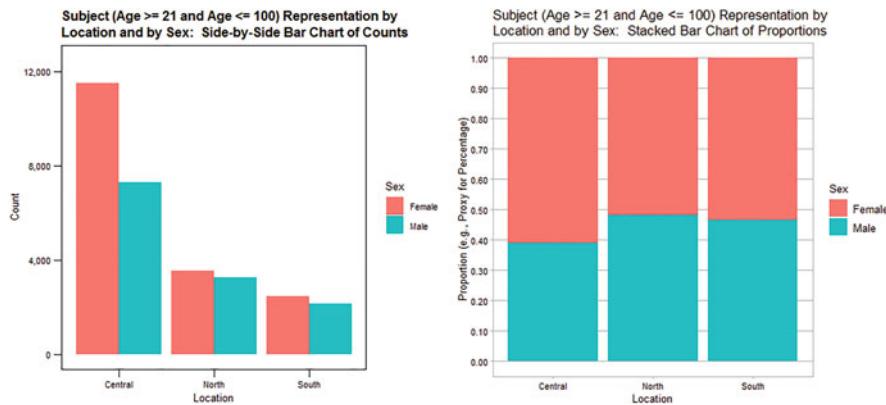


Figure 8.139: ggplot2 package—barchart 4

Box Plot

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=Sex, y=DBP, fill=Sex)) +
  geom_boxplot() +
  stat_summary(fun=mean, geom="point", shape=1, size=6,
  col="black") +
  # Add a circle to represent the mean, along with the median
  # which shows as the solid line
  ggttitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure by Sex: Side-by-Side Box Plot With
Superimposed Mean as a Circle") +
  xlab("Sex") +
  ylab("Diastolic Blood Pressure\n") + # \n for Y Axis spacing
  scale_y_continuous(labels=scales::comma, limits=c(0,120),
  breaks=scales::pretty_breaks(n = 10)) +
  theme_economist()
# The blue background may be a distraction when using
# theme_economist(). Note in the next figure how this issue
# is accommodated.
```

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=Race.Recode, y=DBP, fill=Race.Recode)) +
  geom_boxplot() +
  coord_flip()
```

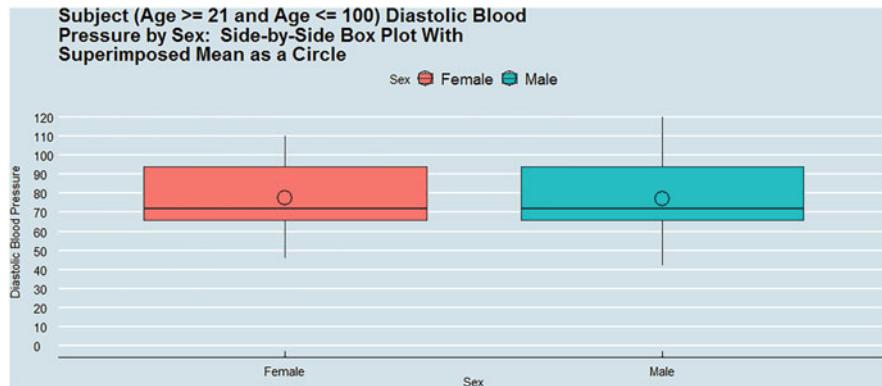


Figure 8.140: ggplot2 package—box plot 1

```

# Flip the X and Y axis, which is often useful for when there
# are many boxplots in the final output -- 7 in this figure
stat_summary(fun=mean, geom="point", shape=1, size=4,
  col="black") +
# Add a circle to represent the mean, along with the median
# which shows as the solid line
scale_fill_manual(values=c("aliceblue", "khaki", "lightblue",
  "orange", "cyan", "pink", "bisque")) +
# Manually declare desired colors for the boxplots to lighter
# shades so that the mean, as a circle, is easily seen
ggttitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure (DBP) by Race: Boxplot With Superimposed
Mean as a Circle on Flipped X and Y Axis") +
xlab("Race\n") +
ylab("\nDiastolic Blood Pressure\n") +
# The X Axis and Y Axis were flipped, so flip placement of
# \n also, to obtain correct spacing.
scale_y_continuous(labels=scales::comma, limits=c(0,120),
  breaks=scales::pretty_breaks(n = 10)) +
# The scale for DBP is purposely set to start at 0 even
# though there are no values, to show the full range.
theme_economist_white(base_size=12, base_family="sans",
  gray_bg=FALSE, horizontal=TRUE)
# For this figure the general approach for presentation using
# theme_economist was retained, but a few changes were made
# to meet specific needs. As stressed throughout the many
# lessons in this text, R and the many packages and functions
# associated with R allow for flexibility and change, when
# needed.

```

The syntax for a singular object variable such as DBP, only, is not the main focus of a boxplot. Boxplots are usually prepared to compare measured outcomes

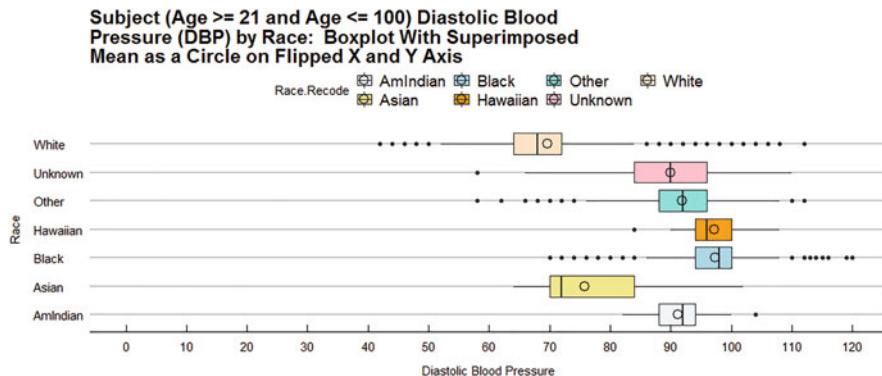


Figure 8.141: ggplot2 package—box plot 2

(e.g., DBP) against breakout groups (e.g., Sex has two breakout groups and Race.Recode has seven breakout groups.). Using the `ggplot2::ggplot()` function with `geom_boxplot()` for a singular object variable, such as DBP, it is necessary to create a dummy (e.g., contrived, fake) grouping variable. This action is needed since there is otherwise no grouping variable to use, in the way that Sex and Race.Recode were previously used. Use the example that follows to prepare a boxplot using the function `ggplot2::ggplot()` for a single measured object variable with no comparisons to other variables (Figs. 8.141–8.147).

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=factor(0), y=DBP)) +
  geom_boxplot() +
  stat_summary(fun=mean, geom="point", shape=1, size=8,
    col="black") +
  ggttitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood Pressure: Boxplot of a Singular Object Variable With Superimposed Mean as a Circle") +
  xlab(" ") +
  # Note how a blank space shows for the X axis label
  ylab("Diastolic Blood Pressure") +
  scale_x_discrete(breaks=NULL) +
  # Note the way the X axis scale is defined, which is needed
  # for this example given how the boxplot is for a singular
  # object value -- only
  # Note how the scale for the Y axis is missing, purposely to
  # show the scale that is produced when scales are not defined
  theme_few()
```

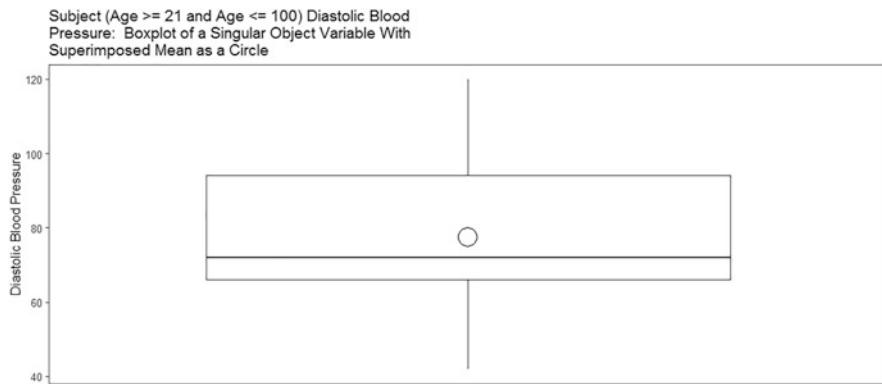


Figure 8.142: ggplot2 package—box plot 3

Density Plot

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=DBP)) +
  geom_density(size=1.5) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood Pressure: Density Plot") +
  xlab("Diastolic Blood Pressure") +
  ylab("Density") +
  theme_fivethirtyeight()
```

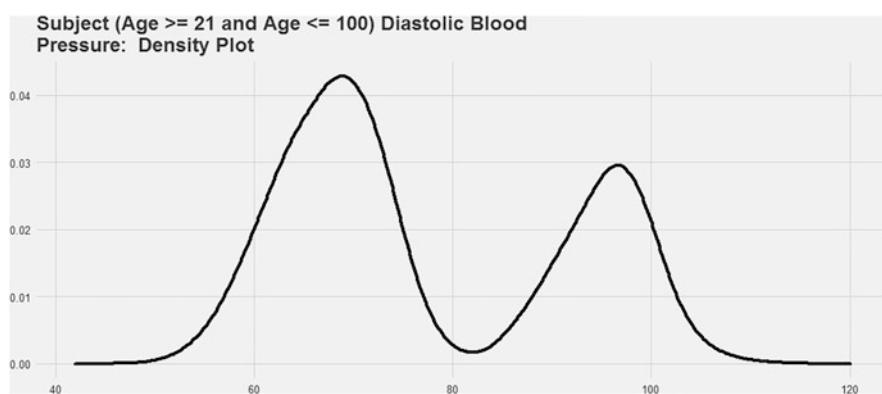


Figure 8.143: ggplot2 package—density plot 1

R Input

```
ggplotdensitydbpsex01 <-
  ggplot2::ggplot(HospitalAge021100.df,
    aes(x=DBP, color=Sex)) +
    geom_density(size=2) +
    ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic
Blood Pressure by Sex: Density Plot") +
    xlab("Diastolic Blood Pressure") +
    ylab("Density") +
    theme_gdocs(base_size=10)

ggplotdensitydbpsex02 <-
  ggplot2::ggplot(HospitalAge021100.df,
    aes(x=DBP)) +
    geom_density(aes(fill=Sex), alpha = 0.5) +
    facet_wrap(~ Sex) +
    ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic
Blood Pressure by Sex: Density Plot") +
    xlab("Diastolic Blood Pressure") +
    ylab("Density") +
    theme_pander()

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotdensitydbpsex01,
  ggplotdensitydbpsex02, ncol=2)
```

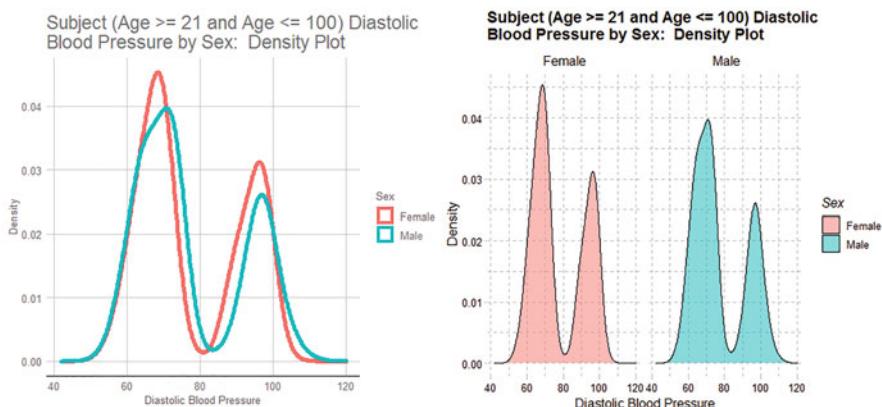


Figure 8.144: ggplot2 package—density plot 2

R Input

```
par.ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
aes(x=DBP)) +
geom_density(aes(fill=Sex), alpha = 0.5) +
facet_wrap(~ PrimaryPayer) +
ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood Pressure by Sex: Density Plot") +
xlab("Diastolic Blood Pressure") +
ylab("Density") +
theme_solarized()
```

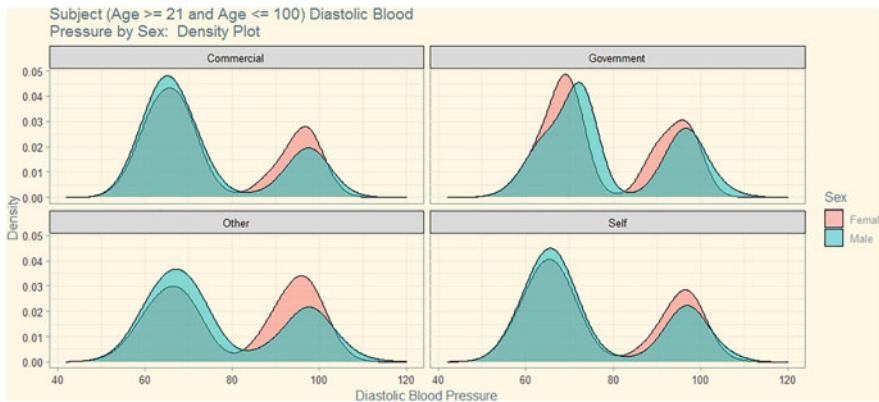


Figure 8.145: ggplot2 package—density plot 3

R Input

```
par.ask=TRUE)
ggplot2::ggplot(data = HospitalAge021100.df,
aes(x=Lbs, color=Sex)) +
geom_density(size=1.25) +
facet_grid(~ Race.Recode) +
ggtitle("Subject (Age >= 21 and Age <= 100) Weight by Race and by Sex: Density Plot") +
xlab("Weight (Lbs)") +
ylab("Density") +
# The arguments for the theme strip.text accommodate desired
# size, color, and angle for the facet text.
# The arguments for the theme strip.background accommodate
# desired color for the facet rectangle.
theme_stata() +
scale_x_continuous(labels=scales::comma, limits=c(75,300)) +
```

```
theme(axis.text.x=element_text(face="bold", size=10,
                               angle=45, hjust=1.25))
# Note the added arguments for theme(axis.text.x), allowing
# for a 45 degree angle and a slight adjustment away from
# the X axis, allowing more space and improved presentation.
```

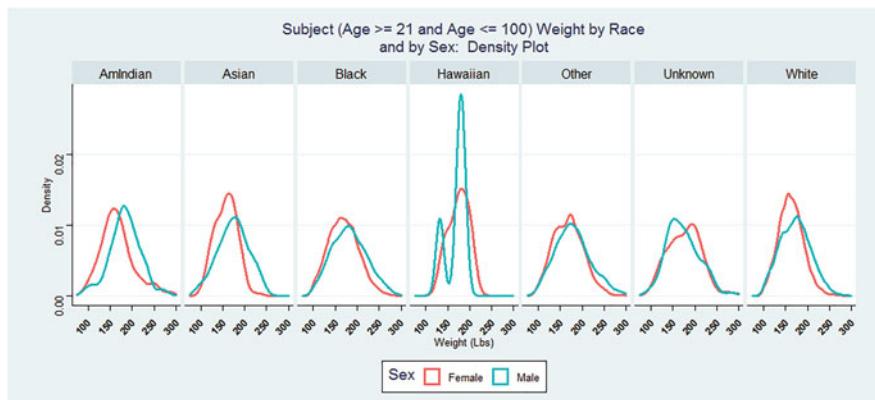


Figure 8.146: ggplot2 package—density plot 4

Dot Plot

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
aes(x=Lbs)) +
  geom_dotplot(binwidth=0.1, stackratio=1.25) +
  ggttitle("Subject (Age >= 21 and Age <= 100) Weight:
Dot Plot") +
  xlab("Weight (Lbs)") +
  ylab("Scale") +
  theme_tufte()
# The theme_tufte() follows along with this approach to data
# visualization, where emphasis is on the data and supporting
# parts of the figure are minimized.
```

The leading themes available from the ggthemes package have been demonstrated and many consider theme_economist() and theme_stata() the most useful. For the remaining ggplot2 figures, theme_MacYates() will be used—or a slight variation for when it is necessary to rotate text, as will be evident when shown (Figs. 8.148, 8.149, and 8.150).

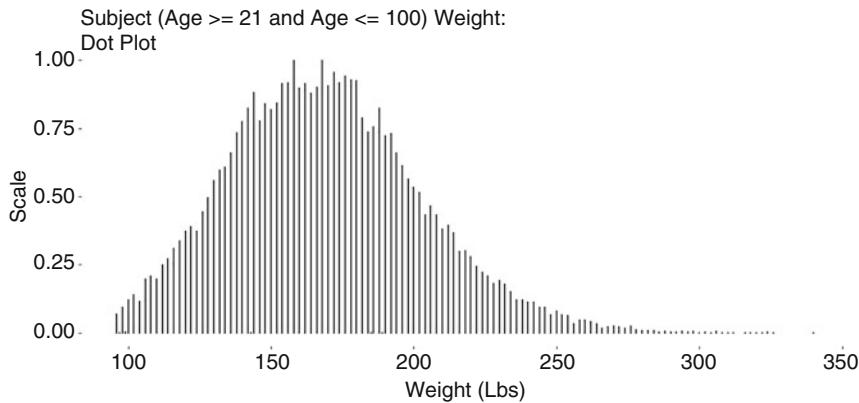


Figure 8.147: ggplot2 package—dot plot 1

R Input

```
ggplotlbs01 <-  
  ggplot2::ggplot(HospitalAge021100.df,  
    aes(x=Lbs)) +  
    geom_dotplot(binwidth=0.1, stackratio=1.25,  
      stackdir="center") +  
    ggtitle("Subject (Age >= 21 and Age <= 100) Weight:  
Dot Plot") +  
    xlab("Weight (Lbs)") +  
    ylab("Scale") +  
    scale_x_continuous(labels=scales::comma, limits=c(0,325),  
      breaks=scales::pretty_breaks(n = 10)) +  
    theme_MacYates()  
  
ggplotlbssex01 <-  
  ggplot2::ggplot(HospitalAge021100.df,  
    aes(x=Lbs)) +  
    geom_dotplot(binwidth=0.1, stackratio=1.25,  
      stackdir="center") +  
    facet_grid(~ Sex) +  
    ggtitle("Subject (Age >= 21 and Age <= 100) Weight by Sex:  
Dot Plot") +  
    xlab("Weight (Lbs)") +  
    ylab("Scale") +  
    scale_x_continuous(labels=scales::comma, limits=c(0,325),  
      breaks=scales::pretty_breaks(n = 10)) +  
    theme(strip.text.x=element_text(face="bold", size=12,  
      color="navyblue")) +
```

```

theme(strip.background=element_rect(fill="wheat1")) +
theme_MacYates()
# Note how theme(strip.text) and theme(strip.background) are
# placed before theme_MacYates()

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotlbs01, ggplotlbssex01, ncol=2)

```

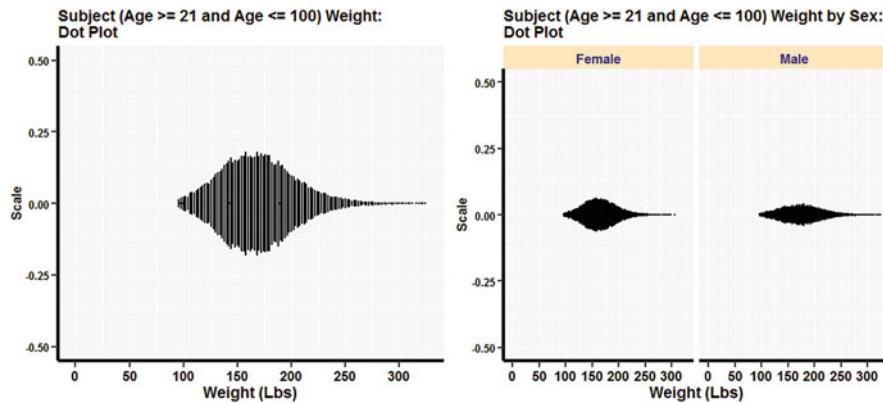


Figure 8.148: ggplot2 package—dot plot 2

R Input

```

ggplotdbpvertical01 <-
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=DBP)) +
  geom_dotplot(binwidth=0.01, stackratio=1.05) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure: Dot Plot") +
  xlab("Diastolic Blood Pressure") +
  ylab("Scale") +
  theme(plot.title=element_text(face="bold", size=14)) +
  theme_MacYates()

ggplotdbphorizontal01 <-
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=DBP)) +
  geom_dotplot(binwidth=0.01, stackratio=1.05,
  stackdir="center") +
  ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure: Dot Plot") +
  xlab("Diastolic Blood Pressure") +

```

```

ylab("Scale") +
scale_x_continuous(labels=scales::comma, limits=c(30,125),
  breaks=scales::pretty_breaks(n = 10)) +
theme_MacYates()

par.ask=TRUE); gridExtra::grid.arrange(
  ggplotdbpvertical01,
  ggplotdbphorizontal01, ncol=2)

```

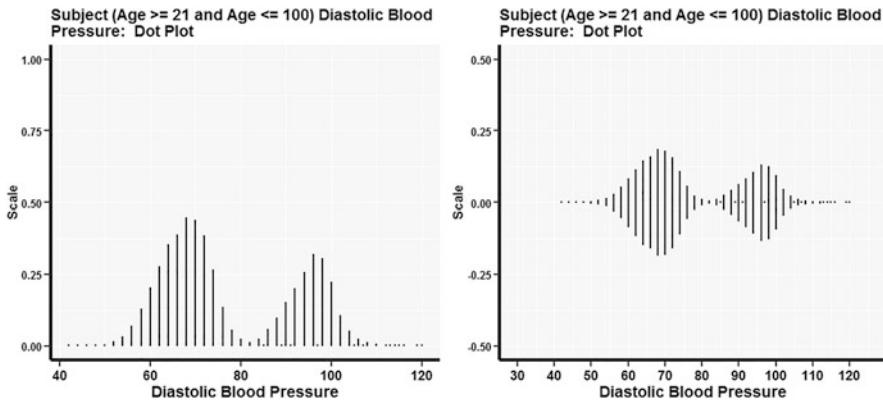


Figure 8.149: ggplot2 package—dot plot 3

R Input

```

par.ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=DBP)) +
  geom_dotplot(binwidth=0.05, stackratio=0.80) +
  # Notice how stackdir="center" was not used.
  # Experiment with different values for arguments such as
  # binwidth and stackratio to see their impact on the
  # figure and desired outcome.
  facet_grid(~ Sex) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure by Sex: Dot Plot") +
  xlab("Diastolic Blood Pressure") +
  ylab("Scale") +
  scale_x_continuous(labels=scales::comma, limits=c(30,125),
    breaks=scales::pretty_breaks(n = 10)) +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1"))

```

```
theme_MacYates()
```

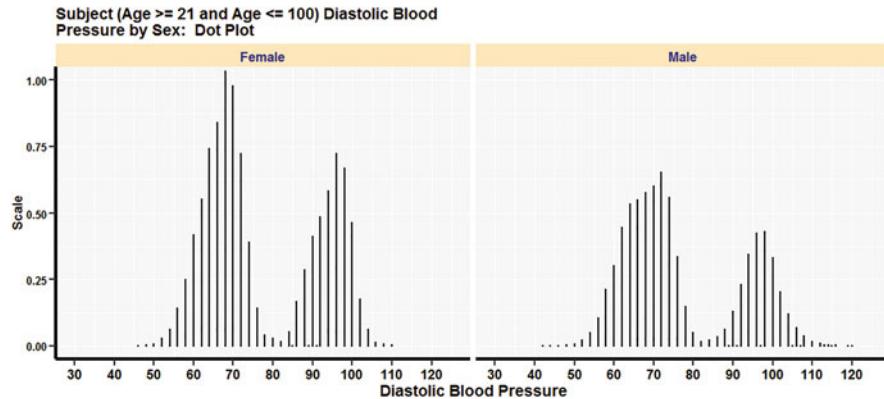


Figure 8.150: ggplot2 package—dot plot 4

Scatter Plot

Pearson's Coefficient of Correlation is the default correlation method for the `cor()` function. The `coef()` function is used to extract model coefficients, which is useful to manually generate the intercept and slope for a correlation model.

R Input

```
stats::cor(HospitalAge021100.df$SBP, HospitalAge021100.df$DBP)
```

R Output

```
[1] 0.854961
```

R Input

```
stats::coef(lm(SBP ~ DBP, data=HospitalAge021100.df))
```

R Output

(Intercept)	DBP
19.35115	1.37295

R Input

```
stats::cor(HospitalAge021100.df$DBP, HospitalAge021100.df$SBP)
```

R Output

```
[1] 0.854961
```

R Input

```
stats::coef(lm(DBP ~ SBP, data=HospitalAge021100.df))
```

R Output

(Intercept)	SBP
10.558317	0.532401

R Input

```
ggplotdbpvsbp01 <-
ggplot2::ggplot(HospitalAge021100.df,
aes(x=DBP, y=SBP)) +
geom_point() +
ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood Pressure by Systolic Blood Pressure: Scatter Plot") +
xlab("Diastolic Blood Pressure (DBP)") +
ylab("Systolic Blood Pressure (SBP)") +
theme_MacYates()

ggplotdbpvsbp02 <-
ggplot2::ggplot(HospitalAge021100.df,
aes(x=DBP, y=SBP)) +
geom_point() +
geom_smooth(method="lm", se=FALSE, size=3,
color="royalblue4") + # abline
ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood Pressure by Systolic Blood Pressure: Scatter Plot") +
xlab("Diastolic Blood Pressure (DBP)") +
ylab("Systolic Blood Pressure (SBP)") +
#####
annotate(geom="text", x=001, y=060, color="darkred",
family="mono", lineheight=1.1, fontface="bold", size=3,
label=
> cor(HospitalAge021100.df$SBP, HospitalAge021100.df$DBP)
0.854961",
hjust=0) +
```

```

#####
##### annotate(geom="text", x=001, y=205, color="darkred",
#####   family="mono", lineheight=1.1, fontface="bold", size=3,
#####   label=
##### > cor(HospitalAge021100.df$DBP, HospitalAge021100.df$SBP)
##### 0.854961",
#####   hjust=0) +
#####
##### theme_MacYates()
# Note how annotate was used to add useful information to
# this figure. Give special notice to the many arguments
# used along with annotate, to position the annotation as
# desired and to make it visually prominent: (1) the geom
# was purposely set as text, (2) the position was set at the
# desired x and y Cartesian coordinates, for this figure, (3)
# the color was set to darkred, (4) the font was set to a
# Courier-type mono (e.g., fixed-spaced) font, (5) the
# lineheight was made somewhat smaller by being set to 1.1
# instead of the default lineheight=1.2 setting, (6) the font
# was made bold, to make the text easy-to-read, (7) the size
# of the annotated text was made somewhat smaller by being
# set to 3 instead of the default size=5 setting, (8) the
# label shows use of the cor() function for the two object
# variables in question, with the output on the line
# immediately below, and (9) hjust was set to 0 so that the
# text would be flush left at the declared coordinates.

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotdbpvsbp01,
  ggplotdbpvsbp02, ncol=2)

```

Before preparing the R syntax needed to display the association between Diastolic Blood Pressure (DBP—X axis) and Systolic Blood Pressure (SBP—Y axis) by Sex (Female and Male), it is first helpful to look at a few summary statistics (Fig. 8.151):

R Input

```
base::summary(HospitalAge021100.df$Sex)
```

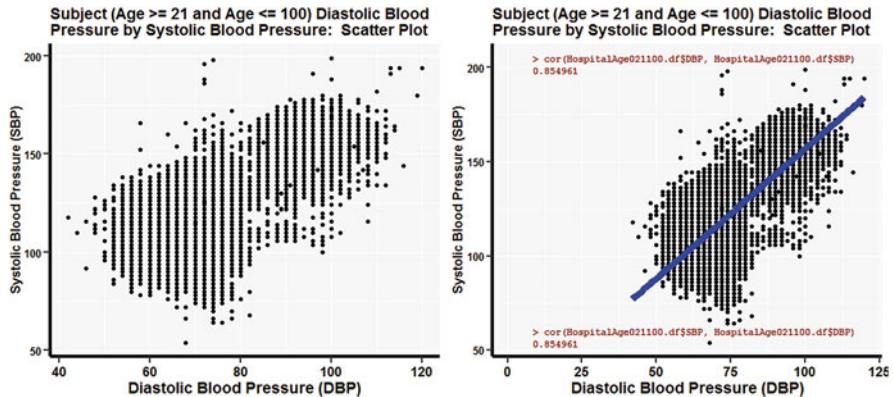


Figure 8.151: ggplot2 package—scatter plot 1

R Output

```
Female    Male
 17512   12715
```

R Input

```
base:::summary(HospitalAge021100.df$Sex=="Female")
```

R Output

Mode	FALSE	TRUE
logical	12715	17512

R Input

```
base:::summary(HospitalAge021100.df$Sex=="Male")
```

R Output

Mode	FALSE	TRUE
logical	17512	12715

R Input

```
base:::table(HospitalAge021100.df$Sex)
```

R Output

```
Female    Male
17512   12715
```

Although the overall correlation between DBP and SBP has been established (Pearson's $r = 0.854961$), it would be helpful to calculate Pearson's r of SBP and DBP for Females and to also calculate Pearson's r of SBP and DBP for Males. There are many ways to calculate these breakout coefficients of correlation, but for this example observe how two new datasets are created, one dataset (`HospitalAge021100Female.df`) consisting of only Females and another dataset (`HospitalAge021100Male.df`) consisting of only Males. The `stats::cor()` function and the `stats::coef()` function are then applied, but first for Females in the Female-only dataset and then a second time for Males in the Male-only dataset.

R Input

```
HospitalAge021100Female.df <- HospitalAge021100.df[ which(
  HospitalAge021100.df$Sex=="Female"), ]
attach(HospitalAge021100Female.df)

summary(HospitalAge021100Female.df$Sex)
# Female only dataset, based on the which() function
```

R Output

```
Female    Male
17512      0
```

R Input

```
stats::cor(HospitalAge021100Female.df$SBP,
  HospitalAge021100Female.df$DBP)
```

R Output

```
[1] 0.879456
```

R Input

```
stats::coef(lm(SBP ~ DBP, data=HospitalAge021100Female.df))
# Female Pearson's r and model coefficients
```

R Output

(Intercept)	DBP
16.44562	1.41271

R Input

```
HospitalAge021100Male.df <- HospitalAge021100.df[ which(
  HospitalAge021100.df$Sex=="Male"), ]
attach(HospitalAge021100Male.df)
```

R Input

```
base::summary(HospitalAge021100Male.df$Sex)
# Male only dataset, based on the which() function
```

R Output

Female	Male
0	12715

R Input

```
stats::cor(HospitalAge021100Male.df$SBP,
  HospitalAge021100Male.df$DBP)
```

R Output

[1] 0.8223

R Input

```
stats::coef(lm(SBP ~ DBP, data=HospitalAge021100Male.df))
# Male Pearson's r and model coefficients
```

R Output

(Intercept)	DBP
23.22271	1.31958

These correlation-related statistics will give greater context to the graphical output than either the statistics or figures alone (Fig. 8.152).

R Input

```
ggplotdbpvsbpsex01 <-
  ggplot2::ggplot(HospitalAge021100.df,
    aes(x=DBP, y=SBP)) +
    geom_point(aes(color=Sex)) +
    geom_smooth(method="lm", se=FALSE, size=3,
      color="royalblue4") + # abline
    ggttitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure by Systolic Blood Pressure and
by Sex: Scatter Plot") +
    xlab("Diastolic Blood Pressure (DBP)") +
    ylab("Systolic Blood Pressure (SBP)") +
    theme_MacYates()

ggplotdbpvsbpsex02 <-
  ggplot2::ggplot(HospitalAge021100.df,
    aes(x=DBP, y=SBP)) +
    geom_point(aes(color=Sex)) +
    geom_smooth(method="lm", se=FALSE, size=3,
      color="royalblue4") + # abline
    facet_grid(~ Sex) + # Breakout panels by Sex using facet
    ggttitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure by Systolic Blood Pressure and
by Sex: Scatter Plot") +
    xlab("Diastolic Blood Pressure (DBP)") +
    ylab("Systolic Blood Pressure (SBP)") +
    scale_x_continuous(labels=scales::comma, limits=c(40,120),
      breaks=scales::pretty_breaks(n = 5)) +
    scale_y_continuous(labels=scales::comma, limits=c(50,200),
      breaks=scales::pretty_breaks(n = 5)) +
    theme(strip.text.x=element_text(face="bold", size=10,
      color="navyblue")) +
    theme(strip.background=element_rect(fill="wheat1")) +
    theme_MacYates()
```

```
par(aspect=TRUE); gridExtra::grid.arrange(
  ggplotdbpvsbpsex01,
  ggplotdbpvsbpsex02, ncol=2)
```

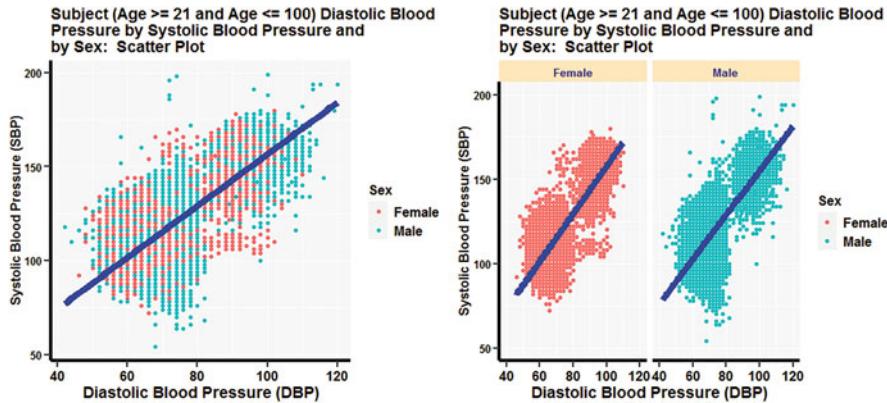


Figure 8.152: ggplot2 package—scatter plot 2

The ggplot2 package can also be used to construct a correlation matrix, using the GGally package, which is an extension to the ggplot2 package. A full set of plotting-type functions are available in GGally, to increase functionality of the ggplot2 package. Observe the syntax of GGally and how it is used as a supplement to the ggplot2 package.

For context, review the Pearson's r correlation coefficients for SBP v DBP, SBP v Lbs, and DBP v Lbs.²¹

R Input

```
cor(HospitalAge021100.df$SBP, HospitalAge021100.df$DBP)
```

R Output

```
[1] 0.854961
```

R Input

```
cor(HospitalAge021100.df$SBP, HospitalAge021100.df$Lbs)
```

²¹Although the expressions SBP v DBP, SBP v Lbs, and DBP v Lbs were used to identify the correlation-focused object variables, it is not uncommon to also see the expressions SBP:DBP, SBP:Lbs, and DBP:Lbs to express the same constructs.

R Output

```
[1] 0.0742602
```

R Input

```
cor(HospitalAge021100.df$DBP, HospitalAge021100.df$Lbs)
```

R Output

```
[1] 0.0670301
```

Use the GGally::ggcorr() function, coupled with options from the ggplot2 package to create a correlation matrix, where the correlation values show in a set of squares (e.g., tiles) set off by a color-coded legend of potential Pearson's r values, ranging from -1.0 to +1.0 (Figs. 8.153 and 8.154).

R Input

```
install.packages("GGally", dependencies=TRUE)
library(GGally)                      # Load the GGally package.
help(package=GGally)                  # Show the information page.
sessionInfo()                        # Confirm all attached packages.
# GGally is an extension to ggplot2.
# Be patient and allow time for GGally to download.

par(ask=TRUE)
GGally::ggcorr(HospitalAge021100.df[, 18:20],      # Columns
               geom="tile",                         # Squares
               method=c("pairwise", "pearson"),       # Correlation
               hjust=0.75, size=5, color="black",      # SBP DBP Lbs
               low="dodgerblue", mid="wheat", high="cyan", # Color scale
               name="Pearson's r", label=TRUE, label_size=3, # Legend
               label_color="black", label_round=3) +
  ggtile(
    "Subject (Age >= 21 and Age <= 100) Systolic Blood Pressure,
     Diastolic Blood Pressure, and Weight: Colored Correlation
     Matrix -- Blue (-1.0) to Gray (0.0) to Cyan (+1.0)") +
  theme_MacYates()
# The expression [, 18:20] refers to columns 18, 19, and 20,
# or SBP (column 18), DBP (column 19), and Lbs (column 20).
```

**Subject (Age >= 21 and Age <= 100) Systolic Blood Pressure,
Diastolic Blood Pressure, and Weight: Colored Correlation
Matrix -- Blue (-1.0) to Gray (0.0) to Cyan (+1.0)**



Figure 8.153: ggplot2 package and GGally package—scatter plot 1

The GGally::ggpairs() function is used to produce a correlation matrix that displays a density curve, a plot of X v Y, and the Pearson's r statistic for each X v Y correlation.

R Input

```
par(ask=TRUE)
GGally::ggpairs(HospitalAge021100.df[, 18:20],
  axisLabels="show") +
  ggttitle("Subject (Age >= 21 and Age <= 100) Systolic Blood
Pressure, Diastolic Blood Pressure, and Weight:
Correlation Matrix (Density Curve, Plot,
and Pearson's r)") +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme(strip.text.y=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme_MacYates()
```

Using a slightly different approach to how the R-based syntax is used with the GGally::ggpairs() function, the correlation matrix can also visually present differences between the two Sexes, Female and Male. The Pearson's r coefficient

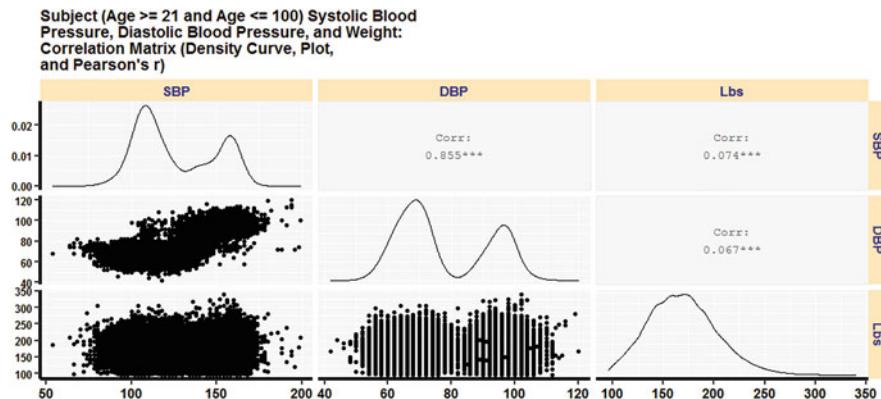


Figure 8.154: ggplot2 package and GGally package—scatter plot 2

of correlation is provided overall and then for the two Sex breakouts, Female and Male (Figs. 8.155, 8.156, and 8.157).

R Input

```
par.ask=TRUE)
GGally::ggpairs(data=HospitalAge021100.df, columns=18:20,
  axisLabels="show", ggplot2::aes(color=Sex),
  ggttitle("Subject (Age >= 21 and Age <= 100) Systolic Blood
Pressure, Diastolic Blood Pressure, and Weight
by Sex: Correlation Matrix") +
  theme(strip.text.x=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme(strip.text.y=element_text(face="bold", size=12,
    color="navyblue")) +
  theme(strip.background=element_rect(fill="wheat1")) +
  theme_MacYates()
# Note the different approach to how columns 18 to 20 were
# identified.
```

Violin Plot

R Input

```
par.ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=Sex, y=DBP, fill=Sex)) +
  geom_violin() +
  stat_summary(fun=mean, geom="point", shape=1, size=6,
```

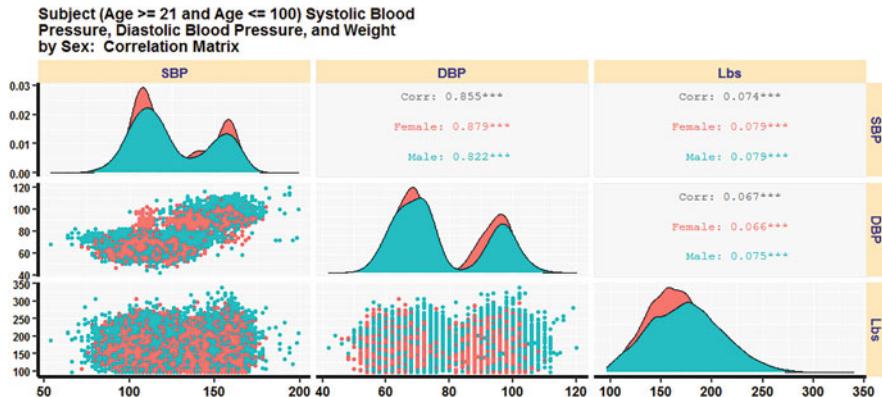


Figure 8.155: ggplot2 package and GGally package—scatter plot 3

```

col="black") +
# Add a circle to represent the mean, along with the median
# which shows as the solid line
ggttitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure) by Sex: Side-by-Side Violin Plot With
Superimposed Mean as a Circle") +
xlab("Sex") +
ylab("Diastolic Blood Pressure") +
scale_y_continuous(labels=scales::comma, limits=c(0,120),
breaks=scales::pretty_breaks(n = 10)) +
theme_MacYates()

```

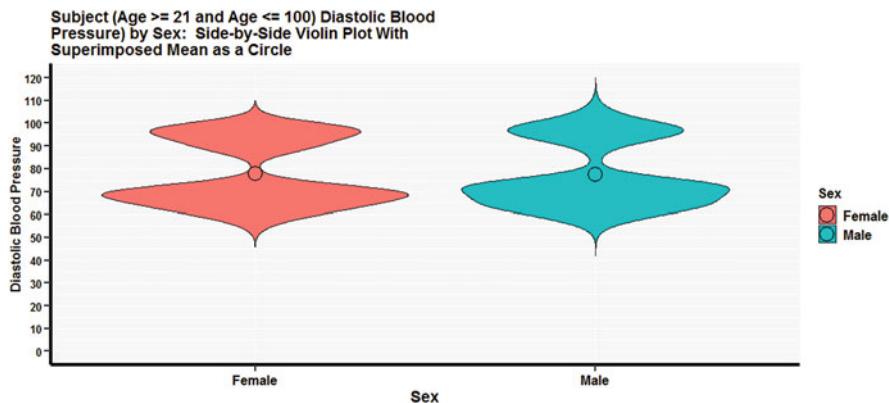


Figure 8.156: ggplot2 package—violin plot 1

R Input

```

par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
aes(x=Race.Recode, y=DBP, fill=Race.Recode)) +

```

```

geom_violin() +
coord_flip() +
# Flip the X and Y axis, which is often useful for when there
# are many boxplots in the final output -- 7 in this figure
stat_summary(fun=mean, geom="point", shape=1, size=4,
             col="black") +
# Add a circle to represent the mean, along with the median
# which shows as the solid line
scale_fill_manual(values=c("aliceblue", "khaki", "lightblue",
                           "orange", "cyan", "pink", "bisque")) +
# Manually declare desired colors to lighter shades so that
# the mean, as a circle, is easily seen
ggttitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure by Race: Violin Plot With Superimposed Mean
as a Circle on Flipped X and Y Axis") +
xlab("Race") +
ylab("Diastolic Blood Pressure") +
scale_y_continuous(labels=scales::comma, limits=c(0,120),
                   breaks=scales::pretty_breaks(n = 10)) +
# The scale for DBP is purposely set to start at 0 even
# though there are no values, to show the full range.
theme_MacYates()

```

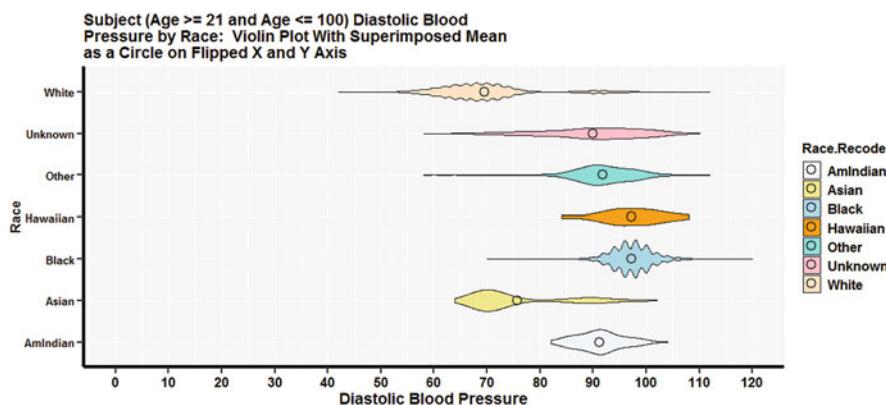


Figure 8.157: ggplot2 package—violin plot 2

The syntax for a singular object variable such as DBP, only, is not the main focus of a violin plot. Violin plots are usually prepared to compare measured outcomes (e.g., DBP) against breakout groups (e.g., There are two breakout groups for Sex and seven breakout groups for Race.). Using the `ggplot2::ggplot()` function with `geom_violin()` for a singular object variable, such as DBP, it is necessary to create a dummy (e.g., contrived, fake) grouping variable. This action is needed since there is otherwise no grouping variable to use, in the way that Sex and

Race.Recode were previously used. Use the example that follows to prepare a violin plot using `ggplot2::ggplot()` for a single measured object variable, showing as a violin plot, with no comparisons to other variables (Fig. 8.158).

R Input

```
par(ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
  aes(x=factor(0), y=DBP)) +
  geom_violin(fill="aliceblue", color="black", size=1.5) +
  stat_summary(fun="mean", geom="point", shape=8, size=10,
    col="red") +           # A star is made by using shape=8.
  ggtitle("Subject (Age >= 21 and Age <= 100) Diastolic Blood
Pressure (DBP): Violin Plot of a Singular Object
Variable with Superimposed Mean as a Star") +
  xlab(" ") +
  # A blank space shows for the X axis label.
  ylab("Diastolic Blood Pressure") +
  scale_x_discrete(breaks=NULL) +
  # Note the way the x axis scale is defined, which is needed
  # for this example given how the violin plot is for a
  # singular object value -- only
  # Note how the scale for the Y axis is missing, purposely to
  # show the scale that is produced when scales are not defined
  theme_MacYates()
```

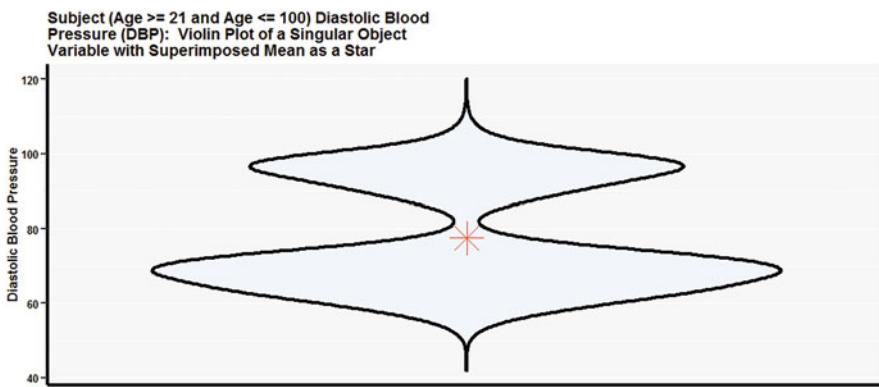


Figure 8.158: `ggplot2` package—violin plot 3

Violin plots can also be used to compare multiple breakouts using `facet_grid()`. With proper design, this multi-level comparison can also have comparisons within each part of the grid. In the example that follows, the figure supports 14 presentations of Weight (Lbs), with seven violin plot Race groups showing for Female and seven violin plot Race groups showing for Male (Fig. 8.159).

R Input

```
par.ask=TRUE)
ggplot2::ggplot(HospitalAge021100.df,
aes(Sex, Lbs)) +
geom_violin(aes(fill=Race), draw_quantiles=c(0.25, 0.5,
0.75), size=1.25) +
ggtitle("Subject (Age >= 21 and Age <= 100) Weight by Race
and by Sex: Violin Plot") +
xlab("Sex") +
ylab("Weight (Lbs)") +
scale_y_continuous(labels=scales::comma, limits=c(0,350),
breaks=scales::pretty_breaks(n = 10)) +
theme_MacYates()
# Note how size=1.25 was used to make the lines thicker and
# darker, making them easier to see -- especially in a group
# presentation.
```

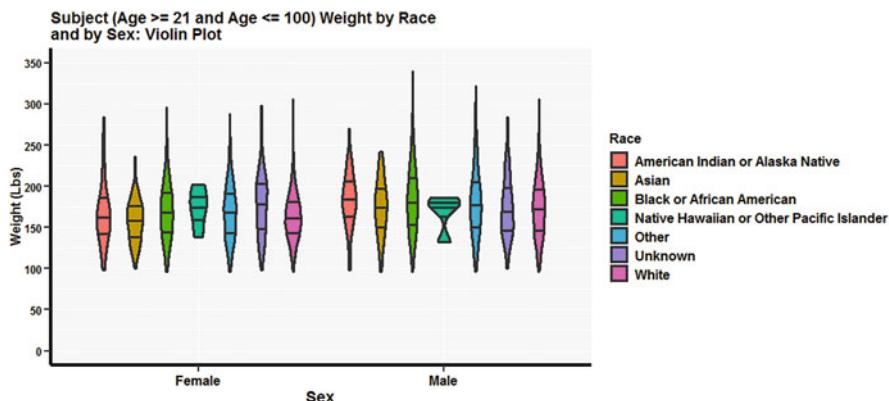


Figure 8.159: ggplot2 package—violin plot 4

However, the prior figure is somewhat difficult to understand since there is no clear break in the figure. Instead, look at the following syntax and use of `facet_grid()`, once again for Race, and how this approach makes for a figure that is far easier to follow and in turn compare weights between Race breakout groups for Females and Males. Note also how lines have been superimposed on the violin plots indicating the 25th percentile, median (50th percentile), and the 75th percentile. This figure is highly-detailed but still easy to follow for comparative purposes (Figs. 8.160–8.165).

R Input

```
par.ask=TRUE)
ggplot2::ggplot(data = HospitalAge021100.df,
  aes(Sex, Lbs, fill=Sex)) +
  geom_violin(draw_quantiles=c(0.25, 0.5, 0.75), size=1.25) +
  facet_grid(~ Race.Recode) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Weight by Race and by Sex: Violin Plot") +
  xlab("\nSex") +
  ylab("Weight (Lbs)\n") +
  scale_y_continuous(labels=scales::comma, limits=c(0,350),
    breaks=scales::pretty_breaks(n = 5)) +
  theme_economist_white(base_size=12, base_family="sans",
    gray_bg=FALSE, horizontal=TRUE)
# Note how theme_economist_white() was used, instead of
# theme_MacYates(), to allow for another view of this figure.
# Also, size=1.25 was used to make the lines thicker and
# darker, making them easier to see -- especially in a group
# presentation.
```

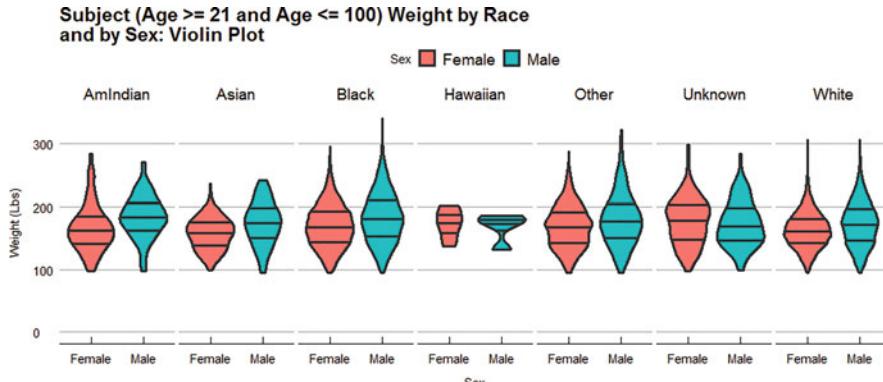


Figure 8.160: ggplot2 package—violin plot 5

Histogram

R Input

```
par.ask=TRUE)
ggplot2::ggplot(data = HospitalAge021100.df,
  aes(x=Lbs)) +
  geom_histogram(binwidth=1.0) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Weight:
```

```
Histogram") +
  xlab("Weight (Lbs)") +
  ylab("Count") +
  scale_x_continuous(labels=scales::comma, limits=c(0,350),
    breaks=scales::pretty_breaks(n = 10)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,800),
    breaks=scales::pretty_breaks(n = 10)) +
  theme_MacYates()
# Experiment with the binwidth setting, to communicate to
# the reader the most appealing and informative graphical
# representation of data distribution as a histogram when
# using geom_histogram(). By design, the default for both
# binwidth and bins is only rarely desirable, calling for
# experimentation that eventually enhances output.
```

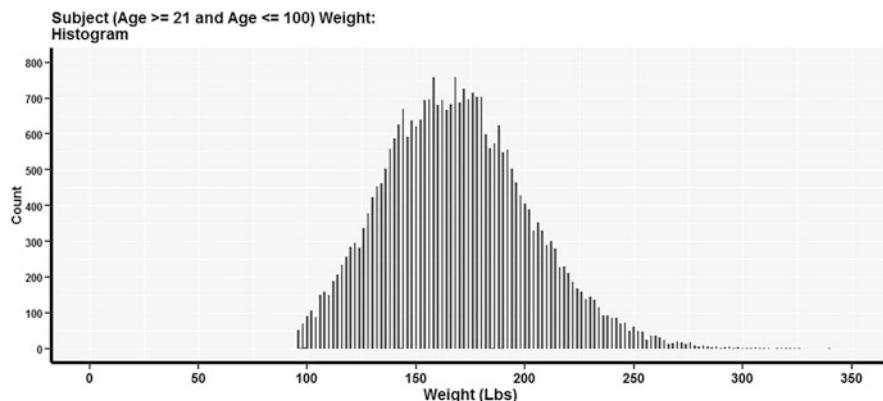


Figure 8.161: ggplot2 package—histogram 1

R Input

```
base::mean(HospitalAge021100.df$Lbs)
```

R Output

```
[1] 169.035
```

R Input

```
stats::median(HospitalAge021100.df$Lbs)
```

R Output

```
[1] 168
```

R Input

```
par(ask=TRUE)
ggplot2::ggplot(data = HospitalAge021100.df,
  aes(x=Lbs)) +
  geom_histogram(binwidth=2, color="black", fill="white") +
  geom_vline(aes(xintercept=mean(Lbs, na.rm=TRUE)),
    color="darkred", linetype="dashed", size=1.15) +
  geom_vline(aes(xintercept=median(Lbs, na.rm=TRUE)),
    color="dodgerblue", linetype="dotted", size=1.15) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Weight:
Histogram, with Mean Showing as a Red
Dashed Vertical Line and Median
Showing as a Blue Dotted Line") +
  xlab("\nWeight (Lbs)") +
  ylab("Count\n") +
  scale_x_continuous(labels=scales::comma, limits=c(0,350),
    breaks=scales::pretty_breaks(n = 10)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,800),
    breaks=scales::pretty_breaks(n = 10)) +
  theme_MacYates()
# Use geom_vline() with the appropriate arguments (above) to
# display the mean and median for Weight (Lbs):
#   Mean ..... darkred dashed vertical line    169.0347
#   Median ... dodgerblue dotted vertical line  168
# Note how the mean (169.035) and median (168) are in such
# close proximity. Equally, note how the histogram begins to
# approximate the bell-shaped curve typically associated with
# the assumption of normal distribution. Even so, it is best
# to use the Quantile-Quantile (Q-Q) plot to further examine
# distribution and to then subject the data (e.g., Weight
# (Lbs) in this example) to the appropriate statistical test
# (Anderson-Darling Test for Normality, Jarque-Bera Test for
# Normality, Lilliefors (Kolmogorov-Smirnov) Test for
# Normality, Shapiro-Wilk Test for Normality) to make an
# informed judgment about data distribution. The histogram
# is only one of many tools for inquiry into the general
# nature data distribution patterns.
```

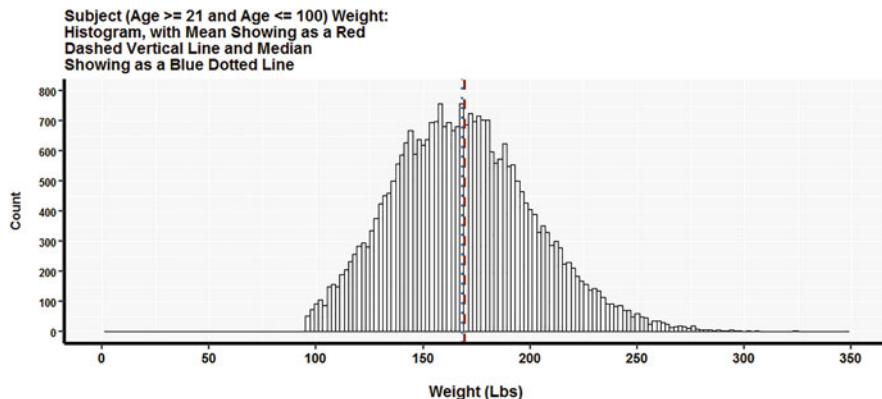


Figure 8.162: ggplot2 package—histogram 2

R Input

```
ggplot.histogramlbs01 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(x=Lbs)) +
geom_histogram(binwidth=2.0, color="cyan",
fill="red") +
ggttitle("Subject (Age >= 21 and Age <= 100) Weight:
Histogram") +
xlab("Weight (Lbs)") +
ylab("Count") +
scale_x_continuous(labels=scales::comma, limits=c(0,350),
breaks=scales::pretty_breaks(n = 10)) +
scale_y_continuous(labels=scales::comma, limits=c(0,800),
breaks=scales::pretty_breaks(n = 10)) +
theme_MacYates()
# Along with experimentation of the binwidth setting, it may
# be useful to also experiment with different colors for the
# outer color of each bar (e.g., color) and the filled color
# of each bar (e.g., fill).

ggplot.histogramlbs02 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(x=Lbs)) +
geom_histogram(binwidth=1.0, color="cyan", fill="red",
aes(y=..density..)) +
stat_density(geom = "line", size=3, aes(color="red")) +
ggttitle("Subject (Age >= 21 and Age <= 100) Weight:
Histogram and Overlay Density Curve") +
```

```

xlab("Weight (Lbs)") +
ylab("Density") +
scale_x_continuous(labels=scales::comma, limits=c(0,350),
  breaks=scales::pretty_breaks(n = 10)) +
scale_y_continuous(labels=scales::comma, limits=c(0,0.026),
  breaks=scales::pretty_breaks(n = 10)) +
theme(legend.position="none") +
theme_MacYates()
# Note how stat_density() was used to overlay, as a red line,
# a kernel density curve. This overlay of a kernel density
# curve could have been produced by using geom_density(),
# also. As has been mentioned many times, R is exceptionally
# flexible and there are often many ways to produce desired
# output.
#
# Note also how theme(legend.position="none") was used to
# remove the legend, which is otherwise not helpful in this
# figure since there is only one object variable for display.

par(aspect=TRUE); gridExtra::grid.arrange(
  ggplothistogramlbs01,
  ggplothistogramlbs02, ncol=2)

```

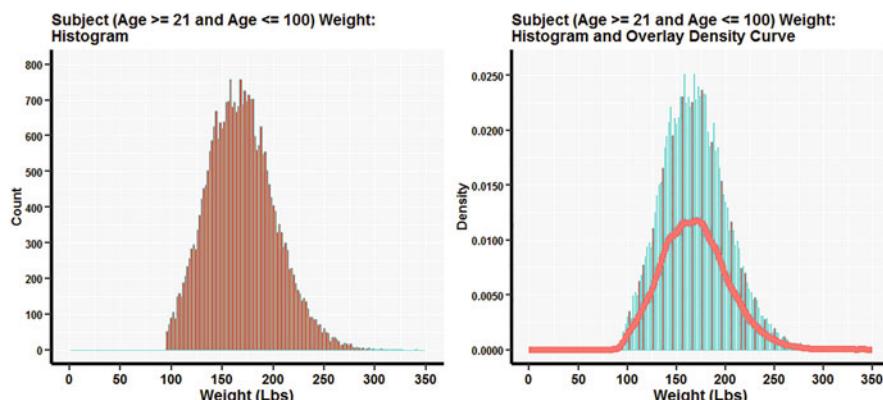


Figure 8.163: ggplot2 package—histogram 3

R Input

```

par(aspect=TRUE)
ggplot2::ggplot(data = HospitalAge021100.df,
  aes(x=Lbs)) +

```

```

geom_histogram(binwidth=1, aes(fill=..count..)) +
  scale_fill_gradient("Count", low="coral",
    high="dodgerblue") +
  ggtitle("Subject (Age >= 21 and Age <= 100) Weight:
Histogram With Color Gradient Legend of
Counts, Coral (Low) to Blue (High)") +
  xlab("Weight (Lbs)") +
  ylab("Count") +
  scale_x_continuous(labels=scales::comma, limits=c(0,350),
    breaks=scales::pretty_breaks(n = 10)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,800),
    breaks=scales::pretty_breaks(n = 10)) +
  theme_MacYates()
# With additions to geom_histogram(), this histogram shows a
# color gradient legend, giving a greater sense of counts
# (Y axis) for each Weight (X axis) value.
#
# Differentiation in the color gradient is achieved in this
# histogram by using scale_fill_gradient(), with the lowest
# counts showing as coral and the highest counts showing in
# blue. Equally, look at the way a title (e.g., Count) was
# easily added to the color gradient legend.

```

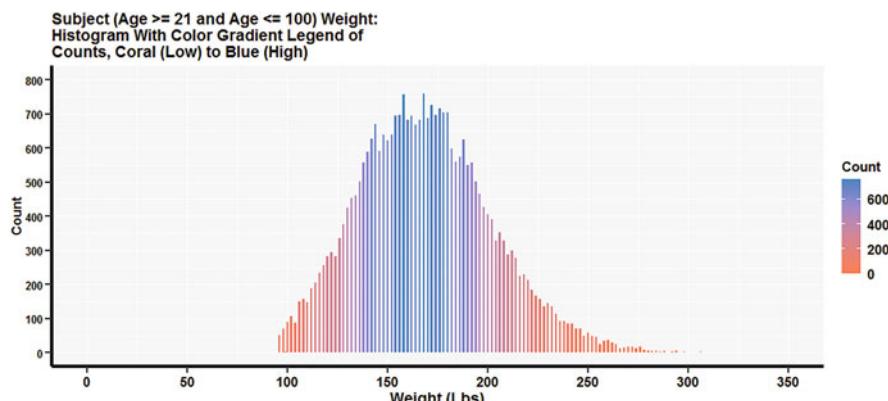


Figure 8.164: ggplot2 package—histogram 4

R Input

```

par(ask=TRUE)
ggplot2::ggplot(data = HospitalAge021100.df,
  aes(x=Lbs)) +
  geom_histogram(binwidth=1, aes(fill=..count..)) +

```

```

facet_grid(Sex ~ Ethnicity) +
scale_fill_gradient("Count", low="coral",
high="dodgerblue") +
ggttitle("Subject (Age >= 21 and Age <= 100) Weight by Sex and
by Ethnicity: Histogram With Color Gradient Legend
of Counts, Coral (Low) to Blue (High)") +
xlab("Weight (Lbs)") +
ylab("Count") +
scale_x_continuous(labels=scales::comma, limits=c(0,350),
breaks=scales::pretty_breaks(n = 10)) +
scale_y_continuous(labels=scales::comma, limits=c(0,351),
breaks=scales::pretty_breaks(n = 10)) +
theme(strip.text.x=element_text(face="bold", size=12,
color="navyblue", angle=0)) +
theme(strip.text.y=element_text(face="bold", size=12,
color="navyblue", angle=45)) +
theme(strip.background=element_rect(fill="wheat1")) +
theme_MacYates()

# Note how facet_grid() was used to add even more information
# about data distribution for Weight (Lbs), now with breakouts
# histograms of Weight (Lbs) differentiated Sex and by
# Ethnicity.

#
# For this figure, it was necessary to accommodate Y axis
# breakouts using theme(strip.text.y=element_text(), to go
# along with theme(strip.text.x=element_text()).

```

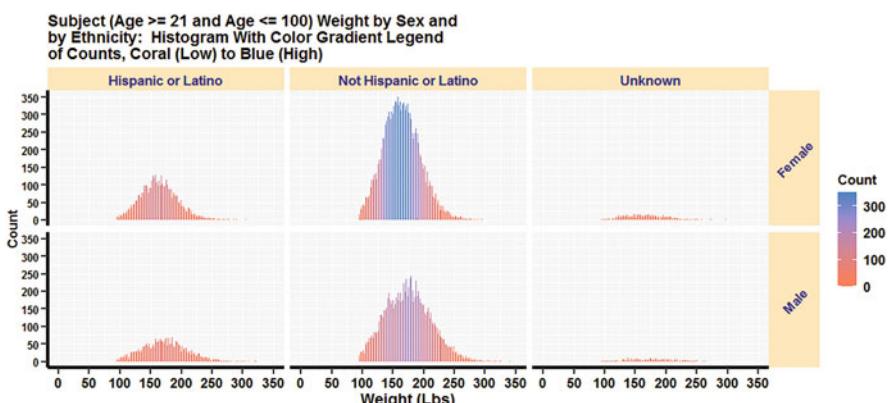


Figure 8.165: ggplot2 package—histogram 5

Use `base::length()` and `base::table()` to supplement the R-based syntax that follows, showing frequency-type information about totals and breakouts. The six histograms obtained by using `facet_grid()` along with `geom_histogram()`, as

well as the color gradient legend, add the potential for far more understanding about data distribution for Weight (Lbs) than could ever be achieved by producing sterile statistics.

R Input

```
base::length(HospitalAge021100.df$Sex)
```

R Output

```
[1] 30227
```

R Input

```
base::length(HospitalAge021100.df$Ethnicity)
```

R Output

```
[1] 30227
```

R Input

```
base::table(HospitalAge021100.df$Sex,
            HospitalAge021100.df$Ethnicity)
```

R Output

	Hispanic or Latino	Not Hispanic or Latino	Unknown
Female	4121	12829	562
Male	2598	9848	269

Although `geom_histogram()` is used far more often than `geom_freqpoly()`, the use of `geom_freqpoly()` produces an interesting graphic that should be considered. When comparing `geom_freqpoly()` to `geom_histogram()`, consider how a frequency polygon is a graphic that is similar to what is obtained if a line were drawn from the top of each histogram bar to the next histogram bar, until all histogram bars are connected by a continuous line.

A simple graphic, using `geom_freqpoly()`, follows. Experiment with this geom to see how it can possibly complement the more frequent use of histograms. Later, look at the way two frequency polygons are shown in the same graphic, allowing for another visual differentiation for the two breakouts of Sex (Fig. 8.166).

R Input

```
ggplotlbsfreqpoly01 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(x=Lbs)) +
geom_freqpoly(binwidth=10, color="black", size=2) +
ggtitle("Subject (Age >= 21 and Age <= 100) Weight:
Frequency Polygon") +
xlab("Weight (Lbs)") +
ylab("Count") +
scale_x_continuous(labels=scales::comma, limits=c(0,350),
breaks=scales::pretty_breaks(n = 10)) +
scale_y_continuous(labels=scales::comma, limits=c(0,4000),
breaks=scales::pretty_breaks(n = 5)) +
theme_MacYates()
# A scale for Count (Y axis) is avoided, to allow R to set
# the scale.

ggplotlbsfreqpoly02 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(x=Lbs, color=Sex)) +
geom_freqpoly(binwidth=10, size=2) +
ggtitle("Subject (Age >= 21 and Age <= 100) Weight
by Sex: Frequency Polygon") +
xlab("Weight (Lbs)") +
ylab("Count") +
scale_x_continuous(labels=scales::comma, limits=c(0,350),
breaks=scales::pretty_breaks(n = 10)) +
scale_y_continuous(labels=scales::comma, limits=c(0,4000),
breaks=scales::pretty_breaks(n = 5)) +
theme_MacYates()

par(ask=TRUE); gridExtra::grid.arrange(
ggplotlbsfreqpoly01,
ggplotlbsfreqpoly02, ncol=2)
```

Stripchart

R Input

```
ggplotsexlbsjitter01 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(x=Sex, y=Lbs)) +
```

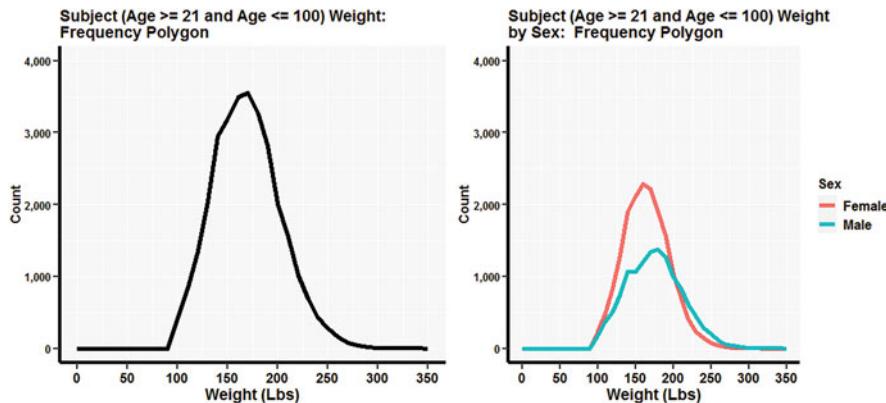


Figure 8.166: ggplot2 package—frequency polygon

```

geom_jitter(aes(color=Sex), position="jitter") +
facet_grid(~ Race.Recode) +
ggtitle("Subject (Age >= 21 and Age <= 100) Weight by Race
and by Sex: Stripchart Using geom_jitter") +
xlab("Sex") +
ylab("Weight (Lbs)") +
scale_y_continuous(labels=scales::comma, limits=c(0,350),
breaks=scales::pretty_breaks(n = 5)) +
theme(strip.text.x=element_text(face="bold", size=10,
color="navyblue", angle=45)) +
theme(strip.background=element_rect(fill="wheat1")) +
theme(legend.position="bottom") +
theme_MacYates() +
# The arguments for the theme strip.text accommodate desired
# size, color, and angle for the facet text.
# The arguments for the theme strip.background accommodate
# desired color for the facet rectangle.
theme(axis.text.x=element_text(face="bold", size=10,
angle=45, hjust=1.25))
# Note the added arguments for theme(axis.text.x), allowing
# for a 45 degree angle and a slight adjustment away from the
# X axis, allowing more space and improved presentation.
#
# IMPORTANT: The arguments for theme(axis.text.x) were
# placed after theme_MacYates() so that the 45 degree angle
# would replace the otherwise horizontal placement of Sex on
# the X axis.

ggplotsexlbspoint01 <-
ggplot2::ggplot(data = HospitalAge021100.df,

```

```

aes(x=Sex, y=Lbs)) +
geom_point(aes(color=Sex), size=0.75, position="jitter") +
facet_grid(~ Race.Recode) +
ggtitle("Subject (Age >= 21 and Age <= 100) Weight by Race  
and by Sex: Stripchart Using geom_point") +
xlab("Sex") +
ylab("Weight (Lbs)") +
scale_y_continuous(labels=scales::comma, limits=c(0,350),
breaks=scales::pretty_breaks(n = 5)) +
theme(strip.text.x=element_text(face="bold", size=10,
color="navyblue", angle=45)) +
theme(strip.background=element_rect(fill="wheat1")) +
theme(legend.position="bottom") +
theme_MacYates() +
theme(axis.text.x=element_text(face="bold", size=10,
angle=45, hjust=1.25))
# Another way to approach the use of jitter in a stripchart
# is to use geom_point instead of geom_jitter. The point
# size was put to 0.75 and this serves as an improvement.

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotsexlbsjitter01,
  ggplotsexlbspoint01, ncol=2)

```

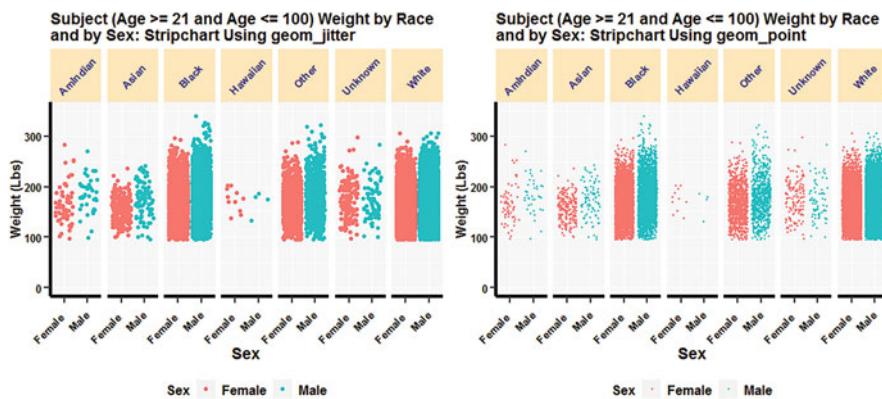


Figure 8.167: ggplot2 package—stripchart

Whatever approach is used for point size, the main advantage of the stripchart is that there is an indication of the number of points (e.g., a proxy for number of subjects) in each breakout group and there is also a sense of frequency and range along the scale (e.g., a proxy for data distribution). Thus, the stripchart should be seen as a complement to the histogram, boxplot, density curve, etc., and certainly has value for personal use as a diagnostic tool (Fig. 8.167).

Quantile-Quantile (Q-Q) Plot

The `ggplot2::ggplot()` function, using selected tools such as `stat_qq()`, provides many ways to achieve the same graphical output. The syntax that follows provides a few ideas on how data distribution and adherence to or deviation away from normal distribution can be presented (Fig. 8.168).

R Input

```
ggplotlbsstat_qq01 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(sample=Lbs)) +
  stat_qq() +           # Notice geom_point is NOT used here
  ggtitle("Theoretical Distribution of Subject (Age >= 21 and
Age <= 100) Weight: Quantile-Quantile (Q-Q) Plot
Using stat_qq()") +
  xlab("Theoretical Distribution") +
  ylab("Weight (Lbs)") +
  scale_x_continuous(labels=scales::comma, limits=c(-4.0,4.0),
  breaks=scales::pretty_breaks(n=10)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,350),
  breaks=scales::pretty_breaks(n=10)) +
  theme_MacYates() +
  theme(plot.title = element_text(size=08)) +
  theme(axis.text.x=element_text(face="bold", size=10,
  angle=45, hjust=1.25))
# Use additional themes after theme_MacYates() to alter
# title size and the direction of X axis labels.

ggplotlbsgeompointstatqq01 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(sample=Lbs)) +
  geom_point(stat="qq") +      # Notice geom_point is used here
  ggtitle("Theoretical Distribution of Subject (Age >= 21 and
Age <= 100) Weight: Quantile-Quantile (Q-Q) Plot
Using geom_point(stat=\"qq\")") +
  # The \ character was placed immediately before the quote
  # marks in the title -- \"qq\".
  # character
  xlab("Theoretical Distribution") +
  ylab("Weight (Lbs)") +
  scale_x_continuous(labels=scales::comma, limits=c(-4.0,4.0),
  breaks=scales::pretty_breaks(n=10)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,350),
```

```

breaks=scales::pretty_breaks(n=10)) +
theme_MacYates() +
theme(plot.title = element_text(size=08)) +
theme(axis.text.x=element_text(face="bold", size=10,
angle=45, hjust=1.25))

ggplotlbstatqqsamplelbs01 <-
ggplot2::ggplot(data = HospitalAge021100.df) +
  stat_qq(aes(sample=Lbs)) +      # Notice the difference here
  ggttitle("Theoretical Distribution of Subject (Age >= 21 and
Age <= 100) Weight: Quantile-Quantile (Q-Q) Plot
Using stat_qq(aes(sample=Lbs))) +
  xlab("Theoretical Distribution") +
  ylab("Weight (Lbs)") +
  scale_x_continuous(labels=scales::comma, limits=c(-4.0,4.0),
  breaks=scales::pretty_breaks(n=10)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,350),
  breaks=scales::pretty_breaks(n=10)) +
  theme_MacYates() +
  theme(plot.title = element_text(size=08)) +
  theme(axis.text.x=element_text(face="bold", size=10,
angle=45, hjust=1.25))

par(ask=TRUE); gridExtra::grid.arrange(
  ggplotlbsstat_qq01,
  ggplotlbsgeompointstatqq01,
  ggplotlbstatqqsamplelbs01, ncol=3)

```

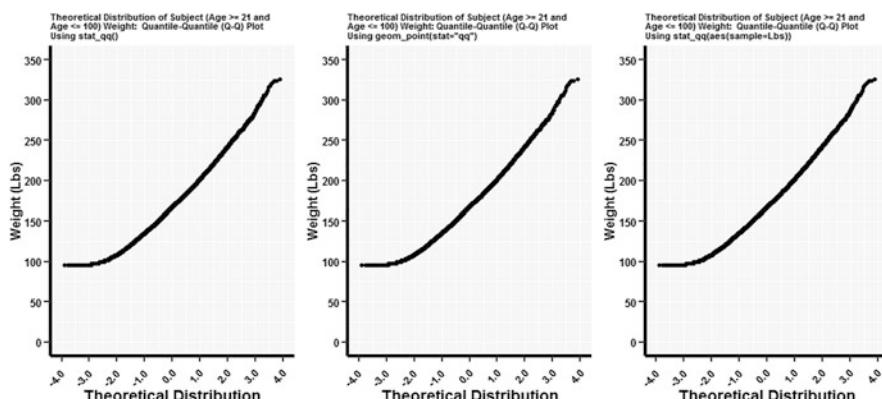


Figure 8.168: ggplot2 package—QQ plot 1

R Input

```

ggplotlbssexstat_qq01 <-
  ggplot2::ggplot(data = HospitalAge021100.df) +
    stat_qq(aes(sample=Lbs, color=(Sex))) + # More variation here
    ggttitle(
      "Theoretical Distribution of Subject (Age >= 21 and Age <= 100)
      Weight by Sex: Quantile-Quantile (Q-Q) Plot with Both
      Genders in One Plot\n\n") +
      xlab("Theoretical Distribution") +
      ylab("Weight (Lbs)") +
      scale_x_continuous(labels=scales::comma, limits=c(-4.0,4.0),
        breaks=scales::pretty_breaks(n=10)) +
      scale_y_continuous(labels=scales::comma, limits=c(0,350),
        breaks=scales::pretty_breaks(n=10)) +
      theme(legend.position="bottom") +
      theme_MacYates() +
      theme(plot.title = element_text(size=10)) +
      theme(axis.text.x=element_text(face="bold", size=10,
        angle=45, hjust=1.25))
    # Use additional themes after theme_MacYates() to alter
    # title size and the direction of X axis labels.

ggplotlbsfacetsexqq01 <-
  ggplot2::ggplot(data = HospitalAge021100.df) +
    stat_qq(aes(sample=Lbs)) +
    facet_grid(~ Sex) +           # Breakouts by using facet_grid
    ggttitle(
      "Theoretical Distribution of Subject (Age >= 21 and Age <= 100)
      Weight by Sex: Quantile-Quantile (Q-Q) Plot with Genders
      in facet_grid Breakouts") +
      xlab("Theoretical Distribution") +
      ylab("Weight (Lbs)") +
      scale_x_continuous(labels=scales::comma, limits=c(-4.0,4.0),
        breaks=scales::pretty_breaks(n=10)) +
      scale_y_continuous(labels=scales::comma, limits=c(0,350),
        breaks=scales::pretty_breaks(n=10)) +
      theme(strip.text.x=element_text(face="bold", size=12,
        color="navyblue", angle=0)) +
      theme(strip.background=element_rect(fill="wheat1")) +
      theme_MacYates() +
      theme(plot.title = element_text(size=10)) +
      theme(axis.text.x=element_text(face="bold", size=10,

```

```
angle=45, hjust=1.25))

par.ask=TRUE); gridExtra::grid.arrange(
  ggplotlbssexstat_qq01,
  ggplotlbsfacetsexqq01, ncol=2)
```

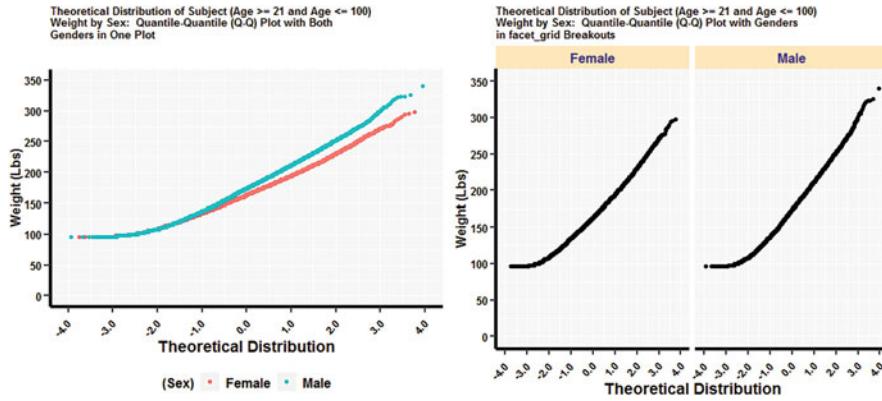


Figure 8.169: ggplot2 package—QQ plot 2

It is also possible to use ggplot2 syntax to prepare figures that combine separate facet_grid plots and breakouts within these breakout plots. As an additional feature, notice how the title for the following ggplot2 figure is centered, as opposed to the default setting where ggplot2 titles are left justified (Figs. 8.169 and 8.170).

R Input

```
par.ask=TRUE)
ggplot2::ggplot(data = HospitalAge021100.df) +
  stat_qq(aes(sample=Lbs, color=(Sex))) +
  facet_grid( ~ Race.Recode) +
  ggttitle(
  "Theoretical Distribution of Subject (Age >= 21 and Age <= 100)
  Weight by Sex and by Race: Quantile-Quantile (Q-Q) Plot with
  Sex and Race in facet_grid Breakouts") +
  xlab("Theoretical Distribution") +
  ylab("Weight (Lbs)") +
  scale_x_continuous(labels=scales::comma, limits=c(-4.0,4.0),
    breaks=scales::pretty_breaks(n=10)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,350),
    breaks=scales::pretty_breaks(n=10)) +
  theme(strip.text.x=element_text(face="bold", size=12,
```

```

    color="navyblue", angle=45)) +
theme(strip.background=element_rect(fill="wheat1")) +
theme(legend.position="bottom") +
theme_MacYates() +
theme(plot.title = element_text(hjust = 0.5)) +
theme(axis.text.x=element_text(face="bold", size=06,
    angle=45, hjust=1.25))
# Notice the placement of theme(plot.title), after
# theme_MacYates().

```

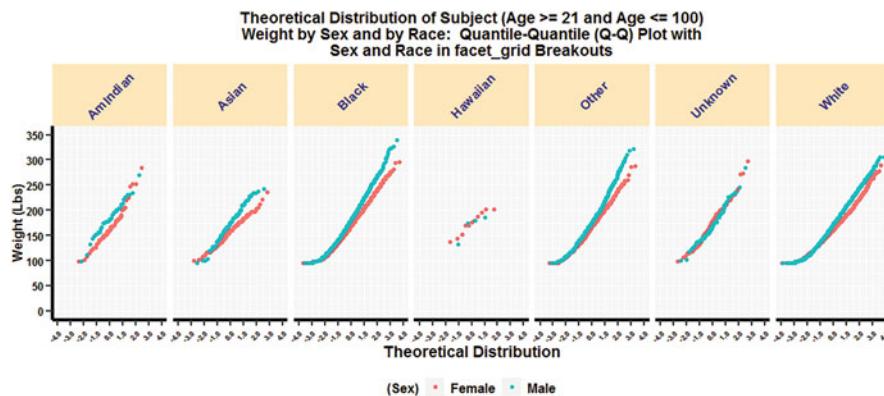


Figure 8.170: ggplot2 package—QQ plot 3

Interaction Plot

There are many ways to display moderator effects using the `ggplot2::ggplot()` function. Use the syntax that follows as an initial guide on how to use the `ggplot2::ggplot()` function to prepare an interaction plot (Fig. 8.171).

R Input

```

ggplottracesbpsexinteraction01 <-
ggplot2::ggplot(data = HospitalAge021100.df,
  aes(x=Race.Recode, y=SBP, group=Sex, color=Sex)) +
  stat_summary(fun=mean, geom="point", size=4.5, shape=18) +
  stat_summary(fun=mean, geom="line", size=1.5) +
  ggtitle("Subject (Age >= 21 and Age <= 100) Mean Systolic
Blood Pressure by Sex and by Race:
Interaction Plot") +
  xlab("Race") +
  ylab("Mean Systolic Blood Pressure (SBP)") +
  scale_y_continuous(labels=scales::comma, limits=c(0,200),
    breaks=scales::pretty_breaks(n=10)) +

```

```

theme_MacYates() +
theme(legend.position="bottom") +
theme(text=element_text(face="bold", size=8, )) +
theme(axis.text.x=element_text(face="bold", size=10,
angle=45, hjust=1.25))
# Give attention to the composition and placement of the
# three themes placed below theme_MacYates(). This type of
# adjustment, such as font size, angle of presentation, etc.,
# is used to make specific improvements to a ggplot-based
# figure and demonstrates the flexibility of the ggplot()
# function.

ggplottracesbpethnicityinteraction01 <-
ggplot2::ggplot(data = HospitalAge021100.df,
aes(x=Race.Recode, y=SBP, group=Ethnicity,
color=Ethnicity)) +
stat_summary(fun=mean, geom="point", size=4.5, shape=18) +
stat_summary(fun=mean, geom="line", size=1.5) +
ggtitle("Subject (Age >= 21 and Age <= 100) Mean Systolic
Blood Pressure by Ethnicity and by Race:
Interaction Plot") +
xlab("Race") +
ylab("Mean Systolic Blood Pressure (SBP)") +
scale_y_continuous(labels=scales::comma, limits=c(0,200),
breaks=scales::pretty_breaks(n=10)) +
theme_MacYates() +
theme(legend.position="bottom") +
theme(text=element_text(face="bold", size=8)) +
theme(axis.text.x=element_text(face="bold", size=10,
angle=45, hjust=1.25))

par(ask=TRUE); gridExtra::grid.arrange(
ggplottracesbpsexinteraction01,
ggplottracesbpethnicityinteraction01, ncol=2)

```

Line Chart

The line chart is often used with data that focus on change over time, specifically using some measure of dates for time. The dataset associated with this lesson does not have a time-specific variable, so to show a line chart consider the dataset that follows, `SpringCounts.df`.

For context, the data represent forest floor counts of an invasive foreign insect that was inadvertently introduced into a deciduous forest more than 50 years

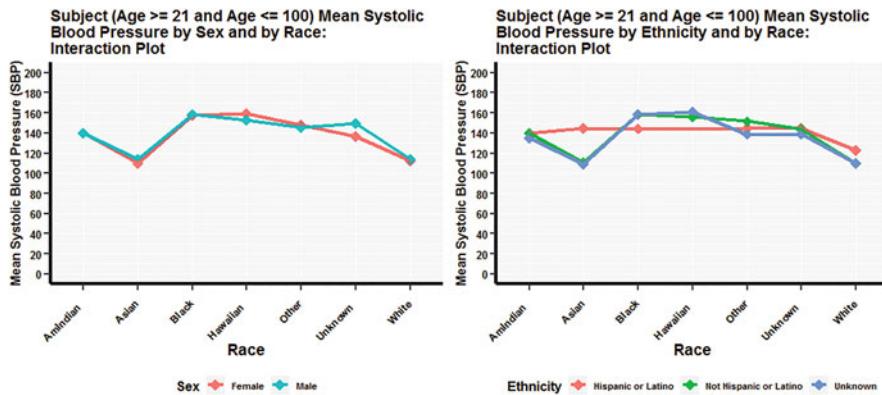


Figure 8.171: ggplot2 package—interaction plot

ago and an estimate of the number of observations (e.g., counts) each year on or about May 15 (the approximate time period when larva are fully grown and easy to spot and count while on the forest floor) in the same one-acre plot. This simple dataset provides no sense of human interaction (e.g., pesticides) or natural conditions (e.g., fire, pathogens, predators, weather, etc.) and their impact on insect counts, either increase or decrease (Fig. 8.172).

R Input

```
SpringInsectCounts.df <- read.table(textConnection("
Year      Observations
1966      19
1967      22
1968      31
1969      49
1970      86
1971      298
1972      971
1973      1983
1974      2879
1975      3039
1976      6863
1977      7338
1978      6923
1979      5204
1980      4905
1981      5543
1982      5756
1983      6096
1984      6497"))
```

```
1985    6930
1986    7873
1987    8776
1988    9905
1989    10331
1990    10789
1991    11527
1992    11856
1993    12339
1994    14068
1995    14602
1996    15092
1997    16801
1998    16821
1999    17908
2000    18567
2001    19067
2002    21339
2003    23422
2004    25550
2005    26025
2006    25931
2007    27338
2008    28418
2009    29872
2010    29041
2011    28377
2012    26651
2013    25359
2014    24044
2015    23091
2016    22009
2017    20017
2018    19556
2019    19217
2020    20038"), header=TRUE)

attach(SpringInsectCounts.df)      # Attach
str(SpringInsectCounts.df)        # Structure
summary(SpringInsectCounts.df )   # Summary
```

R Output

Year	Observations
Min. :1966	Min. : 19
1st Qu.:1980	1st Qu.: 5926
Median :1993	Median :12339
Mean :1993	Mean :13692
3rd Qu.:2006	3rd Qu.:21674
Max. :2020	Max. :29872

R Input

```

par(ask=TRUE)
ggplot2::ggplot(data = SpringInsectCounts.df,
  aes(x=Year, y=Observations)) +
  geom_point(shape=16, color="blue", size=4) +
  geom_line(size=1.75, color="red") +
  ggtitle("Counts of a Specific Foreign Insect in a Deciduous
Forest at a Selected One-Acre Plot, From 1966
to 2020: Line Chart") +
  xlab("\nYear") +
  ylab("Number of Observations (e.g., Counts)\n") +
  scale_x_continuous(limits=c(1966, 2020),
    breaks=scales::pretty_breaks(n=28)) +
  scale_y_continuous(labels=scales::comma, limits=c(0,30000),
    breaks=scales::pretty_breaks(n=10)) +
  theme_MacYates() +
  theme(axis.text.x=element_text(face="bold", size=10,
    angle=45, hjust=1.25))
# Note the added arguments for theme(axis.text.x), allowing
# for a 45 degree angle and a slight adjustment away from
# the X axis, allowing more space and improved presentation.
#
# IMPORTANT: The arguments for theme(axis.text.x) were
# placed after theme_MacYates() so that the 45 degree angle
# would replace the otherwise horizontal placement of Year on
# the X axis.

```

Tile Map

The tile map, when using the `ggplot2::ggplot()` function, uses the center of the tile and its size (`x`, `y`, `width`, `height`) to produce a rectangle that provides a sense of a continuous numeric object variable (e.g., either SBP or DBP) in relation to the breakout groups of factor-type object variables. There are some similarities between use of `geom_tile` and `geom_rect` (Fig. 8.173).

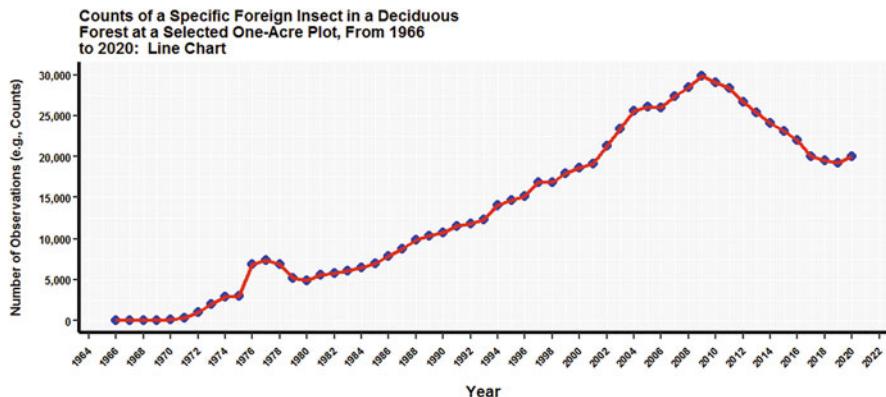


Figure 8.172: ggplot2 package—line chart

R Input

```
ggplotsbpracetilemap01 <-
  ggplot2::ggplot(HospitalAge021100.df,
    aes(x=SBP, y=Race.Recode)) +
    geom_tile(aes(fill=Sex)) +
    ggtitle(
      "Subject (Age >= 21 and Age <= 100)
      SBP by Sex and by Race:
      Tile Map") +
    labs(x="\nSystolic Blood Pressure", y="Race\n") +
    theme_MacYates() +
    theme(plot.title=element_text(size=10, face="bold")) +
    theme(axis.text.x=element_text(size=10, face="bold"),
          axis.text.y=element_text(size=10, face="bold"),
          axis.title.x=element_text(size=12, face="bold"),
          axis.title.y=element_text(size=12, face="bold")) +
    theme(legend.position="bottom") +
    theme(legend.title=element_text(size=06), legend.text=
      element_text(size=06)) +
    theme(legend.key.size=unit(0.15, "cm"), legend.key.width=
      unit(0.15, "cm"))
  # Give attention to adjustments for themes that come after
  # theme_MacYates() and how these adjustments were needed to
  # have three separate figures fit into this one unified
  # figure. If needed, of course, the three figures could be
  # presented individually. Yet, side-by-side comparisons have
  # value and should be considered.
```

```

ggplotsbpracesextilemap01 <-
  ggplot2::ggplot(HospitalAge021100.df,
    aes(x=SBP, y=Race.Recode)) +
    geom_tile(aes(fill=Ethnicity)) +
    ggtitle(
      "Subject (Age >= 21 and Age <= 100)
      SBP by Ethnicity and by Race:
      Tile Map") +
    labs(x="\nSystolic Blood Pressure", y="Race\n") +
    theme_MacYates() +
    theme(plot.title=element_text(size=10, face="bold")) +
    theme(axis.text.x=element_text(size=10, face="bold"),
          axis.text.y=element_text(size=10, face="bold"),
          axis.title.x=element_text(size=12, face="bold"),
          axis.title.y=element_text(size=12, face="bold")) +
    theme(legend.position="bottom") +
    theme(legend.title=element_text(size=06), legend.text=
      element_text(size=06)) +
    theme(legend.key.size=unit(0.15, "cm"), legend.key.width=
      unit(0.15, "cm"))
  # There are subjects who have an unknown race and an unknown
  # ethnicity. Race and ethnicity are two separate constructs,
  # as defined in the Code Book.

ggplotsbpraceprimarypayertilemap01 <-
  ggplot2::ggplot(HospitalAge021100.df,
    aes(x=SBP, y=Race.Recode)) +
    geom_tile(aes(fill=PrimaryPayer)) +
    ggtitle(
      "Subject (Age >= 21 and Age <= 100)
      SBP by Primary Payer and by Race:
      Tile Map") +
    labs(x="\nSystolic Blood Pressure", y="Race\n") +
    theme_MacYates() +
    theme(plot.title=element_text(size=10, face="bold")) +
    theme(axis.text.x=element_text(size=10, face="bold"),
          axis.text.y=element_text(size=10, face="bold"),
          axis.title.x=element_text(size=12, face="bold"),
          axis.title.y=element_text(size=12, face="bold")) +
    theme(legend.position="bottom") +
    theme(legend.title=element_text(size=06), legend.text=
      element_text(size=06)) +
    theme(legend.key.size=unit(0.15, "cm"), legend.key.width=

```

```
unit(0.15, "cm"))

par(aspect=TRUE); gridExtra::grid.arrange(
  ggplotsbpracetilemap01,
  ggplotsbpracesextilemap01,
  ggplotsbpraceprimarypayertilemap01, ncol=3)
```

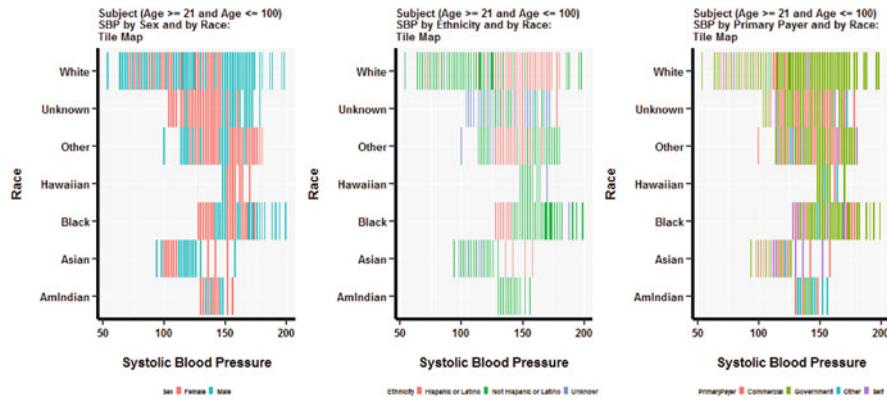


Figure 8.173: ggplot2 package—tile map

This addendum on the many possible *Beautiful Graphics* that can be prepared through use of the `ggplot2::ggplot()` function is only a beginning on what can be produced, either for diagnostic purposes or for professional publication. Review online resources, review the professional literature to see how figures are approached in different areas of biostatistics, and then experiment with the many features available with the `ggplot2::ggplot()` function.

8.12 Addendum 4: Beyond an Introduction to R—Use the tidyverse to Create Subsets of Original Datasets

It was stated in the main part of this lesson that Boolean-type operations were purposely demonstrated in an attempt to emphasize the heuristics (e.g., the actual process) of how data are selected from a large dataset, in an attempt to subset data and create a new dataset. Review the previously demonstrated Boolean-specific syntax and note how it is somewhat verbose, but clearly identifies the many selections for Age, Race, and Sex that had to be accommodated to generate the desired graphic:

```
graphics::hist(subset(HospitalAge021100.df$SBP,      # Boolean
  (HospitalAge021100.df$AgeLastBirthday >= 40 &      # selections
  HospitalAge021100.df$AgeLastBirthday <= 49) &      # of Age,
  (HospitalAge021100.df$Race.Recode == "Black") &      # Race, and
  (HospitalAge021100.df$Sex.Recode == "Female")),      # Sex
```

```

main=
"Systolic Blood Pressure (SBP) of Hospital Patients Who
Are Black, Female, and 40 to 49 Years Old",
xlab="Systolic Blood Pressure", ylab="Number of Patients",
breaks=25, col="red", font=2, xlim=c(100,200), ylim=c(0,125))
# By using the Boolean selections, the output will represent
# the SBP of only of those subjects who (1) range in age from
# 40 to 49 on their last birthday, inclusive, (2) are Black,
# and (3) are Female.
#
# Typical to any Boolean-type comparison for equality when
# using R, look at the way == and not = is used for the
# comparison of equivalence.

```

For those who wish to go beyond introductory-level R syntax, consider how the tidyverse is used to produce syntax that is considered far more concise, or *tidy*. To achieve this aim, two packages from the tidyverse are briefly demonstrated in this addendum: dplyr and magrittr. The syntax presented in this addendum is just a mere preview of the power of the tidyverse, but more details on the tidyverse would go far beyond the purpose of this text.

8.12.1 Use the dplyr::filter() Function to Subset a Large Dataset

The dplyr::filter() function is often the first choice when data scientists need to create a new dataset that represents a subset of a larger dataset. In the syntax that follows, the length() function and the summary() function have purposely been placed around each selection process to show the progression of the four-tiered (e.g., Location, PrimaryPayer, AgeLastBirthday, Race) selection process, from a dataset of 33,859 subjects to a dataset of 1287 subjects. Notice also the dataframe naming process and how self-documentation was used to clearly identify the progression of selection (Figs. 8.174 and 8.175):

- Hospital.df, N = 33,859
- HospitalCentral.df, N = 21,120
- HospitalCentralCommercial.df, N = 3783
- HospitalCentralCommercialAge040069.df, N = 1708
- HospitalCentralCommercialAge040069WhiteAsian.df, N = 1287

R Input

```
install.packages("dplyr", dependencies=TRUE)
library(dplyr)                      # Load the dplyr package.
help(package=dplyr)                  # Show the information page.
sessionInfo()                       # Confirm all attached packages.

length(Hospital.df$Ticket)
```

R Output

```
[1] 33859
```

R Input

```
summary(Hospital.df$Location)
```

R Output

Central	North	South
21120	7542	5197

R Input

```
HospitalCentral.df <-
  dplyr::filter(Hospital.df,
  Location == "Central")
# 1st filter -- Location

length(HospitalCentral.df$Ticket)
```

R Output

```
[1] 21120
```

R Input

```
summary(HospitalCentral.df$Location)
```

R Output

Central	North	South
21120	0	0

R Input

```
HospitalCentralCommercial.df <-
  dplyr::filter(HospitalCentral.df,
  PrimaryPayer == "Commercial")
  # 2nd filter -- PrimaryPayer

length(HospitalCentralCommercial.df$Ticket)
```

R Output

[1] 3783

R Input

```
summary(HospitalCentralCommercial.df$Location)
```

R Output

Central	North	South
3783	0	0

R Input

```
summary(HospitalCentralCommercial.df$PrimaryPayer)
```

R Output

Commercial	Government	Other	Self
3783	0	0	0

R Input

```
HospitalCentralCommercialAge040069.df <-
  dplyr::filter(HospitalCentralCommercial.df,
  AgeLastBirthday >= 40 & AgeLastBirthday <= 69)
```

```
# 3rd filter -- AgeLastBirthday
length(HospitalCentralCommercialAge040069.df$Ticket)
```

R Output

```
[1] 1708
```

R Input

```
summary(HospitalCentralCommercialAge040069.df$Location)
```

R Output

Central	North	South
1708	0	0

R Input

```
summary(HospitalCentralCommercialAge040069.df$PrimaryPayer)
```

R Output

Commercial	Government	Other	Self
1708	0	0	0

R Input

```
summary(HospitalCentralCommercialAge040069.df$AgeLastBirthday)
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
40.0	47.0	54.0	53.2	59.0	69.0

R Input

```
HospitalCentralCommercialAge040069WhiteAsian.df <-
  dplyr::filter(HospitalCentralCommercialAge040069.df,
  Race == "White" | Race == "Asian")
```

```
# 4th filter -- Race
# The pipe character is used to represent OR.

length(HospitalCentralCommercialAge040069WhiteAsian.df$Ticket)
```

R Output

```
[1] 1287
```

R Input

```
summary(
  HospitalCentralCommercialAge040069WhiteAsian.df$Location)
```

R Output

Central	North	South
1287	0	0

R Input

```
summary(
  HospitalCentralCommercialAge040069WhiteAsian.df$PrimaryPayer)
```

R Output

Commercial	Government	Other	Self
1287	0	0	0

R Input

```
summary(
  HospitalCentralCommercialAge040069WhiteAsian.df$AgeLastBirthday)
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
40.0	47.0	54.0	53.4	59.0	69.0

R Input

```
summary(
  HospitalCentralCommercialAge040069WhiteAsian.df$Race)
```

R Output

[Output was organized to save space.]	
American Indian or Alaska Native	0
Asian	25
Black or African American	0
Native Hawaiian or Other Pacific Islander	0
Other	0
Unknown	0
White	1262

R Input

```
par(ask=TRUE)
par(mfrow=c(2,3)) # 6 cells into a 2 row by 3 column grid
hist(Hospital.df$SBP,
  main="SBP", col="red", xlim=c(0,200))
hist(HospitalCentral.df$SBP,
  main="SBP", col="red", xlim=c(0,200))
hist(HospitalCentralCommercial.df$SBP,
  main="SBP", col="red", xlim=c(0,200))
hist(HospitalCentralCommercialAge040069.df$SBP,
  main="SBP", col="red", xlim=c(0,200))
hist(HospitalCentralCommercialAge040069WhiteAsian.df$SBP,
  main="SBP", col="red", xlim=c(0,200))
# Demonstrate the progression from Hospital.df to
# HospitalCentralCommercialAge040069WhiteAsian.df.
# To avoid unnatural line breaks, the hist()
# function was used instead of graphics::hist().
#
# Be sure to observe the SBP histogram for
# HospitalCentralCommercialAge040069WhiteAsian.df
# given the very restrictive selections that went
# into the creation of this final dataset.
```

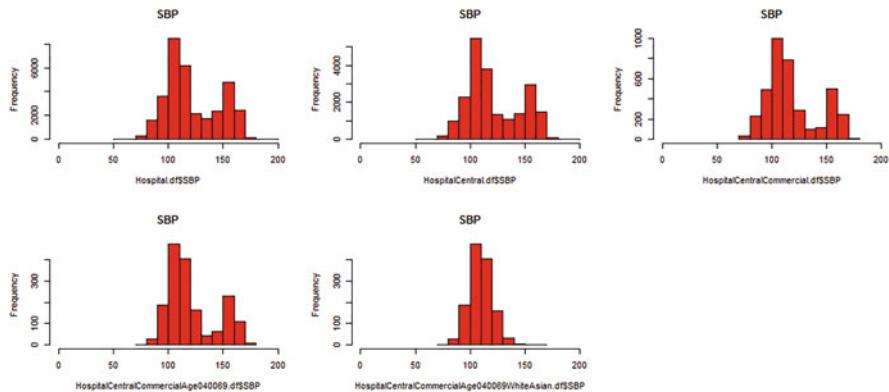


Figure 8.174: Multiple subsets of systolic blood pressure

8.12.2 Use the magrittr Package Pipe-Like Operator

The `%>%` pipe-like operator, which is associated with the `magrittr` package, provides substantial improvements in the development of well-written R syntax. By using `%>%`, the values on the left side of an expression are piped forward to the right side of the expression. Although the `%>%` operator may not be necessary to achieve desired results, for anyone who needs to use advanced R skills the `%>%` operator will be a frequently used tool.

R Input

```
install.packages("magrittr", dependencies=TRUE)
library(magrittr)          # Load the magrittr package.
help(package=magrittr)      # Show the information page.
sessionInfo()               # Confirm all attached packages.

HospitalNorthGovernmentAge030045Female.df <-
  Hospital.df %>%
  subset(
    (Hospital.df$Location == "North") &
    (Hospital.df$PrimaryPayer == "Government") &
    (AgeLastBirthday >= 30 & AgeLastBirthday <= 45) &
    (Sex == "Female")
  )
  # Create HospitalNorthGovernmentAge030045Female.df, which
  # is a purposeful subset of Hospital.df. Note how the %>%
  # operator is used to direct subset() output.

length(Hospital.df$Ticket)
```

R Output

```
[1] 33859
```

R Input

```
length(HospitalNorthGovernmentAge030045Female.df$Ticket)
```

R Output

```
[1] 326
```

R Input

```
summary(Hospital.df$Location)
```

R Output

Central	North	South
21120	7542	5197

R Input

```
summary(HospitalNorthGovernmentAge030045Female.df$Location)
```

R Output

Central	North	South
0	326	0

R Input

```
summary(Hospital.df$PrimaryPayer)
```

R Output

Commercial	Government	Other	Self
7465	21324	1148	3922

R Input

```
summary(HospitalNorthGovernmentAge030045Female.df$PrimaryPayer)
```

R Output

Commercial	Government	Other	Self
0	326	0	0

R Input

```
summary(Hospital.df$AgeLastBirthday)
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0.0	33.0	54.0	51.8	71.0	111.0

R Input

```
summary(
HospitalNorthGovernmentAge030045Female.df$AgeLastBirthday)
```

R Output

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
30.0	33.0	37.0	37.1	41.0	45.0

R Input

```
summary(Hospital.df$Sex)
```

R Output

Female	Male
19421	14438

R Input

```
summary(HospitalNorthGovernmentAge030045Female.df$Sex)
```

R Output

Female	Male
326	0

R Input

```

par(ask=TRUE)
par(mfrow=c(2,4)) # 8 cells into a 2 row by 3 column grid
plot(Hospital.df$Location, main="Location", col=rainbow(3))
plot(HospitalNorthGovernmentAge030045Female.df$Location,
     main="Location", col=rainbow(3))
plot(Hospital.df$PrimaryPayer, main="PrimaryPayer",
     col=rainbow(4), las=2)
plot(HospitalNorthGovernmentAge030045Female.df$PrimaryPayer,
     main="Primary Payer", col=rainbow(4), las=2)
hist(Hospital.df$AgeLastBirthday, main="Age", col="red")
hist(HospitalNorthGovernmentAge030045Female.df$AgeLastBirthday,
     main="Age", col="red")
plot(Hospital.df$Sex, main="Sex", col=rainbow(2))
plot(HospitalNorthGovernmentAge030045Female.df$Sex, main="Sex",
     col=rainbow(2))
# Compare and contrast the object variables used to subset
# Hospital.df into HospitalNorthGovernmentAge030045Female.df.

```

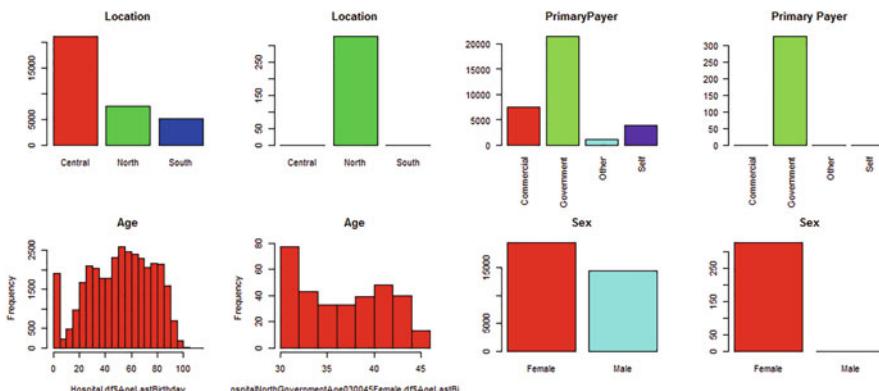


Figure 8.175: Comparison of selected variables—original dataset and dataset after multiple subsets

8.13 Prepare to Exit, Save, and Later Retrieve This R Session

It is common to prepare R syntax in a separate file, using a simple ASCII text editor. If time permits, experiment with Crimson Editor, Tinn-R, or vim, but there are many other possible selections.

R Input

```
getwd()           # Identify the current working directory.  
ls()              # List all objects in the working  
                  # directory.  
ls.str()          # List all objects, with finite detail.  
list.files()      # List files at the PC directory.  
  
save.image("R_Lesson_LargeComplexDatasets.rdata")  
  
getwd()           # Identify the current working directory.  
ls()              # List all objects in the working  
                  # directory.  
ls.str()          # List all objects, with finite detail.  
list.files()      # List files at the PC directory.  
  
alarm()           # Alarm, notice of upcoming action.  
q()               # Quit this session.  
                  # Prepare for Save workspace image? query.
```

Use the R Graphical User Interface (GUI) to load the saved rdata file: **File** and then **Load Workspace**. Otherwise, use the `load()` function, keying the full pathname, to load the .rdata file and retrieve the session.

Recall, however, that it may be just as useful to simply use a R script file (typically saved as a .txt ASCII-type file) and recreate the analyses and graphics, provided the data files remain available.

8.14 External Data and/or Data Resources Used in This Lesson

The publisher's Web site associated with this text includes the file `Hospital.csv` which was the only external file directly imported into this lesson. Use this file to practice and replicate the outcomes used in this lesson.

All other data in this lesson were enumerated (e.g., user-created) directly in the R session, using a variety of R-based functions. The user-created data in this lesson are **not** included at the publisher's Web site associated with this text and they are instead easily created in R by following the syntax presented in this lesson.



Chapter 9

Future Actions and Next Steps

Abstract

The purpose of this lesson is to provide a brief summary of this text and to introduce a few topics that should be considered once initial skills in the use of R for biostatistics have been mastered. A brief introduction to R Markdown is provided, in an attempt to demonstrate how R syntax, R output, and report generation can be integrated into one common set of actions. There is also an initial discussion of how R fits into the Big Data paradigm and from that aim how R, as well as other statistical analysis software, can be used to meet challenges of the constantly evolving field of BioInformatics. A few ideas on external resources that may be useful for learning more about R are also identified. Finally, contact information for the authors is also provided.

Keywords: Big Data, BioInformatics, biostatistics, Code Book, comma-separated values (.csv) file, Command Line Interface (CLI), Comprehensive R Archive Network (CRAN), data analysis, Graphical User Interface (GUI), open source software, R, R Commander, R Markdown, R Studio.

9.1 Use of This Text

Scientists use empiricism to guide and validate decisions. Precision, orderliness, reproducibility, analysis, and a sound background in statistics are directly

associated with informed judgment, transparent decision-making, and the subsequent allocation of human, physical, and fiscal resources – all to improve the human condition. The purpose of this text is to provide an introduction to the use of R software as a platform for analyses related to biostatistics. Data identification, data organization, use of a Code Book, graphical presentation of phenomena, assessment of data distribution patterns, and statistical tests through the use of R are all inherent to this text.

This text serves as an introduction to the use of R in biostatistics. This text has specifically been structured to demonstrate the use of R syntax at the Command Line Interface (CLI) as opposed to the use of a point-and-select Graphical User Interface (GUI). This approach, using syntax at the Command Line Interface, gives the most flexibility to the user in the attempt to reach desired outcomes. With even only minimal (if any) experience with programming, the examples in this text empower the user to organize data, prepare graphical images as desired, assess descriptive statistics and measures of central tendency, and complete a wide range of statistical analyses typically found in biostatistics.

Again, this text covered descriptive statistics and measures of central tendency (e.g., mode, median, mean, N, NA, SD, quartiles, range, etc.) and then provided an introduction to some of the leading inferential tests in biostatistics: Student's t-Test, ANOVA, Correlation and Regression. Along with the standard statistical tests (both those tests that assume an appropriate use of parametric analyses as well as confirming nonparametric tests), a fair degree of emphasis was placed on the preparation of supporting graphics. Statistical tests are needed and should never be avoided, but when making public presentations graphical images are the media that generate interest and further attention to final outcomes.

9.2 R and *Beautiful Reporting* with R Markdown

R can be used as stand-alone software. With a stand-alone approach to the use of R, data are somehow or another brought into the R session, analyses are generated, figures are produced, etc. At the end of the session both the syntax and the many outcomes are saved. If there were a desire to put selected outcomes into a memorandum, lab report, journal article, thesis or dissertation, book, etc., it would be common to use a copy and paste approach, copying R output (e.g., tables, figures, etc.) from the saved session file(s) and pasting this output into the final word-processed file(s).

Although a (1) copy from saved R session file(s) to (2) paste into word processed file(s) approach toward report generation is acceptable, there is a growing movement to more fully integrate R activities and report generation into one common operation. The advantage of this approach is transparency and reproducibility.

R Markdown has been developed to meet this aim of transparency and reproducibility, with R Markdown ostensibly serving as a tool used to facilitate the seamless integration of operations, where one file contains R syntax, R output (statistics, tables, and figures), and text associated with the final report. R Markdown can be used to make static reports and it can also be used to make dynamic reports.

9.2.1 Static Reports (e.g., Documents)

Whether printed to paper or viewed in electronic format, static reports are the most common way scientific outcomes are put into final form and shared with others. Static reports can be as simple as a one page summary memorandum or they can grow in length to a 1,000 page book. Regardless of the length, a static report is in fixed-format when viewed by the reader.

R Commander and R Studio are graphically-enabled R-specific software, sometimes referred to as middleware, that are often used to accommodate R Markdown. It is common to bring additional software into a session when using these programs, including utilities such as knitr and Pandoc, with L^AT_EXan additional possibility. Various file formats such as the common use of .html are possible for report output.

Syntax for a simple *Hello World* R Markdown file follows, with output in .html format. For this example, the syntax was generated using R Studio, where **File -> New File -> R Markdown** were selected and then HTML was selected for default output. Once the syntax was completed it was only necessary to click on the *Knit* icon, located near the top bar of the R Studio interface, to generate the HTML file.¹

```
---
```

```
title: "Hello World Using R Markdown"
author: "Dr. MacFarland and Dr. Yates"
date: "6/25/2020"
output: html_document
---
```

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```

```
# Hello World! This is a simple first try at R Markdown.
```

¹If using R Commander, click on the *Generate report* icon to put the output into desired format.

```
### Purpose of this R Markdown Demonstration
```

- * Produce a simple **Hello World** file, using R markdown.
- * Syntax is put into a .Rmd file.
- * By default, the output will be put into .html format.

```
### Create the object variable SystolicBP
```

```
```{r}
SystolicBP <- rnorm(10000, mean=120, sd=10)
N 10,000
Mean ... 120
SD 10

summary(SystolicBP)
```

```{r}

hist(SystolicBP, col="red", breaks=75,
 main="Systolic Blood Pressure
N = 10,000, Mean = 120, SD = 10")
```

The simple report generated by this syntax should be in .html format. The .html file will include text such as the *Hello World* statement, a simple table showing output from the summary() function, and an attractive histogram of Systolic Blood Pressure.

### 9.2.2 Dynamic Output (e.g., Presentations)

R Markdown can also be used to create Web-based reports that are interactive. That is to say, unlike a traditional static report, an interactive report can be changed to some degree by the reader, typically by changing the parameters associated with the data such that new statistics, new tables, new figures are created *on the fly*. As an example, look at the prior R Markdown syntax where the hist() function used the argument breaks=75. As a static report, the reader has no control over the number of breaks for the generated histogram. With a dynamic (e.g., interactive) approach toward reporting, the reader can interact with the Web page and possibly change breaks=75 to breaks=100 or any other desired value. This degree of flexibility is not possible in a static approach

toward reporting but is possible with R Markdown in dynamic fashion.

The Shiny package is the main supplemental software used for the dynamic use of R Markdown and the integration of data, R syntax, and reporting that interacts with the reader. There are many resources available on how Shiny and the Web are joined into a dynamic reporting experience – resources that go far beyond the purpose of this introductory text.

### 9.3 Future Use of R for Biostatistics

The R user community is international and it is growing. R is free and at first the notion of free software may be the compelling consideration for the use of R. Before too long, however, the power and flexibility of R becomes apparent. For those who regularly engage in research, and subsequently engage with statistical analyses, the freedom to explore different analyses and to share ideas with like-minded researchers, through the use of user-generated packages, simply cannot be ignored. Eventually the power and flexibility of R overrides cost issues for those with advanced needs.

Going beyond the limits of an introductory text, those with further interest in the use of R for biostatistics should consider additional topics such as:

- Central limit theorem and the law of large numbers.
- Confidence intervals.
- Data serving as markers for date and time.
- Dose-response analysis.
- Missing data and data imputation.
- Power.
- Risk assessment.
- Sampling and required sample size.
- Survey analysis.
- Survival analysis.

Small and easy-to-follow confidence-building examples have been used throughout this text. Complexity, often through the use of arguments and functions from external packages, was gradually introduced. The final chapters use a fairly robust approach to the use of R. Even with this gradual approach, the emphasis has been on good programming practices, easy-to-follow actions, and an emphasis on graphical presentations, modularity, and syntax reuse.

## 9.4 Big Data and BioInformatics

The term *Big Data* is now used so frequently in the press that it is possibly used too frequently and out of context. Although Big Data are usually associated with large datasets, the number of subjects is not the only consideration when taking a Big Data approach to analyses. Instead, Big Data also needs to take into consideration how analyses of what may seemingly be disassociated variables, when analyzed appropriately, support the discovery of associations that would not be apparent if the variables had been examined either individually or in union with only similar variables.

Big Data and BioInformatics is an evolving discipline, a discipline that has been made possible by reasonably priced computing systems, low cost software, easy access to data, and nearly unlimited data storage capabilities. Those who work in biostatistics now have the opportunity to go beyond planned analyses, often analyses of a limited number of subjects and a limited number of variables. Instead, with the use of electronic data storage and the desire for transparency and ready access to all different types of data, it is possible to use R as well as other software to examine data for trends that may not be immediately obvious.

The COVID-19 global pandemic is a current example of how Big Data is being used to develop statistics such as  $R_0$  (e.g., reproduction number) rates and projections for the number of days under current conditions needed to *Flatten the Curve*. Indeed, readily available open source data, with daily updates, now include worldwide, national, state, province, county, township, borough, etc., data on COVID-19, including data on: country and ISO code, province or state, region, latitude and longitude, total cases, daily updates of new cases, total deaths, daily updates of new deaths, total tests, daily updates of new tests, etc. Further, breakouts of many COVID-19 statistics are now available by gender, by age groups, by smoking behavior, by prevalence of diabetes, by pregnancy status, by other underlying health issues, local life expectancy, etc. The free availability of these open source datasets, with the data of extreme size given the global nature of the data, are used in support of public health initiatives and, as desired, the eventual development of effective COVID-19 therapies and vaccines. Big Data and BioInformatics will have a prime role in answering critical questions in the quest to promote public health.

## 9.5 External Resources

The Web page for this text, provided by the publisher, should be a first resource for access to sample sections of selected chapters, access to the .csv and other format datasets, etc. External resources, such as R-specific listserv e-mail discussion groups, should also be considered. Use the many resources provided by the Comprehensive R Archive Network (CRAN, <http://cran.us.r-project.org/>)

also. Of course, what may seem to be countless resources on R are also available by using RSeek (<http://rseek.org>).

## 9.6 Contact the Authors

Both authors first used S on a UNIX-based host computer in the 1980s, prior to the development of R, when S was then freely available to educational institutions. Dr. Thomas W. MacFarland is Senior Research Associate (Office of Institutional Effectiveness) and Associate Professor (College of Computing and Engineering), Nova Southeastern University, Fort Lauderdale, Florida, USA, [tommac@nova.edu](mailto:tommac@nova.edu). Dr. Jan M. Yates is Professor Emerita (Abraham S. Fischler College of Education), Nova Southeastern University, Fort Lauderdale, Florida, USA, [yates@nova.edu](mailto:yates@nova.edu).

The many imported datasets used in this text are identified at the end of each lesson and are available on the publisher's Web page associated with this text. Contact the authors if there are any questions about the use of this text, details on the accompanying datasets, or if additional pointers on R for biostatistics are needed. Questions, comments, and feedback are always appreciated. When sending e-mail to the authors, use a meaningful and descriptive term in the subject header so that the message does not get sent to a rarely monitored SPAM folder or is otherwise ignored. We appreciate feedback.

# Index

## A

Agriculture, 1, 2, 28, 488  
Analysis of variance (ANOVA)  
    comparison of group means, 54  
    oneway analysis (*see* Oneway analysis of variance (ANOVA))  
parametric oneway, 681–685  
    descriptive statistics, 673–676  
    determination of normal distribution, 676–681  
download the new file, 670–671  
graphical images, 671–673  
new dataset from an existing dataset, 667–670  
Tukey HSD Approach, 681–685  
    descriptive statistics, 673–676  
    determination of normal distribution, 676–681  
download the new file, 670–671  
graphical images, 671–673  
new dataset from an existing dataset, 667–670  
twoway analysis (*see* Twoway analysis of variance (ANOVA))  
Anderson-Darling normality test, 178, 181, 219, 220, 236  
ANOVA, *see* Analysis of variance (ANOVA)

## Association

additional practice datasets  
data that do not exhibit normal distribution patterns, 574–583  
data with normal distribution patterns, 569–574  
exit, save and R session, 583–584  
external data/resources, 584  
commonalities, 87  
data  
    description, 429–430  
    distribution, 465–488  
import data in comma-separated values, 431–434  
null hypothesis, 431  
plot, 726–727  
QA, 465–488  
second null hypothesis, 589, 666–667, 725  
statistical test(s), 488–532  
tests for normality, 465–488  
visual data check using graphics  
    descriptive statistics for initial analysis, 449–465  
factor object variables, 443–444  
numeric object variables, 444–449  
*See also* Correlation-association  
Association plot, 585, 726–727, 731

**B**

- Bagplot, 720–723, 740–741  
 Balanced design, 294, 362  
 Bar chart, 464, 565, 585, 728, 817, 821, 822  
 Bar plot, 728–729, 817–823  
   race-ethnicity, 134  
   section and gender, 67  
 Beanplot, 446–448, 741–743, 752–755  
*Beautiful Graphics*, 46–50, 132, 155, 372, 379, 726, 817, 870  
 Beeswarm plot, 585, 704, 705, 708, 743–746  
 Between-subjects factors, 227, 237, 295, 340, 352, 362  
 Big Data, 488, 587, 696, 888  
 Binary logistic regression, 428, 429, 523–529, 531, 550, 551, 580, 583  
 Bioinformatics, 888  
 Biostatistics  
   CLI, 3  
   correlation and regression, 532  
   development of, 3–6  
   GUI, 3  
   IDE, 3  
   numeric-type datum, 138  
   research designs  
     association, 54  
     case study and clinical trial, 52  
     comparison of group means, 54  
     correlation, 54  
     factorial data organization, 53  
     fixed group comparative analysis, 53  
     Goodness of Fit, 53–54  
     likelihood, 54  
     posttest only for control group, 53  
     prediction, 54  
     pretest–posttest for control group, 52

- pretest–posttest for one group, 52  
 regression, 54  
 R use, 887  
 table format, 725

Bivariate boxplot, 721, 740–741

Boole, George, 632

Boolean-based breakouts

- Boolean-based data selection process, 632
- download the new dataset, 635–650
- new dataset from an existing object variable, 632–635

Boolean selection, 641, 642, 871

Box-and-whiskers plot, 671, 747–757

Box-percentile plot, 757–760

Boxplot, 48, 50, 65, 68–69, 101–104, 115, 122, 161, 163, 164, 210, 215, 311–313, 385, 459–464, 752–757

Boxplot statistics, 108, 313, 747, 750

**C**

CLI, *see* Command line interface (CLI)

Code book

- dataset, 250
- organize the data, 63–65, 146–150, 207–208, 298–299, 370–371, 435–465, 591–605

Coefficient of correlation, 87, 490, 491, 723, 833

Color gradient plot, 787–791

Command line interface (CLI), 2–3, 42, 884

Comma-separated values (.csv)

- GUI selections, 35–38
- import data, 7–8
- online source, 27–35
- into R, 13–15
- R-based functions, 61–63

- self-generation, R-based functions, 98–99, 144–146, 246–248, 295–298, 364–370, 431–434, 589–591
- Comprehensive R Archive Network (CRAN), 6, 7, 104, 105, 381, 888
- Correlation
- additional practice datasets
  - data that do not exhibit normal distribution patterns, 574–583
  - data with normal distribution patterns, 569–574
  - exit, save and R session, 583–584
  - external data/resources, 584
- data
- description, 429–430
  - distribution, 465–488
- display the code book, 435–443
- import data in comma-separated values, 431–434
- null hypothesis, 431
- organize the data, 435–443
- QA, 465–488
- statistical test(s), 488–532
- tests for normality, 465–488
- visual data check using graphics
- descriptive statistics for initial analysis, 449–465
  - factor object variables, 443–444
  - numeric object variables, 444–449
- Correlation—association
- comparative graphics
    - for systolic blood pressure, 698–701
    - for weight, 701–707
  - correlation null hypothesis, 723
  - download the new file, 697–698
  - graphical images, 698
  - new dataset from an existing dataset, 691–696
  - normal distribution, 713–716
  - parametric Pearson's r, 717–723
- second null hypothesis, 716–717
- singular graphics, 707–711
- statistical analysis, 716
- tests for normal distribution, 711–713
- Correlogram, 790–793
- D**
- Data analysis
- EDA (*see* Exploratory data analysis (EDA))
  - inexperienced researchers, 606
  - nonparametric approach, 407
- Data distribution, 9, 15, 80–86, 112–116, 175–186, 217, 235, 262–267, 314–320, 465–488, 650–666, 849, 854, 884
- Data exploration
- Beautiful Graphics*, 132–138
  - data organization, 63–65
  - description of the data, 58–61
  - display the code book, 63–65
  - exit, save, and later retrieve this R session, 139
  - external data/data resources, 139
  - ggplot2 package, 132–138
  - import data in comma-separated values, 61–63
  - for initial analysis of the data, 70–80
  - multiple numeric and factor object variables in a data frame, 119–132
  - null hypothesis, 61
  - of singular numeric object variables, 116–119
  - visual data check using graphics, 65–70
- Datasets, large and complex
- data description, 586–588
  - descriptive statistics for initial analysis
  - analyses of object variables, 630–632

- Datasets, large and complex (*cont.*)
  - Boolean-based breakouts, 632–650
  - display the code book, 591–605
  - first null hypothesis, 589
  - graphical presentation of grouped data
    - association plot, 726–727
    - bar plot, 728–729
    - mosaic plot, 729–731
    - pie chart, 731–735
    - waffle chart, 735–739
  - graphics for normality
    - density plot, 652
    - histogram, 651
    - Q-Q plot, 652–654
  - import data in comma-separated values, 589–591
  - organize the data, 591–605
  - outcomes
    - for first null hypothesis, 724
    - for second null hypothesis, 725
  - second null hypothesis, 589
  - tests for normality
    - Anderson–Darling test, 654–655
    - Jarque–Bera test, 655–656
    - Lilliefors (Kolmogorov–Smirnov) test, 656–657
    - Shapiro–Wilk test, 657–666
  - visual data check using graphics, common graphical functions EDA, 607–614 for factor-type data, 614–618 for numeric data, 618–625- Data with and without normal distribution patterns, 202–227
- background, 97–98
- descriptive statistics for initial analysis, 104–112
- display the code book, 99–100
- import data in comma-separated values, 98–99
- organize the data, 99–100
- purpose, 97, 201–202, 285
- SBPNormal and SBPNotNormal, 115–116
- visual data check using graphics, 101–104
- See also* Independent samples
- DBP, *see* Diastolic blood pressure (DBP)
- Density plot, 15, 17, 20, 27, 31, 32, 35, 38, 48, 101–103, 166, 195, 196, 224, 232, 265, 267, 652, 760, 826–829
- Dependent variable, 362, 578
- Descriptive statistics
  - Beautiful Graphics*, 132–138
  - data organization, 63–65
  - description of the data, 58–61
  - display the code book, 63–65
  - exit, save, and R session retrieval, 139
  - external data/data resources, 139
  - ggplot2 package, 132–138
  - graphics using R, 43–46
  - import data in comma-separated values, 61–63
  - initial analysis of the data, 8, 70–80, 104–112
  - factor-type object variables, 451–453
  - frequency distributions, 451–453
  - measures of central tendency, 453–465
  - numeric-type object variables, 453–465
  - multiple numeric and factor object variables in a dataframe, 119–132
  - null hypothesis, 61
  - of singular numeric object variables, 116–119
  - visual data check using graphics, 65–70
- Diastolic blood pressure (DBP), 138, 227–230, 232–239, 448, 459, 462, 463, 483, 507–513, 521–

- 523, 525, 526, 528, 554, 559, 601, 620, 671–674, 683–687, 740, 741, 760, 768, 785
- Dotchart, 65, 113, 151–153, 212, 252, 254, 307, 308, 382, 760, 761
- Dotplot, 91, 114, 115, 169, 212, 215, 333, 760, 829–832
- Duncan's multiple range test, 338
- E**
- EDA, *see* Exploratory data analysis (EDA)
- EH plot, *see* Engelmann–Hecker (EH) plot
- Engelmann–Hecker (EH) plot, 762–765
- Exploratory data analysis (EDA)  
examination of the data, 606  
functions, 607–614  
QA, 616–622  
stem-and-leaf plot, 799
- F**
- Factorial design, 53, 381, 382, 386
- Fisher, Ronald, 5, 295
- Fixed-width format text file, 18–24
- G**
- Gantt chart, 803–805
- ggplot2 package  
descriptive statistics, 132–137  
graphics  
bar plot, 817–823  
box plot, 823–826  
density plot, 826–829  
dot plot, 829–833  
histogram, 848–858  
interaction plot, 863–864  
line chart, 864–867  
Q-Q plot, 859–863  
scatter plot, 833–843  
tile map, 867–870  
violin plot, 843–848  
professional publication, 49
- Good programming practices (gpp), 10, 41–43, 432, 434, 887
- Gosset, William Sealy, 5
- Graphical user interface (GUI), 2, 3, 11, 35–38, 139, 239, 291, 881
- GUI, *see* Graphical user interface (GUI)
- H**
- Hexbin plot, 792–794
- Histogram, 651, 766–772, 848–855  
numeric object variable, 101, 303  
peak and width, 119
- I**
- ICD, *see* International classification of diseases (ICD)
- IDE, *see* Integrated development environment (IDE)
- Independent samples  
data  
description, 142–143  
distribution, 175–186  
data that do not exhibit normal distribution patterns  
data description, 227  
data distribution, 235–237  
descriptive statistics for initial analysis, 234–235  
display the code book, 230–231  
import data in comma-separated values, 228–230  
null hypothesis, 227–228  
organize the data, 230–231  
outcome(s), 239  
QA, 235–237  
statistical test(s), 237–238  
tests for normality, 235–237  
visual data check using graphics, 231–234
- data with normal distribution patterns  
data description, 202–203  
data distribution, 217–220  
descriptive statistics for initial analysis, 214–217

- Independent samples (*cont.*)  
 import data in comma-separated values, 203–208  
 null hypotheses, 203  
 outcome(s), 226–227  
 QA, 217–220  
 statistical test(s), 221–226  
 tests for normality, 217–220  
 visual data check using graphics, 208–214  
 descriptive statistics for initial analysis, 167–175  
 display the code book, 146–150  
 exit, save, and R session, 239–240  
 external data/data resources, 240  
 import data in comma-separated values, 144–146  
 null hypothesis, 143–144  
 organize the data, 146–150  
 outcomes, 193–196  
 QA, 175–186  
 statistical test(s), 186–193  
 tests for normality, 175–186  
 t-statistic *v.* z-statistic  
   calculation, 199–200  
   enumerated dataset, 197–199  
   parametric *v.* nonparametric, 200–201  
 visual data check using graphics  
   factor-type barchart, 151  
   numeric-type graphics, 151–167  
 Independent variable, 53  
 Integrated development environment (IDE), 2, 3, 11, 42  
 Interaction  
   ANOVA, 364  
   dairy breed and management practice, 363  
   descriptive statistics, 394  
   gender and drug, 405  
   graphical image, 402  
   plot, 805–807  
 International classification of diseases (ICD), 593, 594, 622, 623  
 Interaction plot, 404, 574, 805–807, 863–864  
 Interquartile range (IQR), 75, 79, 80, 108, 170, 215, 258, 453–458, 629, 645  
 IQR, *see* Interquartile range (IQR)
- L**
- Lactation, 53, 142, 143, 362, 363  
 Lattice package, 194, 810–817  
 Least significant difference (LSD), 338  
 Length, 70, 76, 94, 108, 111, 116, 117, 168, 172–175, 230, 283, 633, 636, 657–660  
 Likelihood  
   additional practice datasets  
   data that do not exhibit normal distribution patterns, 574–583  
   data with normal distribution patterns, 569–574  
   exit, save and R session, 583–584  
   external data/resources, 584  
   binary logistic regression, 551–553  
   data  
     description, 429–430  
     distribution, 465–488  
   import data in comma-separated values, 431–434  
   null hypothesis, 431  
   odds ratio, 550–551  
   diastolic blood pressure by obesity, 557–559  
   parametric *v.* nonparametric, 560–562  
   systolic blood pressure by obesity, 554–557  
   probability, 550  
   QA, 465–488  
   statistical test(s), 488–532  
   tests for normality, 465–488  
   visual data check using graphics  
     descriptive statistics for initial analysis, 449–465  
     factor object variables, 443–444

- numeric object variables, 444–449
- Linear regression, 428, 429, 504–511, 531, 532
- Line chart, 772–778, 864–868
- Line graph, 772–778
- Long-format data, 274, 322
- LSD, *see* Least significant difference (LSD)
- M**
- MAM, *see* Minimal adequate model (MAM)
- Matched pairs
- data
    - description, 242–244
    - distribution, 262–267
  - descriptive statistics for initial analysis, 256–261
- display the code book, 248–252
- exit, save, and R session, 290–291
- external/data resources, 291
- import data in comma-separated values, 246–248
- N impact, 279–282
- normal distribution patterns, 285–290
- null hypothesis, 244
- organize the data, 248–252
- outcomes, 271–272
- parametric *v.* nonparametric, 282–285
- QA, 262–267
- stacked data, 275–279
- statistical test(s), 267–271
- structure, 242
- student's t-test (*see* Student's t-test)
- tests for normality, 262–267
- unstacked data, 244–245, 272–274
- visual data check using graphics, 252–256
- Mean, 106–107, 110–112, 115–119, 124, 138, 171–175, 188, 192, 194, 205, 206, 208, 261, 311, 337–341, 346–348, 394, 395, 403–405, 434, 558
- comparison of group, 54
- density plot, 31
- measures of central tendency, 8
- of operation, 2
- QA, 80
- SBP, 40
- Mean comparison technique, 54, 338, 570, 573, 589, 684, 690, 724
- Measures of central tendency
- biostatistics, 138
  - data organization, 71
  - descriptive statistics, 8, 60
  - graphical tools, 46
  - inferential test, 223
  - numeric-type object variables, 453–465
  - R community, 95
- Median, 14, 17, 31, 34, 59, 70, 72, 108, 109, 162, 168, 190, 248, 405, 434, 442, 647–649, 710, 711, 823, 824
- descriptive statistics, 43
  - measures of central tendency, 60
- Minimal adequate model (MAM), 537–541
- Mode, 72, 73, 105, 106, 111, 631, 836, 838
- measures for average, 59
  - measures of central tendency, 8
- Mosaic plot, 104, 129, 729–731
- Multiple regression
- hand-calculate, 534–537
  - MAM, 537–541
  - stepwise, 541–546
- N**
- Nonparametric approach
- correlation analyses, 484
  - Kruskal–Wallis approach, 685–690
  - normal distribution, 81
  - v.* parametric, 96, 200–201, 282–285, 289, 290, 349–352
  - research design, 58

- Nonparametric Kruskal–Wallis approach  
 first null hypothesis, 690  
 nonparametric statistics, 685–690  
 variance null hypothesis, 690
- Normal distribution  
 data (*see* Data with and without normal distribution patterns)  
 determination, 676–681  
 discernible pattern, 9  
 measured datum, 96  
 normality, 81  
 null hypothesis, 81  
 numeric object variables, 466  
 student’s t-test (*see* Student’s t-test)  
*See also* Independent samples
- Normality  
 Anderson–Darling test, 654–655  
 data distribution, 80–86, 112–115, 175–186, 217–220, 262–267, 314–320, 465–488  
 graphics, 650–651  
 Jarque–Bera test, 655–656  
 Kolmogorov–Smirnov test, 656–657  
 QA, 80–86, 112–115, 175–186, 217–220, 262–267, 314–320, 465–488  
 Shapiro–Wilk test, 657–666  
 tests for, 654
- Null hypothesis ( $H_0$ ), 7, 9, 61, 80, 81, 86, 114, 143–144, 187, 189, 203, 222, 244, 264, 268, 295, 314, 335, 363–364, 431, 466, 589, 654–658, 667
- null hypothesis 1  
 nonparametric Kruskal–Wallis approach, 685–690  
 parametric oneway ANOVA, 667–685
- null hypothesis 2, 690–723  
 outcomes, 724
- Numeric variable, 113, 491, 497, 562, 567, 708, 710, 760
- O**
- Object variable  
 data distribution, 15  
 factor-type, 48, 66, 80, 94, 129, 168, 362, 443–444, 530  
 frequency distributions, 451–453  
 numeric, 81, 179, 180, 183, 262, 444–449, 530
- Odds ratio  
 likelihood, 546–559  
 ordinal logistic regression, 511  
 predicted values, 511  
 statistics, 519, 521, 527
- Oneway analysis of variance (ANOVA) data  
 description, 294–295  
 distribution, 314–320  
 with normal distribution patterns, 354–357  
 that do not exhibit normal distribution patterns, 357–358  
 descriptive statistics for initial analysis, 310–313  
 display of oneway ANOVA, 346–349  
 display the code book, 298–302  
 exit, save and R session, 358–359  
 exploratory oneway, 334–335  
 external/data resources, 359  
 import data in comma-separated values, 297–298  
 null hypothesis, 295  
 oneway method 1, 335–336  
 oneway method 2, 336–340  
 organize the data, 298–302  
 outcomes, 340–346  
 parametric *v.* nonparametric, 349–352  
 QA, 314–320  
 statistical test(s), 320–334  
 tests for normality, 314–320  
 visual data check using graphics, 303–310

Open source software, 6, 7  
Ordinal regression, 515

## P

Parametric approach  
inferential tests, 5, 227  
*v.* nonparametric, 96, 200–201,  
282–285, 349–352, 409–412,  
560–562  
normal distribution, 9  
oneway ANOVA and Tukey HSD  
approach, 667–685  
Pearson’s r, 499, 504  
Pearson, Karl, 5  
Pearson’s r, 428, 429, 488–504,  
529, 530, 716–723, 837, 838,  
840–842  
Pie chart, 731–736, 739  
Pirate plot, 778–780  
Posttest  
comparison of pretest weights,  
254, 255  
distribution of pretest weights,  
262, 267  
posttest only for control group, 53  
pretest–posttest for control group,  
52  
pretest–posttest for one group, 52  
Prediction  
additional practice datasets  
data that do not exhibit normal  
distribution patterns, 574–583  
data with normal distribution  
patterns, 569–574  
exit, save and R session, 583–584  
external data/resources, 584  
binary logistic regression, 551–553  
data  
description, 429–430  
distribution, 465–488  
import data in comma-separated  
values, 431–434  
null hypothesis, 431  
odds ratio, 550–551

diastolic blood pressure by  
obesity, 557–559  
parametric *v.* nonparametric,  
560–562  
systolic blood pressure by  
obesity, 554–557  
probability, 550  
QA, 465–488  
statistical test(s), 488–532  
tests for normality, 465–488  
visual data check using graphics  
descriptive statistics for initial  
analysis, 449–465  
factor object variables, 443–444  
numeric object variables,  
444–449

Predictor variable, 504–509, 515, 525,  
527, 531, 532, 578

## Pretest

comparison of pretest weights,  
254, 255  
distribution of pretest weights,  
256, 267  
pretest–posttest for control group,  
52  
pretest–posttest for one group, 52  
Probability, 4, 519–524, 526–529, 550,  
673, 683

## Q

Quality assurance (QA), 15–17, 20–27,  
30–32, 34, 40–42, 80–86,  
112–115, 175–186, 262–267,  
314–320, 387–394, 465–488,  
530, 608–625, 650–666  
Quantile-quantile plot (Q-Q/QQ plot),  
84, 85, 101, 102, 104, 152, 184,  
211, 218, 233, 254, 289, 317,  
318, 355, 391, 392, 414, 420,  
652–654, 700, 702–705, 782,  
783, 859–863  
Quartiles, 43, 75, 87, 92, 108, 170,  
215, 258–260, 395, 453, 645,  
647, 747, 884

**R**

and *Beautiful graphics*, 46–50  
 dynamic output, 886–887  
 static reports, 885–886  
 big data and bioinformatics, 888  
 for biostatistics, 887  
 descriptive statistics, 43–46  
 development of, 6–7  
 efficient programming, 41–43  
 exit, save and R session, 55  
 external/data resources, 55–56,  
 888–889  
*gpp*, 41–43  
 graphics, 43–46  
 import data into  
   comma-separated values, 10,  
   13–15  
   fixed-width format values, 18–24  
*On the Fly*, 38–41  
 GUI selections, 35–38  
 housekeeping syntax, 11–12  
 online source, 27–35  
 spreadsheet file, 24–27  
 tab-separated values, 16–17  
 project workflow, 41–43  
 structure of, 7–10  
 Range, 60, 73, 75, 94, 100, 108, 112,  
 207, 214, 229–231, 299, 354,  
 405, 436, 453, 509  
 R Commander, 885  
 Regression  
   additional practice datasets  
   data that do not exhibit normal  
     distribution patterns, 574–583  
   data with normal distribution  
     patterns, 569–574  
   exit, save and R session, 583–584  
   external data/resources, 584  
 binary logistic, 523–529  
 data  
   description, 429–430  
   distribution, 465–488  
 import data in comma-separated  
 values, 431–434

multiple, 532–546  
 null hypothesis, 431  
 organize the data and display the  
   code book, 435–443  
 QA, 465–488  
 statistical test(s), 488–532  
 tests for normality, 465–488  
 visual data check using graphics  
   descriptive statistics for initial  
     analysis, 449–465  
   factor object variables, 443–444  
   numeric object variables,  
     444–449

R Markdown, 884–887  
 R Studio, 885

**S**

telecommunications company, 6  
 UNIX operating system, 6  
 SBP, *see* Systolic blood pressure  
 (SBP)  
 Scatter diagram, 783–787  
 Scatter plot, 460, 719, 721,  
 743, 783–789, 793–796,  
 833–843  
 Scatter plot matrix (SPLOM),  
 794–796, 816, 817  
 Scheffe's mean comparison test, 338,  
 573  
 SD, *see* Standard deviation (SD)  
 Shapiro normality test, 477, 484, 664  
 SNK, *see* Student-Newman-Keuls  
 (SNK)  
 Sort, 91, 108, 110, 111, 113, 154, 169,  
 549  
 Spearman's rho, 428, 429, 469,  
 489–504, 529, 561, 562, 575,  
 716–721, 725  
 Specialized external packages and  
 functions, 88–95  
 SPLOM, *see* Scatter plot matrix  
 (SPLOM)  
 Squared pie chart, 129, 586, 735  
 Stacked data, 245, 269, 274–279,  
 321–326

- Staircase plot, 807–809  
Standard deviation (SD), 45, 60, 95, 116, 117, 173–175, 187, 194, 206, 209–212, 223, 238, 301, 386, 406, 650, 884, 886  
Statistical test(s), 86–87, 115  
binary logistic regression, 523–529  
linear regression  
multiple predictor variables, 509–511  
single predictor variable, 504–509  
null hypothesis 1  
nonparametric Kruskal–Wallis approach, 685–690  
parametric oneway ANOVA, 667–685  
null hypothesis 2, 690–723  
ordinal logistic regression, 511–523  
outcomes, 529–532  
Pearson’s r correlation, 489–504  
Spearman’s rho correlation, 489–504  
Stem-and-leaf plot, 797–799  
Stepwise regression, 541–546, 576, 578  
Stripchart, 65, 66, 150, 152, 153, 252, 342, 343, 743, 799–800, 856–858  
Student-Newman-Keuls (SNK), 338  
Student’s t-test  
for independent samples (*see* Independent samples)  
matched pairs (*see* Matched pairs)  
Subset, 51, 208–212, 217, 235, 492, 554, 605, 632–635, 658, 661–663, 667, 677–679, 695, 696, 700–704, 761, 870–8807  
Sunflower scatterplot, 796–797  
Systolic blood pressure (SBP), 136–138, 203–206, 209–212, 214, 221–224, 297, 299, 303–310, 314–320, 326–337, 340–352, 372–383, 391–394, 402–410, 505, 510–515, 554–557, 641, 691, 705–708, 718–720, 792, 839, 841
- T**  
Tab-separated values text file, 16–17  
Tests for normality, 80–86, 112–115  
Text file, 16, 18, 38, 41, 55, 363  
Theme, 48–50, 68, 84, 133–137, 156–165, 185, 186, 232–234, 266, 267, 318–320, 333, 334, 373–378, 416, 422–424, 528, 571, 778, 801–803, 819–835, 842–854, 856–864, 867–869  
Tidyverse, 132, 213, 214, 273, 274, 278, 485, 632, 679, 817, 870, 871  
Trellis graphics, 810  
Triangular plot, 808–810  
Trimmed mean, 106, 107, 270, 271, 408, 409  
Tukey’s five number summary, 75, 109  
Tukey’s honestly significant difference (Tukey HSD), 338  
Twoway analysis of variance (ANOVA)  
data  
description, 363  
distribution, 387–394  
with normal distribution  
patterns, 413–418  
that do not exhibit normal distribution patterns, 418–425  
descriptive statistics for initial analysis, 383–387  
display the code book, 370–371, 408–409  
exit, save and R session, 425  
external/data resources, 426  
import data in comma-separated values, 364–370  
null hypothesis, 363–364  
organize the data, 370–371  
outcomes, 405–408  
parametric *v.* nonparametric, 409–412

Two-way analysis of variance (ANOVA)

(*cont.*)

QA, 387–394

statistical test(s), 394–405

tests for normality, 387–394

visual data check using graphics,  
371–383

## U

Unstacked data, 244–245, 254–256,  
262, 267, 269, 272, 321

## V

Violin plot, 101, 208, 214, 233, 254,  
255, 309, 310, 332, 333, 375–  
377, 417, 423, 673, 698, 701,  
800–801, 844–848

## W

Waffle chart, 129–132, 735–739

Winsor mean, 106, 107, 111, 649

## Z

Z-statistic, 143, 196–200