

SUMMARY AND ANALYSIS OF EXTENSION PROGRAM EVALUATION IN R

SALVATORE S. MANGIAFICO

Rutgers Cooperative Extension
New Brunswick, NJ

VERSION 1.18.1

©2016 by Salvatore S. Mangiafico.

Non-commercial reproduction of this content, with attribution, is permitted.
For-profit reproduction without permission is prohibited.

If you use the code or information in this site in a published work, please cite it as a source. Also, if you are an instructor and use this book in your course, please let me know.
mangiafico@njaes.rutgers.edu

Mangiafico, S.S. 2016. Summary and Analysis of Extension Program Evaluation in R, version 1.18.1.
rcompanion.org/documents/RHandbookProgramEvaluation.pdf.

Web version: rcompanion.org/handbook/

List of Chapters

<i>List of Chapters</i>	iii
<i>Table of Contents</i>	vi
<i>Introduction</i>	1
Purpose of This Book.....	1
Author of this Book and Version Notes	2
Using R	3
Statistics Textbooks and Other Resources.....	14
<i>Statistics for Educational Program Evaluation</i>	17
Why Statistics?	17
Evaluation Tools and Surveys	18
<i>Variables, Descriptive Statistics, and Plots</i>	24
Types of Variables.....	24
Descriptive Statistics	30
Confidence Intervals	62
Basic Plots	73
<i>Understanding Statistics and Hypothesis Testing</i>	113
Hypothesis Testing and p-values.....	113
Reporting Results of Data and Analyses	134
Choosing a Statistical Test	138
Independent and Paired Values.....	146
<i>Likert Data</i>	152
Introduction to Likert Data.....	152
Descriptive Statistics for Likert Data	162
Descriptive Statistics with the likert Package	184
Confidence Intervals for Medians	189
Converting Numeric Data to Categories	197
<i>Traditional Nonparametric Tests</i>	208
Introduction to Traditional Nonparametric Tests	208
One-sample Wilcoxon Signed-rank Test.....	212
Sign Test for One-sample Data	218

Two-sample Mann–Whitney U Test.....	221
Mood’s Median Test for Two-sample Data	234
Two-sample Paired Signed-rank Test	237
Sign Test for Two-sample Paired Data.....	245
Kruskal–Wallis Test	248
Mood’s Median Test	261
Friedman Test.....	267
Quade Test	279
Scheirer–Ray–Hare Test	286
Aligned Ranks Transformation ANOVA	292
Nonparametric Regression and Local Regression	304
Nonparametric Regression for Time Series	313
Permutation Tests	320
Introduction to Permutation Tests.....	320
One-way Permutation Test of Independence for Ordinal Data.....	324
One-way Permutation Test of Symmetry for Paired Ordinal Data	333
Permutation Tests for Medians and Percentiles	343
Tests for Ordinal Data in Tables	349
Association Tests for Ordinal Tables	349
Measures of Association for Ordinal Tables	359
Concepts for Linear Models.....	368
Introduction to Linear Models	368
Using Random Effects in Models.....	371
What are Estimated Marginal Means?	376
Estimated Marginal Means for Multiple Comparisons	379
Factorial ANOVA: Main Effects, Interaction Effects, and Interaction Plots	387
p-values and R-square Values for Models	399
Accuracy and Errors for Models	410
Ordinal Tests with Cumulative Link Models.....	416
Introduction to Cumulative Link Models (CLM) for Ordinal Data.....	416
Two-sample Ordinal Test with CLM	418
Two-sample Paired Ordinal Test with CLMM	422
One-way Ordinal Regression with CLM	427

One-way Repeated Ordinal Regression with CLMM	433
Two-way Ordinal Regression with CLM.....	442
Two-way Repeated Ordinal Regression with CLMM	454
Tests for Nominal Data.....	468
Introduction to Tests for Nominal Variables.....	468
Confidence Intervals for Proportions	478
Goodness-of-Fit Tests for Nominal Variables.....	483
Association Tests for Nominal Variables	499
Measures of Association for Nominal Variables	508
Tests for Paired Nominal Data	522
Cochran–Mantel–Haenszel Test for 3-Dimensional Tables.....	534
Cochran’s Q Test for Paired Nominal Data	538
Models for Nominal Data	548
Parametric Tests.....	566
Introduction to Parametric Tests	566
One-sample t-test	585
Two-sample t-test.....	592
Paired t-test.....	600
One-way ANOVA	609
One-way ANOVA with Blocks	621
One-way ANOVA with Random Blocks.....	631
Two-way ANOVA	639
Repeated Measures ANOVA.....	653
Correlation and Linear Regression	664
Advanced Parametric Methods	682
Transforming Data	703
Analysis of Count Data and Percentage Data	722
Regression for Count Data.....	722
Beta Regression for Percent and Proportion Data	738

Table of Contents

List of Chapters	iii
Table of Contents	vi
Introduction	1
Purpose of This Book	1
Goals of this book	1
Specific learning goals	1
Pre-requisites	1
What this book will not cover	1
Designing and conducting surveys	2
Advanced statistical analyses	2
R programming	2
Author of this Book and Version Notes	2
Author of this book	2
Version notes	2
Using R	3
R and RStudio	3
Using the RStudio environment	3
Using the R Console environment	3
Installing R and RStudio	3
Installing R and RStudio	3
Optional: Problems with library folder location in Windows 10	4
What if I don't have my own computer?	5
Portable installation on a usb drive	5
Using university computers	5
Using R online	6
Tests for package installation	6
Required readings	8
About R	8
A Few Notes to Get Started with R	8
Avoiding Pitfalls in R	8
Help with R	8
References	8
Exercises A	8
Exercises Alpha	11
Statistics Textbooks and Other Resources	14
Some free general statistics books	14
Books on statistical analyses and tests	15
Analyses in R	15
Online resources	15
Text books	15
Videos	15
R programming	15
Other useful resources for understanding statistics	16

Videos	16
Blogs	16
Online Learning Modules and Massive Open Online Courses (MOOC's)	16
Statistics for Educational Program Evaluation	17
Why Statistics?	17
The importance of statistics in program evaluation	17
Applicability of statistical tests to other fields and situations	18
Evaluation Tools and Surveys	18
Extension programs	18
Designing a questionnaire	18
Some principles of questionnaire design for program evaluation	19
Examples of evaluation tools	21
Optional Readings	22
Optional additional resources	23
References for this chapter	23
Acknowledgements	23
Variables, Descriptive Statistics, and Plots	24
Types of Variables	24
Organizing data: observations and variables	24
Long-format and wide-format data	24
Types of variables	25
Nominal data	25
Ordinal data	25
Interval/ratio data	26
Levels of measurement	26
Types of Variables in R	27
References	28
Required readings	29
Optional readings	29
Exercises B	29
Descriptive Statistics	30
Packages used in this chapter	30
Descriptive statistics	31
Descriptive statistics for interval/ratio data	31
Statistics of location for interval/ratio data	32
Statistics of variation for interval/ratio data	34
Statistics for grouped interval/ratio data	36
Summaries for data frames	38
Dealing with missing values	40
Statistics of shape for interval/ratio data	43
Descriptive statistics for ordinal data	45
Example of descriptive statistics for ordinal data	45
Statistics of location for ordinal data	47
Statistics of variation for ordinal data	47
Statistics for grouped ordinal data	48
Statistics for ordinal data treated as nominal data	48
Descriptive statistics for nominal data	50
Example of descriptive statistics for nominal data	51
Levels for factor variables	53

Required readings	54
Optional readings	54
Optional note on reporting summary statistics honestly	55
Optional analyses	56
Robust estimators: trimmed mean and Winsorized mean	57
Geometric mean	58
Harmonic mean	60
Exercises C	60
Confidence Intervals	62
Packages used in this chapter	63
Understanding confidence intervals	63
Populations and samples	63
Statistics and parameters	64
Point estimates and confidence intervals	64
Confidence intervals as an alternative to some tests	64
Example for confidence intervals	65
Recommended procedures for confidence intervals for means	66
groupwiseMean function for grouped and ungrouped data	66
Optional: Other functions for traditional confidence intervals for means	68
Optional Analyses: confidence intervals for the mean by bootstrapping	69
Required readings	70
Optional readings	70
References	71
Optional analyses	71
Confidence intervals for geometric mean	71
Confidence intervals for geometric means for groups	71
Exercises D	72
Basic Plots	73
The need to understand plots	73
Packages used in this chapter	74
Some advice on producing plots	74
Producing clear and informative plots	75
Misleading plots	75
Using software to produce plots	75
Exporting plots	76
Examples of basic plots for interval/ratio and ordinal data	76
Histogram	77
Box plots	81
Plot of means and interaction plots	85
Bar plot of means	91
Scatter plot	95
Examples of basic plots for nominal data	96
Bar plot for counts of a nominal variable	97
Mosaic plot	99
Ordering plot categories	100
Optional analyses: Describing histogram shapes	101
Optional analyses: Misleading and disorienting plots	103
The need to include measures of variation	103
For bar charts, start the y-axis at zero	104
Avoid disorienting the audience	105
Use the best model	107

References	109
Optional readings I	109
Optional readings II	109
Exercises E	110
Understanding Statistics and Hypothesis Testing	113
Hypothesis Testing and p-values	113
Initial comments	113
Statistical inference	113
Packages used in this chapter	113
Hypothesis testing	114
The null and alternative hypotheses	114
p-value definition	114
Theory and practice of using p-values	117
Wait, does this make any sense?	117
Statistics is like a jury?	117
Errors in inference	118
The 0.05 alpha value is not dogma	119
Is the p-value every really true?	120
Effect sizes and practical importance	120
Practical importance and statistical significance	120
Sizes of effects	121
p-values and sample sizes	122
Effect size statistics	122
Good practices for statistical analyses	123
Statistics is not like a trial	123
Preplanned tests	124
p-value hacking	124
Clarification of terms and reporting on assignments	125
"Statistically significant"	125
"Size of the effect" / "effect size"	125
"Practical" / "practical importance"	126
A few of xkcd comics	126
Significant	126
Null hypothesis	126
P-values	126
Experiments, sampling, and causation	126
Types of experimental designs	126
Quasi-experiment designs	127
Observational studies	127
Sampling	127
Plan ahead and be consistent	127
Optional discussion: Alternative methods to the Null Hypothesis Significance Test	128
The NHST controversy	128
Alternatives to the NHST approach	129
References and further reading	130
Exercises F	131
Reporting Results of Data and Analyses	134
Packages used in this chapter	134
Reporting analyses, results, and assumptions	134
Procedures	134
Results	134

Notes on different data and analyses	135
Advice on tables and plots	135
Plots	135
Tables	136
Table headings and plot captions	136
Example description of statistical analysis and results	136
Procedures	136
Results	137
Optional technical note on effect sizes	137
References	138
Choosing a Statistical Test	138
Plan your experimental design before you collect data	139
What is the hypothesis?	139
What number and type of variables do you have?	139
Table of tests by variable type	140
Optional discussion: Sometimes it's all about the hypothesis	143
Example	144
References	146
Optional readings	146
Independent and Paired Values	146
Packages used in this chapter	147
An example of paired and unpaired data	147
Box plot and summary statistics by group	148
Bar plot to show paired differences	149
Paired t-test and unpaired t-test	150
Histogram of differences with normal curve	151
Likert Data	152
Introduction to Likert Data	152
Likert data	152
Numbers of responses	152
Symmetry	152
Neutral responses	152
Form of responses	152
Opt-out answers	153
Examples of Likert item responses	153
Likert items and Likert scales	153
Analysis of Likert item data	153
Likert data should be treated as ordinal data	154
Some people treat Likert data as interval/ratio data	154
Analysis of Likert scale data	154
Analysis of Likert data	155
Optional: Simulated comparisons of traditional nonparametric tests and ordinal regression	155
Optional: Simulated comparisons of traditional nonparametric tests and exact tests and Monte Carlo approaches	159
Optional reading	162
Descriptive Statistics for Likert Data	162
Median	162
Count of responses and bar plot	163
Quartiles and percentiles	163
Packages used in this chapter	163

Descriptive statistics for one-sample data	163
One-sample data	164
Variables and functions in this example	164
Summary treating Likert data as nominal data	165
Summary treating Likert data as numeric data	167
Descriptive statistics for one-way or multi-way data	168
One-way data	168
Multi-way data	168
Variables and functions in this example	168
Summary treating Likert data as nominal data	171
Summary treating Likert data as numeric data	173
Interaction plot using medians and quartiles	177
Interaction plot using medians and confidence intervals	178
Descriptive statistics with opt-out responses	179
Analysis for vector data	180
Analysis for long-form data frame	181
Summary for whole data frame	182
Exercises G	183
Descriptive Statistics with the likert Package	184
Packages used in this chapter	184
The <i>likert</i> package	185
Summary statistics and plots with the <i>likert</i> package	186
Plots that treat Likert data like factor data	186
Plots that treat Likert data like numeric data	188
References	189
Confidence Intervals for Medians	189
Packages used in this chapter	190
Medians, quantiles, and confidence intervals for one-sample data	190
Produce median with median, summary, and Summarize functions	191
groupwiseMedian function to produce medians and confidence intervals	192
Medians, quantiles, and confidence intervals for grouped data	192
Summarize function in FSA package for grouped data	193
groupwiseMedian function for grouped data	194
Optional methods and discussion on confidence intervals for medians	194
Optional: Confidence interval for medians with the wilcox.test function	195
Optional: Median and confidence interval with the DescTools package	195
Optional: Confidence interval for median by bootstrap	195
References	196
Exercises H	196
Converting Numeric Data to Categories	197
Categorizing data by a range of values	198
Categorizing data by percentiles	198
Categorizing data with clustering	198
Packages used in this chapter	198
Examples for converting numeric data to categories	199
Categorize data by range of values	200
Categorize data by percentile	201
Categorize data by clustering	202
Traditional Nonparametric Tests	208
Introduction to Traditional Nonparametric Tests	208

Packages used in this chapter	208
Introduction	208
Advantages of nonparametric tests	209
Disadvantages of nonparametric tests	209
Interpretation of nonparametric tests	209
Effect size statistics	210
Optional: Appropriate use of traditional nonparametric tests	211
Using traditional nonparametric tests with ordinal data	211
Using traditional nonparametric tests with interval/ratio data	211
References	212
One-sample Wilcoxon Signed-rank Test	212
Packages used in this chapter	213
One-sample Wilcoxon signed-rank test example	214
Summarize data treating Likert scores as factors	215
Summarize data treating Likert scores as numeric	216
One-sample Wilcoxon signed-rank test	216
Effect size	216
Exercises I	217
Sign Test for One-sample Data	218
Packages used in this chapter	219
One-sample sign test example	219
Sign test with the <i>BSDA</i> package	220
Sign test with the <i>DescTools</i> package	220
Two-sample Mann–Whitney U Test	221
When to use this test	221
Packages used in this chapter	222
Two-sample Mann–Whitney U test example	222
Summarize data treating Likert scores as factors	223
Bar plots of data by group	224
Summarize data treating Likert scores as numeric	225
Two-sample Mann–Whitney U test example	225
Effect size	225
Optional: Comparison among effect size statistics	229
References	232
Exercises J	233
Mood's Median Test for Two-sample Data	234
Packages used in this chapter	235
Example using the <i>RVAideMemoire</i> package	235
Mood's Median Test	236
Two-sample Paired Signed-rank Test	237
When to use this test	237
Packages used in this chapter	238
Two-sample paired signed-rank test example	238
Number of observations per group	239
Plot the paired data	239
Descriptive statistics	242
Two-sample paired signed-rank test	243
Effect size	244
Exercises K	244

Sign Test for Two-sample Paired Data	245
Packages used in this chapter	246
Sign test for paired two-sample data example	246
Two-sample sign test with BSDA package	247
Two-sample sign test with DescTools package	248
Kruskal–Wallis Test	248
Packages used in this chapter	249
Kruskal–Wallis test example	250
Summarize data treating Likert scores as factors	251
Bar plots of data by group	252
Summarize data treating Likert scores as numeric	252
Kruskal–Wallis test example	253
Effect size	253
Post-hoc test: Dunn test for multiple comparisons of groups	255
Post-hoc test: pairwise Mann–Whitney U-tests for multiple comparisons	256
Plot of medians and confidence intervals	258
References	259
Exercises L	259
Mood's Median Test	261
Packages used in this chapter	262
Example using the <i>RVAideMemoire</i> package	262
Mood's median test	264
Post-hoc test: pairwiseMedianTest function	265
Post-hoc test: pairwiseMedianMatrix function	266
Friedman Test	267
Packages used in this chapter	268
Friedman test example	269
Summarize data treating Likert scores as factors	270
Bar plots by group	271
Bar plots of differences between groups	272
pairwiseDifferences to produce bar plots of differences between all groups	273
Summarize data treating Likert scores as numeric	274
Friedman test example	275
Effect size	275
Post-hoc test: pairwise sign test for multiple comparisons of groups	276
Post-hoc Conover test	278
Quade Test	279
Packages used in this chapter	280
Quade test example	281
Summarize data treating Likert scores as factors	282
Bar plots by group	283
Summarize data treating Likert scores as numeric	284
Quade test example	284
Post-hoc test: pairwise two-sample paired rank-sum test for multiple comparisons	284
Post-hoc test: Quade post-hoc test	285
Scheirer–Ray–Hare Test	286
Packages used in this chapter	288
Scheirer–Ray–Hare test examples	288
Midichlorians example	288
Example from Sokal and Rohlf	289

Example from Real Statistics Using Excel	290
References	291
Aligned Ranks Transformation ANOVA	292
Introduction	292
Packages used in this chapter	292
Aligned Ranks Transformation ANOVA examples	293
Midichlorians example	293
One-way example	297
Repeated measures example	299
Optional: Plot of medians and confidence intervals for midichlorians data	302
References	303
Nonparametric Regression and Local Regression	304
Packages used in this chapter	304
Nonparametric correlation	305
Nonparametric regression examples	305
Kendall–Theil Sen Siegel nonparametric linear regression	306
Plot with statistics	307
Quantile regression	308
Plot with statistics	309
Local regression	310
Plot	311
Generalized additive models	311
Plot with statistics	312
Nonparametric Regression for Time Series	313
Packages used in this chapter	314
Nonparametric regression examples	314
Decomposing time series objects	316
Mann–Kendall trend test	317
Sen's slope for time series data	318
Pettitt's test for change in value	319
Permutation Tests	320
Introduction to Permutation Tests	320
The coin package	320
Packages used in this chapter	321
Permutation test example	321
Box plot	322
Scatter plot with one-to-one line	322
Permutation test of independence	323
Permutation test of symmetry	324
References	324
One-way Permutation Test of Independence for Ordinal Data	324
When to use this test	324
Packages used in this chapter	325
One-way ordinal permutation test example	326
Summarize data treating Likert scores as factors	327
Bar plots by group	328
Summarize data treating Likert scores as numeric	328
One-way ordinal permutation test	329
Post-hoc test: pairwise permutation tests	329

Plot of medians and confidence intervals	331
Comparison of methods for independence tests for ordinal data	332
One-way Permutation Test of Symmetry for Paired Ordinal Data	333
When to use this test	333
Packages used in this chapter	334
One-way ordinal permutation test of symmetry example	334
Summarize data treating Likert scores as factors	336
Bar plots by group	336
Summarize data treating Likert scores as numeric	337
Permutation symmetry test	338
Post-hoc test: pairwise permutation tests	338
Plot of medians and confidence intervals	340
Comparison of methods for symmetry tests for ordinal data	342
Permutation Tests for Medians and Percentiles	343
Packages used in this chapter	343
Examples	343
Box plots	344
Confidence intervals for percentiles	345
Test for percentiles between two groups	346
Test for percentiles among multiple groups	347
Summary table of results	348
Tests for Ordinal Data in Tables	349
Association Tests for Ordinal Tables	349
Packages used in this chapter	349
Linear-by-linear test for ordered contingency tables	349
Example of linear-by-linear test 1	349
Example of linear-by-linear test 2	351
Extended Cochran–Armitage test	352
Example of extended Cochran–Armitage test 1	352
Example of extended Cochran–Armitage test 2	354
Long-form data and three-dimensional contingency tables	356
Two-dimensional table example	357
Three-dimensional table example	358
References	359
Measures of Association for Ordinal Tables	359
Measures of association for one ordinal variable and one nominal variable	359
Measures of association for two ordinal variables	359
Polychoric and tetrachoric correlation	359
Packages used in this chapter	360
Examples for Freeman's <i>theta</i> and <i>epsilon-squared</i>	360
Freeman's <i>theta</i>	361
<i>Epsilon-squared</i>	361
Freeman's <i>theta</i>	361
<i>Epsilon-squared</i>	362
Additional examples for Freeman's <i>theta</i> and <i>epsilon-squared</i>	362
Perfect association	362
Zero association	362
Examples for Somers' <i>D</i> , Kendall's <i>tau-b</i> , and Goodman and Kruskal's <i>gamma</i>	363
First example	363
Second example	364

Examples for polychoric correlation	365
First example	365
Second example	366
Third example	366
Concepts for Linear Models	368
Introduction to Linear Models	368
Types of linear models	368
Formulae for specifying models in R	369
Data in data frames	369
Random effects	370
Extracting model information from R	370
References	371
Using Random Effects in Models	371
Packages used in this chapter	372
An example of a mixed model	373
Mixed model with lmer	374
Mixed model with nlme	375
What are Estimated Marginal Means?	376
Packages used in this chapter	377
Estimated marginal means example	377
Arithmetic means	378
Estimated marginal means	378
Estimated Marginal Means for Multiple Comparisons	379
p-value adjustments for multiple comparisons	379
Confidence intervals for marginal means	379
Ignore values of emmeans for clm and clmm models	379
Packages used in this chapter	379
Multiple comparisons with <i>emmeans</i>	380
Optional: Interaction plot of estimated marginal means with mean separation letters	383
Input data and specify linear model	383
One-way estimated marginal means and plot	383
Interaction plot of estimated marginal means	385
Factorial ANOVA: Main Effects, Interaction Effects, and Interaction Plots	387
Packages used in this chapter	388
Two-way ANOVA example with interaction effect	389
Simple interaction plot	390
Specify the linear model and conduct an analysis of variance	391
Post-hoc testing with emmeans	391
Extended example with additional country	392
Simple interaction plot	393
Specify the linear model and conduct an analysis of variance	394
Post-hoc testing with emmeans	394
Interaction plot with error bars using ggplot2	395
Interaction plot with the phia package	398
p-values and R-square Values for Models	399
p-values for models	399
R-squared and pseudo R-squared	400
Packages used in this chapter	400

Example of model <i>p</i> -value, <i>R-squared</i> , and pseudo <i>R-squared</i>	401
Linear model	403
Generalized linear model	404
Optional analyses: Confidence intervals for R-squared values	406
Linear model	407
Generalized linear model	408
References	410
Accuracy and Errors for Models	410
Packages used in this chapter	410
Examples of accuracy and error	411
Ordinal Tests with Cumulative Link Models	416
Introduction to Cumulative Link Models (CLM) for Ordinal Data	416
The ordinal package	416
Analysis of deviance	417
Model assumptions for CLM	417
References	417
Two-sample Ordinal Test with CLM	418
Packages used in this chapter	418
Two-sample ordinal model example	419
Summarize data treating Likert scores as factors	420
Bar plots of data by group	420
Summarize data treating Likert scores as numeric	421
Two-sample ordinal model example	421
Two-sample Paired Ordinal Test with CLMM	422
Packages used in this chapter	423
Two-sample paired ordinal model example	423
Plot the paired data	424
Two-sample paired ordinal model	425
One-way Ordinal Regression with CLM	427
Packages used in this chapter	428
One-way ordinal regression example	428
Summarize data treating Likert scores as factors	429
Bar plots by group	430
Summarize data treating Likert scores as numeric	430
One-way ordinal regression	431
One-way Repeated Ordinal Regression with CLMM	433
Packages used in this chapter	433
One-way repeated ordinal regression example	434
Summarize data treating Likert scores as factors	435
Bar plots by group	436
Summarize data treating Likert scores as numeric	436
One-way repeated ordinal regression	437
Post-hoc comparisons with emmeans	438
Optional analyses	439
Post-hoc test: pairwise paired ordinal tests for multiple comparisons	439
Two-way Ordinal Regression with CLM	442
Packages used in this chapter	442
Two-way ordinal regression example	443

Summarize data treating Likert scores as factors	445
Bar plots by group	446
Summarize data treating Likert scores as numeric	448
Produce interaction plot with medians and quantiles	449
Two-way ordinal regression	450
Post-hoc tests with emmeans for multiple comparisons of groups	451
Two-way Repeated Ordinal Regression with CLMM	454
Packages used in this chapter	454
Two-way repeated ordinal regression example	455
Summarize data treating Likert scores as factors	457
Bar plots by group	457
Summarize data treating Likert scores as numeric	459
Plot Likert scores before and after	459
Produce interaction plot with medians and confidence intervals	460
Two-way repeated ordinal regression	461
Post-hoc tests with emmeans for multiple comparisons of groups	464
Interaction plot with group separation letters	466
Tests for Nominal Data	468
Introduction to Tests for Nominal Variables	468
Descriptive statistics and plots for nominal data	468
Contingency tables and matrices	468
Bar plots	469
Mosaic plots	470
Working with proportions	471
Optional analyses: converting among matrices, tables, counts, and cases	472
Long-format with each row as an observation (cases)	472
Long-format with counts of observations (counts)	474
Matrix form	475
Optional analyses: obtaining information about a matrix object	477
References	478
Confidence Intervals for Proportions	478
Packages used in this chapter	479
Example of confidence intervals for a binomial proportion	479
Example of confidence intervals for a multinomial proportion	481
Optional analysis: confidence intervals for a difference in proportions	482
References	483
Goodness-of-Fit Tests for Nominal Variables	483
Packages used in this chapter	484
Goodness-of-fit tests for nominal variables example	484
Exact tests for goodness-of-fit	484
G-test for goodness-of-fit	485
Chi-square test for goodness-of-fit	487
Effect size for goodness-of-fit tests	488
Examples of effect size for goodness-of-fit tests	488
Post-hoc analysis	489
Standardized residuals	489
Multinomial confidence intervals	490
Simple bar plots	490
Race example	491
Ethnicity example	493

Optional: Multinomial test example with plot and confidence intervals	495
Optional readings	498
References	498
Exercises Beta	499
Association Tests for Nominal Variables	499
Low cell counts	500
Fisher's exact test	500
Choosing among tests	500
Packages used in this chapter	501
Association tests for nominal variables example	501
Reading the data as a matrix	501
Expected cell counts	502
Effect size	502
Fisher Exact test of association	502
Post-hoc analysis	502
G-test of association	504
Post-hoc analysis	505
Chi-square test of association	505
Post-hoc analysis	506
Optional readings	506
References	507
Exercises M	507
Measures of Association for Nominal Variables	508
Interpretation of statistics	509
Packages used in this chapter	510
Examples for measures of association for nominal variables	510
Cramér's V	510
phi	511
Odds ratio	512
Cohen's w	513
Cohen's h	513
Goodman Kruskal lambda	514
Tschuprow's T	514
Optional analysis: using effect size statistics when counts are not known	515
Optional analysis: Cohen's h and <i>prop.test</i> for paired data when subject matching isn't known	516
Optional: Comparing phi and Cohen's h for a 2 x 2 table	517
Optional analysis: changing the order of the table	518
Optional analysis: comparing statistics	519
References	521
Tests for Paired Nominal Data	522
Packages used in this chapter	523
McNemar and McNemar–Bowker tests	523
Exact tests	523
Example of tests for paired data nominal data	523
Exact tests for symmetry	525
Effect size	526
McNemar and McNemar–Bowker chi-square tests for symmetry	526
An example without repeated measures, comparing test of symmetry with test of association	527
Optional analysis: a 4 x 4 example with several 0's	528
Optional analyses: conducting exact tests for symmetry	530
Optional: Comparing odds ratio and Cohen's g	531

References	532
Exercises N	533
Cochran–Mantel–Haenszel Test for 3-Dimensional Tables	534
Packages used in this chapter	535
C–M–H test example: long-format with counts	535
C–M–H test example: table format	538
Cochran’s Q Test for Paired Nominal Data	538
Packages used in this chapter	540
Cochran’s Q test example: long-format	540
Cochran’s Q test example: short-format in matrix	546
Models for Nominal Data	548
Packages used in this chapter	548
Log-linear models for tests of association	549
Log-linear model example	549
Logistic regression	552
Logistic regression example	553
Multinomial logistic regression	556
VGAM package	556
mlogit and nnet packages	557
Multinomial regression example	557
Mixed-effects logistic regression example	561
Other tools for categorical analysis	565
Parametric Tests	566
Introduction to Parametric Tests	566
Packages used in this chapter	566
When to use parametric tests	566
Count data may not be appropriate for common parametric tests	566
Percentage and proportion data	567
Assumptions in parametric statistics	567
Random sampling	568
Independent observations	568
Normal distribution of data or residuals	568
Homogeneity of variance	570
Additivity of treatment effects	571
Outliers	571
Parametric tests are somewhat robust	571
Assessing model assumptions	572
Using formal tests to assess normality of residuals	573
Skew and kurtosis	573
Using visual inspection to assess the normality of residuals	573
Steps to handle violations of assumptions	579
Descriptive Statistics for Parametric Statistics	579
Optional readings	580
References	580
Optional analyses: formal tests for normality	580
Shapiro–Wilk normality test	581
Anderson–Darling normality test	582
One-sample Kolmogorov–Smirnov test	582
D’Agostino Normality Test	582
Optional analyses: formal tests for homogeneity of variance	583

Levene's test for homogeneity of variance	584
Brown–Forsythe or robust Levene's test	585
Fligner-Killeen test	585
One-sample t-test	585
Packages used in this chapter	586
One-sample t-test example	586
Histogram of data	587
Normal quantile plot of data	588
One-sample t-test	588
Effect size	589
Interpretation of Cohen's d	589
Cohen's d for one-sample t-test	589
Box plot with default value	590
References	591
Exercises O	591
Two-sample t-test	592
Packages used in this chapter	593
Two-sample t-test example	593
Summarize data by group	595
Histograms for data by group	595
Box plots for data by group	597
Two-sample unpaired t-test	597
Effect size	598
Interpretation of Cohen's d	598
Cohen's d for two-sample t-test	598
Optional readings	598
References	598
Exercises P	599
Paired t-test	600
Packages used in this chapter	601
Paired t-test example	601
Check the number of paired observations	602
Histogram of difference data	602
Plot the paired data	603
Paired t-test	605
Effect size	606
Interpretation of Cohen's d	606
Cohen's d for paired t-test	606
Optional readings	607
References	607
Exercises Q	608
One-way ANOVA	609
Packages used in this chapter	610
One-way ANOVA example	610
Summarize data by group	612
Box plots for data by group	612
Plot of means and confidence intervals	613
Define linear model	614
Conduct analysis of variance	615
Histogram of residuals	615

Post-hoc analysis: mean separation tests	616
Plot of means, confidence intervals, and mean-separation letters	617
References	619
Exercises R	619
One-way ANOVA with Blocks	621
Packages used in this chapter	622
One-way ANOVA with blocks example	622
Use Summarize to check if cell sizes are balanced	624
Define linear model	625
Conduct analysis of variance	625
Histogram and plot of residuals	625
Post-hoc analysis: mean separation tests	626
Plot of means, confidence intervals, and mean-separation letters	627
Exercises S	628
One-way ANOVA with Random Blocks	631
Packages used in this chapter	631
One-way ANOVA with random blocks example	631
Mixed-effects model with lmer	633
Mixed-effects model with nlme	636
Comparison of results from one-way ANOVA with blocks	639
Two-way ANOVA	639
Packages used in this chapter	640
Two-way ANOVA example	641
Use Summarize to check if cell sizes are balanced	643
Define linear model	643
Conduct analysis of variance	643
Histogram of residuals	644
Post-hoc analysis: mean separation tests	645
Plot of means and confidence intervals for main effects	646
Plot of EM means and confidence intervals for interaction	649
Exercises T	651
Repeated Measures ANOVA	653
Packages used in this chapter	653
Repeated measures ANOVA example	654
Optional analysis: determining autocorrelation in residuals	661
Optional discussion on specifying formulae for repeated measures analysis	662
Specifying random effects in models	662
Indicating time and subject variables	663
Correlation and Linear Regression	664
Packages used in this chapter	665
Examples for correlation and linear regression	665
Visualizing correlated variables	666
Effect size statistics	669
Pearson correlation	670
Kendall correlation	671
Spearman correlation	672
Linear regression	673
Polynomial regression	675
A few of xkcd comics	679
Correlation	679

Extrapolating	679
Cat proximity	680
References	680
Exercises U	680
Advanced Parametric Methods	682
Packages used in this chapter	682
More complex experimental designs	682
Analysis of co-variance	683
Nonlinear regression and curvilinear regression	683
Multiple regression	683
Logistic regression	684
Analysis of count data	684
Robust techniques	684
Example of robust linear regression	685
Contrasts in Linear Models	689
Linear plateau and quadratic plateau models	689
Examples of linear plateau and quadratic plateau models	689
Cate–Nelson analysis	702
Transforming Data	703
Transforming data	704
Packages used in this chapter	704
Example of transforming skewed data	705
Square root transformation	706
Cube root transformation	706
Log transformation	707
Tukey's Ladder of Powers transformation	708
Example of Tukey-transformed data in ANOVA	709
Box–Cox transformation	714
Box–Cox transformation for a single variable	715
Example of Box–Cox transformation for ANOVA model	716
Conclusions	721
Analysis of Count Data and Percentage Data	722
Regression for Count Data	722
Introduction	722
Count data	722
Regression approaches for count data	722
Cautionary note	722
Generalized linear regression	722
Packages used in this chapter	722
Count data example	723
Histograms	724
Poisson regression example	725
Negative binomial regression example	726
Zero-inflated regression example	728
Robust Poisson regression example	730
Quasi-Poisson regression	730
Hermite regression	731
Post-hoc analysis: Medians and confidence intervals	732
Optional code for chi-square goodness-of-fit test	734
Omnibus test	734

Post-hoc chi-square tests	734
Optional analysis: Vuong test to compare Poisson, negative binomial, and zero-inflated models	735
References	737
References for count data	737
Beta Regression for Percent and Proportion Data	738
Introduction	738
Proportion data	738
Beta regression	739
Packages used in this chapter	740
Beta regression example with discrete counts	740
Beta regression example	741
Alternate logistic regression	742
Beta regression example with inherently proportional data	743
Beta regression	746
References	750

Introduction

Purpose of This Book

Goals of this book

The goal of this book is to introduce to students interested in extension education, outreach, and public education to the quantitative methods used to assess the evaluation of these activities. Extension education includes a diverse collection of subject matter, including environmental science, home horticulture, agriculture, youth development, nutrition, and financial literacy.

Tools for evaluating educational programs may include in-class surveys that measure the knowledge gain of students in a course or follow-up surveys to determine the behaviors adopted by course participants. Evaluation may also include any number of measured variables, perhaps the age of youth participants, the number of calories eaten daily by students in a nutrition program, or the organic matter content of farm fields managed by participating farmers.

The examples and methods here are chosen specifically to be applicable to the evaluation of extension education programs. That being said, these methods are some of the most common used in the analysis of experiments—techniques used from diverse disciplines from manufacturing to environmental science to psychology, though each of these disciplines has additional methods used in specific situations.

Specific learning goals

One goal of this book is to give readers the skills and abilities to be able to understand the graphs and statistics that you might encounter in a publication such as the *Journal of Extension* or other academic reports of program results. As examples, students will be able to answer the questions: *What can I conclude from this bar plot? How do I interpret this p-value? What is an r-squared value?*

A second goal is for readers to be able to design and analyze their own program evaluation experiments in order to document the impacts of their extension teaching or research. *What analysis would I use to assess knowledge gain with before-and-after surveys? What statistics should I report to convey the results of this analysis? Can I explain the results with a graph?*

Pre-requisites

This book is written for students at the undergraduate level with no prior knowledge of the analysis of experiments, and with no prior knowledge of computer programming. This being said, students with no background in these areas will need to apply care and dedication in order to understand the material and the computer code used in examples. These students may also need to explore the optional readings to obtain a better foundation in statistical thinking and theory.

What this book will not cover

Designing and conducting surveys

There are many skills and considerations that go into conducting competent assessments of education programs. This book will not cover these many of these topics in any depth. For example, good survey design and effective survey questions will be touched on only very briefly. Conducting surveys well, for example by avoiding sampling bias, will also not be covered in any significant way.

Advanced statistical analyses

There are variety of relatively advanced statistical analyses that are used in even relatively simple studies. This book focuses on only the most basic analyses for common designs used in extension evaluation. A solid understanding of these analyses will give the reader the foundation for exploring more complicated analyses as the student wishes or the situation calls for.

R programming

R is a flexible and powerful programming language. Readers of this book will benefit from learning the basics of programming in R; however, descriptions of R programming will be kept to a minimum here. There are books and online resources available to learn R programming. A few places to start include the book by Roger Peng listed in the “R programming” section and the courses offered by the resources listed in “Online Learning Modules and Massive Open Online Courses (MOOC’s)” section in the *Statistics Textbooks and Other Resources* chapter.

Author of this Book and Version Notes

This book is written with the intention of providing users simple and complete examples of analyses common in the evaluation of extension education programs using the *R Project for Statistical Computing*. I have chosen the computer code to both be as clear as possible for a beginner to follow, but also to demonstrate some of the options available in conducting analyses or producing plots of data. In some cases the code is more difficult to follow than in others.

Likewise, discussion of the assumptions and theoretical considerations for the statistical analyses included here is limited. Readers are encouraged to understand these considerations in more depth when using these analyses.

Author of this book

I am neither a statistician nor an R programmer, so all advice and code in this book comes without guarantee. I am happy to accept corrections or suggestions. Send correspondence to mangiafico@njaes.rutgers.edu.

Version notes

Version 1.11 – Code utilizing the *lsmeans* package has been switched to using the *emmeans* package. This appears to have required only minor changes in the code. If you find something that doesn’t work, please let me know.

Using R

R and RStudio

This book will use the software package *R Project for Statistical Computing* to create plots and conduct statistical analyses. It is free to install on a Windows, Mac, or Linux computer. Although it is not required, I also recommend using *RStudio*, which is also free.

Using the *RStudio* environment

RStudio provides a nice work environment because it presents several windows on the screen that make it easy to view code, results, and plots at once.

Program code can be worked on in the upper left *Script* window. If that window isn't displayed, it can be opened with *File > New File > R Script*. Code in the *Script* window is selected, and the *Run* button is used to run the code. The results are reported in the *Console* window on the lower left. Code can be saved as an *.r* or *.R* file, and those files should subsequently open automatically with *RStudio*.

Output can be copied from the *Console* window as text.

Smaller pieces of code can be typed or pasted directly in the *Console* window.

The lower right window will usually show either plot results or help results. Plots can be saved as png or jpg files.

Using the *R Console* environment

If you are not using *RStudio*, it is advisable to keep code stored in a separate text file, and then code can be pasted as small chunks directly into the *R GUI Console*. Results are produced in the *Console* as code is entered, and plots will open in a separate window.

Installing R and RStudio

Installing R and RStudio

In theory it should be easy to install R and R Studio on your computer. It works with Windows, Mac, and Linux operating systems.

First install R.

cran.r-project.org/

Then install RStudio. Choose the free desktop version.

www.rstudio.com/products/rstudio/download/

You may need to tell RStudio where your R installation is. In RStudio this can be changed with:

Tools > Global Options

From there, you simply start RStudio when you want to use R.

Optional: Problems with library folder location in Windows 10

I had some problems with R on Windows 10 when installing additional packages. The issue is that by default R wants to place installed packages ("library" folder) in the same location where R resides. The problem is that Windows doesn't allow programs to install things in the Program Files folder.

One option is to install R somewhere else on the computer. That is, not in the Program Files folder.

A second option is a temporary fix. With R installed in Program Files, run the following code first whenever you start an R session. You will need to change the path indicated.

```
.libPaths("C:/Users/Mangiafico/R/Library")
.libPaths()
```

The first path listed should be the path you added.

A better fix. With R installed in Program Files, you can tell R where to install packages ("library"). Here I'll try to recount what I did that worked.

- Right click on the shortcut icon that you use for RStudio. *Properties > Start In.* Change it to e.g. *C:\Users\Mangiafico\R*
- Change *RStudio > Tools > Global Options > Default Working Directory*. To e.g. *C:\Users\Mangiafico\R*, or whatever you used in a).
- Run the following code. It will tell you where there are *Rprofile* and *Rprofile.site* files. Delete, or move those to somewhere harmless.

```
candidates <- c( sys.getenv("R_PROFILE"),
               file.path(Sys.getenv("R_HOME"), "etc", "Rprofile.site"),
               Sys.getenv("R_PROFILE_USER"), file.path(getwd(), ".Rprofile") )

Filter(file.exists, candidates)
```

- Download my *.Rprofile* file. Open it with a text editor (*Right click > Open with > Notepad*). Note that it doesn't have a file extension (i.e. it doesn't have a .txt extension). Change the path in there to where you want your library folder to be. Include "library" in the path.

rcompanion.org/documents/rprofile.html

Save it and move it to the directory you used for a) and b).

Do the same with the *Rprofile.site* file. Save this one in *Program Files > R > (Version) > etc* , or wherever the *Rprofile.site* file was listed in c).

You can modify these files if you wish. It's important that you don't add file extensions to their names. The *.First* function lists commands to run when R starts up. I just have it report what is the R home directory, working directory, and where the library folder is stored.

e) Exit out of R and RStudio. Start RStudio as you normally would.

Enter the following in the console.

```
.libPaths()
```

And the first path listed should be the path you added to the *Rprofile* file.

If these don't work, you may need to do some googling and find a solution. If you find a good solution, let the class know.

What if I don't have my own computer?

If you don't have your own computer on which to install the software, there are a few options.

Portable installation on a usb drive

One solution is to install *R Portable* on a portable usb drive. You can then run this software on university or other computers directly from that drive, if the computer is set up to give you permission to do so. You should check to see if you will have permission to run portable software from a usb drive on these computers.

Using university computers

R and *RStudio* are installed on Rutgers University computers in computer laboratories. You should check with the individual computer lab, and make sure you will have permission to install additional R packages on those machines.

Installing R packages on a university computer

If there are issues with installing additional R packages on university computers, perhaps the easiest solution would be to always change the location of the library folder at the start of each session. For example, if you were using a usb flash drive assigned as the *D:* drive:

```
### Set location of library folder
.libPaths("D:/R/library")

### Set location of working directory
setwd("D:/R")

### Check these
```

```
.libPaths()
getwd()
```

The first path listed for `.libPaths()` should be the directory you requested. If it is available, it is where R will install new packages.

These commands should be run at the start of every session.

Using R online

There are websites on which you can run *R* in an online environment. At the time of writing, I like the [rdrr.io Snippets](#) site because it has common packages installed. There, you can paste code into the window, and press the *Run* button.

Tests for package installation

If you are unsure if you can install additional R packages in the environment you are working in, try the two examples below.

The *psych* and *FSA* packages may take a while for their initial installation. The code for you to run is in blue, and the output is in red. The output is truncated here.

Don't worry too much about what the code is doing at this point. The main point is to see if you can get output from both the *psych* and *FSA* packages.

The code assigns a vector of numbers to *Score*, and a vector of text strings to *Student*. It then combines those two into a data frame called *Data*, which is then printed. The *summary* function counts the values in *Student*, and determines the median and other statistics for *Score*. The *psych* package is installed, then loaded with the *library* function, and then is used to output summary statistics for *Score* for each *Student*. The same is then done with the *FSA* package.

Remember to run only the blue code. The red code is the (truncated) output R should produce.

```
Score = c(10, 9, 8, 7, 7, 8, 9, 10, 6, 5, 4, 9, 10, 9, 10)
Student = c("Bugs", "Bugs", "Bugs", "Bugs", "Bugs",
           "Daffy", "Daffy", "Daffy", "Daffy", "Daffy",
           "Taz", "Taz", "Taz", "Taz", "Taz")

Data = data.frame(Student, Score)

Data
```

	Student	Score
1	Bugs	10
2	Bugs	9
3	Bugs	8
4	Bugs	7
5	Bugs	7
6	Daffy	8
7	Daffy	9
8	Daffy	10

```

9   Daffy      6
10  Daffy      5
11   Taz       4
12   Taz      9
13   Taz     10
14   Taz      9
15   Taz     10

```

```
summary(Data)
```

Student	Score
Bugs :5	Min. : 4.000
Daffy:5	1st Qu.: 7.000
Taz :5	Median : 9.000
	Mean : 8.067
	3rd Qu.: 9.500
	Max. :10.000

```
if(!require(psych)){install.packages("psych")}
```

```
library(psych)
```

```
describeBy(x = Score,
           group = Student)
```

group: Bugs	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se	
group: Bugs	1	1	5	8.2	1.3	8	8.2	1.48	7	10	3	0.26	-1.96	0.58

group: Daffy	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se	
group: Daffy	1	1	5	7.6	2.07	8	7.6	2.97	5	10	5	-0.11	-2.03	0.93

group: Taz	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se	
group: Taz	1	1	5	8.4	2.51	9	8.4	1.48	4	10	6	-0.97	-1.04	1.12

```
if(!require(FSA)){install.packages("FSA")}
```

```
library(FSA)
```

```
Summarize(Score ~ Student,
          data=Data)
```

Student	n	mean	sd	min	Q1	median	Q3	max	percZero
1 Bugs	5	8.2	1.303840	7	7	8	9	10	0
2 Daffy	5	7.6	2.073644	5	6	8	9	10	0
3 Taz	5	8.4	2.509980	4	9	9	10	10	0

Required readings

The following readings are required for this chapter. You can read them at the individual links below, or as chapters in the pdf version of the *R Companion to the Handbook of Biological Statistics* (rcompanion.org/documents/RCompanionBioStatistics.pdf).

About R

rcompanion.org/rcompanion/a_04.html

A Few Notes to Get Started with R

rcompanion.org/rcompanion/a_06.html

Avoiding Pitfalls in R

rcompanion.org/rcompanion/a_07.html

Help with R

rcompanion.org/rcompanion/a_08.html

References

Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09.

rcompanion.org/rcompanion/. (Pdf version:

rcompanion.org/documents/RCompanionBioStatistics.pdf.)

Exercises A

1. Install *R* and *RStudio* on your computer, or determine how you will access this software. Be sure that you will be able to install additional R packages (as in “Tests for package installation” above).

If you haven't installed the *FSA* and *psych* packages, do so with the following command, or use *Tools > Install packages* in RStudio.

```
if(!require(FSA)){install.packages("FSA")}
if(!require(psych)){install.packages("psych")}
```

Use the following code to import from the internet a data frame of river stage measurements for Greenwich, NJ.

```
Greenwich = read.table("http://rcompanion.org/documents/Greenwich.csv",
header=TRUE, sep=",")
```

- a. Summarize the stage measurements by year and **report the results**.

```
library(FSA)
Summarize(Stage ~ Year,
data = Greenwich)
```

Examine the variables in the data frame. (***Don't report anything.***)

```
str(Greenwich)
```

```
library(psych)
```

```
headTail(Greenwich)
```

```
summary(Greenwich)
```

The *str* function reports the type of each variable in the data frame. *Agency* is treated as a factor variable by R. We'll consider this a nominal variable. It has one level, which is "USGS". The rest of the variables are being treated as integer or numeric variables.

The function *headTail* in the *psych* package shows you the top and bottom of the data frame, arranged in the usual way with variables in columns and observations in rows.

The *summary* function reports some summary statistics for the data frame. In this case, it reports that there are 171 instances of "USGS" for the variable *Agency*. Since it treats *Year* as an integer variable, it tells you that the minimum is 2000 and the maximum is 2014. It tells us that there are 6 *NA* values in the variable *Stage*.

These functions are useful to see what variables are in a data frame, and to be sure that the data frame reflects the data we think it should. For example, this data frame has 171 observations. If we were expecting 1000 observations, we would know something went wrong in our data entry or importing of the data.

We could create a new variable called *Year.f* that would be the same as *Year*, but treated as a factor variable.

```
Greenwich$Year.f = as.factor(Greenwich$Year)
```

Now when we use the *summary* function, it will report the counts for some levels of *Year.f*.

```
summary(Greenwich)
```

b. In future chapters, we will learn to summarize data in more specific ways. In this case, if we want the counts for all the years in *Year.f*, we can do the following. ***Report this result.***

```
xtabs(~ Year.f,
      data = Greenwich)
```

2. The following will create a data frame called *TwoCats* with Pepé's and Penelope's scores from a talent show, and produce summary statistics for Penelope's scores.

```
Input = "
Cat          Score
```

```
'Pepé Le Pew'    8
Penelope        10
Penelope        9
Penelope        9
Penelope        8
Penelope        7
Penelope        7
")
TwoCats = read.table(textConnection(Input), header=TRUE)
library(FSA)
Summarize(Score ~ Cat,
           data = TwoCats,
           exclude = "Pepé Le Pew")
```

a. ***Report the results*** for Penelope.

Change Pepé's scores to 9, 9, 7, 7, 6, 5. Change the *exclude* option in the *Summarize* function to exclude only Penelope. Penelope's name will need to be in simple double quotes.

b. ***Report the results*** for the summary statistics for Pepé's new scores. (The new mean for Pepe should be 7.1667.)

c. Summarize the data below with the *Summarize* function in the *FSA* package. ***Report the results***.

To do this,

- Copy the data below and paste it into to code above. Make sure you retain the *Input* line and the *")* line.

Dog	Snacks
Scooby	4
Scooby	3
Scooby	6
Scooby	18
Scooby	7
Scrappy	8
Scrappy	10
Scrappy	6
Scrappy	5
Scrappy	15
Scrappy	7
Scrappy	9

- Get rid of the *exclude* option in the *Summarize* call. Pay attention to the placement of the commas and parentheses, so it will look like this:

```
Summarize(Score ~ Cat,
          data = TwoCats)
```

- In the code, change *TwoCats* to *TwoDogs*.
- In the *Summarize* function call, change *TwoCats* to *TwoDogs*, change *Score* to *Snacks*, change *Cat* to *Dog*.

You should get summary statistics for both Scooby and Scrappy in one output. The number of observations (*n*) for Scooby should be 5 and for Scrappy should be 7. The mean snacks for Scooby should be 7.6 and the mean snacks for Scrappy should be 8.57.

Exercises Alpha

1. The package *FSA* contains a data set called *ChinookArg* that has the lengths and weights of Chinook salmon at three locations.

```
library(FSA)
data(ChinookArg)
```

We can find some information about the data set with

```
?ChinookArg
```

- a. **Report**, From what county are these data?

We can see the first and last rows of the data with

```
library(psych)
headTail(ChinookArg)
```

Note that the variable *tl* is length, *w* is weight, and *loc* is location. This is explained in the help file pulled up with

```
?ChinookArg
```

We can then get some summary statistics about the data.

```
library(FSA)
Summarize(tl ~ loc,
          data = ChinookArg)
```

This will provide summary statistics about length at each location.

In the output, n is the number of observations for that group. The other summary statistics are mean, standard deviation, minimum, 1st quartile, median, 3rd quartile, and maximum. These statistics will be discussed in a later chapter.

This example uses formula notation, where tl is measurement variable and loc is the grouping variable, and they are separated with a tilde. We could also think of tl as the dependent variable and loc as the independent variable.

Some functions accept formula notation and some do not. Usually asking for help about the function will help you determine what input is required and what other arguments can be passed to the function, e.g.

`?Summarize`

Answer the following:

- b. What is the number of observations for Petrohue?
 - c . What is the mean length for Chinook in Petrohue?
 - d. The minimum length in Petrohue?
 - e. The maximum length in Petrohue?
2. Install the *ggplot2* package with following command, or use *Tools > Install packages* in RStudio.

`if(!require(ggplot2)){install.packages("ggplot2")}`

The following code will plot Chinook length vs. weight. It will add a smooth line to the plot.

```
library(ggplot2)

qplot(x      = w,
      y      = tl,
      data  = ChinookArg,
      geom  = c("point", "smooth"),
      xlab  = "Weight (kg)",
      ylab  = "Length (cm)",
      main  = "Chinook plot by Sal")
```

- a. In the code above, change "Sal" to your name. ***Export the plot and embed it in your assignment.***

In RStudio, in the *Plot* window, you can try the *Export* menu, and *save as image*, *save as pdf*, or *copy to clipboard*.

For assignments, probably the easiest thing is to use *Export, save as image*. The recommended format is .png. Pay attention to the directory where your image was saved.

For high resolution images, I tend to use the .pdf option and then edit the file with Photoshop or GIMP.

If you don't use RStudio, or would like to specify the size and resolution of image files, you can use

```
png(filename = "Rplot03d.png",
    width  = 4,
    height = 3,
    units   = "in",
    res     = 600)

qplot(x    = w,
      y    = tl,
      data = ChinookArg,
      geom = c("point", "smooth"),
      xlab = "Weight (kg)",
      ylab = "Length (cm)",
      main = "Chinook plot by sal")

dev.off()
```

You may use the following to see where the file was saved, if you didn't specify a path in the *filename* argument.

```
getwd()
```

3. In the *qplot* function above, switch the variable for weight, *w*, with the variable for location, *loc*. Remove the whole line with the *geom* argument. And change the *xlab* argument to something appropriate.

a. ***Export the plot and embed it in your assignment.***

4. There's another data set in the *FSA* package called *WhitefishLC*.

Using the command

```
?WhitefishLC
```

Answer the following lettered questions.

a. What are the units for fish length in this data set?

Using this data set, summarize the length of whitefish (*tl*) by their age (*scale1*).

Use the code for *Summarize* that you used in 1.a. Make sure you change the names of the variables in the function and the name of the data frame in the function.

- b. Report the results.
- c. Do you observe anything interesting in these results?

Plot length vs. age for whitefish.

Use the code you used in 2. Be sure to change the name of the variables and of the data frame.

- d Embed this plot with your assignment. Include appropriate axis labels and be sure the units of length are correct. If you include a title, be sure it is appropriate.
- e. How would you describe the relationship of length and age for these fish?

Statistics Textbooks and Other Resources

This book is not intended to be a substitute for an introductory course or text in statistics. The introductory chapters will briefly cover certain key concepts mostly through readings and videos from external sources. Other concepts that are important to having an appreciation for the theory and use of statistical tests are not addressed. For this understanding, readers are encouraged to pursue an introductory undergraduate course, appropriate MOOC, or careful reading of an introductory textbook.

Some free general statistics books

Luckily there are free resources available that cover the key concepts in an introductory statistics course. Some of these will be suggested as recommended reading in this book.

Diez, D.M., C.D. Barr, and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.
Types of data, plots, experimental design, sampling, probability, hypothesis testing, confidence limits, t-test, analysis of variance, chi-square test, linear regression, multiple regression, logistic regression.

Openstax College. 2013. *Introductory Statistics*. Rice University.
openstaxcollege.org/textbooks/introductory-statistics.

Sampling, descriptive statistics, probability, distributions, confidence intervals, hypothesis testing, Type I and Type II errors, t-test, chi-square, linear regression, analysis of variance.

Lane, D.M. (ed.). No date. *Introduction to Statistics*, v. 2.0. onlinestatbook.com/.

Descriptive statistics, plots, correlation, probability, experimental design, confidence intervals, type I and type II errors, t-test, regression, analysis of variance, chi-square test, nonparametric tests, power.

Stockburger, D.W. 2013. *Introductory Statistics: Concepts, Models, and Applications*, 3rd web edition. www.psychstat.missouristate.edu/IntroBook3/sbk.htm.

Distributions, regression, correlation, hypothesis testing, t-test, chi-square test, analysis of variance.

Books on statistical analyses and tests

McDonald, J.H. 2014. *Handbook of Biological Statistics*, 3rd edition.

www.biostathandbook.com/index.html, www.biostathandbook.com/HandbookBioStatThird.pdf

Data analysis steps, kinds of variables, probability, hypothesis testing, confounding variables, descriptive statistics, chi-square test, t-test, analysis of variance, nonparametric tests, linear regression, logistic regression. Analyses in SAS programming language.

Analyses in R

Online resources

Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*.

rcompanion.org/rcompanion/, rcompanion.org/documents/RCompanionBioStatistics.pdf.

Using R, descriptive statistics, chi-square test, t-test, analysis of variance, nonparametric tests, linear regression, logistic regression. Analyses in R programming language.

Kabacoff, R. 2014. *Quick-R: Accessing the Power of R*. www.statmethods.net/

Data input, data management, basic statistics, advanced statistics, basic graphs, advanced graphs.

Venables, W.N., D.M. Smith, and R Core Team. 2015. An Introduction to R. cran.r-project.org/doc/manuals/r-release/R-intro.pdf.

Text books

Crawley, M.J. 2012. *The R Book*. ISBN: 978-0-470-97392-9.

Videos

Joseph, M.B. *The IQUIT R video series*. www.r-bloggers.com/the-iquit-r-video-series/ or www.youtube.com/channel/UCXSzyk3li4bh9ODis_B8daQ.

R programming

Peng, R.D. 2015. R Programming for Data Science. leanpub.com/rprogramming.

Other useful resources for understanding statistics

Videos

Statistics Learning Centre (Videos with Dr. Nic). YouTube channel.

www.youtube.com/channel/UCG32MfGLit1pcqCRXyy9cAg.

Blogs

Learn and Teach Statistics and Operations Research (Articles by Dr. Nic).

learnandteachstatistics.wordpress.com/summary/.

Online Learning Modules and Massive Open Online Courses (MOOC's)

Online learning modules and online courses offer another alternative to learning basic statistical concepts, programming in R, or advanced statistical techniques.

Swirl

swirlstats.com/

Swirl has free, brief lessons for beginners to R programming and analysis. Topics include R programming, data analysis, data visualization, data manipulation, statistical inference.

DataCamp

www.datacamp.com/

DataCamp has short online modules covering topics such as an introduction to R, data manipulation, data visualization, and statistics with R. Limited parts of several courses are free. For the complete courses, there is a subscription charge. There is a discount for students.

Coursera

www.coursera.org/

Coursera hosts MOOC's from various universities on a range of academic topics, including introductory statistics using R. Courses are free, though there is an option to pay for a certificate in a data science specialization.

edX

www.edx.org/

edX also hosts free MOOC's from universities on a variety of topics, including a course on introduction to R programming.

Udemy

www.udemy.com/

Udemy hosts relatively inexpensive online courses on a variety of subjects, including programming and statistics with R. Some are free.

Why Statistics?

Statistics is a general term. Every time you report a mean, median, or standard deviation of your data, you are reporting a statistic. Specifically, this type of statistics are called *descriptive statistics*. We can also use statistical tests and report the statistics from these tests, such as *p*-values or *r-squared* values. In order to convey the implications of these statistics and the underlying data, we might present them in a plot or table.

The importance of statistics in program evaluation

There are several reasons why we want to conduct statistical analyses when looking at our program evaluation data.

Communication

Descriptive statistics and statistical tests can be used to communicate with our colleagues. Once we understand statistical tests—as well common statistics such as *p*-values, confidence intervals, and *r-squared* values—we can communicate our results in a shared language. A single well designed plot with appropriate statistics is an effective tool to convey our findings.

Communication is a two-way street. We also need to be able to understand others when they share their results with us, whether it's in a departmental presentation, a presentation at a professional meeting, in extension literature, or in a peer-reviewed journal article.

Adding rigor

Sometimes educators will simply report the change in median scores before and after a course, or report a best fit line for bivariate data. However, without applying a statistical tests and reporting *p*-values or other appropriate statistics, it is not clear to the reader (or probably the author!) if the reported effect is real.

Getting things right

In order for your statistical analyses to provide correct results, you need to understand when they are appropriate and what the results mean. Also you will want to understand what tests and techniques are available in order to best be able to understand and explain your results.

Presenting results well

Knowledge of statistical tests and plots can inform the best ways to summarize and present results from your data.

Academic recognition

Correct and accepted statistical methods are usually required to get results published in academic journals or proceedings from professional meetings.

Program evaluation

Appropriate statistics help to evaluate programs, for example determining if there was an increase in student knowledge scores was statistically significant or if one teaching technique is better than another. Such information can guide future programming efforts.

Applicability of statistical tests to other fields and situations

The statistical techniques presented in this book are applicable to a wide variety of disciplines, and are some of the most common used across fields. While the analyses presented in this book are common and relatively simple, understanding of these techniques will serve as a basis for more advanced analyses.

Evaluation Tools and Surveys

A complete discussion of the proper design and administration of program evaluation tools and surveys is far beyond the scope of this book. This chapter is intended only to give the reader some context for the remainder of the book, which discusses the analysis of data from these survey tools and other sources.

Extension programs

An extension program is an organized educational endeavor delivered to some segment of the public, such as farmers, students in 4-H, or Master Gardeners. It is based on an *assessments of needs* for that group for the specific topic of the program, and is based on *objectives* for what knowledge students should improve, skills they should learn, or behaviors they should adopt.

An extension program will have multiple activities associated with these goals. The extension educator may be organizing educational events, delivering lectures, writing fact sheets, or reporting on research.

Each of these components is *evaluated* in light of the objectives of the program and the objectives of the individual activities. In each case objectives should be *measureable*.

In some cases, the evaluator can collect “hard data” about the results of some program: number of attendees, number of downloads, measured differences in water quality.

In other cases evaluation tools such as surveys will be developed to evaluate how well program objectives will be met. These might ask participants about an increase in their knowledge, a change in their behavior, or adoption of specific practices, along with other relevant information.

Designing a questionnaire

There is much to be learned about good questionnaire design, and good practices for conducting a survey. This chapter will only scratch the surface of this topic.

Some principles of questionnaire design for program evaluation

Length and audience-appropriateness

Your questionnaire should be short enough and simple enough so that your respondents will be likely to complete it, and complete it thoughtfully.

On the other hand, it should be long enough to gain valuable information. For example, it might not want to ask simply, "Did you learn something?", but ask how much was learned in each topic area. Identifying areas where teaching was successful or not successful helps to improve the program in the future.

It is also important to take into account your audience. Young children may be able to answer only a few questions, simple in scope, and without too many options. On the other end of the spectrum, expert audiences can much more critically assess programs and program component. Non-expert adult audiences fall in the middle of this spectrum: They are capable of assessing their own changes in knowledge, skills, attitudes, behaviors, and implemented practices. Still, questionnaires for them should still be short and clear, with the possibility of some open-ended questions.

Types of impacts

A questionnaire may assess a single type of impact or several types depending on what is appropriate. It may seek to assess knowledge gain, changes in attitude, gained skills, changes in behavior.

These can be assessed for an extension activity as short as single lecture, a day-long workshop, or after a program lasting several years.

Assessment techniques of a questionnaire

Knowledge gain: before-and-after tests

For knowledge gain questions, respondents could be tested in a before-and-after manner, where some specific question is asked on a pre-test before the course, and then on a post-test after the course.

It is helpful if the same respondent can be identified on both tests, so that the before responses and after responses can be paired for each respondent.

Personally, I find before-and-after assessment rather patronizing and annoying for adult audiences. Instead, I prefer a self-assessment knowledge gain format, discussed below.

As an example, [Bakacs et al. \(2013\)](#) on Table 1, reported an increase in knowledge by respondents for individual questions about using rain barrels, conserving water, and reducing stormwater runoff. For example, they asked workshop attendees the following question, both before and after the workshop. They asked respondents for their initials and favorite number for each test so that responses could be paired for each respondent.

10. A 55 gallon rain barrel, when full will weigh approximately
- 200 lbs.
 - 300 lbs.
 - 400 lbs.
 - Not sure

Knowledge gain: self-assessment questions

Another approach for assessing knowledge gain after a program or event is to allow respondents to assess their own knowledge gain. This allows self-reflection, and may therefore be more meaningful than looking at correct and incorrect answers on before-and-after tests.

One form of self-assessment question asks the respondent to assess their knowledge on a topic both before and after the course, but is answered only once, after the course.

As an example, [Mangiafico et al. \(2011\)](#) (Table1) reported self-assessment changes in knowledge gain by respondents for individual questions about environmentally-friendly lawn care. The survey was done following a lecture, so that respondents reflected about their knowledge before and after the lecture from the perspective of having just listened to the lecture.

Before		After
1 2 3 4 5	I understand how to determine how much phosphorus should be applied to my lawn.	1 2 3 4 5

Behavior change and adopted practices

One way to approach assessing changes in behavior is to use follow-up surveys (discussed below) at some time after a program or workshop to see if participants adopted practices or behaviors.

A second method is to question participants at the conclusion of an event concerning their *anticipated* changes in behaviors based on what they learned at the event. The obvious drawback to this approach is that respondents are likely to *anticipate* that they will improve their behaviors, and this anticipation may never materialize in reality. However, this method is often used because it is easier and allows collecting data from a captive audience.

As an example, [Mangiafico et al. \(2011\)](#) (Table2) reported self-assessment anticipated behaviors concerning environmentally-friendly lawn care.

17. I will test my soil or have my soil tested to determine the need to adjust pH and apply phosphorus fertilizer.

disagree	neutral	agree
1 2 3 4 5 6 7 8 9 10	Don't know	Not applicable

Follow-up surveys

Follow-up surveys are conducted at some time after a course or event, and are useful to assess if specific practices were implemented or certain behaviors were changed, to give some time reflection by respondents to consider changes in attitudes or knowledge.

As an example, [Bakacs et al. \(2013\)](#) on Table 3, used a follow-up survey to determine if workshop participants installed rain barrels at their homes or businesses after building one at the workshop.

The follow-up survey in New Jersey was e-mailed to participants three to six months after the workshops, and were completed online.

One drawback to this approach is that it is necessary to collect contact information from participants and take the time to re-survey participants. Another problem is that response rates may be lower than response rates from in-classroom assessments.

Rating of course quality and materials

It is often useful to have program participants rate the quality of instruction, presentation, visual aids, handouts, etc.

Open-ended questions

Open-ended questions can be very valuable to solicit critical comments about a course or program, or assess knowledge gain or other impacts that aren't covered by other assessment questions.

Opened-ended questions can attempt to solicit critical comments—"What could be done to improve this program?"; assess knowledge gain—"What did you learn from this program?"; or gain other information—"Where did you learn about this course?".

These questions can be very valuable, but you should also be cautious since responses may be sparse and not representative of program participants. In particular, critical responses may be unrepresentatively positive or negative. In general, participants may or may not put a lot effort into answering open-ended questions, depending on a variety of factors not necessarily related to program content or quality.

Demographic data

Finally, collecting demographic data of questionnaire respondents can be useful for interpreting and understanding responses for other questions. Demographic data may include profession or stakeholder group; current practices; or age, grade, sex, etc.

Forms of responses

Answers to questions may be in the form of yes/no, multiple choice, ordered response (Likert), or open-ended.

The statistical analysis of these different forms of answers will vary. This will be unpacked over the course of this book.

Examples of evaluation tools

Rutgers Cooperative Extension Program Evaluation, Youth Audience, GRK-3njaes.rutgers.edu/evaluation/youth/docs/RCEYOUTHEVALUATION_GRK-3_FORM.pdf

Notes:

- As a questionnaire written for younger children, most responses to questions are 3- to 5-point Likert items with face emoticons for support, and yes/no questions.
- Questions 1 and 2 are self-assessment knowledge gain. 3-point Likert items.
- Question 3 is anticipated behavior change. Yes/no.
- Questions 5 and 6 are rating of the instructor and program. 5-point Likert items.
- Question 7 is a question about demographics.
- Question 8 is an open-ended question.

Rutgers Cooperative Extension Turf Management for a Healthier Lawn, Program Evaluation Formrcompanion.org/documents/TurfProgramEvaluation.pdf

Notes:

- Questions 1–11 are self-assessment questions of knowledge gain and attitudes, using before-and-after in a single sitting. 5-point Likert items.
- Questions 12–13 are ratings of information and educational materials. 10-point Likert items, with two opt-out options (*Don't know*, and *Not applicable*)
- Questions 17 – 20 are anticipated behaviors. 10-point Likert items, with two opt-out options (*Don't know*, and *Not applicable*)

Rutgers Cooperative Extension Program Evaluation, Older Youth Audience, GR4-13Anjaes.rutgers.edu/evaluation/youth/docs/RCEYOUTHEVALUATION_GR4-13A_FORM.pdf

Notes:

- Questions 1–2 are open-ended questions assessing knowledge gain and behavior change.
- Questions 4–5 are ratings of presenter and program. 5-point Likert items.
- Question 6 is a question about demographics.
- Question 7 is an open-ended question.

Rutgers Cooperative Extension Program Evaluation - Older Youth Audience, GR4-13Bnjaes.rutgers.edu/evaluation/youth/docs/RCEYOUTHEVALUATION_GR4-13B_FORM.pdf

Notes:

- Questions 1–3 are self-assessment questions of knowledge gain, using before-and-after in a single sitting. 4-point Likert items.
- Question 4 is an open-ended question about behaviors.
- Questions 6–7 are ratings of presenter and program. 5-point Likert items.
- Question 8 is a question about demographics.
- Question 9 is an open-ended question.

Optional resource

Poling, R. L. 1999. Example Extension Program Evaluation Tools. Agricultural Extension Service Institute of Agriculture University of Tennessee. web.utk.edu/~aee/evaltools.pdf.

Optional Readings

[Video] Statistics Learning Center (Dr. Nic). 2011. "Designing a Questionnaire".

www.youtube.com/watch?v=FkX-t0Pgzzs.

Optional additional resources

Barkman, S. J. 2001. A Field Guide to Designing Quantitative Instruments to Measure Program Impact. Purdue University.

www.northskynonprofitnetwork.org/sites/default/files/documents/Field%20Guide%20to%20Developing%20Quantitative%20Instruments.pdf.

Chavez, C. No date. Survey Design. Loyola Marymount University.

www.lmu.edu/Assets/Academic+Affairs+Division/Assessment+and+Data+Analysis/Christine%27s+Former/Surveys+Website/Survey+Design+Resource.pdf.

Rutgers Cooperative Extension. 2015. Program Evaluation. njaes.rutgers.edu/evaluation/.

Walnick, D.S. 2010. Designing and Using Questionnaires. StatPac. statpac.com/surveys/surveys.pdf.

References for this chapter

Bakacs, M., M. Haberland, S.S. Mangiafico, A. Winquist, C.C Obropta, A. Boyajian A., and S. Mellor. 2013. Rain Barrels: A Catalyst for Change? Journal of Extension 51(3), article 3RIB6.

www.joe.org/joe/2013june/rb6.php

Mangiafico, S.S., C.C. Obropta, and E. Rossi-Griffin. 2011. A Lawn Care Education Program to Address Water Conservation and Water Pollution Prevention in New Jersey. Journal of the National Association of County Agricultural Agents 4(2). www.nacaa.com/journal/index.php?jid=108.

Acknowledgements

I would like to thank Dan Kluchinski of Rutgers Cooperative Extension and the Rutgers Department of Agricultural and Resource Management Agents for supplying some resources for this chapter.

Types of Variables

Organizing data: observations and variables

In general, collected raw data is organized according to *observations* and *variables*. Variables represent a single measurement or characteristic for each observation. In the Statistics Learning Center video in the *Required Readings* below, Dr. Nic gives an example of a survey where each observation is a separate person, and the variables are age, sex, and chocolate preference for each person.

In education evaluation, observations will commonly be a student or an instructor, but could also be any other experimental unit, such as a single farmer's field.

In reality, a single observation may be more specific. For example, the rating for one instructor by one student on one date. For example, here is part of the example from the *Friedman Test* chapter with some additional hypothetical data.

Each row represents an observation. So each observation contains the rating by a single student on a single date for an instructor. The variables are *Instructor*, *Date*, *Student*, and *Rating*.

<u>Instructor</u>	<u>Date</u>	<u>Student</u>	<u>Rating</u>
Bob Belcher	2015-01-01	a	4
Bob Belcher	2015-01-01	b	5
Bob Belcher	2015-01-01	c	4
Bob Belcher	2015-01-01	d	6
Bob Belcher	2015-02-05	e	6
Bob Belcher	2015-02-05	f	6
Bob Belcher	2015-02-05	g	10
Bob Belcher	2015-02-05	h	6
Linda Belcher	2015-01-01	a	8
Linda Belcher	2015-01-01	b	6
Linda Belcher	2015-01-01	c	8
Linda Belcher	2015-01-01	d	8
Linda Belcher	2015-02-05	e	8
Linda Belcher	2015-02-05	f	7
Linda Belcher	2015-02-05	g	10
Linda Belcher	2015-02-05	h	9

Long-format and wide-format data

The example above has ratings for each instructor on each of two dates and by each of eight students. Because each observation has just one rating value, this data is in *long format*.

In general, it is best to keep data in long format for summary and analyses.

However, to conduct certain analyses in *R*, or to produce certain plots, data will need to be in wide format. The following is an example of the same data, translated into wide format, with a focus on the ratings by date, and in this case ignoring student.

<u>Instructor</u>	<u>Rating</u>	
	<u>2015-01-01</u>	<u>2015-02-05</u>
Bob Belcher	4	6
Bob Belcher	5	6
Bob Belcher	4	10
Bob Belcher	6	6
Linda Belcher	8	8
Linda Belcher	6	7
Linda Belcher	8	10
Linda Belcher	8	9

Types of variables

The most common variables used in data analysis can be classified as one of three types of variables: nominal, ordinal, and interval/ratio.

Understanding the differences in these types of variables is critical, since the variable type will determine which statistical analysis will be valid for that data. In addition, the way we summarize data with statistics and plots will be determined by the variable type.

Nominal data

Nominal variables are data whose levels are labels or descriptions, and which cannot be ordered. Examples of nominal variables are *sex*, *school*, and yes/no questions. They are also called “nominal categorical” or “qualitative” variables, and the levels of a variable are sometimes called “classes” or “groups”.

The levels of categorical variables cannot be ordered. For the variable *sex*, it makes no sense to try to put the levels “female”, “male”, and “other” in any numerical order. If levels are numbered for convenience, the numbers are arbitrary, and the variable can’t be treated as a numeric variable.

Ordinal data

Ordinal variables can be ordered, or ranked in logical order, but the interval between levels of the variables are not necessarily known. Subjective measurements are often ordinal variables. One example would be having people rank four items by preference in order from one to four. A different example would be having people assess several items based on a Likert ranking scale: “On a scale of one to five, do you agree or disagree with this statement?” A third example is level of education for adults, considering for example “less than high school”, “high school”, “associate’s degree”, etc.

Critically, in each case we can order the responses: My first favorite salad dressing is better than second favorite, which is better than my third favorite. But we cannot know if the interval between the levels is equal. For example, the distance between your favorite salad dressing and your second favorite salad dressing may be small, where there may be a large gap between your second and third choices.

We can logically assign numbers to levels of an ordinal variable, and can treat them in order, but shouldn't treat them as numeric: "strongly agree" and "neutral" may not average out to an "agree."

For the purposes of this book, we will consider such Likert item data to be ordinal data under most circumstances.

Ordinal data is sometimes called "ordered categorical".

Interval/ratio data

Interval/ratio variables are measured or counted values: *age, height, weight, number of students*. The interval between numbers is known to be equal: the interval between one kilogram and two kilograms is the same as between three kilograms and four kilograms.

Interval/ratio data are also called "quantitative" data.

Discrete and continuous variables

A further division of interval/ratio data is between *discrete* variables, whose values are necessarily whole numbers or other discrete values, such as population or counts of items. *Continuous* variables can take on any value within an interval, and so can be expressed as decimals. They are often measured quantities. For example, in theory a weight could be measured as 1 kg, 1.01 kg, or 1.009 kg, and so on. Age could also be considered a continuous variable, though we often treat it as a discrete variable, by rounding it to the most recent birthday.

Optional technical note

There is a technical difference between interval and ratio data. For interval data, the interval between measurements is the same, but ratio between measurements is not known. A common case of this is temperature measured in degrees Fahrenheit. The difference between 5° F and 10° F is the same as that between 10° F and 15° F . But 10° F is **not** twice 5° F . This is because the definition of 0° F is arbitrary. 0° F does not equal 0° C .

Measurements where there is a natural zero, such as length or height, or where a zero can be honestly defined, such as time since an event, are considered ratio data.

For the most part, ratio and interval data are considered together. In general, just be careful not to make senseless statements with interval data, such as saying, "The mean temperature in Greenhouse 1 was twice the mean temperature of Greenhouse 2."

Levels of measurement

In general it is advantageous to treat variables as the highest level of measurement for which they qualify. That is, we **could** treat education level as a categorical variable, but usually we will want to treat it as an ordinal variable. This is because treating it as an ordinal variable retains more of the information carried in the data. If we were to reduce it to a categorical variable, we would lose the order of the levels of the variable. By using a higher level of measurement, we will have more options in the way we analyze, summarize, and present data.

This being said, there may be cases when it is advantageous to treat ordinal or count data as categorical. One case is if there are few levels of the variable, or if it makes sense to condense the variable into a couple of broad categories. Another example of when we choose a lower level of measurement is when we use nonparametric statistical analyses which treat interval/ratio data as ordinal, or ranked, data.

Types of Variables in R

R does not use the terms *nominal*, *ordinal*, and *interval/ratio* for types of variables.

In R, nominal variables can be coded as variables with *factor* or *character* classes.

```
Colors = c("Red", "Green", "Blue")
class(Colors)
[1] "character"

Colors.f = factor("Red", "Green", "Blue")
class(Colors.f)
[1] "factor"
```

Interval/ratio data can be coded as variables with *numeric* or *integer* classes. An *L* used with values to tell R to store the data as an integer class.

```
BugCount = c(1, 2, 3, 4, 5)
class(BugCount)
[1] "numeric"

BugCount.int = c(1L, 2L, 3L, 4L, 5L)
class(BugCount.int)
[1] "integer"
```

We can code ordinal data as either numeric or factor variables, depending on how we will be summarizing, plotting, and analyzing it.

Note that by default *read.table* will read text input as a factor variable. Also, *read.table* will also determine if a numeric variable should have an integer or numeric class.

```
Dragons = read.table(header=TRUE, text=
  Tribe      Length.m  SizeRank
```

```

Icewings    6.4      1
Mudwings    6.1      2
Seawings    5.8      3
Skywings    5.5      4
Nightwings  5.2      5
Rainwings   4.9      6
Sandwings   4.6      7
")

str(Dragons)

'data.frame':    7 obs. of  3 variables:
$ Tribe     : Factor w/ 7 levels "Icewings","Mudwings",...: 1 2 6 7 3 4 5
$ Length.m: num  6.4 6.1 5.8 5.5 5.2 4.9 4.6
$ SizeRank: int  1 2 3 4 5 6 7

```

We can convert the variable *SizeRank* into an ordered factor variable. This will help us with some types of summary and analysis, and is helpful in determining the order that the levels of a factor variable will be plotted.

```

Dragons$SizeRank.f = factor(Dragons$SizeRank,
                           ordered = TRUE,
                           levels = c("1", "2", "3", "4", "5", "6", "7"))

str(Dragons)

'data.frame':    7 obs. of  4 variables:
$ Tribe     : Factor w/ 7 levels "Icewings","Mudwings",...: 1 2 6 7 3 4 5
$ Length.m: num  6.4 6.1 5.8 5.5 5.2 4.9 4.6
$ SizeRank: int  1 2 3 4 5 6 7
$ SizeRank.f: Ord.factor w/ 7 levels "1"<"2"<"3"<"4"<..: 1 2 3 4 5 6 7

sapply(Dragons, class)

`$`Tribe` 
[1] "factor"

$Length.m 
[1] "numeric"

$SizeRank 
[1] "integer"

$SizeRank.f
[1] "ordered" "factor"

```

References

Rouxzee. 2014. Wings of Fire: How Big are the Dragons? www.deviantart.com/rouxzee/art/Wings-of-Fire-How-Big-are-the-Dragons-UPDATED-471608178.

Required readings

[Video] “*Types of Data: Nominal, Ordinal, Interval/Ratio*” from Statistics Learning Center (Dr. Nic). 2011. www.youtube.com/watch?v=hZxnzfnt5v8.

Optional readings

“*Types of biological variables*” in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/variabletypes.html.

“*Frequency, frequency tables, and levels of measurement*”, *Chapter 1.3* in Openstax College. 2013. *Introductory Statistics*. Rice University. openstaxcollege.org/textbooks/introductory-statistics.

“*Data basics*”, *Chapter 1.2* in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

Exercises B

1. For the following variables and levels identify the variable as nominal, ordinal, or interval/ratio.
 - a. Political affiliation (very liberal, liberal, independent, conservative, very conservative)
 - b. Political affiliation (Democrat, Republican, Green, Libertarian)
 - c. Student age
 - d. Favorite school subject (Science, History, Math, Physical Education, English Literature)
 - e. How often do you complete your homework (never, sometimes, often, always)
 - f. Gender identity (1-female, 2-male, 3-other)
 - g. Level of education (1-elementary school, 2-junior high school, 3-high school, 4-college)
 - h. Dairy cow weight
 - i. Number of meals and snacks eaten in a day
 - j. (Yes, No)
 - k. (Yes, No, Not applicable)
 - l. Favorite salad dressing (1st, 2nd, 3rd, 4th)
 - m. Favorite salad dressing (Italian, French, Caesar, etc.)
2. Are **each** of the following terms associated with nominal, ordinal, or interval/ratio data.
 - a. Likert
 - b. Categorical
 - c. Continuous
 - d. Qualitative
 - e. Discrete
3. For the following table, identify:
 - a. Number of observations
 - b. Number of variables
 - c. Type of each variable (nominal, ordinal, or interval/ratio)

<u>Student</u>	<u>Grade</u>	<u>Age</u>	<u>Sex</u>	<u>Height</u>	<u>Calories eaten</u>	<u>Attitude</u>	<u>Class rank</u>
A	5	10	M	137	2000	5	4
B	5	11	M	140	1500	4	2
C	4	9	F	120	1200	3	5
D	4	10	F	140	1400	4	1
E	6	12	Other	147	1800	5	6
F	5	10	M	135	1600	4	8
G	4	9	M	130	1200	4	3
H	4	10	F	140	1800	3	7

4. For the following table, identify:

- a. Number of observations
- b. Number of variables
- c. Type of each variable (nominal, ordinal, or interval/ratio)

<u>Farm</u>	<u>Town</u>	<u>Acreage</u>	<u>Product</u>	<u>Sustainability</u>	<u>Sustainability code</u>	<u>Preserved</u>
A	Skara.Brae	21.1	Turnip	Very.unsustainable	1	Yes
B	Arboria	7.0	Parsnip	Sustainable	4	Yes
C	Lucencia	32.5	Rutabaga	Unsusatainable	2	No
D	Tenebrosia	21.0	Daikon	Very.sustainable	5	No
E	Tarmitia	6.3	Yucca	Sustainable	4	Unknown
F	Malefia	18.1	Kholrabi	Neutral	3	Yes

Descriptive Statistics

Packages used in this chapter

The packages used in this chapter include:

- psych
- DescTools
- Rmisc
- FSA
- plyr
- boot

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(Rmisc)){install.packages("Rmisc")}
if(!require(FSA)){install.packages("FSA")}
if(!require(plyr)){install.packages("plyr")}
if(!require(boot)){install.packages("boot")}
```

Descriptive statistics

Descriptive statistics are used to summarize data in a way that provides insight into the information contained in the data. This might include examining the mean or median of numeric data or the frequency of observations for nominal data. Plots can be created that show the data and indicating summary statistics.

Choosing which summary statistics are appropriate depend on the type of variable being examined. Different statistics should be used for interval/ratio, ordinal, and nominal data.

In describing or examining data, you will typically be concerned with measures of location, variation, and shape.

Location is also called *central tendency*. It is a measure of the values of the data. For example, are the values close to 10 or 100 or 1000? Measures of location include mean and median, as well as somewhat more exotic statistics like M-estimators or Winsorized means.

Variation is also called *dispersion*. It is a measure of how far the data points lie from one another. Common statistics include standard deviation and coefficient of variation. For data that aren't normally-distributed, percentiles or the interquartile range might be used.

Shape refers to the distribution of values. The best tools to evaluate the shape of data are histograms and related plots. Statistics include skewness and kurtosis, though they are less useful than visual inspection. We can describe data shape as normally-distributed, log-normal, uniform, skewed, bi-modal, and others.

Descriptive statistics for interval/ratio data

For this example, imagine that Ren and Stimpy have each held eight workshops educating the public about water conservation at home. They are interested in how many people showed up to the workshops.

Because the data are housed in a data frame, we can use the convention *Data\$Attendees* to access the variable *Attendees* within the data frame *Data*.

```
Input = "
Instructor  Location  Attendees
Ren          North      7
Ren          North     22
Ren          North      6
Ren          North     15
Ren          South     12
Ren          South     13
Ren          South     14
Ren          South     16
Stimpy        North    18
Stimpy        North    17
Stimpy        North    15
Stimpy        North     9"
```

```

Stimpy      South     15
Stimpy      South     11
Stimpy      South     19
Stimpy      South     23
")

Data = read.table(textConnection(Input),header=TRUE)

Data          ### will output data frame called Data

str(Data)    ### will be discussed later,
              ### but shows the structure of the data frame

summary(Data)  ### will be discussed later,
                 ### but summarizes variables in the data frame

```

Functions *sum* and *length*

The sum of a variable can be found with the *sum* function, and the number of observations can be found with the *length* function.

```

sum(Data$Attendees)

232

length(Data$Attendees)

16

```

Statistics of location for interval/ratio data

Mean

The mean is the arithmetic average, and is a common statistic used with interval/ratio data. It is simply the sum of the values divided by the number of values. The *mean* function in R will return the mean.

```

sum(Data$Attendees) / length(Data$Attendees)

14.5

mean(Data$Attendees)

14.5

```

Caution should be used when reporting mean values with skewed data, as the mean may not be representative of the center of the data. For example, imagine a town with 10 families, nine of whom have an income of less than \$50,000 per year, but with one family with an income of \$2,000,000 per year. The mean income for families in the town would be \$233,000, but this may not be a reasonable way to summarize the income of the town.

```
Income = c(49000, 44000, 25000, 18000, 32000, 47000, 37000, 45000, 36000, 2000000)
```

```
mean(Income)
```

233300

Median

The median is defined as the value below which are 50% of the observations. To find this value manually, you would order the observations, and separate the lowest 50% from the highest 50%. For data sets with an odd number of observations, the median is the middle value. For data sets with an even number of observations, the median falls half-way between the two middle values.

The median is a robust statistic in that it is not affected by adding extreme values. For example, if we changed Stimpy's last *Attendees* value from 23 to 1000, it would not affect the median.

```
median(Data$Attendees)
```

15

```
### Note that in this case the mean and median are close in value
### to one another. The mean and median will be more different
### the more the data are skewed.
```

The median is appropriate for either skewed or unskewed data. The median income for the town discussed above is \$40,500. Half the families in the town have an income above this amount, and half have an income below this amount.

```
Income = c(49000, 44000, 25000, 18000, 32000, 47000, 37000, 45000, 36000, 2000000)
```

```
median(Income)
```

40500

Note that medians are sometimes reported as the “average person” or “typical family”. Saying, “The average American family earned \$54,000 last year” means that the median income for families was \$54,000. The “average family” is that one with the median income.

Mode

The mode is a summary statistic that is used rarely in practice, but is normally included in any discussion of mean and medians. When there are discrete values for a variable, the mode is simply the value which occurs most frequently. For example, in the Statistics Learning Center video in the *Required Readings* below, Dr. Nic gives an example of counting the number of pairs of shoes each student owns. The most common answer was 10, and therefore 10 is the mode for that data set.

For our Ren and Stimpy example, the value 15 occurs three times and so is the mode.

The *Mode* function can be found in the package *DescTools*.

```
library(DescTools)
Mode(Data$Attendees)
15
```

Statistics of variation for interval/ratio data

Standard deviation

The standard deviation is a measure of variation which is commonly used with interval/ratio data. It's a measurement of how close the observations in the data set are to the mean.

There's a handy rule of thumb that—for normally distributed data—68% of data points fall within the mean \pm 1 standard deviation, 95% of data points fall within the mean \pm 2 standard deviations, and 99.7% of data points fall within the mean \pm 3 standard deviations.

Because the mean is often represented with the letter *mu*, and the standard deviation is represented with the letter *sigma*, saying someone is “a few *sigmas* away from *mu*” indicates they are rather a rare character. (I originally heard this joke on an episode of *Car Talk*, for which I cannot find a reference or transcript.)

```
sd(Data$Attendees)
4.830459
```

Standard deviation may not be appropriate for skewed data.

Standard error of the mean

Standard error of the mean is a measure that estimates how close a calculated mean is likely to be to the true mean of that population. It is commonly used in tables or plots where multiple means are presented together. For example, we might want to present the mean attendees for Ren with the standard error for that mean and the mean attendees for Stimpy with the standard error that mean.

The standard error is the standard deviation of a data set divided by the square root of the number of observations. It can also be found in the output for the *describe* function in the *psych* package, labelled *se*.

```
sd(Data$Attendees) /
sqrt(length(Data$Attendees))
1.207615
```

```
library(psych)
describe(Data$Attendees)
```

```
vars n mean sd median trimmed mad min max range skew kurtosis se
1   1 16 14.5 4.83      15    14.5 4.45    6  23     17 -0.04    -0.88 1.21
```

se indicates the standard error of the mean

Standard error of the mean may not be appropriate for skewed data.

Five-number summary, quartiles, percentiles

The median is the same as the 50th percentile, because 50% of values fall below this value. Other percentiles for a data set can be identified to provide more information. Typically, the 0th, 25th, 50th, 75th, and 100th percentiles are reported. This is sometimes called the five-number summary.

These values can also be called the minimum, 1st quartile, 2nd quartile, 3rd quartile, and maximum.

The five-number summary is a useful measure of variation for skewed interval/ratio data or for ordinal data. 25% of values fall below the 1st quartile and 25% of values fall above the 3rd quartile. This leaves the middle 50% of values between the 1st and 3rd quartiles, giving a sense of the range of the middle half of the data. This range is called the *interquartile range* (IQR).

Percentiles and quartiles are relatively robust, as they aren't affected much by a few extreme values. They are appropriate for both skewed and unskewed data.

```
summary(Data$Attendees)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
6.00	11.75	15.00	14.50	17.25	23.00

The five-number summary and the mean

Optional technical note on calculating percentiles

It may have struck you as odd that the 3rd quartile for *Attendees* was reported as 17.25. After all, if you were to order the values of *Attendees*, the 75th percentile would fall between 17 and 18. But why does R go with 17.25 and not 17.5?

```
sort(Data$Attendees)
```

```
6  7  9 11 12 13 14 15 15 15 16 17 18 19 22 23
```

The answer is that there are several different methods to calculate percentiles, and they may give slightly different answers. For details on the calculations, see *?quantiles*.

For *Attendees*, the default type 7 calculation yields a 75th percentile value of 17.25, whereas the type 2 calculation simply splits the difference between 17 and 18 and yields 17.5. The type 1 calculation doesn't average the two values, and so just returns 17.

```
quantile(Data$Attendees, 0.75, type=7)
```

75%
17.25

```
quantile(Data$Attendees, 0.75, type=2)
```

75%
17.5

```
quantile(Data$Attendees, 0.75, type=1)
```

75%
17

Percentiles other than the 25th, 50th, and 75th can be calculated with the quantiles function. For example, to calculate the 95th percentile:

```
quantile(Data$Attendees, .95)
```

95%
22.25

Confidence intervals

Confidence intervals are covered in the next chapter.

Statistics for grouped interval/ratio data

In many cases, we will want to examine summary statistics for a variable within groups. For example, we may want to examine statistics for the workshops lead by Ren and those lead by Stimpy.

Summarize in FSA

The *Summarize* function in the *FSA* package returns the number of observations, mean, standard deviation, minimum, 1st quartile, median, 3rd quartile, and maximum for grouped data.

Note the use of formula notation: *Attendees* is the dependent variable (the variable you want to get the statistics for); and *Instructor* is the independent variable (the grouping variable). *Summarize* allows you to summarize over the combination of multiple independent variables by listing them to the right of the ~ separated by a plus sign (+).

```
library(FSA)

Summarize(Attendees ~ Instructor,
          data=Data)

Instructor n  nvalid   mean        sd  min    Q1  median    Q3  max percZero
1       Ren  8      8 13.125 5.083236    6 10.75  13.5 15.25  22        0
2     Stimpy  8      8 15.875 4.454131    9 14.00  16.0 18.25  23        0
```

```
Summarize(Attendees ~ Instructor + Location,
  data=Data)
```

	Instructor	Location	n	invalid	mean	sd	min	Q1	median	Q3	max	perczero
1	Ren	North	4	4	12.50	7.505554	6	6.75	11.0	16.75	22	0
2	Stimpy	North	4	4	14.75	4.031129	9	13.50	16.0	17.25	18	0
3	Ren	South	4	4	13.75	1.707825	12	12.75	13.5	14.50	16	0
4	Stimpy	South	4	4	17.00	5.163978	11	14.00	17.0	20.00	23	0

summarySE in Rmisc

The *summarySE* function in the *Rmisc* package outputs the number of observations, mean, standard deviation, standard error of the mean, and confidence interval for grouped data. The *summarySE* function allows you to summarize over the combination of multiple independent variables by listing them as a vector, e.g. *c("Instructor", "Student")*.

```
library(Rmisc)
```

```
summarySE(data=Data,
  "Attendees",
  groupvars="Instructor",
  conf.interval = 0.95)
```

	Instructor	N	Attendees	sd	se	ci
1	Ren	8	13.125	5.083236	1.797195	4.249691
2	Stimpy	8	15.875	4.454131	1.574773	3.723747

```
summarySE(data=Data,
  "Attendees",
  groupvars = c("Instructor", "Location"),
  conf.interval = 0.95)
```

	Instructor	Location	N	Attendees	sd	se	ci
1	Ren	North	4	12.50	7.505553	3.7527767	11.943011
2	Ren	South	4	13.75	1.707825	0.8539126	2.717531
3	Stimpy	North	4	14.75	4.031129	2.0155644	6.414426
4	Stimpy	South	4	17.00	5.163978	2.5819889	8.217041

describeBy in psych

The *describeBy* function in the *psych* package returns the number of observations, mean, median, trimmed means, minimum, maximum, range, skew, kurtosis, and standard error of the mean for grouped data. *describeBy* allows you to summarize over the combination of multiple independent variables by combining terms with a colon (:).

```
library(psych)
```

```
describeBy(Data$Attendees,
```

```

group = Data$Instructor,
digits= 4)

group: Ren
  vars n mean   sd median trimmed mad min max range skew kurtosis se
1    1 8 13.12 5.08    13.5 13.12 2.97    6 22    16 0.13 -1.08 1.8
-----
group: Stimpy
  vars n mean   sd median trimmed mad min max range skew kurtosis se
1    1 8 15.88 4.45     16 15.88 3.71    9 23    14 -0.06 -1.26 1.57

describeBy(Data$Attendees,
           group = Data$Instructor : Data$Location,
           digits= 4)

group: Ren:North
  vars n mean   sd median trimmed mad min max range skew kurtosis se
1    1 4 12.5 7.51     11 12.5 6.67    6 22    16 0.26 -2.14 3.75
-----
group: Ren:South
  vars n mean   sd median trimmed mad min max range skew kurtosis se
1    1 4 13.75 1.71    13.5 13.75 1.48   12 16     4 0.28 -1.96 0.85
-----
group: Stimpy:North
  vars n mean   sd median trimmed mad min max range skew kurtosis se
1    1 4 14.75 4.03    16 14.75 2.22    9 18     9 -0.55 -1.84 2.02
-----
group: Stimpy:South
  vars n mean   sd median trimmed mad min max range skew kurtosis se
1    1 4 17 5.16      17    17 5.93 11 23    12 0    -2.08 2.58

```

Summaries for data frames

Often we will want to summarize variables in a whole data frame, either to get some summary statistics for individual variables, or to check that variables have the values we expected in inputting the data, to be sure there wasn't some error.

The *str* function

The *str* function in the native *utils* package will list variables for a data frame, with their types and levels. *Data* is the name of the data frame, created above.

```

str(Data)

'data.frame':   16 obs. of  3 variables:
 $ Instructor: Factor w/ 2 levels "Ren", "Stimpy": 1 1 1 1 1 1 1 1 2 2 ...
 $ Location   : Factor w/ 2 levels "North", "South": 1 1 1 1 2 2 2 2 1 1 ...
 $ Attendees  : int  7 22 6 15 12 13 14 16 18 17 ...

### Instructor is a factor (nominal) variable with two levels.
### Location is a factor (nominal) variable with two levels.
### Attendees is an integer variable.

```

The *summary* function

The *summary* function in the native *base* package will summarize all variables in a data frame, listing the frequencies for levels of nominal variables; and for interval/ratio data, the minimum, 1st quartile, median, mean, 3rd quartile, and maximum.

```
summary(Data)
```

Instructor	Location	Attendees
Ren :8	North:8	Min. : 6.00
Stimpy:8	South:8	1st Qu.:11.75
		Median :15.00
		Mean :14.50
		3rd Qu.:17.25
		Max. :23.00

The *headTail* function in *psych*

The *headTail* function in the *psych* package reports the first and last observations for a data frame.

```
library(psych)
```

```
headTail(Data)
```

	Instructor	Location	Attendees
1	Ren	North	7
2	Ren	North	22
3	Ren	North	6
4	Ren	North	15
...	<NA>	<NA>	...
13	Stimpy	South	15
14	Stimpy	South	11
15	Stimpy	South	19
16	Stimpy	South	23

The *describe* function in *psych*

The *describe* function in the *psych* package reports number of observations, mean, standard deviation, trimmed means, minimum, maximum, range, skew, kurtosis, and standard error for variables in a data frame.

Note that factor variables are labeled with an asterisk (*), and the levels of the factors are coded as 1, 2, 3, etc.

```
library(psych)
```

```
describe(Data)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
Instructor*	1	16	1.5	0.52	1.5	1.5	0.74	1	2	1	0.00	-2.12	0.13
Location*	2	16	1.5	0.52	1.5	1.5	0.74	1	2	1	0.00	-2.12	0.13
Attendees	3	16	14.5	4.83	15.0	14.5	4.45	6	23	17	-0.04	-0.88	1.21

Dealing with missing values

Sometimes a data set will have missing values. This can occur for a variety of reasons, such as a respondent not answering a specific question, or a researcher being unable to make a measurement due to temporarily malfunctioning equipment.

In R, a missing value is indicated with *NA*.

By default, different functions in R will handle missing values in different ways. But most have options to change how they treat missing data.

In general, you should scan your data for missing data, and think carefully about the best way to handle observations with missing values.

```
Input = "
Instructor  Location  Attendees
Ren          North      7
Ren          North     22
Ren          North      6
Ren          North     15
Ren          South     12
Ren          South     13
Ren          South    NA
Ren          South     16
Stimpy        North     18
Stimpy        North     17
Stimpy        North    NA
Stimpy        North      9
Stimpy        South     15
Stimpy        South     11
Stimpy        South     19
Stimpy        South     23
")
Data2 = read.table(textConnection(Input),header=TRUE)

### Note: This data frame will be called Data2 to distinguish it
###         from Data above.

Data2
```

The *na.rm* option for missing values with a simple function

Many common functions in R have a *na.rm* option. If this option is set to *FALSE*, the function will return an *NA* result if there are any *NA*'s in the data values passed to the function. If set to *TRUE*, observations with *NA* values will be discarded, and the function will continue to calculate its output.

Note that the *na.rm* option operates only on the data values actually passed to the function. In the following example with *median*, only *Attendees* is passed to the function; if there were *NA*'s in other variables, this would not affect the function.

Not all functions have the same default for the `na.rm` option. To determine the default, use e.g. `?median`, `?mean`, `?sd`.

```
median(Data2$Attendees,
       na.rm = FALSE)

NA

### na.rm=FALSE. Since there is an NA in the data, report NA.

median(Data2$Attendees,
       na.rm = TRUE)

15

### na.rm=TRUE. Drop observations with NA and then calculate the median.
```

Missing values with Summarize in FSA

`Summarize` in `FSA` will indicate invalid values, including NA's, with the count of valid observations outputted as the variable `nvalid`.

```
library(FSA)

Summarize(Attendees ~ Instructor,
          data=Data2)

  Instructor n nvalid mean      sd min   Q1 median   Q3 max percZero
1        Ren  8     7 13.5477226  6.9.5    13 15.5  22      0
2      Stimp  8     7 16.4795832  9.13.0    17 18.5  23      0

### This function removes missing values, but indicates the number of
### missing values by not including them in the count for nvalid.
```

Missing values indicated with summary function

The `summary` function counts NA's for numeric variables.

```
summary(Data2)

  Instructor Location Attendees
Ren      :8 North:8 Min.   : 6.00
Stimp:8 South:8 1st Qu.:11.25
                  Median :15.00
                  Mean   :14.50
                  3rd Qu.:17.75
                  Max.   :23.00
                  NA's   :2

### Indicates two NA's in Attendees.
```

Missing values in the describe function in psych

The *describe* function is the *psych* package removes NA's by default.

```
library(psych)

describe(Data2$Attendees)

  vars n mean   sd median trimmed mad min max range skew kurtosis se
1    1 14 14.5 5.19      15 14.5 5.19   6 23    17 -0.04 -1.17 1.39

### Note that two NA's were removed by default, reporting an n of 14.
```

Missing values in the summarySE function in Rmisc

By default, the *summarySE* function does not remove NA's, but can be made to do so with the *na.rm=TRUE* option.

```
library(Rmisc)

summarySE(data=Data2,
          "Attendees")

  .id N Attendees sd se ci
1 <NA> 16        NA NA NA NA

### Note an N of 16 is reported, and statistics are reported as NA.

library(Rmisc)

summarySE(data=Data2,
          "Attendees",
          na.rm=TRUE)

  .id N Attendees      sd      se      ci
1 <NA> 14     14.5 5.185038 1.38576 2.993752

### Note an N of 14 is reported, and statistics are calculated with
### NA's removed.
```

Advanced techniques*Discarding subjects*

Certain psychology studies use scales that are calculated from responses of several questions. Because answers to all questions are needed to reliably calculate the scale, any observation (subject, person) with missing answers will simply be discarded. Some types of analyses in other fields also follow this approach.

Subjects can be deleted with the *subset* function. The following code creates a new data frame, *Data3*, with all observations with *NA* in the variable *Attendees* removed from *Data2*.

```
Data3 = subset(Data2,
  !is.na(Attendees))

Data3
```

Imputation of values

Missing values can be assigned a likely value through the process of imputation. An algorithm is used that determines a value based on the values of other variables for that observation relative to values for other variables. The *mice* package in R can perform imputation.

Optional code: removing missing values in vectors

If functions don't have a *na.rm* option, it is possible to remove NA observations manually. Here we'll create a vector called *valid* that just contains those values of *Attendees* that are not NA's.

The *!* operator is a logical *not*. The brackets serve as a list of observations to include for the preceding variable. So, the code essentially says, "Define a vector *valid* as the values of *Attendees* where the values of *Attendees* are *not NA*."

```
valid = Data2$Attendees[!is.na(Data2$Attendees)]

summary(valid)

Min. 1st Qu. Median Mean 3rd Qu. Max.
6.00 11.25 15.00 14.50 17.75 23.00
```

Statistics of shape for interval/ratio data

The most common statistics for shape are skewness and kurtosis.

Understanding skewness and kurtosis are important as they are ways in which a distribution of data varies from a normal distribution. This will be important in assessing the assumptions of certain statistical tests.

However, I rarely see skewness and kurtosis values reported. Instead, normality is usually assessed visually with plot, or using certain statistical tests. One problem with using skewness and kurtosis values is that there is not agreement in what values constitute meaningful deviations from the normal curve.

Skewness

Skewness indicates the degree of asymmetry in a data set. If there are relatively more values that are far greater than the mean, the distribution is positively skewed or right skewed, with a tail stretching to the right. Negative or left skew is the opposite.

A symmetric distribution has a skewness of 0. The skewness value for a positively skewed distribution is positive, and a negative value for a negatively skewed distribution. Sometimes a skew with an absolute value greater than 1 or 1.5 or 2 is considered highly skewed. There is not agreement on this interpretation.

There are different methods to calculate skew and kurtosis. The *describe* function in the *psych* package has three options for how to calculate them.

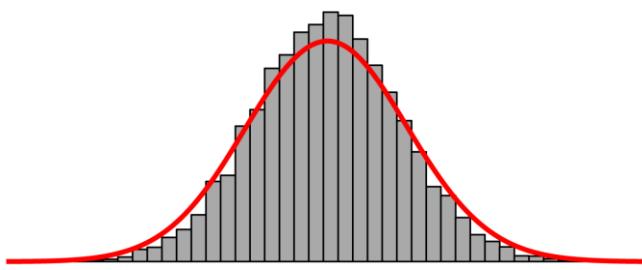
```
library(psych)

describe(Data$Attendees,
  type=3)      ### Type of calculation for skewness and kurtosis

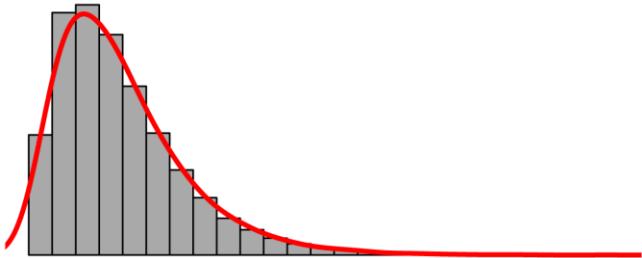
vars n mean sd median trimmed mad min max range skew kurtosis se
1   1 16 14.5 4.83      15    14.5 4.45    6  23     17 -0.04 -0.88 1.21

### Skewness and kurtosis among other statistics
```

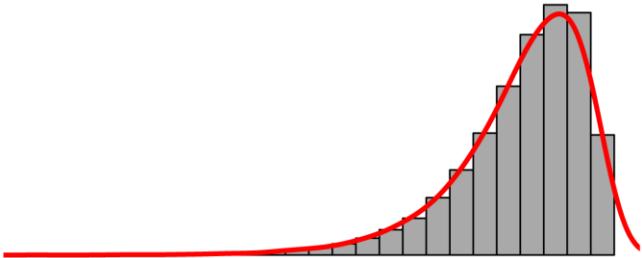
Normal curve



Positive (right) skewed



Negative (left) skewed



The normal curve is symmetrical around its center. The positively-skewed distribution has a longer, thicker tail to the right. The negatively-skewed distribution has a longer, thicker tail to the left.

For more information on the normal curve, see the video on “Normal Distribution” from Statistics Learning Center in the “Optional Readings” section. Additional thoughts on the normal distribution and real-world data distributions are in the article by Dr. Nic in the “Optional Readings” section.

Kurtosis

Kurtosis measures the degree to which the distribution has either fewer and less extreme outliers, or more and more extreme outliers. In general, the higher the kurtosis, the sharper the peak and the

longer the tails. This is called leptokurtic, and is indicated by positive kurtosis values. The opposite—platykurtosis—has negative kurtosis values.

Confusingly, the kurtosis for a normal distribution is sometimes defined as 0, and sometimes defined as 3. The former is often called “excess kurtosis”.

Sometimes an excess kurtosis with an absolute value greater than 2 or 3 is considered a high deviation from being mesokurtic. There is not agreement on this interpretation.

Descriptive statistics for ordinal data

Descriptive statistics for ordinal data are more limited than those for interval/ratio data. You’ll remember that for ordinal data, the levels can be ordered, but we can’t say that the intervals between the levels are equal. For example, we can’t say that an Associate’s degree and Master’s degree somehow average out to a Bachelor’s degree. This concept is discussed in more detail in the chapters on Likert data.

Because of this fact, several common descriptive statistics are usually inappropriate for use with ordinal data. These include mean, standard deviation, and standard error of the mean.

Ordinal data can be described by either: 1) treating the data as numeric and using appropriate statistics such as median and quartiles; or, 2) treating the data as nominal, and looking at counts of the data for each level.

A more-complete discussion of descriptive statistics for ordinal data can be found in the *Descriptive Statistics for Likert Data* chapter.

Example of descriptive statistics for ordinal data

For this example, imagine that Arthur and Baxter have each held a workshop educating the public about preventing water pollution at home. They are interested in the education level of people who attended the workshops. We can consider education level an ordinal variable. They also collected information on the sex and county of participants.

For this data, we have manually coded *Education* with numbers in a separate variable *Ed.code*. These numbers list the education in order: High school < Associate’s < Bachelor’s < Master’s < Ph.D.

Of course we could have had R do this coding for us, but the code is slightly messy, so I did the coding manually, and listed the optional R code below.

Ideally we would want to treat *Education* as an *ordered factor* variable in R. But unfortunately most common functions in R won’t handle ordered factors well. Later in this book, ordered factor data will be handled directly with cumulative link models (CLM), permutation tests, and tests for ordered tables.

Optional R code is shown below for converting a factor to an ordered factor.

```

Input = "
Date      Instructor   Student  Sex    County   Education Ed.code
'2015-11-01' 'Arthur Read'  a       female  'Elwood'  BA        3
'2015-11-01' 'Arthur Read'  b       female  'Bear Lake' PHD       5
'2015-11-01' 'Arthur Read'  c       male   'Elwood'  BA        3
'2015-11-01' 'Arthur Read'  d       female  'Elwood'  MA        4
'2015-11-01' 'Arthur Read'  e       male   'Elwood'  HS        1
'2015-11-01' 'Arthur Read'  f       female  'Bear Lake' MA        4
'2015-11-01' 'Arthur Read'  g       male   'Elwood'  HS        1
'2015-11-01' 'Arthur Read'  h       female  'Elwood'  BA        3
'2015-11-01' 'Arthur Read'  i       female  'Elwood'  BA        3
'2015-11-01' 'Arthur Read'  j       female  'Elwood'  BA        3
'2015-12-01' 'Buster Baxter' k       male   'Elwood'  MA        4
'2015-12-01' 'Buster Baxter' l       male   'Bear Lake' MA        4
'2015-12-01' 'Buster Baxter' m       female  'Elwood'  AA        2
'2015-12-01' 'Buster Baxter' n       male   'Elwood'  AA        2
'2015-12-01' 'Buster Baxter' o       other   'Elwood'  BA        3
'2015-12-01' 'Buster Baxter' p       female  'Elwood'  BA        3
'2015-12-01' 'Buster Baxter' q       female  'Bear Lake' PHD       5
")
)

Data = read.table(textConnection(Input), header=TRUE)

### Check the data frame

Data

str(Data)

'data.frame': 17 obs. of 7 variables:
 $ Date      : Factor w/ 2 levels "2015-11-01","2015-12-01": 1 1 1 1 1 1 1 1 1 1 ...
 $ Instructor: Factor w/ 2 levels "Arthur Read",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ Student   : Factor w/ 17 levels "a","b","c","d",...: 1 2 3 4 5 6 7 8 9 10 ...
 $ Sex       : Factor w/ 3 levels "female","male",...: 1 1 2 1 2 1 2 1 1 1 ...
 $ County    : Factor w/ 2 levels "Bear Lake","Elwood": 2 1 2 2 2 1 2 2 2 2 ...
 $ Education : Factor w/ 5 levels "AA","BA","HS",...: 2 5 2 4 3 4 3 2 2 2 ...
 $ Ed.code   : int 3 5 3 4 1 4 1 3 3 3 ...

summary(Data)

      Date           Instructor      Student      Sex      County
2015-11-01:10  Arthur Read :10     a       : 1  female:10  Bear Lake: 4
2015-12-01: 7   Buster Baxter: 7    b       : 1   male : 6   Elwood   :13
                           c       : 1   other : 1
                           d       : 1
                           e       : 1
                           f       : 1
                           (Other):11

Education   Ed.code
AA :2      Min.   :1.000
BA :7      1st Qu.:3.000
HS :2      Median :3.000
MA :4      Mean    :3.118

```

```
PHD:2      3rd Qu.:4.000
          Max.   :5.00
```

```
### Remove unnecessary objects
```

```
rm(Input)
```

Optional code to assign values to a variable based on another variable

```
Data$Ed.code[Data$Education=="HS"] = 1
Data$Ed.code[Data$Education=="AA"] = 2
Data$Ed.code[Data$Education=="BA"] = 3
Data$Ed.code[Data$Education=="MA"] = 4
Data$Ed.code[Data$Education=="PHD"] = 5
```

Optional code to change a factor variable to an ordered factor variable

```
Data$Education.ordered = factor(Data$Education,
                                 ordered = TRUE,
                                 levels = c("HS", "AA", "BA", "MA", "PHD"))
str(Data$Education.ordered)
summary(Data$Education.ordered)

HS  AA  BA  MA  PHD
2   2   7   4   2
```

```
median(Data$Education.ordered)
```

```
[1] "BA"
```

Statistics of location for ordinal data

Using the mean is usually not appropriate for ordinal data. Instead the median should be used as a measure of location. We will use our numerically-coded variable *Ed.code* for the analysis.

```
median(Data$Ed.code)

3
### Remember that 3 meant Bachelor's degree in our coding
```

Statistics of variation for ordinal data

Statistics like standard deviation and standard error of the mean are usually inappropriate for ordinal data. Instead the five-number summary can be used.

```
summary(Data$Ed.code)
```

```
Min. 1st Qu. Median Mean 3rd Qu. Max.
1.000 3.000 3.000 3.118 4.000 5.000
```

Remember to ignore the mean value for ordinal data.

Remember that 1 meant High school, 3 meant Bachelor's,
4 meant Master's, and 5 meant Ph.D.

Statistics for grouped ordinal data

Because the *Summarize* function in the *FSA* package reports the five-number summary, it is a useful function for descriptive statistics for grouped ordinal data.

```
library(FSA)
```

```
Summarize(Ed.code ~ Sex,
          data=Data)
```

	Sex	n	nvalid	mean	sd	min	Q1	median	Q3	max	percZero
1	female	10	10	3.5	0.9718253	2	3.00	3.0	4.00	5	0
2	male	6	6	2.5	1.3784049	1	1.25	2.5	3.75	4	0
3	other	1	1	3.0	NA	3	3.00	3.0	3.00	3	0

Remember to ignore the mean and sd values for ordinal data.

```
library(FSA)
```

```
Summarize(Ed.code ~ Sex + County,
          data=Data)
```

	Sex	County	n	nvalid	mean	sd	min	Q1	median	Q3	max	percZero
1	female	Bear Lake	3	3	4.6666667	0.5773503	4	4.5	5	5	5	0
2	male	Bear Lake	1	1	4.0000000	NA	4	4.0	4	4	4	0
3	female	Elwood	7	7	3.0000000	0.5773503	2	3.0	3	3	4	0
4	male	Elwood	5	5	2.2000000	1.3038405	1	1.0	2	3	4	0
5	other	Elwood	1	1	3.0000000	NA	3	3.0	3	3	3	0

Remember to ignore the mean and sd values for ordinal data.

Statistics for ordinal data treated as nominal data

The *summary* function in the native *base* package and the *xtabs* function in the native *stats* package provide counts for levels of a nominal variable.

The *prop.table* function translates a table into proportions. The *margin=1* option indicates that the proportions are calculated for each row.

First we will order the levels of *Education*, otherwise R with report results in alphabetical order.

```
Data$Education = factor(Data$Education,
                        levels = c("HS", "AA", "BA", "MA", "PHD"))
```

```
### Order factors, otherwise R will alphabetize them
```

```
summary(Data$Education)
```

HS	AA	BA	MA	PHD
2	2	7	4	2

```
### Counts of each level of Education
```

```
XT = xtabs(~ Education,  
          data=Data)
```

```
XT
```

Education

HS	AA	BA	MA	PHD
2	2	7	4	2

```
prop.table(XT)
```

Education

HS	AA	BA	MA	PHD
0.1176471	0.1176471	0.4117647	0.2352941	0.1176471

Grouped data

```
XT = xtabs(~ Sex + Education,  
          data = Data)
```

```
XT
```

Education

Sex	HS	AA	BA	MA	PHD
female	0	1	5	2	2
male	2	1	1	2	0
other	0	0	1	0	0

```
prop.table(XT,  
          margin = 1)
```

Education

Sex	HS	AA	BA	MA	PHD
female	0.0000000	0.1000000	0.5000000	0.2000000	0.2000000
male	0.3333333	0.1666667	0.1666667	0.3333333	0.0000000
other	0.0000000	0.0000000	1.0000000	0.0000000	0.0000000

```
### Proportion of responses for each row
```

Two-way grouped data

```

XT = xtabs(~ Sex + Education + County,
           data=Data)

XT

### Note that the dependent variable, Education, is the middle
### of the variable list.

, , County = Bear Lake

      Education
Sex      HS AA BA MA PHD
female   0  0  0  1  2
male     0  0  0  1  0
other    0  0  0  0  0

, , County = Elwood

      Education
Sex      HS AA BA MA PHD
female   0  1  5  1  0
male     2  1  1  1  0
other    0  0  1  0  0

```

Descriptive statistics for nominal data

Descriptive statistics for nominal data consist of listing or plotting counts for levels of the nominal data, often by levels of a grouping variable. Tables of this information are often called *contingency tables*.

For this example, we will look again at Arthur and Buster's data, but this time considering *Sex* to be the dependent variable of interest.

If levels of a nominal variable are coded with numbers, remember that the numbers will be arbitrary. For example, if you assign *female* = 1, and *male* = 2, and *other* = 3, it makes no sense to say that the average sex in Arthur's class was 1.3. Or that the average sex in Buster's class was greater than that in Arthur's. It also makes no sense to say that the median sex was female.

```

Input = "
Date      Instructor    Student  Sex      County       Education Ed.code
'2015-11-01' 'Arthur Read'  a        female   'Elwood'    BA        3
'2015-11-01' 'Arthur Read'  b        female   'Bear Lake'  PHD       5
'2015-11-01' 'Arthur Read'  c        male     'Elwood'    BA        3
'2015-11-01' 'Arthur Read'  d        female   'Elwood'    MA        4
'2015-11-01' 'Arthur Read'  e        male     'Elwood'    HS        1
'2015-11-01' 'Arthur Read'  f        female   'Bear Lake'  MA        4
'2015-11-01' 'Arthur Read'  g        male     'Elwood'    HS        1
'2015-11-01' 'Arthur Read'  h        female   'Elwood'    BA        3
'2015-11-01' 'Arthur Read'  i        female   'Elwood'    BA        3
"

```

```
'2015-11-01' 'Arthur Read' j
'2015-12-01' 'Buster Baxter' k
'2015-12-01' 'Buster Baxter' l
'2015-12-01' 'Buster Baxter' m
'2015-12-01' 'Buster Baxter' n
'2015-12-01' 'Buster Baxter' o
'2015-12-01' 'Buster Baxter' p
'2015-12-01' 'Buster Baxter' q
")
```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
### Check the data frame
```

```
Data
```

```
str(Data)
```

```
summary(Data)
```

```
### Remove unnecessary objects
```

```
rm(Input)
```

Example of descriptive statistics for nominal data

The *summary* function in the native *base* package and the *xtabs* function in the native *stats* package provide counts for levels of a nominal variable.

The *prop.table* function translates a table into proportions. The *margin=1* option indicates that the proportions are calculated for each row.

One-sample data

```
summary(Data$Sex)
```

female	male	other
10	6	1

```
### Counts of each level of sex
```

One-way data

```
xtabs(~ Date + Sex,
      data=Data)
```

Date	Sex		
	female	male	other
2015-11-01	7	3	0
2015-12-01	3	3	1

```

XT = xtabs(~ Date + Sex,
           data=Data)

prop.table(XT,
           margin = 1)

      Sex
Date      female     male   other
2015-11-01 0.7000000 0.3000000 0.0000000
2015-12-01 0.4285714 0.4285714 0.1428571

### Proportion of each level of Sex for each row

sum(XT)

[1] 17

### Sum of observation in the table

rowSums(XT)

2015-11-01 2015-12-01
10          7

### Sum of observation in each row of the table

colSums(XT)

female     male   other
10        6       1

### Sum of observation in each column of the table

```

Two-way data

```

xtabs(~ County + Sex + Date,
      data=Data)

### Note that the dependent variable, Sex, is the middle
### of the variable list.

, , Date = 2015-11-01

      Sex
County      female male other
Bear Lake    2     0     0
Elwood       5     3     0

, , Date = 2015-12-01

```

County	Sex		
	female	male	other
Bear Lake	1	1	0
Elwood	2	2	1

Levels for factor variables

Most of the time in R, nominal variables will be handled by the software as factor variables.

The order of levels of factor variables are important because most functions, including plotting functions, will handle levels of the factor in order. The order of the levels can be changed, for example, to change the order that groups are plotted in a plot, or which groups are at the top of the table.

By default, the *read.table* function in R interprets character data as factor variables. And by default R alphabetizes the levels of the factors.

Looking at the Arthur and Buster data, note that *Instructor*, *Student*, *Sex*, among other variables, are treated as factor variables.

```
str(Data)
```

```
'data.frame': 17 obs. of 7 variables:
 $ Date      : Factor w/ 2 levels "2015-11-01", "2015-12-01": 1 1 1 1 1 1 1 1 1 ...
 $ Instructor: Factor w/ 2 levels "Arthur Read", ...: 1 1 1 1 1 1 1 1 1 ...
 $ Student    : Factor w/ 17 levels "a", "b", "c", "d", ...: 1 2 3 4 5 6 7 8 9 10 ...
 $ Sex        : Factor w/ 3 levels "female", "male", ...: 1 1 2 1 2 1 2 1 1 1 ...
 $ County     : Factor w/ 2 levels "Bear Lake", "Elwood": 2 1 2 2 2 1 2 2 2 2 ...
 $ Education   : Factor w/ 5 levels "AA", "BA", "HS", ...: 2 5 2 4 3 4 3 2 2 2 ...
 $ Ed.code    : int 3 5 3 4 1 4 1 3 3 3 ...
```

Note also that the levels of the factor variables were alphabetized by default. That is, even though *Elwood* was found in the data before *Bear Lake*, R treats *Bear Lake* as the first level in the variable *County*.

```
summary(Data)
```

Date	Instructor	Sex	County
2015-11-01:10	Arthur Read :10	female:10	Bear Lake: 4
2015-12-01: 7	Buster Baxter: 7	male : 6	Elwood :13
		other : 1	

We can order factor levels by the order in which they were read in the data frame.

```
Data$County = factor(Data$County,
```

```
levels=unique(Data$County))
```

```
levels(Data$County)
```

```
[1] "Elwood"    "Bear Lake"
```

We can also order factor levels in an order of our choosing.

```
Data$Sex = factor(Data$Sex,
                  levels=c("male" , "female"))

levels(Data$Sex)

[1] "male"   "female"
```

Note that in the actions above, we are not changing the order in the data frame, simply which level is treated internally by the software as "1" or "2", and so on.

Required readings

[Video] "*Understanding Summary statistics: Mean, Median, Mode*" from Statistics Learning Center (Dr. Nic). 2015. www.youtube.com/watch?v=rAN6DBctgJ0.

Optional readings

Statistics of location: mean and median

"*Statistics of central tendency*" in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/central.html.

"*The median outclasses the mean*" from Dr. Nic. 2013. Learn and Teach Statistics & Operations Research. learnandteachstatistics.wordpress.com/2013/04/29/median/.

"*Measures of the Location of the Data*", **Section 2.3** in Openstax. 2013. *Introductory Statistics*. openstaxcollege.org/textbooks/introductory-statistics.

"*Measures of the Center of the Data*", **Section 2.5** in Openstax. 2013. *Introductory Statistics*. openstaxcollege.org/textbooks/introductory-statistics.

"*Skewness and the Mean, Median, and Mode*", **Section 2.6** in Openstax. 2013. *Introductory Statistics*. openstaxcollege.org/textbooks/introductory-statistics.

Statistics of variation: standard deviation and range

"*Statistics of dispersion*" in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/dispersion.html.

"*Standard error of the mean*" in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/standarderror.html.

"Measures of the Spread of the Data", Section 2.7 in Openstax. 2013. *Introductory Statistics.* openstaxcollege.org/textbooks/introductory-statistics.

The normal distribution

[Video] **"Normal Distribution"** from Statistics Learning Center (Dr. Nic). 2016. www.youtube.com/watch?v=mtH1fmUVkFE.

"The normal distribution – three tricky bits" from Dr. Nic. 2016. Learn and Teach Statistics & Operations Research. learnandteachstatistics.wordpress.com/2016/01/18/the-normal-distribution/.

Optional note on reporting summary statistics honestly

Every time we report a descriptive statistic or the result of a statistical test, we are condensing information, which may have been a whole data set, into one or a few pieces of information. It is the job of the analyst to choose the best way to present descriptive and graphical information, and to choose the correct statistical test or method.

SAT score example

Imagine a set of changes in SAT scores from seven students who took a certain SAT preparation course. The change in scores, *Increase*, represents the difference in score, that is the score after taking the course minus their initial score.

```
Increase = c(50, 60, 120, -80, -10, 10, 0)
```

Our first instinct in assessing the course might be to look at the average (mean) change in score. The result is a mean increase of 21 points, which could be considered a success, perhaps.

```
mean(Increase)
```

```
[1] 21.42857
```

But we would be remiss to not look at other summary statistics and plots.

Using the *Summarize* function in the *FSA* package, we find that the median increase was only 10 points. The increase for the first quartile was negative, suggesting at least 25% of students got a lower score afterwards.

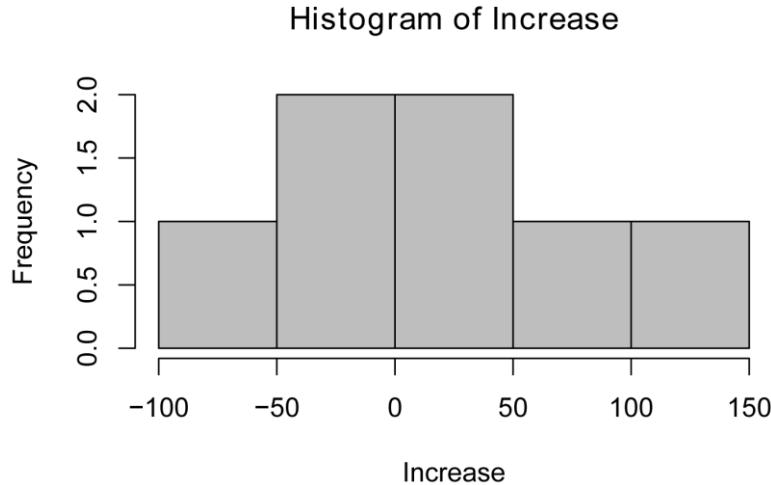
```
library(FSA)
```

```
Summarize(Increase,
          digits = 2)
```

n	nvalid	mean	sd	min	Q1	median	Q3	max
7.00	7.00	21.43	63.09	-80.00	-5.00	10.00	55.00	120.00

A histogram of the changes in scores suggests a slight right skew to the data, but that the mass of the data sits not far from zero.

```
hist(Increase,
  col="gray")
```



Finally, we'll compute the 95% confidence intervals for the mean change in score. Looking at the results for the percentile method, the confidence interval includes zero, suggesting the change in scores for this course were not statistically different from zero.

```
library(rcompanion)

Data = data.frame(Increase)

groupwiseMean(Increase ~ 1,
  data      = Data,
  traditional = FALSE,
  percentile = TRUE)
```

.id	n	Mean	Conf.level	Percentile.lower	Percentile.upper
1	<NA>	7	0.95	-21.4	65.7

Based on the data exploration, it seems that it would be irresponsible or dishonest to simply report that the average increase in test scores was 20 points.

For descriptive statistics, we might report the mean and 95% confidence interval. Or perhaps the 5-point summary for the change in scores, or show the histogram of values.

Optional analyses

Robust estimators: trimmed mean and Winsorized mean

Robust estimators of central tendency are used to describe the location of a data set without undue influence of extreme values. Robust estimators include trimmed means, Winsorized means, and other estimates like M-estimators (not discussed here).

Example

The New Jersey Envirothon is a statewide competition for high school students covering various aspects of natural resources science. In the Team Presentation station, student teams are judged by five judges.

Currently, scorekeepers drop the highest and lowest scores for each team to avoid the effect of aberrant low or high scores. This is an example of a trimmed mean. In this case, the mean is trimmed to 60% (with 20% removed from each side).

Another option to ameliorate the effect of extreme scores is using Winsorized means. A Winsorized mean removes extreme observations, but replaces them with the closest observations in terms of magnitude.

In this example, two teams received five scores for their presentations that have the same mean and median. We will look at robust estimators to determine if one team should be scored higher than the other.

```
Team.A = c(100, 90, 80, 60, 20)

median(Team.A)
[1] 80

mean(Team.A)
[1] 70

mean(Team.A,
      trim = .20)           # This trims to the inner 60% of observations
[1] 76.66667

library(psych)

winsor(Team.A,
       trim = 0.20)          # This winsorizes to the inner 60% of observations
[1] 92 90 80 60 52

### Note that the Winsorized values at the extremes appear to be calculated
### with a function analogous to the quantile(x, probs, type = 7) function.
```

```
winsor.mean(Team.A,
            trim = 0.20) # This Winsorizes to the inner 60% of observations

[1] 74.8

Team.B = c(80, 80, 80, 80, 30)

median(Team.B)

[1] 80

mean(Team.B)

[1] 70

mean(Team.B,
      trim = .20) # This trims to the inner 60% of observations

[1] 80

library(psych)

winsor(Team.B,
       trim = 0.20) # This Winsorizes to the inner 60% of observations

[1] 80 80 80 80 70

### Note that the Winsorized values at the extremes appear to be calculated
### with a function analogous to the quantile(x, probs, type = 7) function.

winsor.mean(Team.B,
            trim = 0.20) # This Winsorizes to the inner 60% of observations

[1] 78
```

In this example, the means and medians for Team A and Team B were identical. However, the trimmed mean for Team A was less than for Team B (77 vs. 80), and the Winsorized mean for A was less than for Team B (75 vs. 78). According to either the trimmed mean method or the Winsorized mean method, Team B had a higher score.

Note also that the Winsorized means were lower than for the trimmed means, for both teams. This is because the Winsorized mean is better able to take into account the low scores for each team.

Geometric mean

The geometric mean is used to summarize certain measurements, such as average return for investments, and for certain scientific measurements, such as bacteria counts in environmental water. It is useful when data are log-normally distributed.

Practically speaking, using the geometric mean ameliorates the effect of outlying values in the data.

To get the geometric mean, the log of each value is taken, these values are averaged, and then the result is the base of the log raised to this value.

Imagine a series of 9 counts of bacteria from lake water samples, called *Bacteria* here. The geometric mean can be calculated with a nested *log*, *mean*, and *exp* functions. Or more simply, the *geometric.mean* function in the *psych* package can be used.

Bacteria example

```
Bacteria = c(20, 40, 50, 60, 100, 120, 150, 200, 1000)
exp(mean(log(Bacteria)))
[1] 98.38887

library(psych)
geometric.mean(Bacteria)
[1] 98.38887

hist(Bacteria,     ### Produce a histogram of values
  col="darkgray",
  breaks="FD")
```

Annual return example

Geometric means are also used to calculate the average annual return for investments. Each value in this vector represents the percent return of the investment each year. The arithmetic mean will not give the correct answer for average annual return.

```
Return = c(-0.80, 0.20, 0.30, 0.30, 0.20, 0.30)
library(psych)
geometric.mean(Return + 1)-1
[1] -0.07344666

### The geometric mean will give the correct answer for average
### annual return
```

This can also be calculated manually. If you start with 100 dollars with this investment, you will end up with 63 dollars, equivalent to a return of - 0.07.

```
100 * (1-0.80) * 1.20 * 1.30 * 1.30 * 1.20 * 1.30
[1] 63.2736
```

Harmonic mean

The harmonic mean is another type of average that is used in certain situations, such as averaging rates or speeds.

To get the harmonic mean, the inverse of each value is taken, these values are averaged, and then the inverse of this result is reported.

Imagine a series of 9 speeds, called *Speed* here. The harmonic mean can be calculated with a nested $1/x$, *mean*, and $1/x$ functions. Or more simply, the *harmonic.mean* function in the *psych* package can be used.

```
Speed = c(20, 40, 50, 60, 100, 120, 150, 200, 1000)
1/mean(1/Speed)
63.08411
library(psych)
harmonic.mean(Speed)
63.08411
```

Exercises C

1. Considering Ren and Stimpy's workshops together,

- a. How many total attendees were there?
- b. How many total workshops were there?
- c. How many locations were there?
- d. What was the mean number of attendees?
- e. What was the median number of attendees?

2. Considering Ren and Stimpy's workshops,

a. Which workshops had a higher mean number of attendees and what was the mean? Ren's or Stimp's?

b. Which workshops had a higher mean number of attendees and what was the mean? Ren North or Ren South?

3. Considering Arthur and Buster's workshops together,

a. How many students attended in total?

b. How many distinct levels of education were reported?

c. What was the median level of education reported?

d. 75% of students had what level of education or lower?

e. What was the median education level for females?

f. What was the median education level for males?

g. What was the median education level for females from Elwood County?

h. What was the median education level for males from Elwood County?

4. Considering Arthur and Buster's workshops together,

a. How many students were male?

b. What percentage is this of total students?

c. How many students in the November workshop were male?

d. What percentage is this of total students in the November workshop?

e. How many male students in the November workshop were from Bear Lake County?

5. As part of a nutrition education program, extension educators had students keep diaries of what they ate for a day and then calculated the calories students consumed. The following data are the result. *Rating* indicates the score, from 1 to 5, that students gave as to the usefulness of the program. It should be considered an ordinal variable.

Student	Teacher	Sex	Calories	Rating
a	Tetsuo	male	2300	3
b	Tetsuo	female	1800	3
c	Tetsuo	male	1900	4
d	Tetsuo	female	1700	5
e	Tetsuo	male	2200	4
f	Tetsuo	female	1600	3

g	Tetsuo	male	1800	3
h	Tetsuo	female	2000	3
i	Kaneda	male	2100	4
j	Kaneda	female	1900	5
k	Kaneda	male	1900	4
l	Kaneda	female	1600	4
m	Kaneda	male	2000	4
n	Kaneda	female	2000	5
o	Kaneda	male	2100	3
p	Kaneda	female	1800	4

- a. What are the variables in this data set and what type of variable is each? (Types are “nominal”, “ordinal”, and “interval/ratio”, not how their types are reported by R.)

For each of the following, answer the question, and ***show the output from the analyses you used to answer the question.***

- b. How many students were involved in this data set?
 - c. What was the mean caloric intake?
 - d. What was the median caloric intake?
 - e. What was the standard deviation of caloric intake?
 - f. What was the mean caloric intake for females?
 - g. What was the mean caloric intake for females in Kaneda’s class?
 - h. How many males are in Kaneda’s class?
6. What do you conclude in practical terms about the caloric intake data for Tetsuo’s and Kaneda’s classes? You might consider the mean or median values between males and females or between Tetsuo and Kaneda. Think about the real world. Are the differences you are looking at important in a practical sense?

Be sure to pay attention to the spread of data as you think about this, for example using minimums and maximums, Q1 and Q3 values, or standard deviation values.

Be quantitative in your answer. That is, if you are talking about a difference in medians, mention the actual numerical difference. If expressing values as a percentage makes sense, feel free to do so.

Confidence Intervals

Packages used in this chapter

The packages used in this chapter include:

- Rmisc
- DescTools
- plyr
- boot
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(Rmisc)){install.packages("Rmisc")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(plyr)){install.packages("plyr")}
if(!require(boot)){install.packages("boot")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Understanding confidence intervals

Confidence intervals are used to indicate how accurate a calculated statistic is likely to be. Confidence intervals can be calculated for a variety of statistics, such as the mean, median, or slope of a linear regression. This chapter will focus on confidences intervals for means. This book contains a separate chapter, *Confidence Intervals for Medians*, which addresses confidence intervals for medians. There is also a chapter *Confidence Intervals for Proportions* in this book.

The Statistics Learning Center video in the *Required Readings* below gives a good explanation of the meaning of confidence intervals.

Populations and samples

Most of the statistics we use assume we are analyzing a sample which we are using to represent a larger population. If extension educators want to know about the caloric intake of 7th graders, they would be hard-pressed to get the resources to have every 7th grader in the U.S. keep a food diary. Instead they might collect data from one or two classrooms, and then treat the data *sample* as if it represents a larger *population* of students.

The mean caloric intake could be calculated for this sample, but this mean will not be exactly the same as the mean for the larger population. If we collect a large sample and the values aren't too variable, then the sample mean should be close to the population mean. But if we have few observations, or the values are highly variable, we are less confident our sample mean is close to the population mean.

We will use confidence intervals to give a sense of this confidence.

It's best not to overthink the discussion on populations and samples. We aren't necessarily actually extending our statistics to a larger population. That is, we shouldn't think our measurements from two classrooms are actually indicative of the whole country. There are likely many factors that would change the result school to school and region to region. But even if we are thinking about just the 7th

graders in just these two classrooms, most of our statistics will still be based on the assumption that there is a larger population of 7th graders, and we are sampling just a subset.

Statistics and parameters

When we calculate the sample mean, the result is a *statistic*. It's an estimate of the population mean, but our calculated sample mean would vary depending on our sample. In theory, there is a mean for the population of interest, and we consider this population mean a *parameter*. Our goal in calculating the sample mean is estimating the population parameter.

Point estimates and confidence intervals

Our sample mean is a *point estimate* for the population parameter. A point estimate is a useful approximation for the parameter, but considering the confidence interval for the estimate gives us more information.

As a definition of confidence intervals, if we were to sample the same population many times and calculated a sample mean and a 95% confidence interval each time, then 95% of those intervals would contain the actual population mean.

If this definition of confidence intervals doesn't make much intuitive sense to you at this point, don't worry about it. Working through some of the examples in this book will help you understand their usefulness.

One use of confidence intervals is to give a sense of how accurate our calculated statistic is relative to the population parameter.

An example

Imagine we have a rule of thumb that we consider a town with a mean household income of greater than \$100,000 to be high-income.

For Town A we sample some households, and calculate the mean household income and the 95% confidence interval for this statistic. The mean is \$125,000, but the data are quiet variable, and the 95% confidence interval is from \$75,000 to \$175,000. In this case, we don't have much confidence that Town A is actually a high-income town. The point estimate for the population mean is greater than \$100,000, but the confidence interval extends considerably lower than this threshold.

For Town B, we also get a mean of \$125,000, so the point estimate is the same as for Town A. But the 95% confidence interval is from \$105,000 to \$145,000. Here, we have some confidence that Town B is actually a high-income town, because the whole 95% confidence interval lies higher than the \$100,000 threshold.

Confidence intervals as an alternative to some tests

Most of the statistical tests in this book will calculate a probability (*p*-value) of the likelihood of data and draw a conclusion from this *p*-value. John McDonald, in the *Optional Readings* below, describes how confidence intervals can be used as an alternative approach.

For example, if we want to compare the means of two groups to see if they are statistically different, we will use a *t*-test, or similar test, calculate a *p*-value, and draw a conclusion. An alternative approach

would be to construct 95% or 99% confidence intervals about the mean for each group. If the confidence intervals of the two means don't overlap, we are justified in calling them statistically different.

Likewise, if the 95% confidence interval for some statistic includes zero, we can conclude that the statistic is not significantly different from zero.

As a technical note, non-overlapping confidence intervals for means do not equate exactly to a *t*-test with a *p*-value of 0.05. They are different methods to assess similar questions. The article by Cumming and Finch in the "References" section gives more details on the relationship between overlapping confidence intervals and *p*-values from statistical tests.

Example for confidence intervals

For this example, extension educators had students wear pedometers to count their number of steps over the course of a day. The following data are the result. *Rating* is the rating each student gave about the usefulness of the program, on a 1-to-10 scale.

```
Input = "
Student Sex Teacher Steps Rating
a female Catbus 8000 7
b female Catbus 9000 10
c female Catbus 10000 9
d female Catbus 7000 5
e female Catbus 6000 4
f female Catbus 8000 8
g male Catbus 7000 6
h male Catbus 5000 5
i male Catbus 9000 10
j male Catbus 7000 8
k female Satsuki 8000 7
l female Satsuki 9000 8
m female Satsuki 9000 8
n female Satsuki 8000 9
o male Satsuki 6000 5
p male Satsuki 8000 9
q male Satsuki 7000 6
r female Totoro 10000 10
s female Totoro 9000 10
t female Totoro 8000 8
u female Totoro 8000 7
v female Totoro 6000 7
w male Totoro 6000 8
x male Totoro 8000 10
y male Totoro 7000 7
z male Totoro 7000 7
")
```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
### Check the data frame
```

```

Data
str(Data)
summary(Data)

### Remove unnecessary objects

rm(Input)

```

Recommended procedures for confidence intervals for means

Confidence intervals for means can be calculated by various methods.

The *traditional* method is the most commonly encountered, and is appropriate for normally distributed data or with large sample sizes. It produces an interval that is symmetric about the mean.

For skewed data, confidence intervals by *bootstrapping* may be more reliable.

For routine use, I recommend using bootstrapped confidence intervals, particularly the *BCa* or *percentile* methods. For further discussion, see below *Optional Analyses: confidence intervals for the mean by bootstrapping*.

groupwiseMean function for grouped and ungrouped data

The *groupwiseMean* function in the rcompanion package can produce confidence intervals both by traditional and bootstrap methods, for grouped and ungrouped data.

The data must be housed in a data frame. By default, the function reports confidence intervals by the traditional method.

In the *groupwiseMean* function, the measurement and grouping variables can be indicated with formula notation, with the measurement variable on the left side of the tilde (~), and grouping variables on the right.

The confidence level is indicated by, e.g., the *conf = 0.95* argument. The *digits* option indicates the number of significant digits to which the output is rounded. Note that in the output, the means and other statistics are rounded to 3 significant figures.

Ungrouped data

Ungrouped data is indicated with a 1 one on the right side of the formula, or the *group = NULL* argument.

```

library(rcompanion)

groupwiseMean(Steps ~ 1,
               data   = Data,
               conf   = 0.95,
               digits = 3)

```

```
.id n Mean Conf.level Trad.lower Trad.upper
1 <NA> 26 7690      0.95      7170      8210

### Trad.lower and Trad.upper indicate the confidence interval
### for the mean by traditional method.
```

One-way data

```
library(rcompanion)

groupwiseMean(Steps ~ Sex,
             data = Data,
             conf = 0.95,
             digits = 3)

   Sex n Mean Conf.level Trad.lower Trad.upper
1 female 15 8200      0.95      7530      8870
2 male   11 7000      0.95      6260      7740

### Trad.lower and Trad.upper indicate the confidence interval
### for the mean by traditional method.
```

Two-way data

```
library(rcompanion)

groupwiseMean(Steps ~ Teacher + Sex,
             data = Data,
             conf = 0.95,
             digits = 3)

   Teacher Sex n Mean Conf.level Trad.lower Trad.upper
1 Catbus female 6 8000      0.95      6520      9480
2 Catbus male  4 7000      0.95      4400      9600
3 Satsuki female 4 8500      0.95      7580      9420
4 Satsuki male  3 7000      0.95      4520      9480
5 Totoro female 5 8200      0.95      6360     10000
6 Totoro male  4 7000      0.95      5700      8300

### Trad.lower and Trad.upper indicate the confidence interval
### for the mean by traditional method.
```

Bootstrapped means by group

In the *groupwiseMean* function, the type of confidence interval is requested by setting certain options to *TRUE*. These options are *traditional*, *normal*, *basic*, *percentile* and *bca*. The *boot* option reports an optional statistic, the mean by bootstrap. The *R* option indicates the number of iterations to calculate each bootstrap statistic.

```
library(rcompanion)
```

```
groupwiseMean(Steps ~ Sex,
  data = Data,
  conf = 0.95,
  digits = 3,
  R = 10000,
  boot = TRUE,
  traditional = FALSE,
  normal = FALSE,
  basic = FALSE,
  percentile = FALSE,
  bca = TRUE)

  Sex n Mean Boot.mean Conf.level Bca.lower Bca.upper
1 female 15 8200     8200      0.95      7470      8670
2 male 11 7000     7000      0.95      6270      7550
```

```
library(rcompanion)

groupwiseMean(Steps ~ Teacher + Sex,
  data = Data,
  conf = 0.95,
  digits = 3,
  R = 10000,
  boot = TRUE,
  traditional = FALSE,
  normal = FALSE,
  basic = FALSE,
  percentile = FALSE,
  bca = TRUE)
```

	Teacher	Sex	n	Mean	Boot.mean	Conf.level	Bca.lower	Bca.upper
1	Catbus	female	6	8000	8000	0.95	6830	8830
2	Catbus	male	4	7000	6990	0.95	5500	8000
3	Satsuki	female	4	8500	8500	0.95	8000	8750
4	Satsuki	male	3	7000	7000	0.95	6000	7670
5	Totoro	female	5	8200	8190	0.95	6800	9000
6	Totoro	male	4	7000	7000	0.95	6250	7500

Optional: Other functions for traditional confidence intervals for means

Functions that produce confidence intervals for means by the traditional method include *t.test*, *CI* in *Rmisc*, and *MeanCI* in *DescTools*.

```
t.test(Data$Steps,
  conf.level=0.95)

One Sample t-test

95 percent confidence interval:
7171.667 8212.949

mean of x
7692.308
```

```

library(DescTools)

MeanCI(Data$Steps,
       conf.level=0.95)

  mean   lwr.ci   upr.ci
7692.308 7171.667 8212.949

library(Rmisc)

CI(Data$Steps,
  ci=0.95)

  upper     mean    lower
8212.949 7692.308 7171.667

library(Rmisc)

group.CI(Steps ~ Sex,
         data=Data,
         ci = 0.95)

  Sex Steps.upper Steps.mean Steps.lower
1 female    8868.482      8200    7531.518
2 male      7735.930      7000    6264.070

group.CI(Steps ~ Teacher + Sex,
         data=Data,
         ci = 0.95)

  Teacher   Sex Steps.upper Steps.mean Steps.lower
1 Catbus female    9484.126      8000    6515.874
2 Satsuki female    9418.693      8500    7581.307
3 Totoro female   10041.685      8200    6358.315
4 Catbus male      9598.457      7000    4401.543
5 Satsuki male      9484.138      7000    4515.862
6 Totoro male      8299.228      7000    5700.772

```

Optional Analyses: confidence intervals for the mean by bootstrapping

Bootstrapping is a method that samples the data many times, each time calculating a statistic, and then determining a confidence interval or other statistic from these iterations.

Bootstrapped confidence intervals can be more reliable than those determined by the traditional method for certain data sets. For more details on the different types of bootstrapped confidence intervals, see the Carpenter and Bithel article in the “References” section of this chapter. The BCa method (bias corrected, accelerated) is often cited as the best method, and the percentile method is also cited as typically good.

The *boot* package can calculate confidence intervals for means by bootstrap. In the *boot* function, *R* indicates the number of re-samplings.

The function *groupwiseMean* in the *rcompanion* package allows for calculating confidence intervals for means for grouped data, using the bootstrap procedures from the *boot* package.

Note that for bootstrap procedures, your results may vary slightly from the results reported here.

```
library(boot)

Mboot = boot(Data$Steps,
             function(x,i) mean(x[i]),
             R=10000)

mean(Mboot$t[,1])
7693.381

### The mean based on the bootstrap method.

boot.ci(Mboot,
        conf = 0.95,
        type = c("norm", "basic", "perc", "bca")
      )

Intervals :
Level      Normal          Basic
95%  (7208, 8174 )  (7192, 8154 )

Level      Percentile       BCa
95%  (7231, 8192 )  (7154, 8115 )
Calculations and Intervals on Original Scale

### Other information

Boot

hist(Boot$t[,1],
  col = "darkgray")
```

Required readings

[Video] “*Understanding Confidence Intervals: Statistics Help*” from Statistics Learning Center. (Dr. Nic). 2013. www.youtube.com/watch?v=tFWsu09f74o.

Optional readings

"Confidence limits" in McDonald, J.H. 2014. *Handbook of Biological Statistics*.
www.biostathandbook.com/confidence.html.

[Video] **"Calculating the Confidence interval for a mean using a formula"** from Statistics Learning Center. (Dr. Nic). 2013. www.youtube.com/watch?v=s4SRdaTycaw.

"Confidence Intervals", Chapter 8 in Openstax. 2013. *Introductory Statistics*.
openstaxcollege.org/textbooks/introductory-statistics.

"Confidence intervals", Chapter 4.2 in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

References

Carpenter, J. and J. Bithel. 2000. "Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians". *Statistics in Medicine* 19:1141–1164.

www.tau.ac.il/~saharon/Boot/10.1.1.133.8405.pdf.

Cumming, G. and Finch, S. 2005. Inference by Eye: Confidence Intervals and How to Read Pictures of Data. *American Psychologist* 60:170-180.

Optional analyses

Confidence intervals for geometric mean

The geometric mean was discussed in the previous chapter. Here, the *CI* function in *Rmisc* is used to construct confidence intervals for the geometric mean.

```
Bacteria = c(20, 40, 50, 60, 100, 120, 150, 200, 1000)

library(Rmisc)

exp(CI(log(Bacteria), ci=0.95))

      upper      mean      lower
233.85448 98.38887 41.39484
```

Confidence intervals for geometric means for groups

The function *groupwiseGeometric* in the *rcompanion* package produces the geometric mean and limits for the geometric mean plus and minus the standard deviation, standard error, and confidence interval.

```
Input = "
Site  Bacteria
A      20
A      40
A      50
A      60
A      100
```

```

A      120
A      150
A      200
A     1000

B      100
B      120
B      210
B     300
B     420
B     400
B     500
B     800
B    4000

C      10
C      30
C      40
C      60
C     110
C     100
C     160
C     210
C    1200
")

```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
library(rcompanion)
```

```
groupwiseGeometric(Bacteria ~ Site,
                    data   = Data,
                    digits = 3,
                    na.rm  = TRUE)
```

	Site	n	Geo.mean	sd.lower	sd.upper	se.lower	se.upper	ci.lower	ci.upper
1	A	9	98.4	31.9	303	67.6	143	41.4	234
2	B	9	389.0	129.0	1170	269.0	561	167.0	906
3	C	9	88.1	22.7	341	56.1	138	31.1	249

Exercises D

1. Considering the Catbus, Satsuki, and Totoro data,

- a. What was the mean of *Steps*?
- b. What is the 95% confidence interval for *Steps* (traditional method)?

2. Considering the Catbus, Satsuki, and Totoro data,

- a. What was the mean of *Steps* for females?
- b. What is the 95% confidence interval for *Steps* for females (traditional method)?
3. Looking at the 95% confidence intervals for *Steps* for males and females, are we justified in claiming that the mean *Steps* was statistically different for females and males? Why?
4. As part of a nutrition education program, extension educators had students keep diaries of what they ate for a day and then calculated the calories students consumed.

Student	Teacher	Sex	Calories	Rating
a	Tetsuo	male	2300	3
b	Tetsuo	female	1800	3
c	Tetsuo	male	1900	4
d	Tetsuo	female	1700	5
e	Tetsuo	male	2200	4
f	Tetsuo	female	1600	3
g	Tetsuo	male	1800	3
h	Tetsuo	female	2000	3
i	Kaneda	male	2100	4
j	Kaneda	female	1900	5
k	Kaneda	male	1900	4
l	Kaneda	female	1600	4
m	Kaneda	male	2000	4
n	Kaneda	female	2000	5
o	Kaneda	male	2100	3
p	Kaneda	female	1800	4

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- a. What is the mean of *Calories*?
- b. What is the 95% confidence interval for *Calories* (traditional method)?
- c. What is the mean of *Calories* for females?
- d. What is the 95% confidence interval for *Calories* for females (traditional method)?
- e. Looking at the 95% confidence intervals for *Calories* for males and females, are we justified in claiming that the mean *Calories* was statistically different for females and males? Why?

Basic Plots

The need to understand plots

There are several reasons why it is important to be able to produce and interpret plots of data.

Visualize your data. Plots are essential for visualizing and exploring your data. Plotting a histogram gives a sense of the range, center, and shape of the data. It shows if the data is symmetric, skewed, bimodal, or uniform. Box plots and plots of means, medians, and measures of variation visually indicate the difference in means or medians among groups. Scatter plots show the relationship between two measured variables.

Communicate your results to others. Being able to choose and produce appropriate plots is important for summarizing your data for professional or extension presentations and journal articles. Plots can be very effective for summarizing data in a visual manner and can show statistical results with impact.

Understand other's results. It is essential to understand the plots you find in extension and professional literature.

Packages used in this chapter

The packages used in this chapter include:

- lattice
- plyr
- ggplot2
- FSA
- DescTools
- vcd
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(lattice)){install.packages("lattice")}
if(!require(plyr)){install.packages("plyr")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(FSA)){install.packages("FSA")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(vcd)){install.packages("vcd")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Some advice on producing plots

For some good advice on producing high-quality plots, read the “General tips” section of the McDonald page listed in the “Optional readings I” section.

His basic points:

- “Don’t clutter up your graph with unnecessary junk.”
- “Include all necessary information.”
- “Don’t use color in graphs for publication.”
- “Do use color in graphs for presentations.”

Producing clear and informative plots

There are many ways to make a bad plot. As McDonald notes, one way is to have a cluttered or unclear plot, and another is to use color inappropriately.

I strive for clear, bold text, lines and points. In a presentation, it can be difficult for the audience to understand busy plots, or see thin text, lines, or subtle distinctions of colors. Publications may or may not accept color in plots, and subtle shading or hatching may not be preserved. Sometimes the resolution of images of plots are downsampled, making small text or symbols unclear.

In general, a plot and its caption should more-or-less be able to stand alone. Obviously, you can't describe your whole study in a caption, but it is helpful to give enough information so that the figure is informative if it were separated from the rest of the article or presentation. The amount of information will be tempered by the style of medium you're writing for, journal versus factsheet, for example. In a scientific article, you may wish to include the subjects ("students in seventh grade who had completed a 4-H program in robotics"), location, time period, and other relevant information.

Be sure to label axes as thoroughly as possible, and also accurately. For example, an "increase in student scores" is not the same as "student scores." Indicate if the values are cumulative, total, monthly, etc.

When possible, plots should show some measures of variation, such as standard deviation, standard error of the mean, or confidence interval. By their nature, histograms and box plots show the variation in the data. Plots of means or medians will need to have error bars added to show dispersion or confidence limits.

If applicable, usually the *x*-axis is the independent variable and the *y*-axis is the dependent variable of an analysis.

Misleading plots

You should also avoid making misleading plots. Remember that you are trying to accurately summarize your data and show the relationships among variables. It takes care to do this accurately and in a way that won't confuse or mislead your reader.

The "Optional analyses" section on "Misleading and disorienting plots" near the end of this chapter gives some examples of plots I've seen that display their shortcomings.

Using software to produce plots

The plots in this book will be produced using R. R has the capability to produce informative plots quickly, which is useful for exploring data or for checking model assumptions. It also has the ability to produce more refined plots with more options, quintessentially through using the package *ggplot2*.

Sometimes it is easier to produce plots using software with a graphic user interface like Microsoft Excel, Apple Numbers, or free software like LibreOffice Calc.

In general, you should use whichever tool will best serve your purposes.

Exporting plots

RStudio makes exporting plots from R relatively easy. For lower-resolution images, plots can be exported directly as image files like .jpg or .png. For higher-resolution images, images can be exported as .pdf. See the [Mangiafico](#) reference in the “Optional readings I” section for some brief instructions.

Also, see “Exercises A” in this book for code on exporting higher resolution image files.

Examples of basic plots for interval/ratio and ordinal data

There are many types of plots, and this chapter will present only a few of the most common ones. In later chapters, variants of these plots will be used to explore the data or to check the assumptions of the analyses.

The type of plot you choose will depend upon on the number and types of variables you are trying to present, and what you are trying to show.

For this example, extension educators had students wear pedometers to count their number of steps over the course of a day. The following data are the result. *Rating* is the rating each student gave about the usefulness of the program, on a 1-to-10 scale.

Note the code that orders the levels of the factor variables. In general, R will alphabetize levels of factors, and this may be reflected on plots. The code will order the factor by the order that the levels are encountered in the dataset.

```
Input = "
Student  Sex   Teacher  Steps  Rating
a        female Catbus   8000   7
b        female Catbus   9000   10
c        female Catbus  10000  9
d        female Catbus   7000   5
e        female Catbus   6000   4
f        female Catbus   8000   8
g        male   Catbus   7000   6
h        male   Catbus   5000   5
i        male   Catbus   9000   10
j        male   Catbus   7000   8
k        female Satsuki  8000   7
l        female Satsuki  9000   8
m        female Satsuki  9000   8
n        female Satsuki  8000   9
o        male   Satsuki  6000   5
p        male   Satsuki  8000   9
q        male   Satsuki  7000   6
r        female Totoro  10000  10
s        female Totoro  9000   10
t        female Totoro  8000   8
u        female Totoro  8000   7
v        female Totoro  6000   7
w        male   Totoro  6000   8
x        male   Totoro  8000   10
y        male   Totoro  7000   7"
```

```

z      male    Totoro    7000    7
")

Data = read.table(textConnection(Input),header=TRUE)

### Order the factors, otherwise R will alphabetize them

Data$Sex = factor(Data$Sex,
                   levels=unique(Data$Sex))

Data$Teacher = factor(Data$Teacher,
                      levels=unique(Data$Teacher))

### Check the data frame

Data

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Histogram

A histogram shows the frequency of observations broken down into ranges, to give a sense of the range and distribution of the data.

- A single histogram is used for one interval/ratio or ordinal variable.
- Multiple histograms can be put on one plot to compare among groups.
- If you want to make a histogram for counts of a categorical variable in R, you will want to use a bar plot.

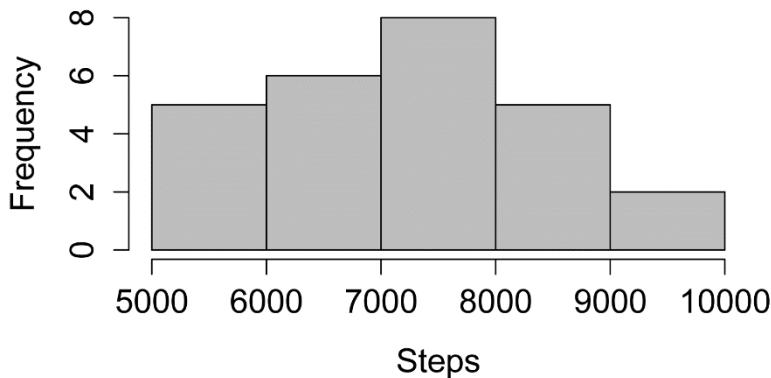
Note that *col* defines the colors of the bars; *main* defines the main plot title; *xlab* defines the label on the *x*-axis.

Simple histogram

```

hist(Data$Steps,
      col="gray",
      main="",
      xlab="Steps")

```



Frequency histogram of the number of steps taken by students in three classes.

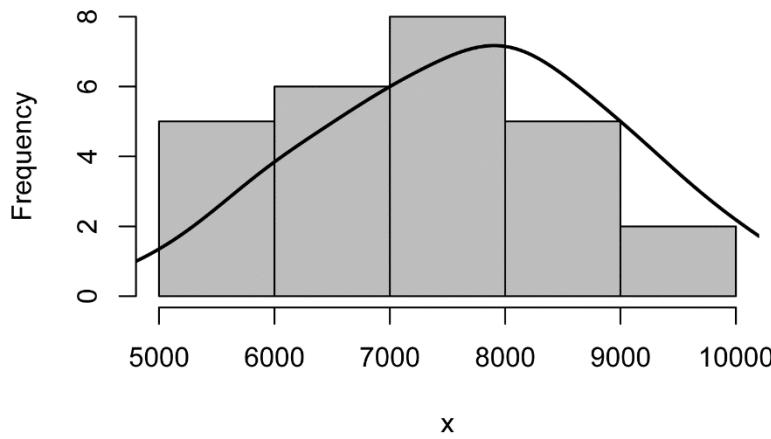
Histogram with density line

This plot will add a curve for the density of the data in the histogram, which is a smoothed-out representation of the distribution of the data.

The function *plotDensityHistogram* in the *rcompanion* package will produce this plot. Options include the options for the *hist* function as well as *adjust*, *bw*, and *kernel*, which is passed to the *density* function.

```
library(rcompanion)
```

```
plotDensityHistogram(Data$Steps,
                     adjust = 1)  ### Decrease this number
                           ### to make line less smooth
```



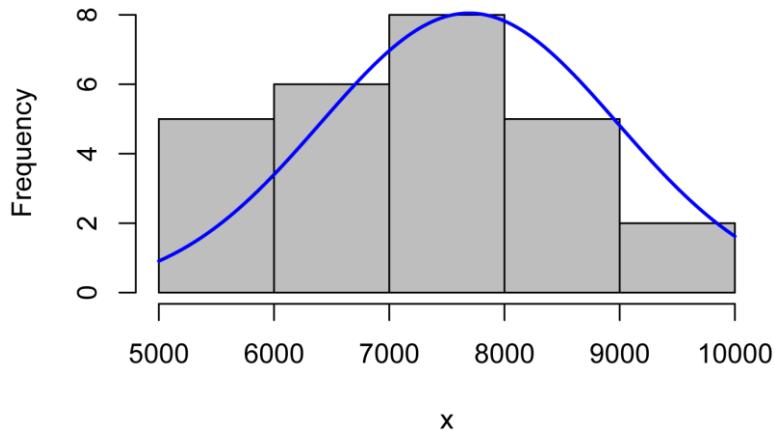
Histogram of the number of steps taken by students in three classes with the smoothed gaussian kernel density shown as a black curve.

Histogram with normal curve

This plot will add a normal curve with the mean and standard deviation of the data in the histogram. This type of plot will be useful to visually determine if a distribution of data is close to normal.

The function *plotNormalHistogram* in the *rcompanion* package will produce this plot. Options include the options for the *hist* function as well as *linecol* for the line color.

```
library(rcompanion)
plotNormalHistogram(Data$Steps)
```

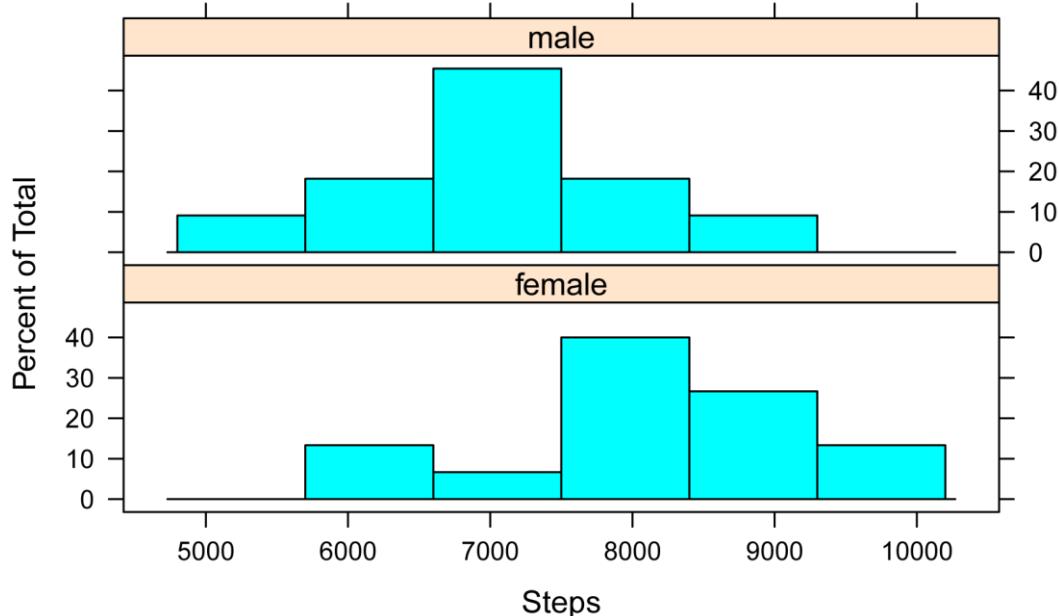


Histogram of the number of steps taken by students in three classes with a normal distribution of the same mean and standard deviation shown in blue.

Histogram for one-way data

The *lattice* package can be used to put multiple histograms on one plot for grouped data. This is useful to compare data distributions among groups. Notice the grammar used in the formula in the *histogram* function, and the need to indicate into how many columns and rows the plots will be placed.

```
library(lattice)
histogram(~ Steps | Sex,
          data=Data,
          layout=c(1,2)      # columns and rows of individual plots
        )
```

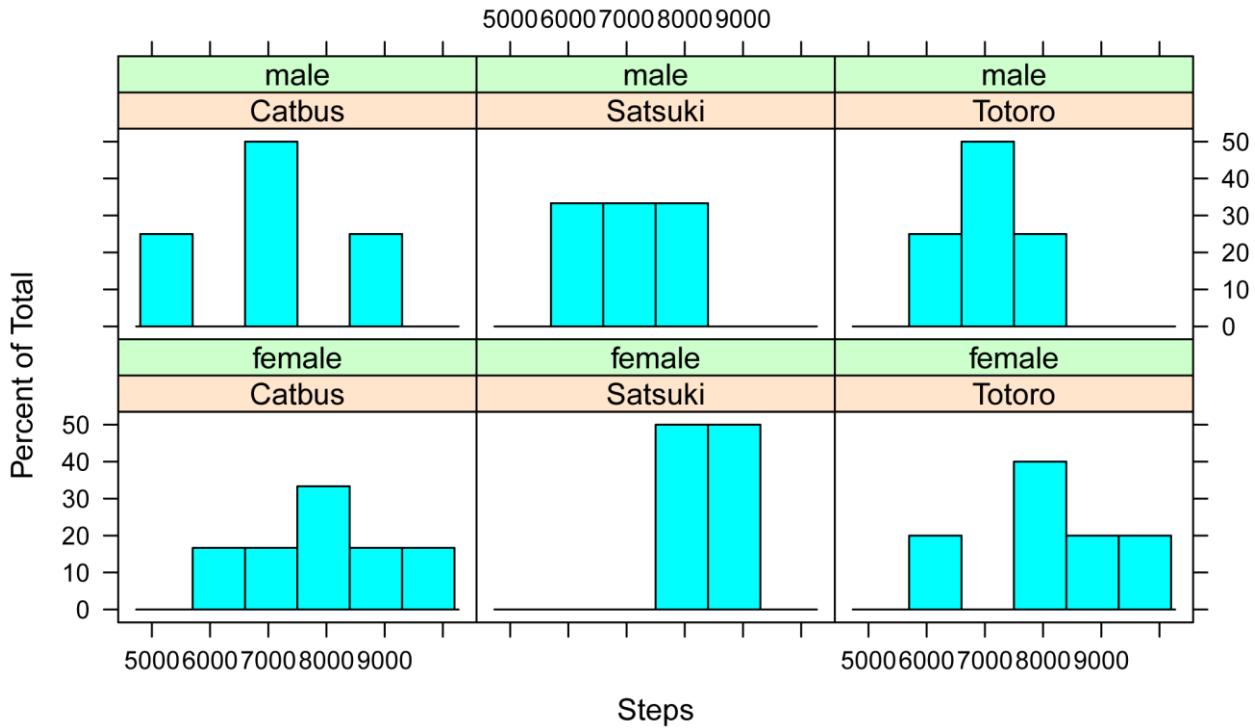


Histograms of the number of steps taken by male and female students across three students.

Histogram for two-way data

```
library(lattice)

histogram(~ Steps | Teacher + Sex,
          data=Data,
          layout=c(3,2)      # columns and rows of individual plots
)
```



Histograms of the number of steps taken by male and female students in each of the classes of Catbus, Satsuki, and Totoro.

Box plots

Box plots graphically present the five-number summary. The main box indicates the 25th and 75th percentile of the plotted data. The median is indicated with a heavy line through the box. The whiskers extending from the box indicate the range of the data, but different software packages may use somewhat different statistics for the extent of the whiskers. For this reason, when you are presenting box plots, you should always specify what the length of whiskers indicate.

By default, the whiskers in the *boxplot* function indicate the range of the data, unless there are data that are further away from the box than 1.5 times the length of the box. In this case, the whiskers extend only 1.5 times the length of the box, and points beyond this distance are indicated with circles. An example of box plots with circles indicating outlying values is shown in the “Box plot for one-way data” section. If you wish to have the whiskers extend to the range of the data, and not display any point with circles, the *range=0* option can be used. See *?boxplots* and *?boxplot.stats* for more details.

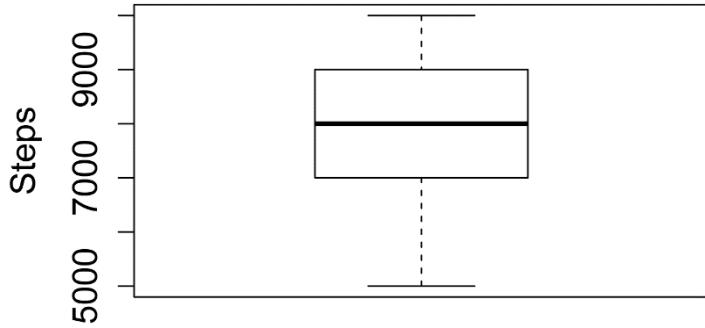
The video from Statistics Learning Center in the “Optional readings II” section may be a helpful introduction to box plots. The “Graphing Distributions” chapter in the Lane textbook in the “Optional readings II” section also has a nice description of box plots.

- A single box plot is used for one interval/ratio or ordinal variable. Don’t use means for ordinal data.
- Multiple box plots can be put on one plot to compare among groups.

Note that *xlab* defines the label on the x-axis; *ylab* defines the label on the y-axis

Simple box plot

```
boxplot(Data$Steps,
        ylab="Steps",
        xlab="")
```



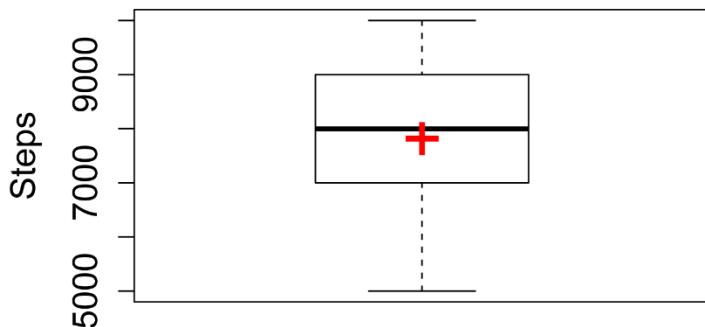
Box plot of the number of steps taken by students in three classes. The box indicates the 1st and 3rd quartiles, and the dark line indicates the median. Whiskers extend to the maximum and minimum of the data.

Box plot with means

```
M = mean(Data$Steps)

boxplot(Data$Steps,
        ylab="Steps",
        xlab="")

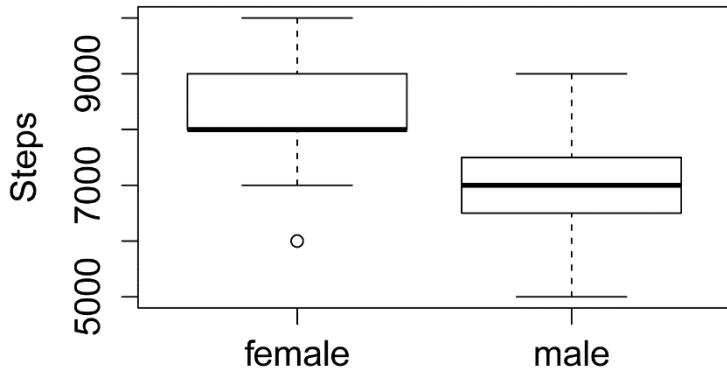
points(M,
       col="red",
       pch="+",    ### symbol to use, see ?points
       cex=2)      ### size of symbol
```



Box plot of the number of steps taken by students in three classes. The box indicates the 1st and 3rd quartiles, and the dark line indicates the median. The mean is indicated with the red cross. Whiskers extend to the maximum and minimum of the data.

Box plot for one-way data

```
boxplot(steps ~ Sex,
       data=Data,
       ylab="Steps")
```



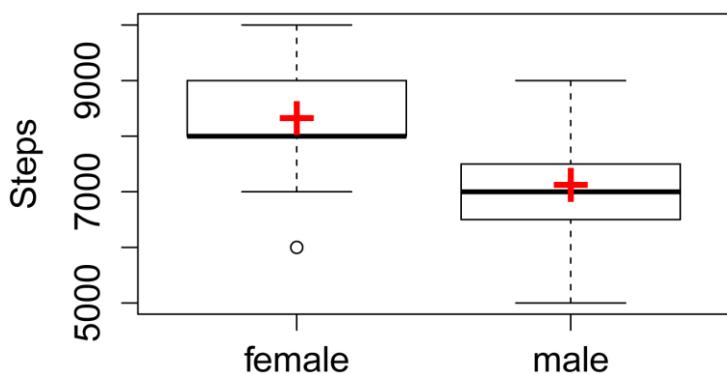
Box plots of the number of steps taken by male and female students across three classes. The boxes indicate the 1st and 3rd quartiles, and the dark lines indicate the median. Points more than 1.5 times the inter-quartile range away from the box are shown with hollow circles.

Box plot for one-way data with means

```
M = tapply(Data$Steps,
            INDEX = Data$Sex,
            FUN    = mean)
```

```
boxplot(steps ~ Sex,
        data=Data,
        ylab="Steps")
```

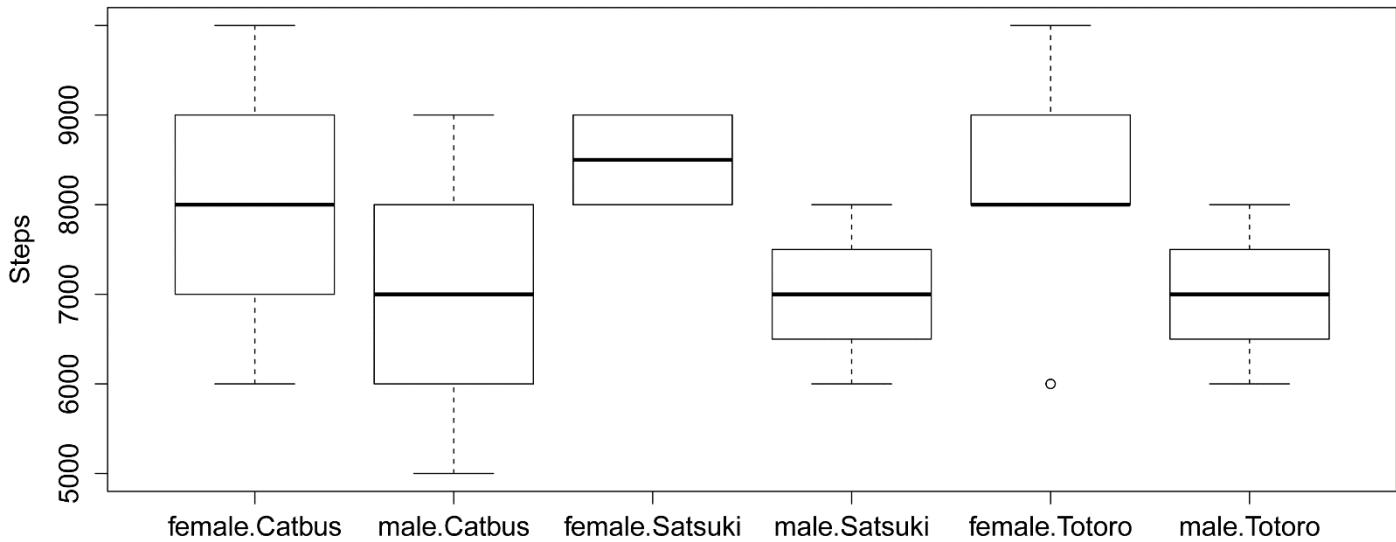
```
points(M,
       col="red",
       pch="+",
       cex=2)
```



Box plots of the number of steps taken by male and female students across three classes. The boxes indicate the 1st and 3rd quartiles, and the dark lines indicate the median. The means are indicated with red crosses. Points more than 1.5 times the inter-quartile range away from the box are shown with hollow circles.

Box plot for two-way data

```
boxplot(steps ~ Sex + Teacher,
       data=Data,
       ylab="Steps")
```



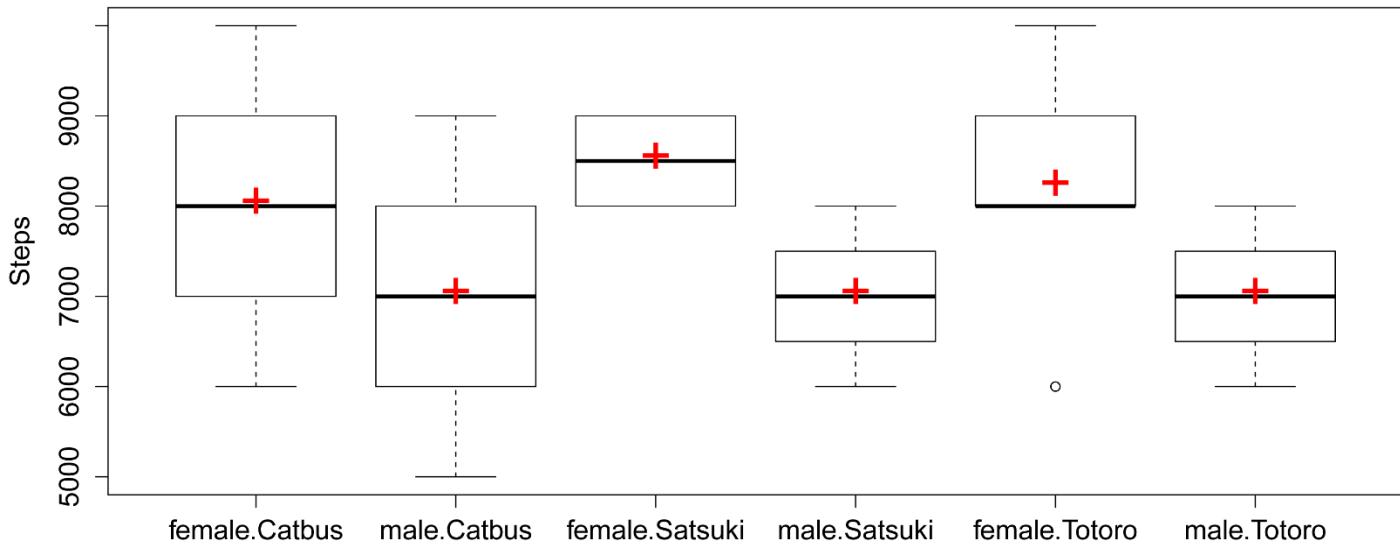
Box plots of the number of steps taken by male and female students in each of three classes. The boxes indicate the 1st and 3rd quartiles, and the dark lines indicate the median. Points more than 1.5 times the inter-quartile range away from the box are shown with hollow circles.

Box plot for two-way data with means

```
M = tapply(Data$Steps,
            INDEX = Data$Teacher : Data$Sex,
            FUN    = mean)

boxplot(steps ~ Sex + Teacher,
        data=Data,
        ylab="Steps")

points(M,
       col="red",
       pch="+",
       cex=2)
```



Box plots of the number of steps taken by male and female students in each of three classes. The boxes indicate the 1st and 3rd quartiles, and the dark lines indicate the median. The means are indicated with red crosses. Points more than 1.5 times the inter-quartile range away from the box are shown with hollow circles.

Plot of means and interaction plots

Means or medians for a group can be plotted with error bars showing standard deviation, standard error, or confidence interval for the statistic.

For two-way data, these plots are sometimes called *interaction plots*.

- Each mean or median represents one group or treatment for an interval/ratio or ordinal variable. Error bars are associated with each mean or median. Don't use means for ordinal data.
- Multiple means or medians can be plotted on the same plot, with groups from one or two independent variables.

The package *ggplot2* will be used for this type of plot. A simple interaction plot can be made with the *qplot* function, and more refined plots can be made with the *ggplot* function.

The *ggplot2* package is very powerful and flexible for making plots. The tradeoff is that the grammar can be difficult to understand. When trying to get a plot to look a certain way, it may be necessary to find an example plot online, and follow the code closely. Good resources include those by Hadley Wickham, Winston Chang, and Robbins and Robbins in the "References" section.

Plot of means for one-way data with *qplot*

In this example, we will plot means and confidence intervals. These will first be calculated with the function *groupwiseMean*. The output of the function is a data frame, we will call *Sum*. *Sum* will then be passed to the *qplot* function, so the variables mentioned in the code are from the *Sum* data frame.

```
library(rcompanion)

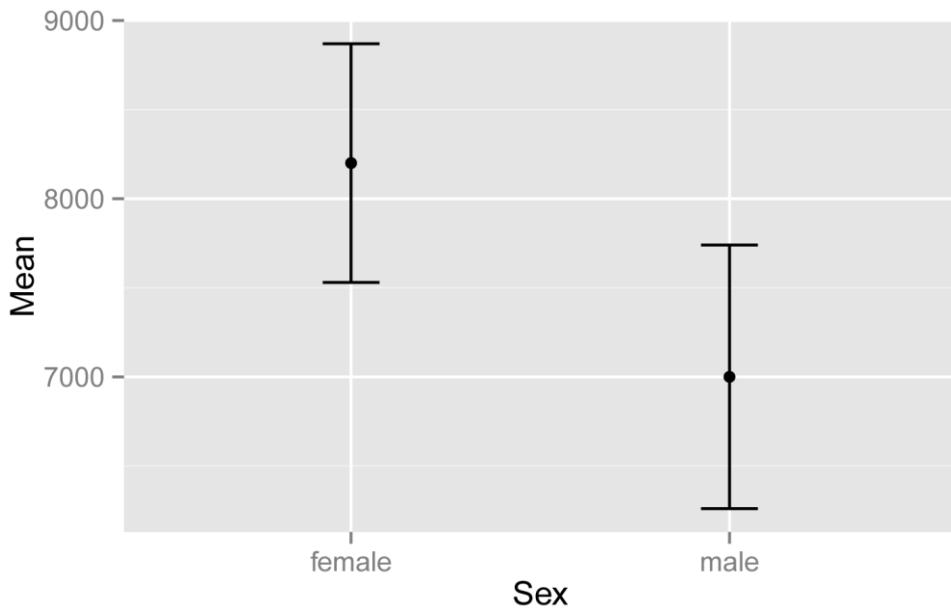
Sum = groupwiseMean(Steps ~ Sex,
                     data   = Data,
                     conf   = 0.95,
                     digits = 3)
```

```
Sum
```

	Sex	n	Mean	Conf.level	Trad.lower	Trad.upper
1	female	15	8200	0.95	7530	8870
2	male	11	7000	0.95	6260	7740

```
library(ggplot2)

qplot(x      = Sex ,
       y      = Mean,
       data = Sum) +
geom_errorbar(aes(
                    ymin  = Trad.lower,
                    ymax  = Trad.upper,
                    width = 0.15))
```



Plot of mean steps taken versus sex, for students in the classes of Catbus, Satsuki, and Totoro. Error bars indicate the traditional 95% confidence intervals for the mean.

Plot of means with *ggplot*

This example will plot the same statistics, but will make a more attractive plot using the *ggplot* function. Note that the variables mentioned in the code are from the *Sum* data frame.

The code invokes the black-and-white visual theme with the option *theme_bw*.

```
library(rcompanion)

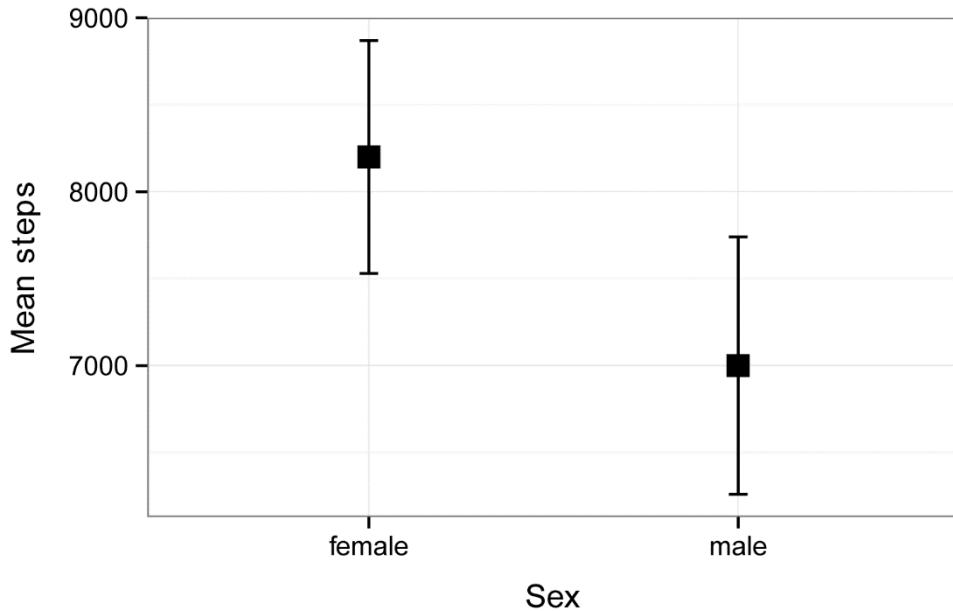
Sum = groupwiseMean(Steps ~ Sex,
                     data   = Data,
                     conf   = 0.95,
                     digits = 3)

Sum

      Sex  n Mean Conf.level Trad.lower Trad.upper
1 female 15 8200      0.95      7530      8870
2   male 11 7000      0.95      6260      7740

library(ggplot2)

ggplot(Sum,                   ### The data frame to use.
       aes(x = Sex,
           y = Mean)) +
  geom_errorbar(aes(ymin = Trad.lower,
                    ymax = Trad.upper),
                width = 0.05,
                size  = 0.5) +
  geom_point(shape = 15,
             size  = 4) +
  theme_bw() +
  theme(axis.title  = element_text(face  = "bold")) +
  ylab("Mean steps")
```



Plot of mean steps taken versus sex, for students in the classes of Catbus, Satsuki, and Totoro. Error bars indicate the traditional 95% confidence intervals for the mean.

Interaction plot of means with qplot

This example will plot means and standard errors for the interaction of two independent variables, *Teacher* and *Sex*.

We will use the *Summarize* function to produce the data frame *Sum*, and will then add a variable *se* to the data frame for the standard error.

Note that the default color palette of *ggplot* uses a pinkish color for the first group in the legend and a blueish color for the second. I wasn't intentionally matching females and males with their traditional colors. These colors could be changed to avoid the appearance of gender essentialism.

```
library(FSA)

Sum = Summarize(steps ~ Teacher + Sex,
                 data=Data)

Sum$se = Sum$sd / sqrt(sum$n)

### This creates a new variable for standard error in our summary data frame
### Replace n with nvalid if there are missing values
```

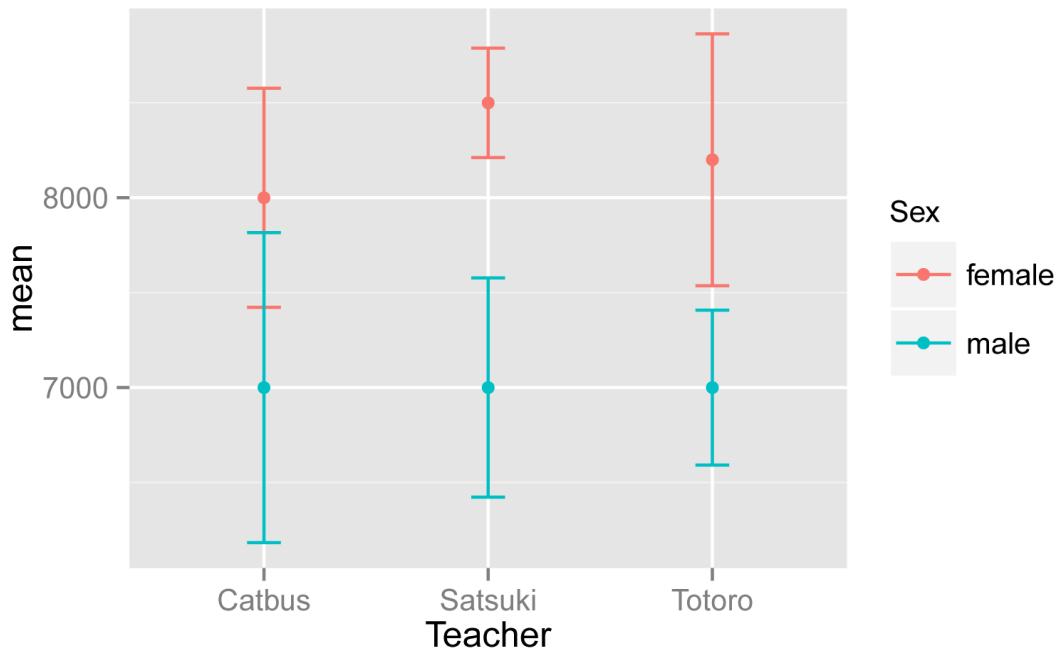
```
Sum
```

Teacher	Sex	n	mean	sd	min	Q1	median	Q3	max	percZero	se
1 Catbus	female	6	8000	1414.2136	6000	7250	8000	8750	10000	0	577.3503
2 Satsuki	female	4	8500	577.3503	8000	8000	8500	9000	9000	0	288.6751
3 Totoro	female	5	8200	1483.2397	6000	8000	8000	9000	10000	0	663.3250
4 Catbus	male	4	7000	1632.9932	5000	6500	7000	7500	9000	0	816.4966

5 Satsuki	male	3	3	7000	1000.0000	6000	6500	7000	7500	8000	0 577.3503
6 Totoro	male	4	4	7000	816.4966	6000	6750	7000	7250	8000	0 408.2483

```
library(ggplot2)

qplot(x      = Teacher,
      y      = mean,
      color = Sex,
      data = Sum) +
  geom_errorbar(aes(ymin = mean - se,
                     ymax = mean + se,
                     width = 0.15))
```



Interaction plot of mean steps taken versus teacher for males and female students. Error bars indicate the standard error of the mean.

Interaction plot of means with ggplot

As in the example above, this example will plot means and standard errors for the interaction of two independent variables, *Teacher* and *Sex*. As noted above, the pink and blue colors are the default color palette in *ggplot*.

```
library(FSA)

Sum = Summarize(steps ~ Teacher + Sex,
                 data=Data)

Sum$se = Sum$sd / sqrt(Sum$n)

### This creates a new variable for standard error in our summary data frame
```

```
### Replace n with nvalid if there are missing values
```

Sum

	Teacher	Sex	n	mean	sd	min	Q1	median	Q3	max	percZero	se
1	Catbus	female	6	8000	1414.2136	6000	7250	8000	8750	10000	0	577.3503
2	Satsuki	female	4	8500	577.3503	8000	8000	8500	9000	9000	0	288.6751
3	Totoro	female	5	8200	1483.2397	6000	8000	8000	9000	10000	0	663.3250
4	Catbus	male	4	7000	1632.9932	5000	6500	7000	7500	9000	0	816.4966
5	Satsuki	male	3	7000	1000.0000	6000	6500	7000	7500	8000	0	577.3503
6	Totoro	male	4	7000	816.4966	6000	6750	7000	7250	8000	0	408.2483

```
library(ggplot2)
```

```
pd = position_dodge(.2)      ### How much to jitter the points on the plot
```

```
ggplot(Sum,                   ### The data frame to use.
```

```
  aes(x      = Teacher,
       y      = mean,
       color = Sex)) +
```

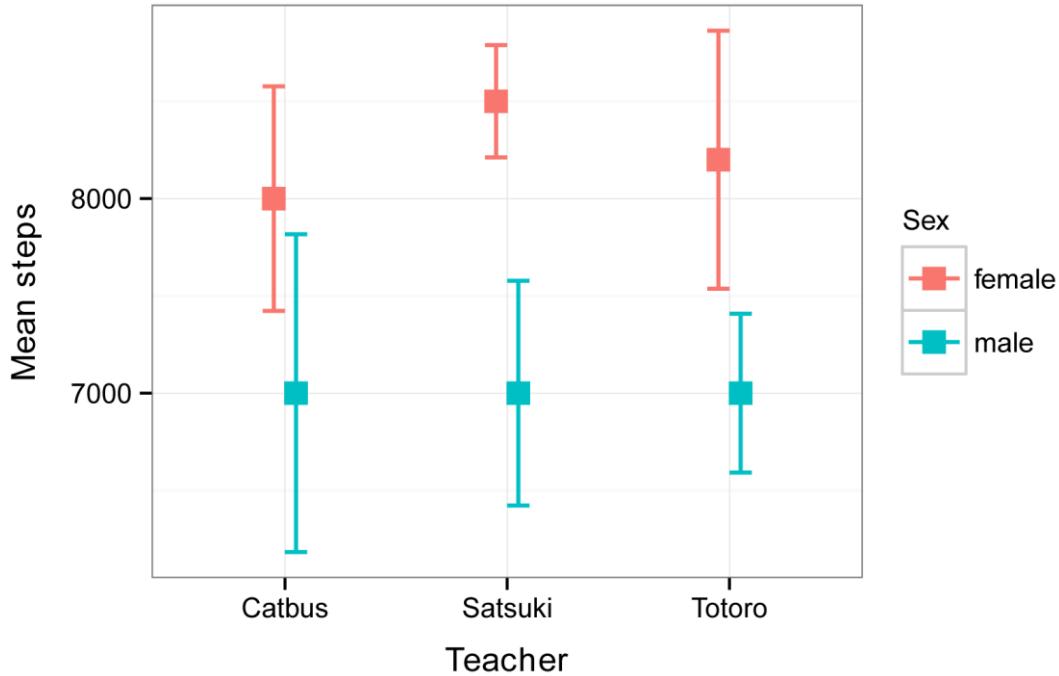
```
  geom_point(shape = 15,
             size  = 4,
             position = pd) +
```

```
  geom_errorbar(aes(ymin = mean - se,
                     ymax  = mean + se),
                 width = 0.2,
                 size  = 0.7,
                 position = pd) +
```

```
  theme_bw() +
```

```
  theme(axis.title = element_text(face = "bold")) +
```

```
  ylab("Mean steps")
```



Interaction plot of mean steps taken versus teacher for males and female students. Error bars indicate the standard error of the mean.

Bar plot of means

Another common method to plot means or medians is with bar plots. Error bars can be added to each bar.

- Each bar represents one group mean or median for an interval/ratio or ordinal variable. Error bars are associated with each mean or median. Don't use means for ordinal data.
- Multiple means or medians can be plotted on the same plot, with groups from one or two independent variables.

The *barplot* function can produce a simple bar plot, from a table of numbers. For more complex bar plots, or to add error bars, the *ggplot2* package can be used.

Simple bar plot of means

```
library(FSA)
```

```
Sum = Summarize(steps ~ Sex,
                 data=Data)
```

```
Sum
```

	Sex	n	nvalid	mean	sd	min	Q1	median	Q3	max	perczero
1	female	15	15	8200	1207.122	6000	8000	8000	9000	10000	0

```
2   male 11      11 7000 1095.445 5000 6500    7000 7500 9000          0
```

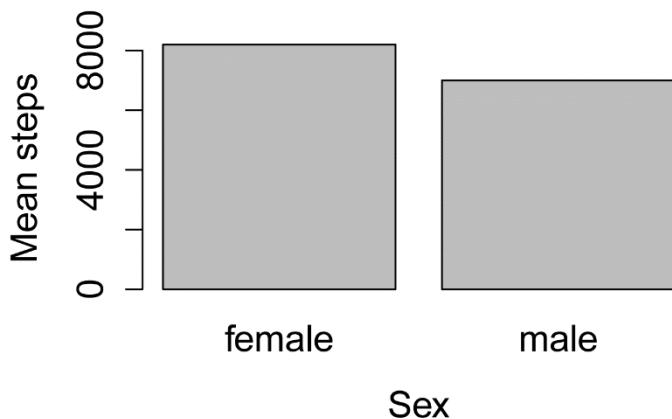
```
Table = as.table(sum$mean)
```

```
rownames(Table) = Sum$Sex
```

```
Table
```

female	male
8200	7000

```
barplot(Table,
        ylab="Mean steps",
        xlab="Sex")
```



Bar plot of the mean steps taken by female and male students across three classes.

Bar plot of means for one-way data with *qplot*

In this example, we will plot means and confidence intervals. These will first be calculated with the function *groupwiseMean*. The output of the function is a data frame, which we will call *Sum*. *Sum* will then be passed to the *qplot* function, so that the variables mentioned in the code are from the *Sum* data frame.

```
library(rcompanion)
```

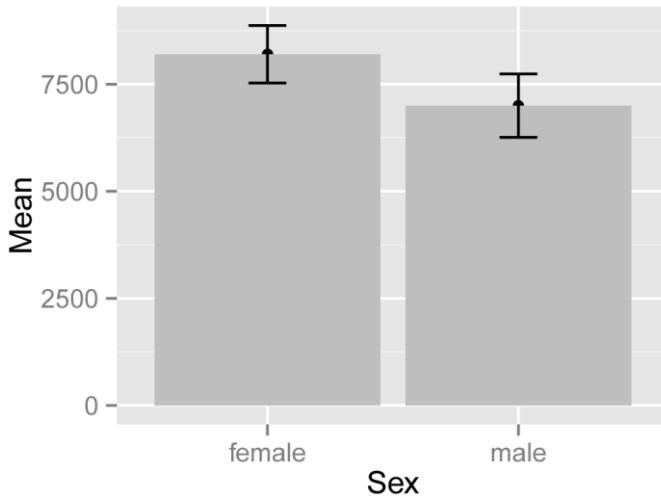
```
Sum = groupwiseMean(Steps ~ Sex,
                     data   = Data,
                     conf   = 0.95,
                     digits = 3)
```

```
Sum
```

	Sex	n	Mean	Conf.level	Trad.lower	Trad.upper
1	female	15	8200	0.95	7530	8870
2	male	11	7000	0.95	6260	7740

```
library(ggplot2)

qplot(x      = Sex ,
      y      = Mean,
      data = Sum) +
  geom_bar(stat ="identity",
            fill = "gray") +
  geom_errorbar(aes(ymin = Trad.lower,
                    ymax  = Trad.upper,
                    width = 0.15))
```



Bar plot of the mean steps taken by female and male students across three classes. Error bars indicate traditional 95% confidence intervals of the means.

Interaction bar plot of means with ggplot

This example will plot means and standard errors for the interaction of two independent variables, *Teacher* and *Sex*. We will use the *Summarize* function to produce the data frame *Sum*, and will then add a variable *se* to the data frame for the standard error.

Note that the default color palette of *ggplot* uses a pinkish color for the first group in the legend and a blueish color for the second. These colors can be changed.

```
library(FSA)

Sum = Summarize(Steps ~ Teacher + Sex,
                data=Data)

Sum$se = Sum$sd / sqrt(Sum$n)

### This creates a new variable for standard error in our summary data frame
```

```
### Replace n with nvalid if there are missing values
```

Sum

	Teacher	Sex	n	mean	sd	min	Q1	median	Q3	max	percZero	se
1	Catbus	female	6	8000	1414.2136	6000	7250	8000	8750	10000	0	577.3503
2	Satsuki	female	4	8500	577.3503	8000	8000	8500	9000	9000	0	288.6751
3	Totoro	female	5	8200	1483.2397	6000	8000	8000	9000	10000	0	663.3250
4	Catbus	male	4	7000	1632.9932	5000	6500	7000	7500	9000	0	816.4966
5	Satsuki	male	3	7000	1000.0000	6000	6500	7000	7500	8000	0	577.3503
6	Totoro	male	4	7000	816.4966	6000	6750	7000	7250	8000	0	408.2483

```
library(ggplot2)
```

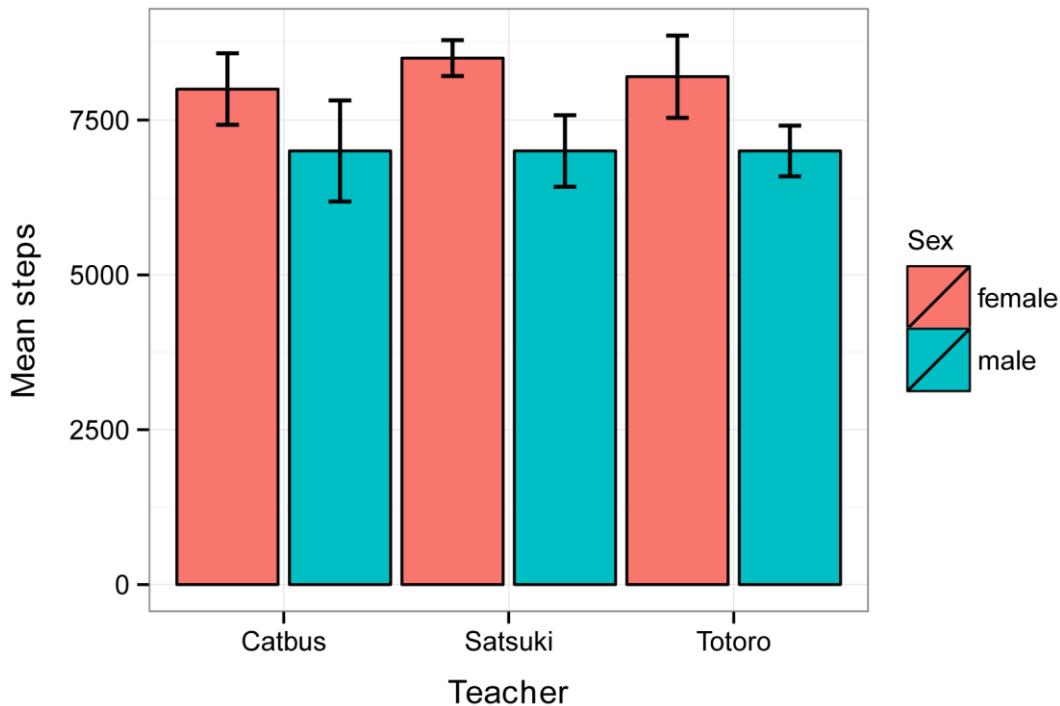
```
pd = position_dodge(1)      ### How much to jitter the bars on the plot,
                            ### 0.5 for overlapping bars

ggplot(Sum,                  ### The data frame to use.
       aes(x      = Teacher,
           y      = mean,
           fill   = Sex)) +

  geom_bar(stat    = "identity",
            color   = "black",
            position = pd) +

  geom_errorbar(aes(ymin = mean - se,
                    ymax   = mean + se),
                width = 0.2,
                size  = 0.7,
                position = pd,
                color = "black"
              ) +

  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Mean steps")
```



Bar plots of the mean steps taken by female and male students in each of three classes. Error bars indicate the standard error of the means.

Scatter plot

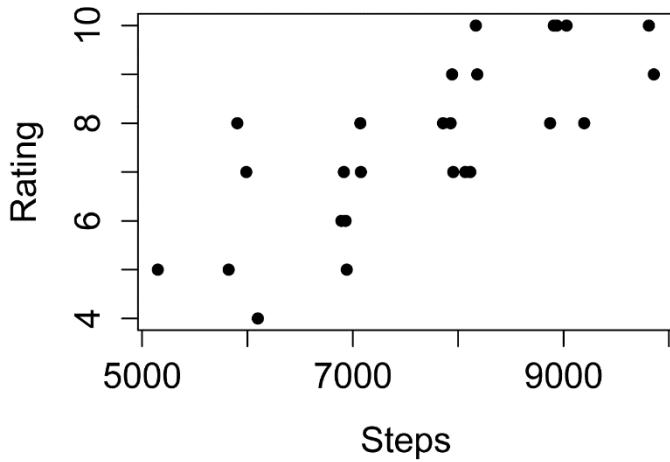
A scatter plot is used for bivariate data, to show the relationship between two interval/ratio or ordinal variables. Each point represents a single observation with one measured variable on the x-axis, and one measured variable is y-axis.

- Data are two interval/ratio or ordinal variables, paired by observation.
- Multiple plots can be shown together to investigate the relationships among multiple measured variables. (Not shown in this chapter.)

Simple scatter plot

The *plot* function can produce a simple bivariate scatter plot. The *xlab* and *ylab* options indicate the labels for the x- and y-axes. The *jitter* function is needed only in cases where individual points will overlap on the plot.

```
plot(Rating ~ jitter(Steps),    # jitter offsets points so you can see them all
      data=Data,
      pch = 16,                  # shape of points
      cex = 1.0,                 # size of points
      xlab="Steps",
      ylab="Rating")
```

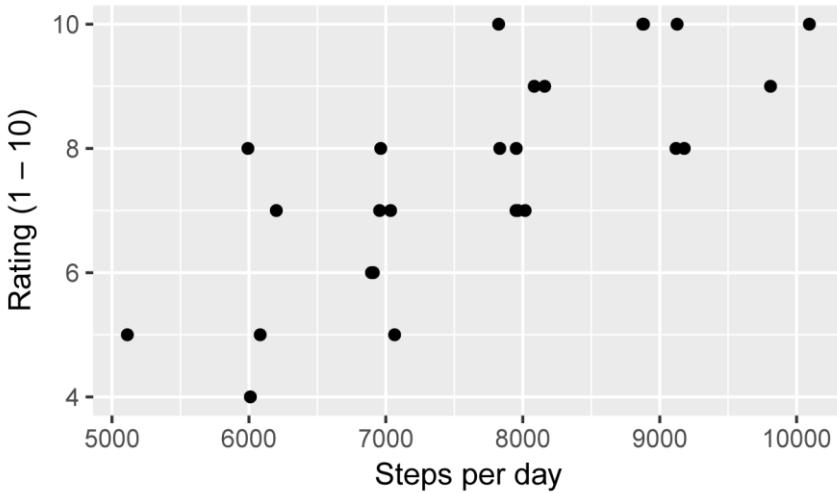


Plot of program rating versus steps taken by students in three classes.

Scatter plot with *qplot*

The *jitter* function is needed only in cases where individual points will overlap on the plot.

```
qplot(x      = jitter(Steps),
      y      = Rating,
      data = Data,
      xlab = "Steps per day",
      ylab = "Rating (1 - 10)")
```



Plot of program rating versus steps taken by students in three classes.

Examples of basic plots for nominal data

Plots for nominal data show the counts of observation for each category of the variable.

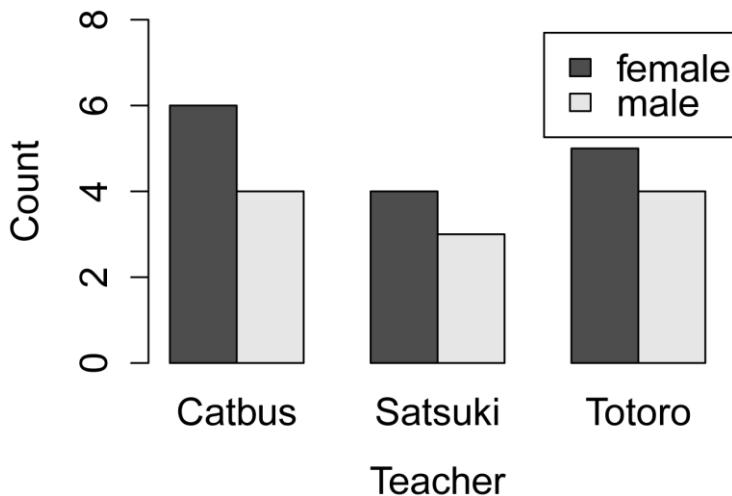
Bar plot for counts of a nominal variable

The `xtabs` function is used to count observations for each level of the nominal variables, producing a contingency table.

```
XT = xtabs(~ Sex + Teacher, data=Data)

XT
      Teacher
Sex      Catbus Satsuki Totoro
female     6       4       5
male       4       3       4

barplot(XT,
        beside = TRUE,
        legend = TRUE,
        ylim   = c(0, 8),    # adjust to remove legend overlap
        xlab   = "Teacher",
        ylab   = "Count")
```



Bar plot for frequencies of observations for female and male students in each of three classes.

Bar plot of counts and confidence intervals with `ggplot`

This example will plot the counts of observations for the interaction of two independent variables, `Teacher` and `Sex`. We will use the `Summarize` function to produce the data frame `Sum`, and will use the variable `n` as the count of observations.

It's not necessary you understand the code completely, but in order to demonstrate error bars on this plot, 95% confidence intervals for the counts will be produced with the `MultinomialCI` function in the `DescTools` package. We call the results of this function `MCI`. These results will then be converted from proportions to counts and added to the `Sum` data frame.

We will use the `scale_fill_manual` option at the end of the code to change the fill on the bars to grays.

Note that this example uses two-way data in a data frame, and extracts the count of observations from the *Summarize* function. A similar plot, using one-way data, and starting with a vector of counts, is shown in the “Multinomial test example with plot and confidence intervals” in the *Goodness-of-Fit Tests for Nominal Variables* chapter.

```

library(FSA)

Sum = Summarize(steps ~ Teacher + Sex,
                 data=Data)

library(DescTools)

MCI = MultinomCI(Sum$n,
                  conf.level=0.95)

Total = sum(Sum$n)

Sum$lower = MCI[, 'lwr.ci'] * Total
Sum$upper = MCI[, 'upr.ci'] * Total

### The few lines above create new variables in the data frame Sum
## for the multinomial confidence intervals for n

Sum

  Teacher   Sex n nvalid mean      sd  min   Q1 median   Q3 max percZero lower upper
1 Catbus female 6     8000 1414.2136 6000 7250    8000 8750 10000      0     2 11.27107
2 Satsuki female 4     8500  577.3503 8000 8000    8500 9000  9000      0     0  9.27107
3 Totoro female 5     8200 1483.2397 6000 8000    8000 9000 10000      0     1 10.27107
4 Catbus male 4      7000 1632.9932 5000 6500    7000 7500  9000      0     0  9.27107
5 Satsuki male 3      7000 1000.0000 6000 6500    7000 7500  8000      0     0  8.27107
6 Totoro male 4      7000  816.4966 6000 6750    7000 7250  8000      0     0  9.27107

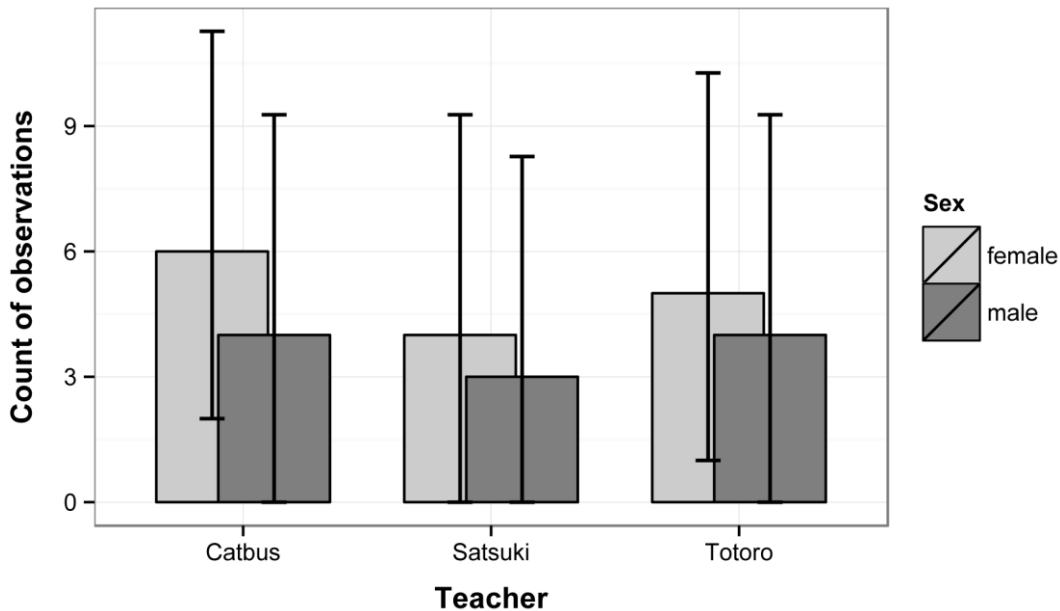
library(ggplot2)

pd = position_dodge(0.5)    ### How much of a gap for the bars on the plot,
                            ## 1 for non-overlapping bars

ggplot(Sum,                   ### The data frame to use.
       aes(x      = Teacher,
            y      = n,
            fill   = Sex)) +
  geom_bar(stat = "identity",
            color = "black",
            position = pd) +
  geom_errorbar(aes(ymin = lower,
                    ymax  = upper),
                width = 0.2,
                size  = 0.7,
                position = pd,
                color = "black")

```

```
) +
theme_bw() +
theme(axis.title = element_text(face = "bold")) +
scale_fill_manual(values= c("gray80", "gray50", "green")) +
ylab("Count of observations")
```



Bar plots of the counts of observations for female and male students in each of three classes. Error bars indicate the 95% multinomial confidence intervals for each count (Sison and Glaz method).

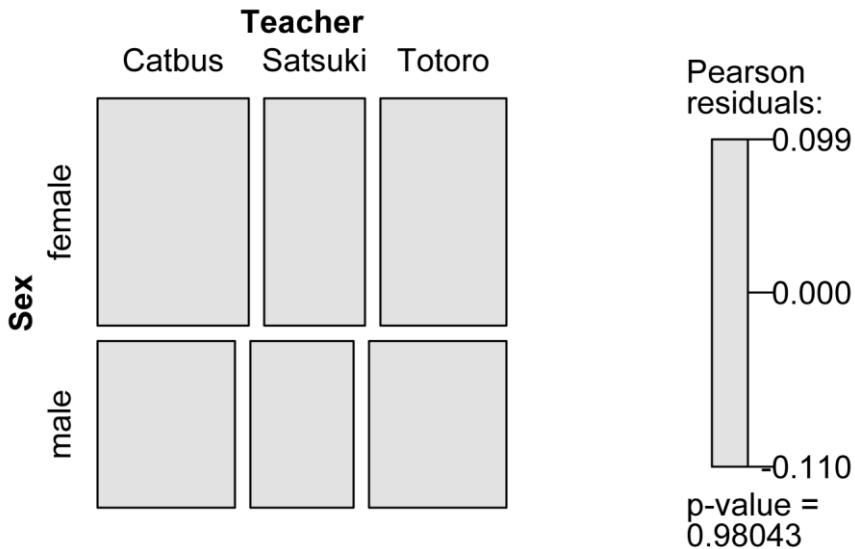
Mosaic plot

A mosaic plot can be constructed with the *mosaic* function in the *vcd* package.

Interpretation of mosaic plots is discussed in the *Introduction to Tests for Nominal Variables* chapter.

```
library(vcd)

mosaic(~ Sex + Teacher,
       data = Data,
       shade = TRUE,
       legend = TRUE)
```



Mosaic plot for frequencies of observations for female and male students in each class.

Ordering plot categories

The order in which factor variable levels are plotted can often be controlled by ordering the levels of the factor, for example,

```
Data$Teacher = factor(Data$Teacher,
                      levels=c("Totoro", "Catbus", "Satsuki"))

levels(Data$Teacher)

[1] "Totoro"  "Catbus"   "Satsuki"
```

If you are going to be using other variables from the data frame, for example for error bars or mean separation letters, it is sometimes best to reorder the whole data frame according to the factor variable you want to use, and then reorder the levels of that variable as well. Sometimes this is overkill, but it prevents any mismatching of the observations from the different variables.

```
library(rcompanion)

Sum = groupwiseMean(Steps ~ Teacher,
                     data    = Data,
                     conf    = 0.95,
                     digits = 3,
                     traditional = FALSE,
                     percentile = TRUE)

Sum

Teacher  n  Mean Conf.level Percentile.lower Percentile.upper

```

1 Totoro 9 7670	0.95	6890	8440
2 Catbus 10 7600	0.95	6700	8400
3 Satsuki 7 7860	0.95	7140	8570

```
Sum = Sum[order(factor(Sum$Teacher,
                         levels= c("Catbus", "Satsuki", "Totoro"))),]

Sum$Teacher = factor(Sum$Teacher,
                      levels= c("Catbus", "Satsuki", "Totoro"))

Sum

  Teacher  n Mean Conf.level Percentile.lower Percentile.upper
2  Catbus 10 7600      0.95        6700          8500
3  Satsuki 7 7860      0.95        7140          8570
1  Totoro 9 7670      0.95        6890          8440
```

Optional analyses: Describing histogram shapes

The following plots are histograms of water quality data for U.S. Geological Survey station 1467016, Rancocas Creek at Willingboro, NJ, from approximately 1970 to 1974 (USGS, 2015).

The distribution of *pH* follows a normal distribution relatively well. It is rather symmetric around a mean of about 6. A normal curve with the same mean and standard deviation as the data has been superimposed as a blue line.

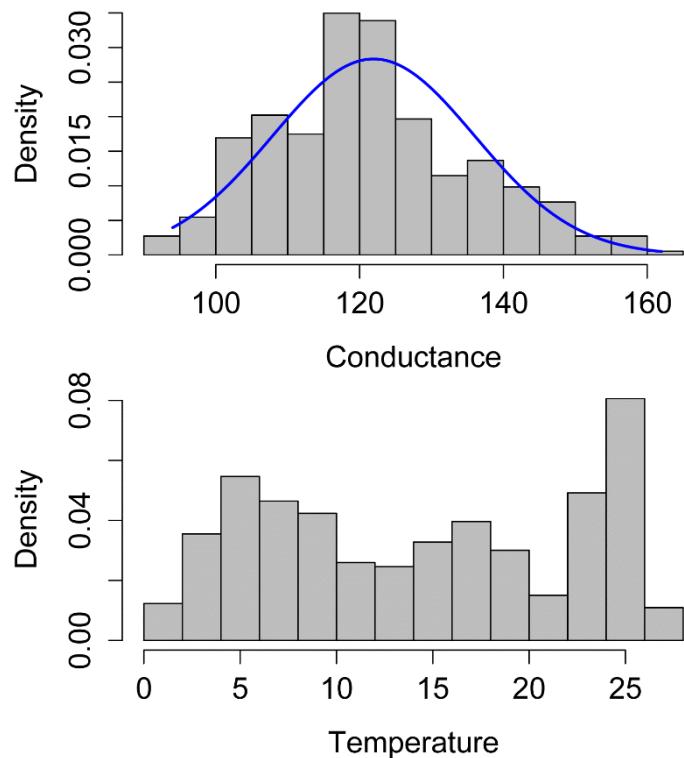
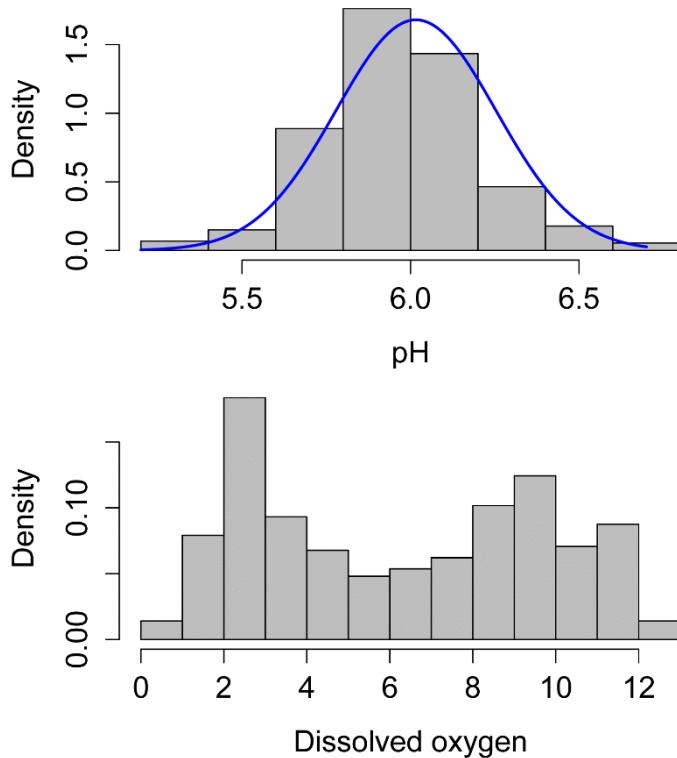
The distribution of *conductance* shows a slight right skew, but otherwise is relatively symmetric around a mean of about 120.

I would describe the distribution of *dissolved oxygen* as bimodal. It seems to have relatively many observations clustered around a value of 3 and around a value of 9.

I'm not sure how I would describe the distribution of *temperature*. If pressed I might say it is relatively uniform. A histogram of a uniform distribution would have all bars of equal height. You might see *temperature* as bimodal, or of indescribable shape.

Results from the *describe* function in the *psych* package are shown below the histograms, and kernel density plots of these distributions are shown below those results.

For examples of skewed histograms, see the “Skewness” sub-section of the “Statistics of shape for interval/ratio data” section in the *Descriptive Statistics* chapter.



```
### describe(pH)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	366	6.02	0.24	6	6.01	0.15	5.2	6.7	1.5	0.11	0.84	0.01

```
### describe(Conductance)
```

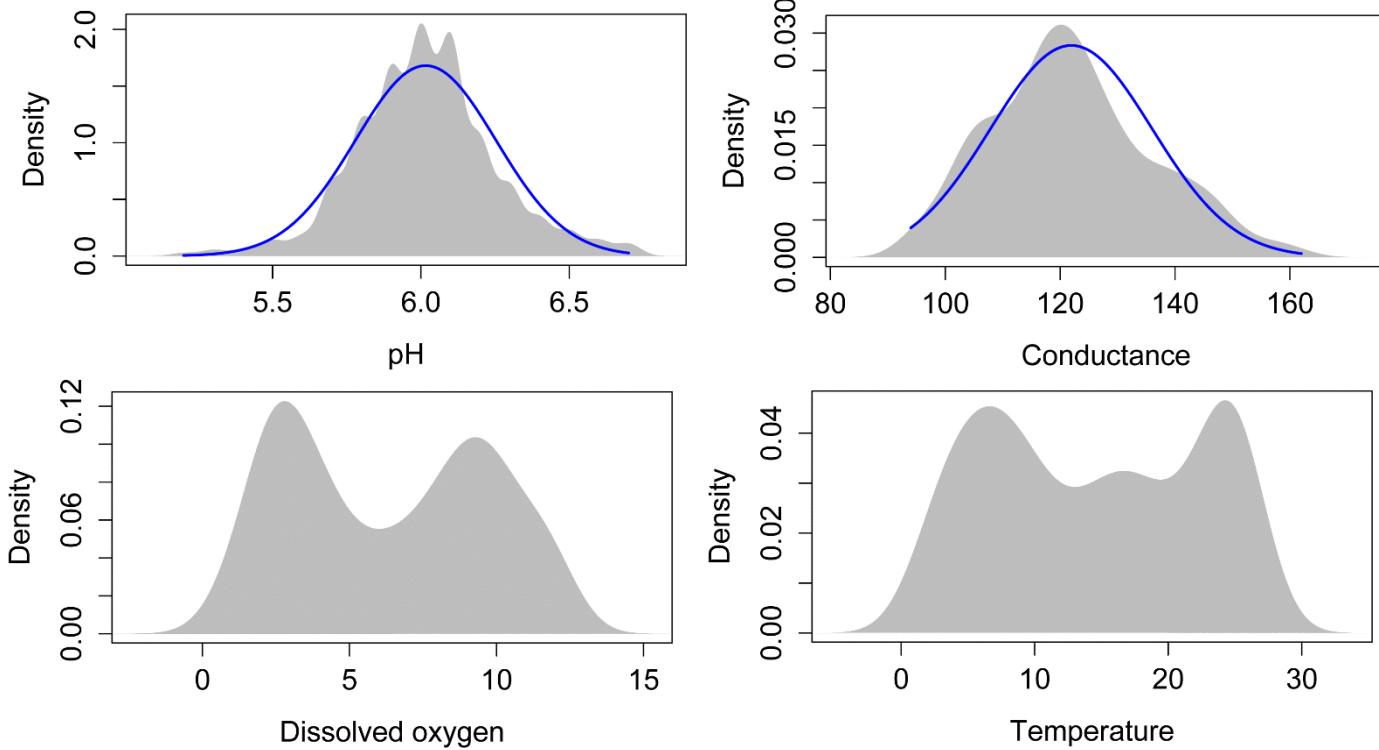
	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	366	121.95	14.08	121	121.33	13.34	94	162	68	0.38	-0.26	0.74

```
### describe(DO)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	354	6.32	3.43	6.35	6.25	4.82	0.5	12.4	11.9	0.09	-1.42	0.18

```
### describe(Temperature)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	366	14.48	7.98	14.55	14.54	11.64	1.5	27.1	25.6	0.02	-1.43	0.42



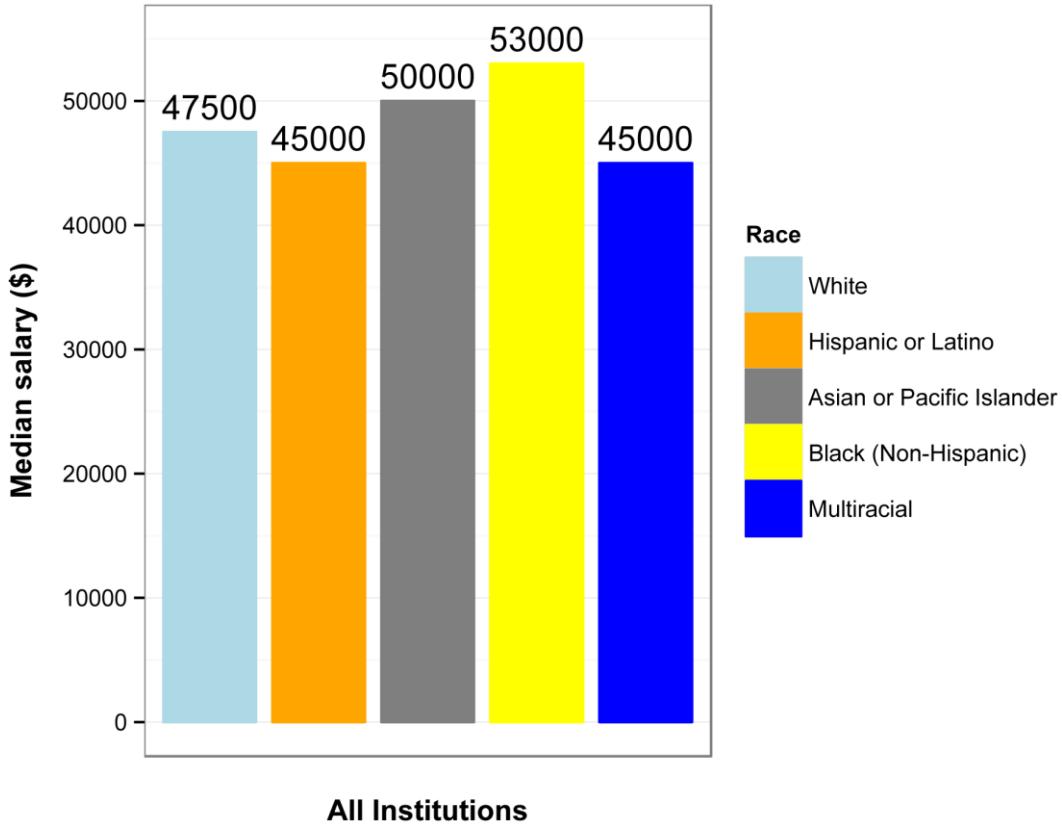
Optional analyses: Misleading and disorienting plots

The need to include measures of variation

The following plot mimics one I've seen, showing median salary for non-tenure-track university faculty by race or ethnicity. One minor complaint I have about it is that the *x*-axis label doesn't really describe the *x*-axis. Really, it should be labeled "Race or ethnicity", and the categories should be listed along the *x*-axis.

The larger complaint I have is that there are no measures of variation indicated for the medians. As presented, one might say, "Blacks have a higher median salary than do Whites, whose salary is higher still than that of Hispanics." But I suspect that the variation within each demographic category is much greater than the variation between them. If we had error bars on each bar, we might be less inclined to think that there is a real (statistically significant) difference among groups. Of course, it's impossible to draw any conclusions without their including measures of variation or statistical analyses.

As a final note, usually if there is no statistical difference among groups, there is no need to plot their values separately. In the example of this plot, if there were no statistical difference among racial groups, we could just report one median for the whole lot, with a five-number summary, or a single box plot. There are exceptions to this; for example, in this case if salaries among racial groups were of some particular interest, we might show statistics for each group even if there were no statistical differences.



Original caption: "Median academic-year salary for full-time, non-tenure-track faculty by race or ethnicity, Fall 2010."

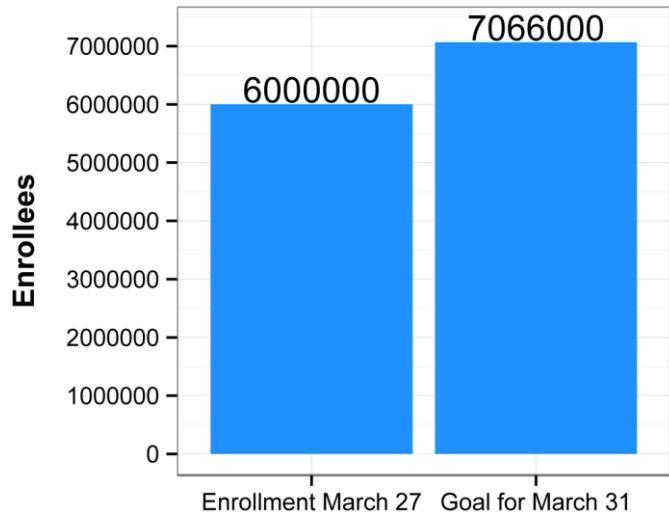
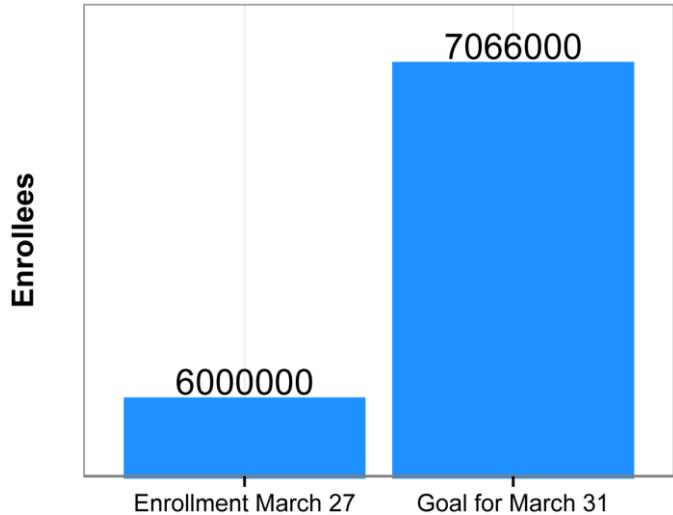
Original source: American Association of University Professors. 2013. Annual Report on the Economic Status of the Profession, 2012–2013. www.aaup.org/13EconomicStatus.

For bar charts, start the y-axis at zero

The following plots mimic plots I saw for the same data on *Fox News* and *The Guardian*. They compare the enrollment for the Affordable Care Act ("Obamacare") in March 2014 to the stated goal for enrollment in March 2014.

The main deficiency of the plot on the left is that the bottom of the *y*-axis doesn't represent zero, but 5.75 million, making the difference in the bars look large compared with the size of first bar. For good measure, units on the *y*-axis are left off.

The plot on the right more honestly depicts the relationship between the difference in the bars and the size of the bars by setting the bottom of the *y*-axis to zero. Also, units are included on the *y*-axis.



Avoid disorienting the audience

The following plot mimics one released by Reuters in 2014. I didn't try to mimic the aesthetics of the plot, but you can see the original in the article cited in "Original source" at the end of the section.

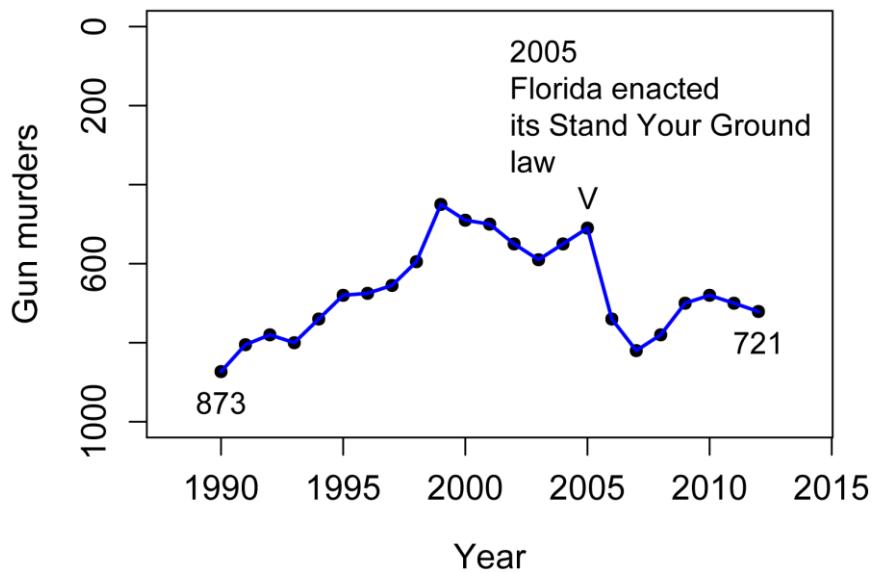
The original plot had the confusing heading and sub-heading, "Gun deaths in Florida: number of murders committed using firearms". Of course, murders aren't the same as gun deaths, so it's not clear what is being measured on the *y*-axis.

A quick look at the plot would suggest that gun murders dropped after 2005. But a more careful inspection reveals that the *y*-axis is inverted, in that smaller numbers are on the top and larger numbers are on the bottom. Therefore, gun murders went up after 2005!

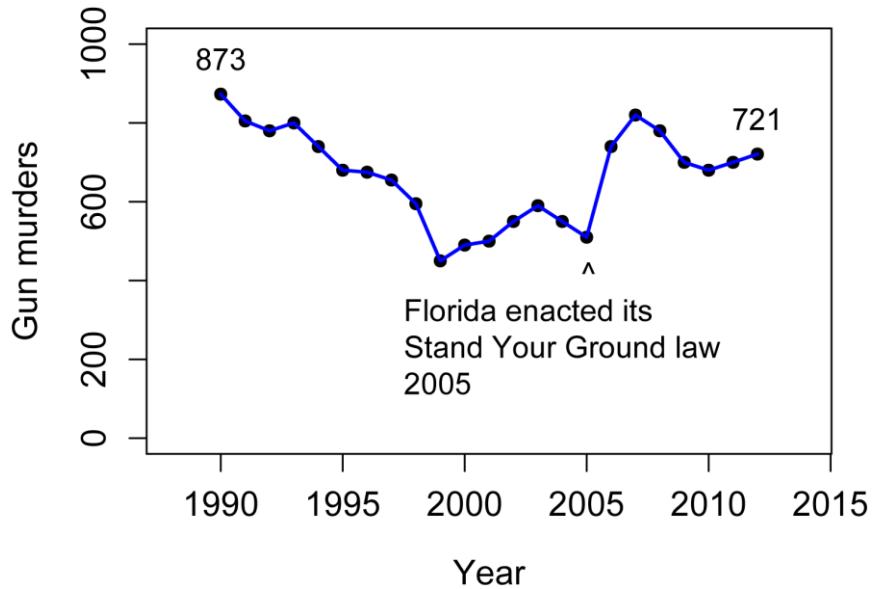
At first I found it difficult to believe that this plot could have been made in good faith. Most commonly we put larger numbers toward the top of the *y*-axis. I suspect, though, that the author was using a less-common convention that positive outcomes are put toward the top of the *y*-axis. It makes sense if you think of zeros murders as the goal, and increasing murders are further below that goal. An example of this convention is the Hickey article in the "Reference" the end of the section. In that article, actors with a low billing score are at the top of the *y*-axis, since first billing is the best.

Reuters did release a second plot with the *y*-axis in its normal orientation, which is mimicked in the second plot.

As a final note, there are cases where the *y*-axis is traditionally inverted. For example when plotting the depth to water table in groundwater hydrology, the surface (0 depth) is plotted at the top, so as the depth increases, the value is vertically lower on the plot.



First plot, with y-axis reversed.



Second plot, with y-axis in its common orientation.

Original source: Engel, P. 2014. This chart shows an alarming rise in Florida gun deaths after 'Stand Your Ground' was enacted. *Business Insider*. www.businessinsider.com/gun-deaths-in-florida-increased-with-stand-your-ground-2014-2.

Reference: Hickey, W. 2015. Congratulations, You've Been Cast In Star Wars! Will You Ever Work Again? *Five Thirty Eight*. fivethirtyeight.com/features/congratulations-youve-been-cast-in-star-wars-will-you-ever-work-again/.

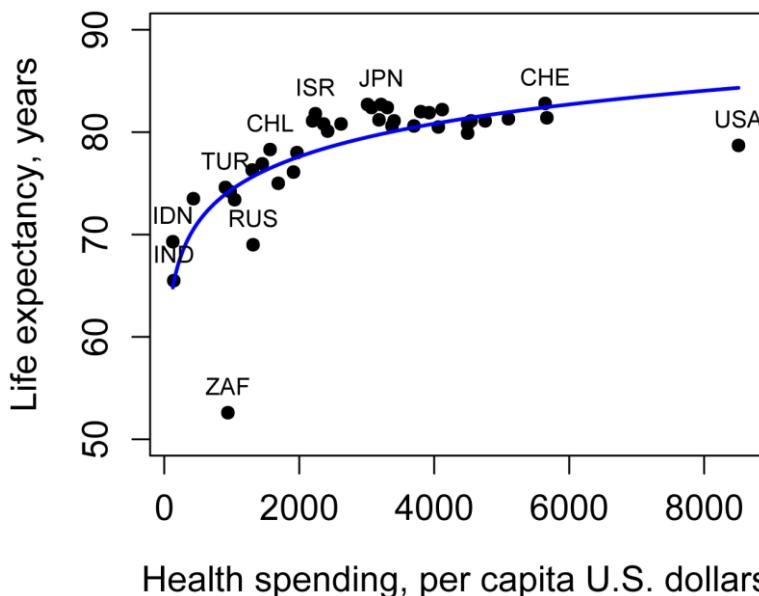
Use the best model

The following plot mimics one published by the Organisation for Economic Co-operation and Development. The model they fit to the data is shown as the blue curve. It's subtle, but the model makes assumptions that may not fit the data well. In particular, the author chose a simple logarithmic model for which y continues to increase as x increases. [In reality, I think they chose a model from the limited options in Excel, probably without thinking about it much.] One implication in this plot is that life expectancy in the United States, *USA*, is notably lower than what the model predicts.

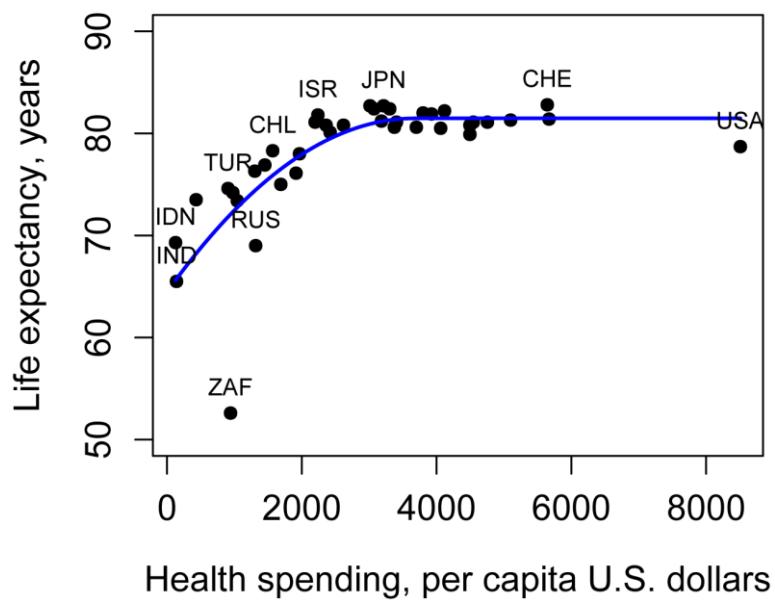
But I don't think there's any good reason to suspect that life expectancy would continue to increase as health care spending increases, beyond some limit. Life expectancy is probably limited by human biology, limits in medical science, and intractable human behaviors. Looking at just the data points and ignoring the blue curve, the data suggest to me that life expectancy increases as spending increases up to about 2500 dollars (about where Israel, *ISR*, is) and then plateaus. Spending beyond this critical value results in no further increase in life expectancy.

I created the model in the second plot, which uses a *quadratic plateau* model. With this model, a quadratic curve rises to a plateau. This model predicts a critical level of about 3500 dollars, about where Japan, *JPN*, is. It also suggests that life expectancy in the United States is lower than the model predicts, but not much more than Switzerland's, *CHE*, or Japan's is above it. But more importantly, this model and other similar alternatives predict a critical value in healthcare spending, above which we should not expect an increase in life expectancy. Of course, there are other factors that would affect life expectancy, and maybe spending above this critical value is warranted in some cases.

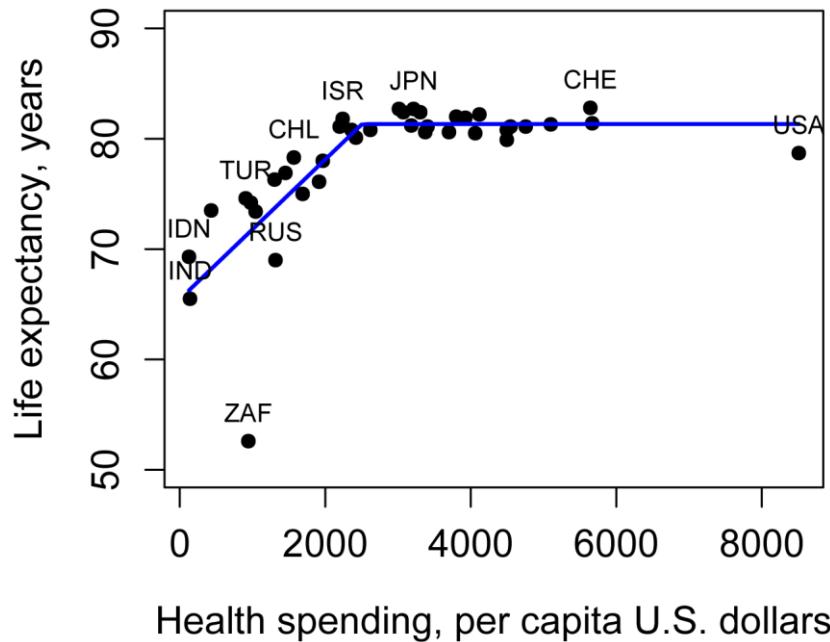
The third plot uses a *linear plateau* model, which is composed of a linear increasing segment and a plateau segment. It fits the data slightly better than the quadratic plateau model, and indicates a critical level near 2500 dollars.



First plot with logarithmic model.



Second plot with quadratic plateau model.



Third plot with linear plateau model.

Original source: OECD. 2013. *Health at a Glance 2013*. www.oecd.org/els/health-systems/Health-at-a-Glance-2013.pdf.

References

- Apple 2015. "Numbers for Mac." www.apple.com/mac/numbers/.
- Chang, W. "Graphs with ggplot2." No date. *Cookbook for R*. www.cookbook-r.com/Graphs/index.html.
- LibreOffice. 2015. "Calc." www.libreoffice.org/discover/calc/.
- Microsoft. 2015. "Excel." products.office.com/en-us/excel.
- Robbins, N.B. and J. Robbins. 2016. Effective graphs with Microsoft R Open. www.joyce-robbins.com/wp-content/uploads/2016/04/effectivegraphsmro1.pdf.
- [USGS] U.S. Geological Survey. 2015. National Water Information System: Web Interface. waterdata.usgs.gov/nwis/.
- Wickham, H. 2013. ggplot2 0.9.3.1. docs.ggplot2.org/current/.

Optional readings I

"General tips for all graphs" in "Guide to fairly good graphs" in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/graph.html.

"Exporting graphics" in "A Few Notes to Get Started with R" in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/a_06.html.

Optional readings II

[Video] **"Understanding summary statistics: The box plot"** from Statistics Learning Center (Dr. Nic). 2015. www.youtube.com/watch?v=bhkqq0w60Gc.

"Graphing Distributions", Chapter 2 in Lane, D.M. (ed.). No date. *Introduction to Statistics*, v. 2.0. onlinestatbook.com/. (*Stem and Leaf Displays, Histograms, Frequency Polygons, Box Plots, Bar Charts, Line Graphs, Dot Plots*.)

"Stem-and-Leaf Graphs (Stemplots), Line Graphs, and Bar Graphs", Chapter 2.1 in Openstax. 2013. *Introductory Statistics*. openstaxcollege.org/textbooks/introductory-statistics.

"Histograms, Frequency Polygons, and Time Series Graphs", Chapter 2.2 in Openstax. 2013. *Introductory Statistics*. openstaxcollege.org/textbooks/introductory-statistics.

"Box Plots", Chapter 2.4 in Openstax. 2013. *Introductory Statistics*. openstaxcollege.org/textbooks/introductory-statistics.

“Scatterplots for paired data”, **Chapter 1.6.1** in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

“Dot plots and the mean”, **Chapter 1.6.2** in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

“Histograms and shape”, **Chapter 1.6.3** in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

“Box plots, quartiles, and the median”, **Chapter 1.6.5** in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

“Contingency tables and bar plots”, **Chapter 1.7.1** in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

“Segmented bar and mosaic plot”, **Chapter 1.7.3** in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

Exercises E

1. Look at the histograms of the steps separated by male and female students.
 - a. Would you describe the distribution of each as symmetrical, skewed, bimodal, or uniform?
 - b. If skewed, in which direction?
2. Look at the box plot of the steps taken by all students.
 - a. Estimate the minimum, 25th percentile, median, 75th percentile, and maximum.
 - b. Write a few sentences interpreting these results. For example, what is the meaning of the median? Is the range of the data large relative to its median value? Are there outlying values? What can you say about the middle 50% of students?
3. In box plots including the mean, a symmetrical distribution will have the mean close to the median, and the box will be symmetrical on either side of the median line. Looking at the box plot of the steps taken by female and male students, would you describe the distributions of each as symmetrical or skewed?
4. Looking at the plot of mean steps for female and male students, estimate the mean of each.
5. It is justified to say that group means with non-overlapping 95% confidence intervals are statistically different. Looking at the plot of mean steps with confidence intervals for female and male students, are the two means statistically different?
- 6.

- a. Looking at the interaction plot for mean steps for female and male students by each class, do the error bars for standard error overlap or not for each teacher?
- b. Write a couple of sentences interpreting these results. Include the practical importance of the results. (For example, is the difference in means in a certain class large enough to be practically important? You might consider the difference in means as a percentage of the absolute value of the mean. Also, consider the spread of the data or the standard errors of the means.)

7. Look at the scatter plot.

- a. Does it look like there is a correlation or relationship between *Steps* and *Rating*, or not?
- b. If so, how would you describe it? ("As *Steps* increases, *Rating*...")
- c. Is the relationship of practical importance? (Consider, for example, how much *Rating* changes over the range of *Steps*.)

8. Look at any of the plots for nominal data.

- a. Are there more females or males in these classes?
- b. In which class is the greatest disparity?
- c. Write a couple of sentences reporting these results. (For example, are there big differences among the classes? What are the sum for male and female students? Do you conclude something interesting from these data?)

9–11. As part of a nutrition education program, extension educators had students keep diaries of what they ate for a day and then calculated the calories students consumed.

Student	Teacher	Sex	Calories	Rating
a	Tetsuo	male	2300	3
b	Tetsuo	female	1800	3
c	Tetsuo	male	1900	4
d	Tetsuo	female	1700	5
e	Tetsuo	male	2200	4
f	Tetsuo	female	1600	3
g	Tetsuo	male	1800	3
h	Tetsuo	female	2000	3
i	Kaneda	male	2100	4
j	Kaneda	female	1900	5
k	Kaneda	male	1900	4
l	Kaneda	female	1600	4
m	Kaneda	male	2000	4
n	Kaneda	female	2000	5
o	Kaneda	male	2100	3
p	Kaneda	female	1800	4

For each of the following, answer the question, and ***show the plot or other output from the analyses you used to answer the question.***

9. Plot a histogram of the calories consumed for males and females.

- a. Write a caption for this plot. Submit the plot and caption.
- b. Would you describe the distribution of each as symmetrical, skewed, bimodal, or uniform?

10. Produce box plots of the calories consumed for males and females.

- a. Write a caption for this plot. Submit the plot and caption.
- b. Report the 25th percentile, median, and 75th percentile. (Estimate from the plot, or report more precise statistics.)

11. Plot means of the calories consumed for males and females. Include error bars for either confidence intervals or standard error of the means. You can use either *qplot* or *ggplot*, and you can use either the bar chart style or the "plot of means" style.

- a. Write a caption for this plot. Submit the plot and caption.
- b. Add a sentence reporting the respective means in the caption.
- c. Are the calories consumed likely to be statistically different?

Understanding Statistics and Hypothesis Testing

Hypothesis Testing and p-values

Initial comments

Traditionally when students first learn about the analysis of experiments, there is a strong focus on hypothesis testing and making decisions based on p -values. Hypothesis testing is important for determining if there are statistically significant effects. However, readers of this book should not place undue emphasis on p -values. Instead, they should realize that p -values are affected by sample size, and that a low p -value does not necessarily suggest a large effect or a practically meaningful effect. Summary statistics, plots, effect size statistics, and practical considerations should be used. The goal is to determine: a) statistical significance, b) effect size, c) practical importance. These are all different concepts, and they will be explored below.

Statistical inference

Most of what we've covered in this book so far is about producing descriptive statistics: calculating means and medians, plotting data in various ways, and producing confidence intervals. The bulk of the rest of this book will cover statistical inference: using statistical tests to draw some conclusion about the data. We've already done this a little bit in earlier chapters by using confidence intervals to conclude if means are different or not among groups.

As Dr. Nic mentions in her article in the "References and further reading" section, this is the part where people sometimes get stumped. It is natural for most of us to use summary statistics or plots, but jumping to statistical inference needs a little change in perspective. The idea of using some statistical test to answer a question isn't a difficult concept, but some of the following discussion gets a little theoretical. The video from the Statistics Learning Center in the "References and further reading" section does a good job of explaining the basis of statistical inference.

One important thing to gain from this chapter is an understanding of how to use the p -value, α , and decision rule to test the null hypothesis. But once you are comfortable with that, you will want to return to this chapter to have a better understanding of the theory behind this process.

Another important thing is to understand the limitations of relying on p -values, and why it is important to assess the size of effects and weigh practical considerations.

Packages used in this chapter

The packages used in this chapter include:

- lsr

The following commands will install these packages if they are not already installed:

```
if(!require(lsr)){install.packages("lsr")}
```

Hypothesis testing

The null and alternative hypotheses

The statistical tests in this book rely on testing a null hypothesis, which has a specific formulation for each test. The null hypothesis always describes the case where e.g. two groups are not different or there is no correlation between two variables, etc.

The alternative hypothesis is the contrary of the null hypothesis, and so describes the cases where there is a difference among groups or a correlation between two variables, etc.

Notice that the definitions of null hypothesis and alternative hypothesis have nothing to do with what you want to find or don't want to find, or what is interesting or not interesting, or what you expect to find or what you don't expect to find. If you were comparing the height of men and women, the null hypothesis would be that the height of men and the height of women were not different. Yet, you might find it surprising if you found this hypothesis to be true for some population you were studying. Likewise, if you were studying the income of men and women, the null hypothesis would be that the income of men and women are not different, in the population you are studying. In this case you might be *hoping* the null hypothesis is true, though you might be *unsurprised* if the alternative hypothesis were true. In any case, the null hypothesis will take the form that there is no difference between groups, there is no correlation between two variables, or there is no effect of this variable in our model.

p-value definition

Most of the tests in this book rely on using a statistic called the *p*-value to evaluate if we should reject, or fail to reject, the null hypothesis.

Given the assumption that the null hypothesis is true, the *p*-value is defined as the probability of obtaining a result equal to or more extreme than what was actually observed in the data.

We'll unpack this definition in a little bit.

Decision rule

The *p*-value for the given data will be determined by conducting the statistical test.

This *p*-value is then compared to a pre-determined value *alpha*. Most commonly, an *alpha* value of 0.05 is used, but there is nothing magic about this value.

If the *p*-value for the test is less than *alpha*, we reject the null hypothesis.

If the *p*-value is greater than or equal to *alpha*, we fail to reject the null hypothesis.

Coin flipping example

For an example of using the *p*-value for hypothesis testing, imagine you have a coin you will toss 100 times. The null hypothesis is that the coin is fair—that is, that it is equally likely that the coin will land on heads as land on tails. The alternative hypothesis is that the coin is not fair. Let's say for this

experiment you throw the coin 100 times and it lands on heads 95 times out of those hundred. The p -value in this case would be the probability of getting 95, 96, 97, 98, 99, or 100 heads, or 0, 1, 2, 3, 4, or 5 heads, *assuming that the null hypothesis is true.*

This is what we call a two-sided test, since we are testing both extremes suggested by our data: getting 95 or greater heads or getting 95 or greater tails. In most cases we will use two sided tests.

You can imagine that the p -value for this data will be quite small. If the null hypothesis is true, and the coin is fair, there would be a low probability of getting 95 or more heads or 95 or more tails.

Using a binomial test, the p -value is < 0.0001 .

(Actually, R reports it as $< 2.2\text{e-}16$, which is shorthand for the number in scientific notation, 2.2×10^{-16} , which is 0.000000000000022, with 15 zeros after the decimal point.)

Assuming an α of 0.05, since the p -value is less than α , we reject the null hypothesis. That is, we conclude that the coin is not fair.

```
binom.test(5, 100, 0.5)

Exact binomial test

number of successes = 5, number of trials = 100, p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal to 0.5
```

Passing and failing example

As another example, imagine we are considering two classrooms, and we have counts of students who passed a certain exam. We want to know if one classroom had statistically more passes or failures than the other.

In our example each classroom will have 10 students. The data is arranged into a contingency table.

<u>Classroom</u>	<u>Passed</u>	<u>Failed</u>
A	8	2
B	3	7

We will use Fisher's exact test to test if there is an association between *Classroom* and the counts of passed and failed students. The null hypothesis is that there is no association between *Classroom* and *Passed/Failed*, based on the relative counts in each cell of the contingency table.

```
Input ="
  Classroom  Passed  Failed
  A          8        2
  B          3        7
")

Matrix = as.matrix(read.table(textConnection(Input),
```

```
header=TRUE,
row.names=1))
```

Matrix

	Passed	Failed
A	8	2
B	3	7

```
fisher.test(Matrix)
```

Fisher's Exact Test for Count Data

p-value = 0.06978

The reported *p*-value is 0.070. If we use an *alpha* of 0.05, then the *p*-value is greater than *alpha*, so we fail to reject the null hypothesis. That is, we did not have sufficient evidence to say that there is an association between *Classroom* and *Passed/Failed*.

More extreme data in this case would be if the counts in the upper left or lower right (or both!) were greater.

<u>Classroom</u>	<u>Passed</u>	<u>Failed</u>
A	9	1
B	3	7

<u>Classroom</u>	<u>Passed</u>	<u>Failed</u>
A	10	0
B	3	7

and so on, with Classroom B...

In most cases we would want to consider as "extreme" not only the results when Classroom A has a high frequency of passing students, but also results when Classroom B has a high frequency of passing students. This is called a two-sided or two-tailed test. If we were only concerned with one classroom having a high frequency of passing students, relatively, we would instead perform a one-sided test. The default for the *fisher.test* function is two-sided, and usually you will want to use two-sided tests.

<u>Classroom</u>	<u>Passed</u>	<u>Failed</u>
A	2	8
B	7	3

<u>Classroom</u>	<u>Passed</u>	<u>Failed</u>
A	1	9
B	7	3

<u>Classroom</u>	<u>Passed</u>	<u>Failed</u>
------------------	---------------	---------------

A	0	10
B	7	3

and so on, with Classroom B...

In both cases, "extreme" means there is a stronger association between *Classroom* and *Passed/Failed*.

Theory and practice of using *p*-values

Wait, does this make any sense?

Recall that the definition of the *p*-value is:

Given the assumption that the null hypothesis is true, the *p*-value is defined as the probability of obtaining a result equal to or more extreme than what was actually observed in the data.

The astute reader might be asking herself, "If I'm trying to determine if the null hypothesis is true or not, why would I start with the assumption that the null hypothesis is true? And why am I using a probability of getting certain data given that a hypothesis is true? Don't I want to instead determine the probability of the hypothesis given my data?"

The answer is *yes*, we *would* like a method to determine the likelihood of our hypothesis being true given our data, but we use the *Null Hypothesis Significance Test* approach since it is relatively straightforward, and has wide acceptance historically and across disciplines.

In practice we do use the results of the statistical tests to reach conclusions about the null hypothesis.

Technically, the *p*-value says nothing about the alternative hypothesis. But logically, if the null hypothesis is rejected, then its logical complement, the alternative hypothesis, is supported. Practically, this is how we handle significant *p*-values, though this practical approach generates disapproval in some theoretical circles.

Statistics is like a jury?

Note the language used when testing the null hypothesis. Based on the results of our statistical tests, we either *reject* the null hypothesis, or *fail to reject* the null hypothesis.

This is somewhat similar to the approach of a jury in a trial. The jury either finds sufficient evidence to declare someone guilty, or fails to find sufficient evidence to declare someone guilty.

Failing to convict someone isn't necessarily the same as declaring someone innocent. Likewise, if we fail to reject the null hypothesis, we shouldn't assume that the null hypothesis is true. It may be that we didn't have sufficient samples to get a result that would have allowed us to reject the null hypothesis, or maybe there are some other factors affecting the results that we didn't account for. This is similar to an "innocent until proven guilty" stance.

Errors in inference

For the most part, the statistical tests we use are based on probability, and our data could always be the result of chance. Considering the coin flipping example above, if we did flip a coin 100 times and came up with 95 heads, we would be compelled to conclude that the coin was not fair. But 95 heads *could* happen with a fair coin strictly by chance.

We can, therefore, make two kinds of errors in testing the null hypothesis:

- A *Type I error* occurs when the null hypothesis really is true, but based on our decision rule we reject the null hypothesis. In this case, our result is a *false positive*; we think there is an effect (unfair coin, association between variables, difference among groups) when really there isn't. The probability of making this kind of error is *alpha*, the same *alpha* we used in our decision rule.
- A *Type II error* occurs when the null hypothesis is really false, but based on our decision rule we fail to reject the null hypothesis. In this case, our result is a *false negative*; we have failed to find an effect that really does exist. The probability of making this kind of error is called *beta*.

The following table summarizes these errors.

Reality		
<u>Decision of Test</u>	<u>Null is true</u>	<u>Null is false</u>
Reject null hypothesis	Type I error (prob. = alpha)	Correctly reject null (prob. = 1 - beta)
Retain null hypothesis	Correctly retain null (prob. = 1 - alpha)	Type II error (prob. = beta)

Statistical power

The statistical power of a test is a measure of the ability of the test to detect a real effect. It is related to the effect size, the sample size, and our chosen *alpha* level.

The effect size is a measure of how unfair a coin is, how strong the association between two variables, or how large the difference among groups. As the effect size increases or as the number of observations we collect increases, or as the *alpha* level decreases, the power of the test increases.

Statistical power in the table above is indicated by $1 - \beta$, and power is the probability of correctly rejecting the null hypothesis.

An example should make these relationships clear. Imagine we are sampling a large group of 7th grade students for their height. That is, the group is the population, and we are sampling a sub-set of these

students. In reality, for students in the population, the girls are taller than the boys, but the difference is small (that is, the effect size is small), and there is a lot of variability in students' heights. You can imagine that in order to detect the difference between girls and boys that we would have to measure many students. If we fail to sample enough students, we might make a Type II error. That is, we might fail to detect the actual difference in heights between sexes.

If we had a different experiment with a larger effect size—for example the weight difference between mature hamsters and mature hedgehogs—we might need fewer samples to detect the difference.

Note also, that our chosen *alpha* plays a role in the power of our test, too. All things being equal, across many tests, if we decrease our *alpha*, that is, insist on a lower rate of Type I errors, we are more likely to commit a Type II error, and so have a lower power. This is analogous to a case of a meticulous jury that has a very high standard of proof to convict someone. In this case, the likelihood of a false conviction is low, but the likelihood of a letting a guilty person go free is relatively high.

The 0.05 alpha value is not dogma

The level of *alpha* is traditionally set at 0.05 in some disciplines, though there is sometimes reason to choose a different value.

One situation in which the *alpha* level is increased is in preliminary studies in which it is better to include potentially significant effects even if there is not strong evidence for keeping them. In this case, the researcher is accepting an inflated chance of Type I errors in order to decrease the chance of Type II errors.

Imagine an experiment in which you wanted to see if various environmental treatments would improve student learning. In a preliminary study, you might have many treatments, with few observations each, and you want to retain any potentially successful treatments for future study. For example, you might try playing classical music, improved lighting, complimenting students, and so on, and see if there is any effect on student learning. You might relax your *alpha* value to 0.10 or 0.15 in the preliminary study to see what treatments to include in future studies.

On the other hand, in situations where a Type I, false positive, error might be costly in terms of money or people's health, a lower *alpha* can be used, perhaps, 0.01 or 0.001. You can imagine a case in which there is an established treatment for cancer, and a new treatment is being tested. Because the new treatment is likely to be expensive and to hold people's lives in the balance, a researcher would want to be very sure that the new treatment is more effective than the established treatment. In reality, the researchers would not just lower the *alpha* level, but also look at the effect size, submit the research for peer review, replicate the study, be sure there were no problems with the design of the study or the data collection, and weigh the practical implications.

The 0.05 alpha value is almost dogma

In theory, as a researcher, you would determine the *alpha* level you feel is appropriate. That is, the probability of making a Type I error when the null hypothesis is in fact true.

In reality, though, 0.05 is almost always used in most fields for readers of this book. Choosing a different *alpha* value will rarely go without question. It is best to keep with the 0.05 level unless you have good justification for another value, or are in a discipline where other values are routinely used.

Practical advice

One good practice is to report actual p -values from analyses. It is fine to also simply say, e.g. “The dependent variable was significantly correlated with variable A ($p < 0.05$).” But I prefer when possible to say, “The dependent variable was significantly correlated with variable A ($p = 0.026$).”

It is probably best to avoid using terms like “marginally significant” or “borderline significant” for p -values less than 0.10 but greater than 0.05, though you might encounter similar phrases. It is better to simply report the p -values of tests or effects in straight-forward manner. If you had cause to include certain model effects or results from other tests, they can be reported as e.g., “Variables correlated with the dependent variable with $p < 0.15$ were A, B, and C.”

Is the p-value every really true?

Considering some of the examples presented, it may have occurred to the reader to ask if the null hypothesis is ever really true. For example, in some population of 7th graders, if we could measure everyone in the population to a high degree of precision, then there must be *some* difference in height between girls and boys. This is an important limitation of null hypothesis significance testing. Often, if we have many observations, even small effects will be reported as significant. This is one reason why it is important to not rely too heavily on p -values, but to also look at the size of the effect and practical considerations. In this example, if we sampled many students and the difference in heights was 0.5 cm, even if significant, we might decide that this effect is too small to be of practical importance, especially relative to an average height of 150 cm. (Here, the difference would be 0.3% of the average height).

Effect sizes and practical importance

Practical importance and statistical significance

It is important to remember to not let p -values be the only guide for drawing conclusions. It is equally important to look at the size of the effects you are measuring, as well as take into account other practical considerations like the costs of choosing a certain path of action.

For example, imagine we want to compare the SAT scores of two SAT preparation classes with a t -test.

```
Class.A = c(1500, 1505, 1505, 1510, 1510, 1510, 1515, 1515, 1520, 1520)
Class.B = c(1510, 1515, 1515, 1520, 1520, 1520, 1525, 1525, 1530, 1530)
t.test(Class.A, Class.B)

Welch Two Sample t-test

t = -3.3968, df = 18, p-value = 0.003214

mean of x mean of y
1511      1521
```

The p -value is reported as 0.003, so we would consider there to be a significant difference between the two classes ($p < 0.05$).

But we have to ask ourselves the practical question, is a difference of 10 points on the SAT large enough for us to care about? What if enrolling in one class costs significantly more than the other class? Is it worth the extra money for a difference of 10 points on average?

Sizes of effects

It should be remembered that p -values do not indicate the size of the effect being studied. It shouldn't be assumed that a small p -value indicates a large difference between groups, or vice-versa.

For example, in the SAT example above, the p -value is fairly small, but the size of the effect (difference between classes) in this case is relatively small (10 points, especially small relative to the range of scores students receive on the SAT).

In converse, there could be a relatively large size of the effects, but if there is a lot of variability in the data or the sample size is not large enough, the p -value could be relatively large.

In this example, the SAT scores differ by 100 points between classes, but because the variability is greater than in the previous example, the p -value is not significant.

```
Class.C = c(1000, 1100, 1200, 1250, 1300, 1300, 1400, 1400, 1450, 1500)
Class.D = c(1100, 1200, 1300, 1350, 1400, 1400, 1500, 1550, 1600)
```

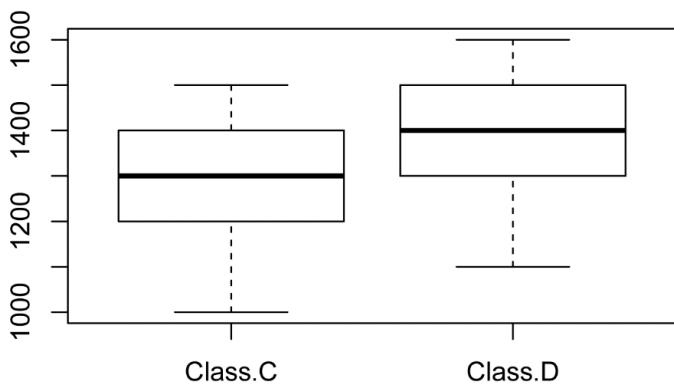
```
t.test(Class.C, Class.D)

Welch Two Sample t-test

t = -1.4174, df = 18, p-value = 0.1735

mean of x mean of y
1290      1390
```

```
boxplot(cbind(Class.C, Class.D))
```



p-values and sample sizes

It should also be remembered that *p*-values are affected by sample size. For a given effect size and variability in the data, as the sample size increases, the *p*-value is likely to decrease. For large data sets, small effects can result in significant *p*-values.

As an example, let's take the data from *Class.C* and *Class.D* and double the number of observations for each without changing the distribution of the values in each, and rename them *Class.E* and *Class.F*.

```
Class.E = c(1000, 1100, 1200, 1250, 1300, 1300, 1400, 1400, 1450, 1500,
          1000, 1100, 1200, 1250, 1300, 1300, 1400, 1400, 1450, 1500)
Class.F = c(1100, 1200, 1300, 1350, 1400, 1400, 1500, 1500, 1550, 1600,
          1100, 1200, 1300, 1350, 1400, 1400, 1500, 1500, 1550, 1600)

t.test(Class.E, Class.F)

Welch Two Sample t-test

t = -2.0594, df = 38, p-value = 0.04636

mean of x mean of y
1290      1390

boxplot(cbind(Class.E, Class.F))
```

Notice that the *p*-value is lower for the *t*-test for *Class.E* and *Class.F* than it was for *Class.C* and *Class.D*. Also notice that the means reported in the output are the same, and the box plots would look the same.

Effect size statistics

One way to account for the effect of sample size on our statistical tests is to consider effect size statistics. These statistics reflect the size of the effect in a standardized way, and are unaffected by sample size.

An appropriate effect size statistic for a *t*-test is Cohen's *d*. It takes the difference in means between the two groups and divides by the pooled standard deviation of the groups. Cohen's *d* equals zero if the means are the same, and increases to infinity as the difference in means increases relative to the standard deviation.

In the following, note that Cohen's *d* is not affected by the sample size difference in the *Class.C* / *Class.D* and the *Class.E* / *Class.F* examples.

```
library(lsrr)
cohensD(Class.C, Class.D,
        method = "raw")

[1] 0.668
```

```
cohensD(Class.E, Class.F,
method = "raw")
```

[1] 0.668

Effect size statistics are standardized so that they are not affected by the units of measurements of the data. This makes them interpretable across different situations, or if the reader is not familiar with the units of measurement in the original data. A Cohen's d of 1 suggests that the two means differ by one pooled standard deviation. A Cohen's d of 0.5 suggests that the two means differ by one-half the pooled standard deviation.

For example, if we create new variables—*Class.G* and *Class.H*—that are the SAT scores from the previous example expressed as a proportion of a 1600 score, Cohen's d will be the same as in the previous example.

```
Class.G = Class.E / 1600
Class.H = Class.F / 1600
```

```
Class.G
Class.H
```

```
cohensD(Class.G, Class.H,
method="raw")
```

[1] 0.668

Good practices for statistical analyses

Statistics is not like a trial

When analyzing data, the analyst should not approach the task as would a lawyer for the prosecution. That is, the analyst should not be searching for significant effects and tests, but should instead be like an independent investigator using lines of evidence to find out what is most likely to true given the data, graphical analysis, and statistical analysis available.

The problem of multiple p-values

One concept that will be important in the following discussion is that when there are multiple tests producing multiple p -values, that there is an inflation of the Type I error rate. That is, there is a higher chance of making false-positive errors.

This simply follows mathematically from the definition of α . If we allow a probability of 0.05, or 5% chance, of making a Type I error for any one test, as we do more and more tests, the chances that at least one of them having a false positive becomes greater and greater.

p-value adjustment

One way we deal with the problem of multiple p -values in statistical analyses is to adjust p -values when we do a series of tests together (for example, if we are comparing the means of multiple groups).

Don't use Bonferroni adjustments

There are various *p*-value adjustments available in R. In some cases, we will use FDR, which stands for *false discovery rate*, and in R is an alias for the Benjamini and Hochberg method. There are also cases in which we'll use Tukey range adjustment to correct for the family-wise error rate.

Unfortunately, students in analysis of experiments courses often learn to use Bonferroni adjustment for *p*-values. This method is simple to do with hand calculations, but is excessively conservative in most situations, and, in my opinion, antiquated.

There are other *p*-value adjustment methods, and the choice of which one to use is dictated either by which are common in your field of study, or by doing enough reading to understand which are statistically most appropriate for your application.

Preplanned tests

The statistical tests covered in this book assume that tests are preplanned for their *p*-values to be accurate. That is, in theory, you set out an experiment, collect the data as planned, and then say "I'm going to analyze it with kind of model and do these post-hoc tests afterwards", and report these results, and that's all you would do.

Some authors emphasize this idea of preplanned tests. In contrast is an exploratory data analysis approach that relies upon examining the data with plots and using simple tests like correlation tests to suggest what statistical analysis makes sense.

If an experiment is set out in a specific design, then usually it is appropriate to use the analysis suggested by this design.

p-value hacking

It is important when approaching data from an exploratory approach, to avoid committing *p*-value hacking. Imagine the case in which the researcher collects many different measurements across a range of subjects. The researcher might be tempted to simply try different tests and models to relate one variable to another, for all the variables. He might continue to do this until he found a test with a significant *p*-value.

But this would be a form of *p*-value hacking.

Because an *alpha* value of 0.05 allows us to make a false-positive error five percent of the time, finding one *p*-value below 0.05 after several successive tests may simply be due to chance.

Some forms of *p*-value hacking are more egregious. For example, if one were to collect some data, run a test, and then continue to collect data and run tests iteratively until a significant *p*-value is found.

Publication bias

A related issue in science is that there is a bias to publish, or to report, only significant results. This can also lead to an inflation of the false-positive rate. As a hypothetical example, imagine if there are currently 20 similar studies being conducted testing a similar effect—let's say the effect of glucosamine supplements on joint pain. If 19 of those studies found no effect and so were discarded,

but one study found an effect using an *alpha* of 0.05, and was published, is this really any support that glucosamine supplements decrease joint pain?

Clarification of terms and reporting on assignments

"Statistically significant"

In the context of this book, the term "significant" means "statistically significant".

Whenever the decision rule finds that $p < \alpha$, the difference in groups, the association, or the correlation under consideration is then considered "statistically significant" or "significant".

No effect size or practical considerations enter into determining whether an effect is "significant" or not. The only exception is that test assumptions and requirements for appropriate data must also be met in order for the *p*-value to be valid.

What you need to consider:

- The null hypothesis
- *p*, α , and the decision rule,
- Your result. That is, whether the difference in groups, the association, or the correlation is significant or not.

What you should report on your assignments:

- The *p*-value
- The conclusion, e.g. "There was a significant difference in the mean heights of boys and girls in the class." It is best to preface this with the "reject" or "fail to reject" language concerning your decision about the null hypothesis.

"Size of the effect" / "effect size"

In the context of this book, I use the term "size of the effect" to suggest the use of summary statistics to indicate how large an effect is. This may be, for example the difference in two medians. I try reserve the term "effect size" to refer to the use of effect size statistics. This distinction isn't necessarily common.

Usually you will consider an effect in relation to the magnitude of measurements. That is, you might look at the difference in medians as a percent of the median of one group or of the global median. Or, you might look at the difference in medians in relation to the range of answers. For example, a one-point difference on a 5-point Likert item. Counts might be expressed as proportions of totals or subsets.

What you should report on assignments:

- The size of the effect. That is, the difference in medians or means, the difference in counts, or the proportions of counts among groups.
- Where appropriate, the size of the effect expressed as a percentage or proportion.
- If there is an effect size statistic—such as *r*, ϵ^2 , *phi*, Cramér's *V*, or Cohen's *d*—: report this and its interpretation (small, medium, large), and incorporate this into your conclusion.

"Practical" / "practical importance"

If there is a significant result, the question of practical importance asks if the difference or association is large enough to matter in the real world.

If there is no significant result, the question of practical importance asks if the a difference or association is large enough to warrant another look, for example by running another test with a larger sample size or that controls variability in observations better.

What you should report on assignments:

- Your conclusion as to whether this effect is large enough to be important in the real world.
- The context, explanation, or support to justify your conclusion.
- In some cases you might include considerations that aren't included in the data presented.
Examples might include the cost of one treatment over another, including time investment, or whether there is a large risk in selecting one treatment over another (e.g., if people's lives are on the line).

A few of xkcd comics***Significant***

xkcd.com/882/

Null hypothesis

xkcd.com/892

P-values

xkcd.com/1478/

Experiments, sampling, and causation***Types of experimental designs*****Experimental designs**

A true experimental design assigns treatments in a systematic manner. The experimenter must be able to manipulate the experimental treatments and assign them to subjects. Since treatments are randomly assigned to subjects, a causal inference can be made for significant results. That is, we can say that the variation in the dependent variable is caused by the variation in the independent variable.

For interval/ratio data, traditional experimental designs can be analyzed with specific parametric models, assuming other model assumptions are met. These traditional experimental designs include:

- Completely random design
- Randomized complete block design
- Factorial
- Split-plot

- Latin square

Quasi-experiment designs

Often a researcher cannot assign treatments to individual experimental units, but can assign treatments to groups. For example, if students are in a specific grade or class, it would not be practical to randomly assign students to grades or classes. But different classes could receive different treatments (such as different curricula). Causality can be inferred cautiously if treatments are randomly assigned and there is some understanding of the factors that affect the outcome.

Observational studies

In observational studies, the independent variables are not manipulated, and no treatments are assigned. Surveys are often like this, as are studies of natural systems without experimental manipulation. Statistical analysis can reveal the relationships among variables, but causality cannot be inferred. This is because there may be other unstudied variables that affect the measured variables in the study.

Sampling

Good sampling practices are critical for producing good data. In general, samples need to be collected in a random fashion so that bias is avoided.

In survey data, bias is often introduced by a self-selection bias. For example, internet or telephone surveys include only those who respond to these requests. Might there be some relevant difference in the variables of interest between those who respond to such requests and the general population being surveyed? Or bias could be introduced by the researcher selecting some subset of potential subjects, for example only surveying a 4-H program with particularly cooperative students and ignoring other clubs. This is sometimes called "convenience sampling".

In election forecasting, good pollsters need to account for selection bias and other biases in the survey process. For example, if a survey is done by landline telephone, those being surveyed are more likely to be older than the general population of voters, and so likely to have a bias in their voting patterns.

Plan ahead and be consistent

It is sometimes necessary to change experimental conditions during the course of an experiment. Equipment might fail, or unusual weather may prevent making meaningful measurements.

But in general, it is much better to plan ahead and be consistent with measurements.

Consistency

People sometimes have the tendency to change measurement frequency or experimental treatments during the course of a study. This inevitably causes headaches in trying to analyze data, and makes writing up the results messy. Try to avoid this.

Controls and checks

If you are testing an experimental treatment, include a *check* treatment that almost certainly will have an effect and a *control* treatment that almost certainly won't. A *control* treatment will receive no treatment and a *check* treatment will receive a treatment known to be successful. In an educational

setting, perhaps a control group receives no instruction on the topic but on another topic, and the check group will receive standard instruction.

Including checks and controls helps with the analysis in a practical sense, since they serve as standard treatments against which to compare the experimental treatments. In the case where the experimental treatments have similar effects, controls and checks allow you say, for example, “Means for the all experimental treatments were similar, but were higher than the mean for control, and lower than the mean for check treatment.”

Include alternate measurements

It often happens that measuring equipment fails or that a certain measurement doesn’t produce the expected results. It is therefore helpful to include measurements of several variables that can capture the potential effects. Perhaps test scores of students won’t show an effect, but a self-assessment question on how much students learned will.

Include covariates

Including additional independent variables that might affect the dependent variable is often helpful in an analysis. In an educational setting, you might assess student age, grade, school, town, background level in the subject, or how well they are feeling that day.

The effects of covariates on the dependent variable may be of interest in itself. But also, including covariates in an analysis can better model the data, sometimes making treatment effects more clear or making a model better meet model assumptions.

Optional discussion: Alternative methods to the Null Hypothesis Significance Test

The NHST controversy

Particularly in the fields of psychology and education, there has been much criticism of the null hypothesis significance test approach. From my reading, the main complaints against NHST tend to be:

- Students and researchers don’t really understand the meaning of *p*-values.
- *p*-values don’t include important information like confidence intervals or parameter estimates.
- *p*-values have properties that may be misleading, for example that they do not represent effect size, and that they change with sample size.
- We often treat an *alpha* of 0.05 as a magical cutoff value.

Personally, I don’t find these to be very convincing arguments against the NHST approach.

The first complaint is in some sense pedantic: Like so many things, students and researchers learn the definition of *p*-values at some point and then eventually forget. This doesn’t seem to impact the usefulness of the approach.

The second point has weight only if researchers use *only p*-values to draw conclusions from statistical tests. As this book points out, one should always consider the size of the effects and practical considerations of the effects, as well present finding in table or graphical form, including confidence intervals or measures of dispersion. There is no reason why parameter estimates, goodness-of-fit statistics, and confidence intervals can't be included when a NHST approach is followed.

The properties in the third point also don't count much as criticism if one is using *p*-values correctly. One should understand that it is possible to have a small effect size and a small *p*-value, and vice-versa. This is not a problem, because *p*-values and effect sizes are two different concepts. We shouldn't expect them to be the same. The fact that *p*-values change with sample size is also in no way problematic to me. It makes sense that when there is a small effect size or a lot of variability in the data that we need many samples to conclude the effect is likely to be real.

(One case where I think the considerations in the preceding point are commonly problematic is when people use statistical tests to check for the normality or homogeneity of data or model residuals. As sample size increases, these tests are better able to detect small deviations from normality or homoscedasticity. Too many people use them and think their model is inappropriate because the test can detect a small effect size, that is, a small deviation from normality or homoscedasticity).

The fourth point is a good one. It doesn't make much sense to come to one conclusion if our *p*-value is 0.049 and the opposite conclusion if our *p*-value is 0.051. But I think this can be ameliorated by reporting the actual *p*-values from analyses, and relying less on *p*-values to evaluate results.

Overall it seems to me that these complaints condemn poor practices that the authors observe: not reporting the size of effects in some manner; not including confidence intervals or measures of dispersion; basing conclusions solely on *p*-values; and not including important results like parameter estimates and goodness-of-fit statistics.

Alternatives to the NHST approach

Estimates and confidence intervals

One approach to determining statistical significance is to use estimates and confidence intervals. Estimates could be statistics like means, medians, proportions, or other calculated statistics. This approach can be very straightforward, easy for readers to understand, and easy to present clearly.

Bayesian approach

The most popular competitor to the NHST approach is Bayesian inference. Bayesian inference has the advantage of calculating the probability of the hypothesis *given the data*, which is what we thought we should be doing in the "Wait, does this make any sense?" section above. Essentially it takes *prior* knowledge about the distribution of the parameters of interest for a population and adds the information from the measured data to reassess some hypothesis related to the parameters of interest. If the reader will excuse the vagueness of this description, it makes intuitive sense. We start with what we suspect to be the case, and then use new data to assess our hypothesis.

One disadvantage of the Bayesian approach is that it is not obvious in most cases what could be used for legitimate prior information. A second disadvantage is that conducting Bayesian analysis is not as straightforward as the tests presented in this book.

References and further reading

[Video] “Understanding statistical inference” from Statistics Learning Center (Dr. Nic). 2015. www.youtube.com/watch?v=tFRXsngz4UQ.

[Video] “Hypothesis tests, p-value” from Statistics Learning Center (Dr. Nic). 2011. www.youtube.com/watch?v=0zZYBALbZgg.

[Video] “Understanding the p-value” from Statistics Learning Center (Dr. Nic). 2011. www.youtube.com/watch?v=eyknGvncKLw.

[Video] “Important statistical concepts: significance, strength, association, causation” from Statistics Learning Center (Dr. Nic). 2012. www.youtube.com/watch?v=FG7xnWmZlPE.

“**Understanding statistical inference**” from Dr. Nic. 2015. Learn and Teach Statistics & Operations Research. learnandteachstatistics.wordpress.com/2015/11/09/understanding-statistical-inference/.

“**Basic concepts of hypothesis testing**” in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/hypothesistesting.html.

“**Hypothesis testing**”, section 4.3, in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

“**Hypothesis Testing with One Sample**”, sections 9.1–9.2 in Openstax. 2013. *Introductory Statistics*. openstaxcollege.org/textbooks/introductory-statistics.

“**Proving causation**” from Dr. Nic. 2013. Learn and Teach Statistics & Operations Research. learnandteachstatistics.wordpress.com/2013/10/21/proving-causation/.

[Video] “Variation and Sampling Error” from Statistics Learning Center (Dr. Nic). 2014. www.youtube.com/watch?v=y3A0lUkpAko.

[Video] “Sampling: Simple Random, Convenience, systematic, cluster, stratified” from Statistics Learning Center (Dr. Nic). 2012. www.youtube.com/watch?v=be9e-Q-jC-0.

“**Confounding variables**” in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/confounding.html.

“**Overview of data collection principles**”, section 1.3, in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

"Observational studies and sampling strategies", section 1.4, in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

"Experiments", section 1.5, in Diez, D.M., C.D. Barr , and M. Çetinkaya-Rundel. 2012. *OpenIntro Statistics*, 2nd ed. www.openintro.org/.

Exercises F

1. Which of the following pair is the null hypothesis?

- A) The number of heads from the coin is not different from the number of tails.
- B) The number of heads from the coin is different from the number of tails.

2. Which of the following pair is the null hypothesis?

- A) The height of boys is different than the height of girls.
- B) The height of boys is not different than the height of girls.

3. Which of the following pair is the null hypothesis?

- A) There is an association between classroom and sex. That is, there is a difference in counts of girls and boys between the classes.
- B) There is no association between classroom and sex. That is, there is no difference in counts of girls and boys between the classes.

4. We flip a coin 10 times and it lands on heads 7 times. We want to know if the coin is fair.

- a. What is the null hypothesis?
- b. Looking at the code below, and assuming an *alpha* of 0.05,

What do you decide (use the *reject* or *fail to reject* language)?

- c. In practical terms, what do you conclude?

```
binom.test(7, 10, 0.5)
```

Exact binomial test

number of successes = 7, number of trials = 10, p-value = 0.3438

5. We measure the height of 9 boys and 9 girls in a class, in centimeters. We want to know if one group is taller than the other.

- a. What is the null hypothesis?
- b. Looking at the code below, and assuming an *alpha* of 0.05,

What do you decide (use the *reject* or *fail to reject* language)?

- c. In practical terms, what do you conclude? Address the practical importance of the results.

```
Girls = c(152, 150, 140, 160, 145, 155, 150, 152, 147)
Boys = c(144, 142, 132, 152, 137, 147, 142, 144, 139)
t.test(Girls, Boys)

Welch Two Sample t-test

t = 2.9382, df = 16, p-value = 0.009645

mean of x mean of y
150.1111 142.1111

mean(Boys)
sd(Boys)
quantile(Boys)
mean(Girls)
sd(Girls)
quantile(Girls)

boxplot(cbind(Girls, Boys))
```

6. We count the number of boys and girls in two classrooms. We are interested to know if there is an association between the classrooms and the number of girls and boys. That is, does the proportion of boys and girls differ statistically across the two classrooms?

- a. What is the null hypothesis?
- b. Looking at the code below, and assuming an *alpha* of 0.05,

What do you decide (use the *reject* or *fail to reject* language)?

- c. In practical terms, what do you conclude?

<u>Classroom</u>	<u>Girls</u>	<u>Boys</u>
A	13	7

B	5	15
---	---	----

```

Input =("
  Classroom Girls Boys
  A          13    7
  B          5     15
")

Matrix = as.matrix(read.table(textConnection(Input),
  header=TRUE,
  row.names=1))

fisher.test(Matrix)

Fisher's Exact Test for Count Data

p-value = 0.02484

Matrix

rowSums(Matrix)

colSums(Matrix)

prop.table(Matrix,
  margin=1)

### Proportions for each row

barplot(t(Matrix),
  beside = TRUE,
  legend = TRUE,
  ylim   = c(0, 25),
  xlab   = "Class",
  ylab   = "Count")

```

7. Why should you not rely solely on *p*-values to make a decision in the real world? (You should have at least two reasons.)

8. Create your own example to show the importance of considering the ***size of the effect***. Describe the scenario: what the research question is, and what kind of data were collected. You may make up data and provide real results, or report hypothetical results.

9. Create your own example to show the importance of weighing ***other practical considerations***. Describe the scenario: what the research question is, what kind of data were collected, what statistical results were reached, and what other practical considerations were brought to bear.

10. What is 5e-4 in common decimal notation?

Reporting Results of Data and Analyses

Given the variety of experimental designs, potential types of data, and analytical approaches, it is relatively impossible to develop a cookbook approach to reporting data summaries and analyses. That being said, it is the intent of this chapter to give some broad and practical advice for this task.

Packages used in this chapter

The packages used in this chapter include:

- FSA

The following commands will install these packages if they are not already installed:

```
if(!require(FSA)){install.packages("FSA")}
```

Reporting analyses, results, and assumptions

The following bullets list some of the information you should include in reporting summaries and analyses. These can be included in a combination of text, plots, tables, plot captions, and table headings.

A variety of plots are shown in the *Basic Plots* chapter, as well as in the chapters for specific statistical analyses. Simple tables for grouped data are shown in the chapters on *Descriptive Statistics*, *Confidence Intervals*, *Descriptive Statistics for Likert Data*, and *Confidence Intervals for Medians*.

Procedures

- A description of the analysis. Give the test used, indicate the dependent variables and independent variables, and describe the experimental design or model in as much detail as is appropriate. Mention the experimental units, dates, and location.
- Model assumptions checks. Describe what assumptions about the data or residuals are made by the test and how you assessed your data's conformity with these assumptions.
- Cite the software and packages you used.
- Cite a reference for statistical procedures or code.

Results

- A measure of the central tendency or location of the data, such as the mean or median for groups or populations

- A measure of the variation for the mean or median for each group, such standard deviation, standard error, first and third quartile, or confidence intervals
- Size of the effect. Often this is conveyed by presenting means and medians in a plot or table so that the reader can see the differences. Sometimes, specific statistics are reported to indicate “effect size”. See the “Optional technical note on effect sizes” below for more information.
- Number of observations, either for each group, or for total observations
- *p*-value from the analysis
- Goodness-of-fit statistics, if appropriate, such as *r-squared*, or pseudo *R-squared*.
- Any other relevant statistics, such as the predictive equation from linear regression or the critical *x* value from a linear plateau model.
- Results from post-hoc analyses. Summarize the differences among groups with letters, in a plot or table. Include the *alpha* value for group separations

Notes on different data and analyses

Interval/ratio or ordinal data analyzed by group

The bullet points above should be appropriate for this kind of analysis. Data can be summarized in tables or with in a variety of plots including “plot of means”, “plot of medians”, “interaction plot”, box plot, histogram, or bar plot. For some of these, error bars can indicate measures of variation. Mean separations can be indicated with letters within plots.

Nominal data arranged in contingency tables

Nominal data arranged in contingency tables can be presented as frequencies in contingency tables, or in plots. Bar plots of frequencies can be used. Confidence intervals can be added as error bars to plots. Another alternative is using mosaic plots.

Bivariate data analyses

Bivariate relationships can be shown with a scatterplot of the two variables, often with the best fit model drawn in. These models include those from linear regression or curvilinear regression. It is usually important to present the *p*-value for the model and the *r-squared* or pseudo *R-squared* value.

Advice on tables and plots

Plots

Some advice for producing plots is given in the “Some advice on producing plots” section of the *Basic Plots* chapter.

For additional advice on presenting data in plots, see McDonald (2014a) in the “References” section.

Tables

Table style and format will vary widely from publication to publication. As general advice, the style and format should follow that of the journals or extension publications in your field or institution, or where you hope to get published.

For additional advice on presenting data in tables, see McDonald (2014b) in the “References” section.

Table headings and plot captions

Both table headings and plot captions should give as much information as is practical to make the table or plot understandable if it were separated from the rest of the publication.

Example elements

Element	Example
Description of data	“Mean of reading scores for program curricula”
Experimental units	“for 8 th grade students”
Location	“in Watkins Glen, NY”
Date	“1999–2000”
Statistics	“Error bars represent standard error of the mean. The effect of curricula on mean reading score was significant by one-way ANOVA ($p = 0.027$). Means sharing a letter not significantly different by Tukey-adjusted mean separations ($\alpha = 0.05$). Total observations = 24.”

Example description of statistical analysis and results

The citation information for the R software can be found with:

```
citation()
```

Citation information for individual packages can be found with, e.g.:

```
library(FSA)
```

```
citation("FSA")
```

Procedures

A study of student learning was conducted in 1999–2000 in Watkins Glen, NY. Students were randomly assigned to one of four curricula, which they studied for one-hour per week under teacher-supervised conditions, and their scores on an assessment exam were recorded at the end of the study. A one-way analysis of variance was conducted with student score as the dependent variable and curriculum as the independent variable (Mangiafico, 2016).

Treatment means were separated by Tukey-adjusted comparisons. Model residuals were

checked for normality and homoscedasticity by visual inspection of residual plots. Analysis of variance and post-hoc tests were conducted in R (R Core Team, 2016) with the *car* and *emmeans* packages. Data summary was conducted with the *FSA* package.

Results

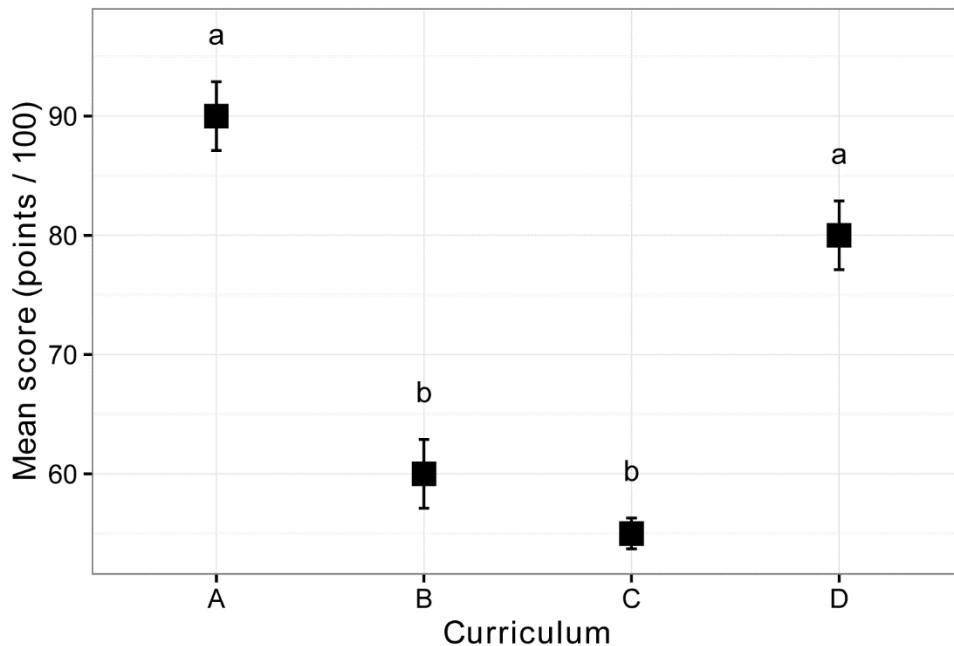


Figure 1. Mean of reading scores for program curricula for 8th grade students in Watkins Glen, NY, 1999–2000. Error bars represent standard error of the mean. The effect of curricula on mean reading score was significant by one-way ANOVA ($p < 0.0001$). Means sharing a letter not significantly different by Tukey-adjusted mean separations ($\alpha = 0.05$). Total observations = 24.

Table 1. Mean of reading scores for program curricula for 8th grade students in Watkins Glen, NY, 1999–2000. The effect of curricula on mean reading score was significant by one-way ANOVA ($p < 0.0001$). Means sharing a letter not significantly different by Tukey-adjusted mean separations ($\alpha = 0.05$). *n* indicates number of observations. *Std. err.* indicates the standard error of the mean.

Curriculum	n	Mean score (points / 100)	Std. err.	Tukey group
A	6	90	2.89	a
B	6	60	2.89	b
C	6	55	1.29	b
D	6	80	2.89	a

Optional technical note on effect sizes

In this chapter, I use the term “size of the effect” in a general sense of the magnitude of differences among groups or the degree of association of variables. “Effect size” usually describes specific statistics for a given analysis, such as Cohen's d , *eta-squared*, or odds ratio.

For some examples of these statistics, see Sullivan and Feinn (2012) or IDRE (2015) in the “References” section of this chapter.

The *pwr* package can calculate the effect size for proportions, *chi-square* goodness-of-fit, and *chi-square* test of association. For the effect size for one-way ANOVA (Cohen's f), see Mangiafico (2015) or IDRE (2015). For effect sizes for nonparametric tests, see Tomczak and Tomczak (2014), and King and Rosopa (2010).

References

“One-way Anova” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_05.html.

“Guide to fairly good graphs” in McDonald, J.H. 2014a. *Handbook of Biological Statistics*. www.biostathandbook.com/graph.html.

“Presenting data in tables” in McDonald, J.H. 2014b. *Handbook of Biological Statistics*. www.biostathandbook.com/table.html.

Sullivan, G.M. and R. Feinn. 2012. Using Effect Size—or Why the P Value is Not Enough. *Journal of Graduate Medical Education* 4(3): 279–282.
www.ncbi.nlm.nih.gov/pmc/articles/PMC3444174/.

Tomczak, M. and Tomczak, E. 2014. The need to report effect size estimates revisited. An overview of some recommended measures of effect size. *Trends in Sports Sciences* 1(21):1–25. www.tss.awf.poznan.pl/files/3_Trends_Vol21_2014_no1_20.pdf.

[IDRE] Institute for Digital Research and Education. 2015. How is effect size used in power analysis? UCLA.
www.ats.ucla.edu/stat/mult_pkg/faq/general/effect_size_power/effect_size_power.htm.

King, B.M., P.J. Rosopa, and E.W. Minium. 2018. Some (Almost) Assumption-Free Tests. In *Statistical Reasoning in the Behavioral Sciences*, 7th ed. Wiley.

Choosing a Statistical Test

Choosing a statistical test can be a daunting task for those starting out in the analysis of experiments. This chapter provides a table of tests and models covered in this book, as well as some general advice for approaching the analysis of your data.

Plan your experimental design before you collect data

It is important to have an experimental design planned out before you start collecting data, and to have some idea of how you plan on analyzing the data. One of the most common mistakes people make in doing research is collecting a bunch of data without having thought through what questions they are trying to answer, what specific hypotheses they want to test, and what statistical tests they can use to test these hypotheses.

What is the hypothesis?

The most important consideration in choosing a statistical test is determining what hypothesis you want to test. Or, more generally, what question are you trying to answer.

Often people have a notion about the purpose of the research they are conducting, but haven't formulated a specific hypothesis. It is possible to begin with exploratory data analysis, to see what interesting secrets the data wish to say. But ultimately, choosing a statistical test relies on having in mind a specific hypothesis to test.

For example, we may know that our goal is to determine if one curriculum works better than another. But then we must be more specific in our hypothesis. Perhaps we wish to compare the mean of scores that students get on an exam across the different curricula. Then a specific null hypothesis is, There is no difference among the mean of student scores across curricula.

In this example, we identified the dependent variable as *Student scores*, and the independent variable as *Curriculum*.

Of course, we might make things more complicated. For example, if the curricula were used in different classrooms, we might want to include *Classroom* as an independent blocking variable.

What number and type of variables do you have?

To a large extent, the appropriate statistical test for your data will depend upon the number and types of variables you wish to include in the analysis.

Consider the type of dependent variable you wish to include.

- If it is of interval/ratio type, you can consider parametric tests or nonparametric tests.
- However, if it is an ordinal variable, you would look toward ordinal regression models, permutation tests, nonparametric tests, or tests for ordinal tables.
- Nominal variables arranged in contingency tables can be analyzed with chi-square and similar tests. Nominal dependent variables can be related to independent variables with logistic regression.

- Count data dependent variables can be related to independent variables with Poisson regression and related models. If the dependent variable is a proportion or percentage, beta regression might be appropriate.

The number and type of independent variables will also be taken into account. As will whether there are paired observations or random blocking variables.

The table below lists the tests in this book according to their number and types of variables.

Note that each test has its own set of assumptions for appropriate data, which should be assessed before proceeding with the analysis.

Also note that the tests in this book cover cases with a single dependent variable only. There are other statistical tests, included under the umbrella of *multivariate statistics* that can analyze multiple dependent variables simultaneously. These include multivariate analysis of variance (MANOVA), canonical correlation, and discriminant function analysis.

The “References” and “Optional readings” sections of this chapter includes a few other guides to choosing statistical tests.

Table of tests by variable type

Test	DV type, or variable type when there is no DV	DV	IV type	Number of IV	Levels in IV	Test type
One-sample Wilcoxon	Ordinal or interval/ratio	Independent	Single default value	N/A	N/A	Nonparametric
Sign test for one-sample	Ordinal or interval/ratio	Independent	Single default value	N/A	N/A	Nonparametric
Two-sample Mann-Whitney	Ordinal or interval/ratio	Independent	Nominal	1	2	Nonparametric
Mood's median test for two-sample	Ordinal or interval/ratio	Independent	Nominal	1	2	Nonparametric
Two-sample paired rank-sum	Ordinal or interval/ratio	Paired	Nominal	1, or 2 when one is blocking	2	Nonparametric
Sign test for two-sample paired	Ordinal or interval/ratio	Paired	Nominal	1, or 2 when one is blocking	2	Nonparametric
Kruskal-Wallis	Ordinal or interval/ratio	Independent	Nominal	1	2 or more	Nonparametric
Mood's median	Ordinal or interval/ratio	Independent	Nominal	1	2 or more	Nonparametric
Friedman	Ordinal or interval/ratio	Independent blocked, or paired	Nominal	2 when one is blocking, in unreplicated complete block design	2 or more	Nonparametric

Quade	Ordinal or interval/ratio	Independent blocked, or paired	Nominal	2 when one is blocking, in unreplicated complete block design	2 or more	Nonparametric
One-way Permutation Test of Independence	Ordinal or interval/ratio	Independent	Nominal	1	2 or more	Permutation
One-way Permutation Test of Symmetry	Ordinal or interval/ratio	Independent blocked, or paired	Nominal	2 when one is blocking	2 or more	Permutation
Two-sample CLM	Ordinal	Independent	Nominal	1	2	Ordinal regression
Two-sample paired CLMM	Ordinal	Paired	Nominal	2 when one is blocking	2	Ordinal regression
One-way ordinal regression CLM	Ordinal	Independent	Nominal	1	2 or more	Ordinal regression
One-way repeated ordinal regression CLMM	Ordinal	Independent	Nominal	2 when one is blocking	2 or more	Ordinal regression
Two-way ordinal regression CLM	Ordinal	Independent	Nominal	2	2 or more	Ordinal regression
Two-way repeated ordinal regression CLMM	Ordinal	Independent	Nominal	3 when one is blocking	2 or more	Ordinal regression
Goodness-of-fit tests for nominal variables • binomial test • multinomial test • G-test goodness-of-fit • Chi-square test goodness-of-fit	Nominal	Independent	Expected counts	N/A	Overall: vector of counts and expected proportions	Nominal
Association tests for nominal variables • Fisher exact test of association • G-test of association • Chi-square test of association	Nominal	Independent	Nominal	N/A	Overall: 2-way contingency table	Nominal
Tests for paired nominal data • McNemar • McNemar-Bowker	Nominal	Paired	Nominal	N/A	Overall: 2-way marginal contingency table	Nominal
Cochran-Mantel-Haenszel	Nominal	Independent	Nominal	N/A	Overall: 3-way contingency table	Nominal
Cochran's Q	Nominal (2 levels only)	Paired	Nominal	2 when one is blocking	2 or more	Nominal

Linear-by-linear	Ordered nominal (ordinal)	Independent	Ordered nominal (ordinal)	N/A	Overall: 2-way or 3-way contingency table	Nominal
Cochran-Armitage (extended)	Ordered nominal (ordinal)	Independent	Nominal	N/A	Overall: 2-way or 3-way contingency table	Nominal
Log-linear model (multiway frequency analysis)	Nominal	Independent	Nominal	N/A	Overall: contingency table with 2 or dimensions	Generalized linear model
Logistic regression (standard)	Nominal with 2 levels	Independent	Interval/ratio or nominal	1 or more	2 or more	Generalized linear model
Multinomial logistic regression	Nominal with 2 or more levels	Independent	Interval/ratio or nominal	1 or more	2 or more	Generalized linear model
Mixed-effects logistic regression	Nominal with 2 levels	Independent or paired	Interval/ratio or nominal	1 or more when one is blocking or random	2 or more	Generalized linear model
One-sample t-test	Interval/ratio	Independent	Single default value	N/A	N/A	Parametric
Two-sample t-test	Interval/ratio	Independent	Nominal	1	2	Parametric
Paired t-test	Interval/ratio	Paired	Nominal	1, or 2 when one is blocking	2	Parametric
One-way ANOVA	Interval/ratio	Independent	Nominal	1	2 or more	Parametric
One-way ANOVA with blocks	Interval/ratio	Independent	Nominal	2 when one is blocking	2 or more	Parametric
One-way ANOVA with random blocks	Interval/ratio	Independent	Nominal	2 when one is blocking	2 or more	Parametric
Two-way ANOVA	Interval/ratio	Independent	Nominal	2	2 or more	Parametric
Repeated measures ANOVA	Interval/ratio	Paired across time	Nominal	2 or more when one is time effect	2 or more	Parametric
Multiple correlation	Interval/ratio or ordinal, depending on type selected	Independent	Interval/ratio or ordinal, depending on type selected	1 or more	Overall: multiple vectors of interval/ratio or ordinal data	Parametric or nonparametric depending on type selected
Pearson correlation	Interval/ratio	Independent	Interval/ratio	1	Overall: two vectors of interval/ratio data	Parametric
Kendall correlation	Interval/ratio or ordinal	Independent	Interval/ratio or ordinal	1	Overall: two vectors of interval/ratio or ordinal data	Nonparametric

Spearman correlation	Interval/ratio or ordinal	Independent	Interval/ratio or ordinal	1	Overall: two vectors of interval/ratio or ordinal data	Nonparametric
Linear regression	Interval/ratio	Independent	Interval/ratio	1	N/A	Parametric
Polynomial regression	Interval/ratio	Independent	Interval/ratio	2 or more that are polynomial terms	N/A	Parametric
Nonlinear regression and curvilinear regression	Interval/ratio	Independent	Interval/ratio	1	N/A	Parametric
Multiple regression	Interval/ratio	Independent	Interval/ratio	2 or more	N/A	Parametric
Robust linear regression	Interval/ratio	Independent	Interval/ratio	1	N/A	Robust parametric
Kendall-Theil regression	Interval/ratio	Independent	Interval/ratio	1	N/A	Nonparametric
Linear plateau and quadratic plateau models	Interval/ratio	Independent	Interval/ratio	1	N/A	Parametric
Cate–Nelson analysis	Interval/ratio	Independent	Interval/ratio	1	N/A	Mostly nonparametric
Poisson and related regression • Hermite regression • Poisson regression • Negative binomial regression • Zero-inflated regression	Count	Independent	Interval/ratio or nominal	1 or more	2 or more	Generalized linear model
Beta regression	Proportion or percentage	Independent	Interval/ratio or nominal	1 or more	2 or more	Generalized linear model

Optional discussion: Sometimes it's all about the hypothesis

Tests that have analogous purposes, like comparing a measurement variable across two groups, may test very different hypotheses.

For example, imagine you are investigating the income of two towns. Let's say the income of Town A is normally distributed about a mean and median of \$48,000. The income of Town B has a similar median, but has right skew, with some observations close to \$1 million.

What test or statistic would you use to compare the income of these two towns?

You might be tempted to compare the means of the two towns with a *t*-test. In this case, however, means may not be the best statistic for skewed data, and this data may not meet the assumptions of the *t*-test.

You might be interested in comparing the median of the income of the two towns, for example with Mood's median test. This might make sense for some regulatory purpose that is concerned with medians.

On the other hand, looking for a systemic change in the income across the two towns may make more sense. For example, the higher incomes in Town B may give the town a different character, for example, some streets with larger homes or upscale stores. For this, you might use the Mann–Whitney test.

Another approach is to use a permutation test.

Or you might compare the overall distributions of incomes for the two towns using the Kolmogorov–Smirnov test.

Finally, you might want to compare at the 75th percentile of income for the two towns. This could be done using quantile regression.

Example

The following code compares some of these results for a hypothetical data set of income in two towns.

Note that the assumptions and pitfalls of these tests are not discussed here, but should be considered in real situations.

```
### Load required packages

if(!require(FSA)){install.packages("FSA")}
if(!require(psych)){install.packages("psych")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(coin)){install.packages("coin")}
if(!require(quantreg)){install.packages("quantreg")}

### Read the data frame

TwoTown = read.table("http://rcompanion.org/documents/TwoTown.csv",
                     header=TRUE, sep=",")

### Check the data frame

library(psych)

headTail(TwoTown)

summary(TwoTown)

### Summarize the data

library(FSA)
```

```
Summarize(Income ~ Town,
  data=TwoTown,
  digits=3)
```

	Town	n	mean	sd	min	Q1	median	Q3	max
1	Town.A	101	48146.43	10851.67	23560	40970	48010	56420	77770
2	Town.B	101	115275.22	163878.17	29050	34140	47220	108200	880000

```
boxplot(Income ~ Town,
  data=TwoTown)
```

Mood's median test

```
library(RVAideMemoire)
```

```
mood.medtest(Income ~ Town,
  data = TwoTown)
```

Mood's median test

X-squared = 0, df = 1, p-value = 1

Mann-Whitney test

```
wilcox.test(Income ~ Town,
  data=TwoTown)
```

Wilcoxon rank sum test with continuity correction

w = 4672, p-value = 0.3029

alternative hypothesis: true location shift is not equal to 0

Permutation test

```
library(coin)
```

```
independence_test(Income ~ Town,
  data = TwoTown)
```

Asymptotic General Independence Test

Z = -3.9545, p-value = 7.669e-05

Kolmogorov-Smirnov test

```
library(FSA)
```

```
ksTest(Income ~ Town,
  data = TwoTown)
```

```
Two-sample Kolmogorov-Smirnov test
```

```
D = 0.35644, p-value = 5.349e-06
```

```
### quantile regression considering the 75th percentile
```

```
library(quantreg)
```

```
model.q = rq(Income ~ Town,
             data = TwoTowns,
             tau = 0.75)
```

```
model.null = rq(Income ~ 1,
                 data = TwoTowns,
                 tau = 0.75)
```

```
anova(model.q, model.null)
```

```
Quantile Regression Analysis of Deviance Table
```

Df	Resid Df	F value	Pr(>F)
1	200	5.7342	0.01756 *

References

[IDRE] Institute for Digital Research and Education. 2015. What statistical analysis should I use? UCLA. stats.idre.ucla.edu/other/mult-pkg/whatstat/

“Choosing a statistical test” in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/testchoice.html.

Optional readings

[Video] **“Choosing which statistical test to use”** from Statistics Learning Center (Dr. Nic). 2014. www.youtube.com/watch?v=rullUAN0U3w.

Independent and Paired Values

An assumption of many statistical tests is that observations are *independent* of one another. This means that the value for one observation is unlikely to be influenced by the value of another observation. If we pick students at random from a class and measure their height, we can assume the height of the first student will not affect the height of the next student. These observations would be independent. If, however, we measured the height of the same students across years, we would expect that a student who is tall this year would likely be tall the next, and so on. These observations would

not be independent. We might call this second set of observations *non-independent, paired, dependent, or correlated*.

Dependent samples commonly arise in a few situations. One is *repeated measures*, in which the same subject is measured on multiple dates. This is like the student height example described above.

A second is when we are taking multiple measurements of the *same individual*. An example of this might be if we are testing students on multiple concepts; we might suspect that if a student scores well in one section, that she is likely to score well in the other sections. Another example would be measuring the length of people's hands. We would suspect that someone with a large left hand is likely to have a large right hand. A final example would be if student raters were measuring multiple instructors. We might suspect that a rater who scores one instructor low might be likely to score another instructor low.

A related concept is that of *blocks*. If observations can be broken into meaningful groups where values are likely to be different, this should be taken into account. For example, if we are measuring students' scores from two classes, and we suspect scores would be lower for one class than the other. If we were testing instructional methods, we may care about the effect of the instructional methods, and not care at all about the classes *per se*, but we want to take differences due to the different classes into account.

Packages used in this chapter

The packages used in this chapter include:

- FSA
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(FSA)){install.packages("FSA")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

An example of paired and unpaired data

In this example we measure the length in centimeters of both the left hand and the right hand for each of 16 individuals.

```
Input = "
Individual Hand Length
A Left 17.5
B Left 18.4
C Left 16.2
D Left 14.5
E Left 13.5
F Left 18.9
G Left 19.5
H Left 21.1
I Left 17.8
J Left 16.8
```

```

K      Left   18.4
L      Left   17.3
M      Left   18.9
N      Left   16.4
O      Left   17.5
P      Left   15.0
A      Right  17.6
B      Right  18.5
C      Right  15.9
D      Right  14.9
E      Right  13.7
F      Right  18.9
G      Right  19.5
H      Right  21.5
I      Right  18.5
J      Right  17.1
K      Right  18.9
L      Right  17.5
M      Right  19.5
N      Right  16.5
O      Right  17.4
P      Right  15.6
")

```

```

Data = read.table(textConnection(Input), header=TRUE)

### Note: for the paired test below, data must be ordered so that
### the first observation of Group 1
### is the same subject as the first observation of Group 2

### The following will order the data frame by Hand, and then by Individual

Data = Data[order(Data$Hand, Data$Individual),]

### Check the data frame

Data
str(Data)
summary(Data)

### Remove unnecessary objects

rm(Input)

```

Box plot and summary statistics by group

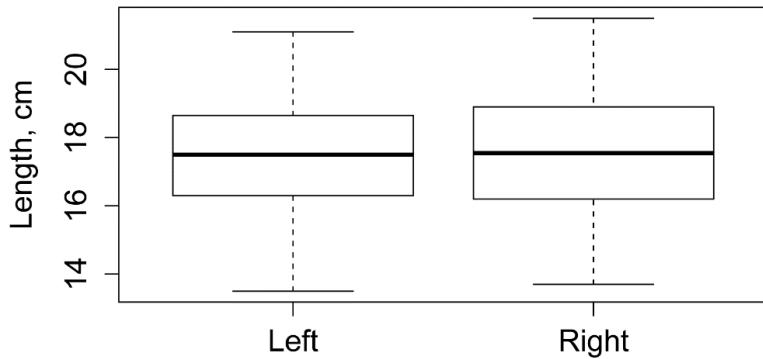
Below, the descriptive statistics suggest that left hands and right hands had similar means, medians, and standard deviations for *Length*.

```
library(FSA)

Summarize(Length ~ Hand,
  data=Data,
  digits=3)

boxplot(Length ~ Hand,
  data=Data,
  ylab="Length, cm")

Hand n   mean    sd  min   Q1 median   Q3 max percZero
1 Left 16 17.356 1.948 13.5 16.35 17.50 18.52 21.1      0
2 Right 16 17.594 1.972 13.7 16.35 17.55 18.90 21.5      0
```



Bar plot to show paired differences

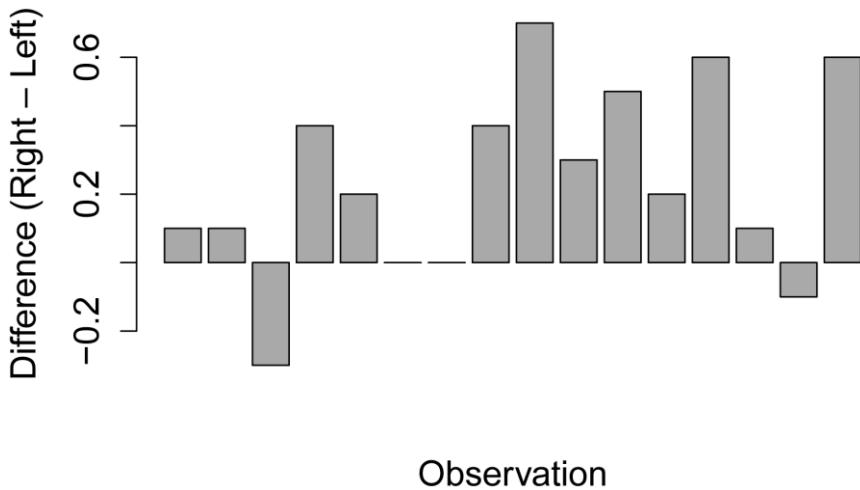
The previous summary statistics, however, do not capture the paired nature of the data. Instead, we want to investigate the difference between left hand and right hand for each individual. We can calculate this difference, and use a bar plot to visualize the difference. For most observations, the right hand was larger, with *Right – Left* being greater than zero.

```
Left_hand = Data$Length[Data$Hand=="Left"]

Right_hand = Data$Length[Data$Hand=="Right"]

Difference = Right_hand - Left_hand

barplot(Difference,
  col="dark gray",
  xlab="Observation",
  ylab="Difference (Right – Left)")
```

**Paired t-test and unpaired t-test**

t-tests are discussed later in this book. It isn't important that you understand the test fully at this point. In this example, a t-test that ignores the pairing of observations found no difference between the mean length for left hand and right hand, whereas the t-test that accounts for the paired observations found a significant difference. On average the right hands were about 0.2 cm longer than their paired left hands.

```
t.test(Length ~ Hand,
       data = Data,
       paired = FALSE)

Welch Two Sample t-test

t = -0.3427, df = 29.996, p-value = 0.7342
### No difference between left hand and right if length treated as not paired

t.test(Length ~ Hand,
       data = Data,
       paired = TRUE)

Paired t-test

t = -3.3907, df = 15, p-value = 0.004034
mean of the differences
-0.2375
### Significant difference between left hand and right
### if length treated as paired
```

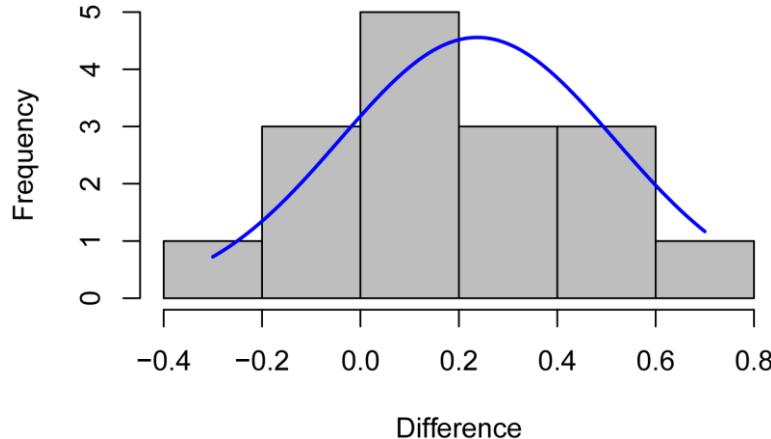
Histogram of differences with normal curve

To be sure our paired *t*-test was valid, we'll plot the differences in hands to be sure their distribution is approximately normal.

```
Left_hand = Data$Length[Data$Hand=="Left"]
Right_hand = Data$Length[Data$Hand=="Right"]
Difference = Right_hand - Left_hand

library(rcompanion)
```

```
plotNormalHistogram(Difference,
                    xlab = "Difference")
```



```
### Distribution of differences is probably close enough to normal
### for paired t-test
```

Introduction to Likert Data

Likert data—properly pronounced like “LICK-ert”—are ordered responses to questions or ratings. Responses could be descriptive words, such as “agree”, “neutral”, or “disagree,” or numerical, such as “On a scale of 1 to 5, where 1 is ‘not interested’ and 5 is ‘very interested’...” Likert data is commonly collected from surveys evaluating education programs, as well in a variety of opinion surveys and social science surveys.

Likert data

Numbers of responses

Most commonly, a 5- or 7- point scale is used for Likert items. It is believed that most people can think about or visualize 5 or 7 ordered options easily. Younger children, however, may do better with a 3-point scale or a simple dichotomous question. On the other hand, if the audience is educated about a subject and trained in the evaluation, a 10-point scale could be used.

Symmetry

Responses to Likert items are usually symmetrical. That is, if there are options for “agree” and “strongly agree”, there should be options for “disagree” and “strongly disagree”.

Neutral responses

Responses to Likert items also tend to have a neutral option, such as “neutral”, “neither agree nor disagree”. Neutral responses may also be terms like “sometimes” or “occasionally” if “never” and “rarely” on one side are balanced with “often” and “always”.

Form of responses

Numbered responses are typically described with descriptive terms, either for every number, for just the end points, or for the end points and the middle points, for example:

Strongly agree	Agree	Neutral	Disagree	Strongly disagree
1	2	3	4	5

Strongly agree				Strongly disagree
1	2	3	4	5

Strongly agree	Neutral	Strongly disagree
1	3	4
2		5

Other options for Likert responses include faces (smiley face, neutral face, frowny face), and a line on which respondents mark their response.

Opt-out answers

Questions may also include opt-out responses, like “Don’t know” or “Not applicable”. These are included outside the Likert responses.



I am in favor of including opt-out responses as it tends to encourage more honest responses. It seems to me it is better to allow a respondent to opt out of answering a question rather than force an inauthentic response. It is possible that a respondent has no opinion or doesn’t understand a question, or that a question is not applicable for them.

That being said, the opt-out answer “Don’t know” may not be a great choice, simply because respondents and researchers may interpret “Don’t know” as a “Neutral” answer. It may be better to choose less ambiguous opt-out answers like “Not applicable”.

Examples of Likert item responses

See:

Vagias, W.M. (2006). Likert-type scale response anchors. Clemson International Institute for Tourism & Research Development, Department of Parks, Recreation and Tourism Management. Clemson University. www.clemson.edu/centers-institutes/tourism/documents/sample-scales.pdf.

Brown, S. Likert Scale Examples for Surveys. Iowa State University Extension.
www.extension.iastate.edu/ag/staff/info/likertscaleexamples.pdf.

Likert items and Likert scales

Technically, a Likert *item* is a single question with Likert responses, whereas a Likert *scale* is a group of items viewed together as a single measure. For example, one could have several Likert items with various questions about religious attitudes or behaviors, and then combine those items to a single Likert scale on religiosity.

When presenting methods and results, it is important to be clear if data were handled as Likert item data or Likert scale data.

This book will treat Likert data as individual Likert items, and will not create Likert scales.

Analysis of Likert item data

Likert data should be treated as ordinal data

There is some agreement that Likert item data should generally be treated as ordinal and not treated as interval/ratio data.

One consideration is that values in interval/ratio data need to be equally spaced. That is, 2 is equally between 1 and 3, and you could average 1 and 3 and the response would be 2. But it is not clear that “agree” is equally spaced between “strongly agree” and “neutral”. Nor is it clear that “strongly agree” and “neutral” could be averaged for a result of “agree”. Simply numbering the response levels does not make the responses interval/ratio data.

This book will treat Likert data as ordinal data. It will avoid using parametric tests, such as *t*-test and ANOVA, with Likert data. Likert data typically do not meet the assumptions of those parametric tests.

Instead, we will use nonparametric tests, permutation tests, and ordinal regression.

A couple other properties suggest that Likert data should not be treated as interval/ratio data. Likert data are not continuous; that is, there typically aren’t any decimal points in Likert responses. Also, the responses in Likert data are constrained at their ends; that is, on a five-point scale, the responses cannot be below 1 or above 5.

Where it is useful, this book will treat Likert data as nominal data for certain types of summaries. In general it is better to *not* treat ordinal data as nominal data in statistical analyses. One reason is that when treating the data as nominal data, the information about the ordered nature of the response categories is lost. However, sometimes it is useful to collapse Likert responses into categories; for example, grouping “strongly agree” and “agree” together as one category and reporting its frequency as a percentage of responses.

Some people treat Likert data as interval/ratio data

Not everyone agrees that Likert item data should not be treated as interval/ratio data. A quick search of the internet will produce plenty of examples of people defending treating Likert data as interval/ratio data.

Cases in which treating Likert responses as interval/ratio data may be reasonable include:

- When there are a high number of response options per question (say 10)
- When only the endpoints of the responses are indicated with text descriptors
- When response options are assumed to be equally spaced
- When respondents mark their answer on a line so that the precise location of the mark can be measured

Analysis of Likert scale data

When several Likert items are combined into a scale, so that there are many possible numeric outcomes, the results are often treated as interval/ratio data.

This is not entirely permissible from a theoretical point of view since Likert scales are made up of Likert items, and so have the same properties. But it is often a reasonable approach if the data meet the assumptions of the analysis. This is particularly the case if the scale data take on many values.

Analysis of Likert data

Ordinal regression

Probably the best tool for the analysis of experiments with Likert item data as the dependent variable is ordinal regression. The *ordinal* package in R provides a powerful and flexible framework for ordinal regression. It can handle a wide variety of experimental designs, including those with paired or repeated observations. Ordinal regression is relatively easy to perform in R, but might be somewhat challenging for the novice in statistical analyses. Occasionally there are problems with fitting models or checking model assumptions. These cases may be frustrating for the novice user.

Tests for ordinal tables

Another appropriate tool for the analysis of Likert item data are tests for ordinal data arranged in contingency table form. These include the linear-by-linear test, which is a test of association between two ordinal variables, and the Cochran-Armitage test, which is a test of association between an ordinal variable and a nominal variable. The major limitation to these tests is that they are limited to data arranged in a two-dimensional table. Also, these tests require the spacing between ordinal categories to be indicated. By default the tests assume that the categories are equally spaced, but the functions in R allow other spacing patterns to be used.

Permutation tests

Another tool appropriate for the analysis of Likert item data are permutation tests. The *coin* package in R provides a relatively powerful and flexible framework for permutation tests with ordinal dependent variables. It can handle models analogous to a one-way analysis of variance with stratification blocks, including paired or repeated observations. This covers more than all the designs that can be handled with the common traditional nonparametric tests.

Traditional nonparametric tests

Traditional nonparametric tests are generally considered appropriate for analyses with ordinal dependent variables. They have the advantages of being widely used and likely to be familiar for readers. One disadvantage of these tests is that the variety of designs they can handle is limited. The Kruskal-Wallis test can analyze a model analogous to a one-way analysis of variance. The Friedman and Quade tests can analyze data in an unreplicated complete block design with paired or repeated observations.

As a technical note, some authors have questioned using traditional nonparametric tests with Likert item data. One consideration is that the underlying statistics for some tests are based on the dependent variable being continuous in nature. Another consideration is that, while these tests have provisions to handle tied values, some authors worry that they may not behave well when there are *many* ties, as is likely for Likert data.

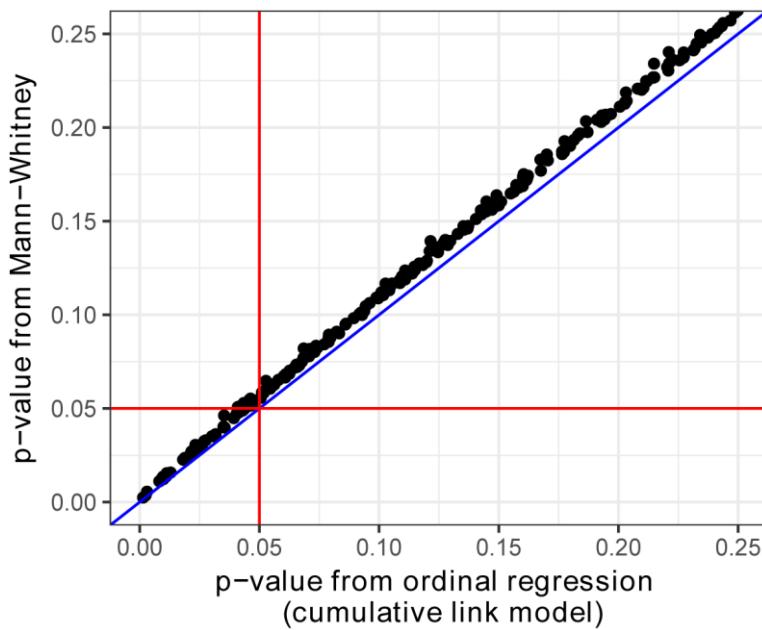
However, the results of the simulation studies below show that traditional nonparametric tests are good approximations for ordinal regression in most cases.

Optional: Simulated comparisons of traditional nonparametric tests and ordinal regression

These simulations used results from a 5-point Likert item as the dependent variable. Here, the results from the ordinal regression are used as the preferred standard.

Mann–Whitney test

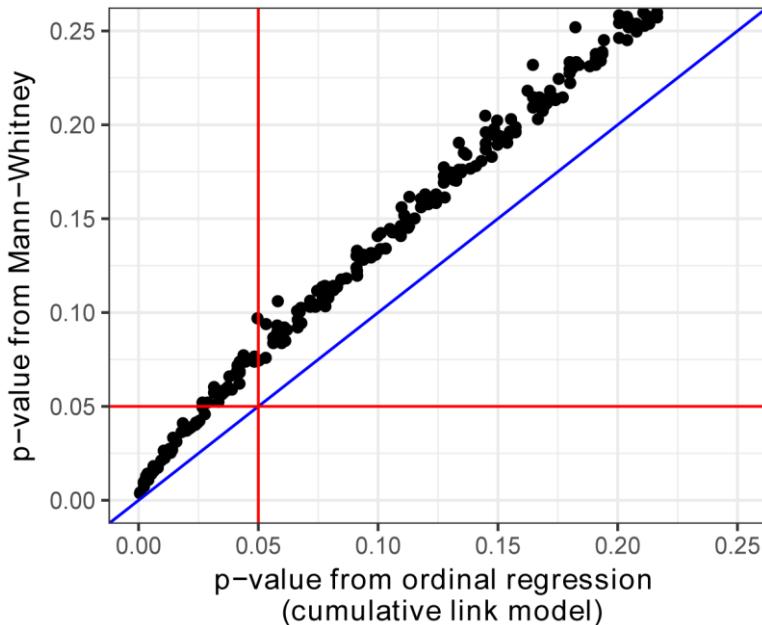
When sample sizes were reasonably large and equal between groups (n per group = 25), p -values from Mann–Whitney were closely related to those from ordinal regression, with the Mann–Whitney test being underpowered only slightly.



p -values from Mann–Whitney test compared to those from ordinal regression (cumulative link model) with simulated data. Dependent variable is 5-point Likert data. Both groups have equal sample sizes (n per group = 25). The blue line is the 1:1 line. The red lines indicate a p -value of 0.05 on each axis.

Small sample size

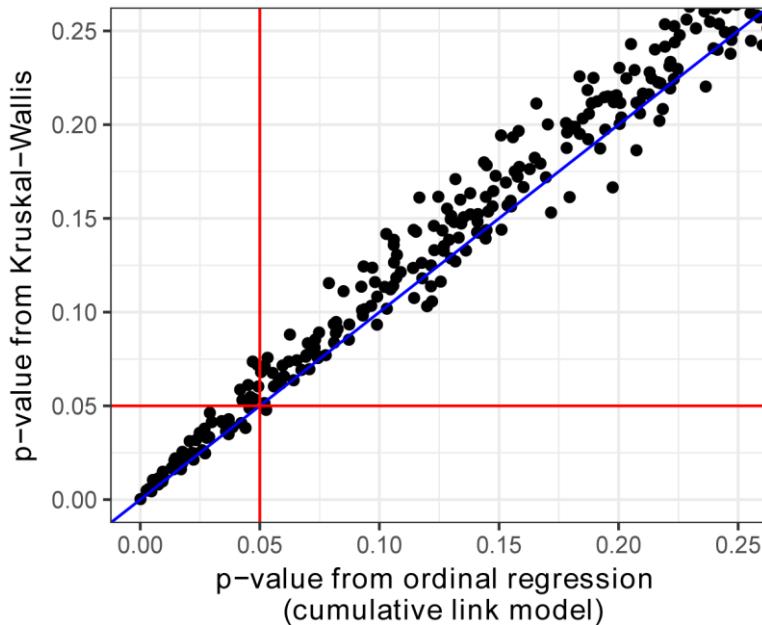
When sample sizes were small (n per group = 8), p -values from Mann–Whitney were still closely related to those from ordinal regression, but the Mann–Whitney test was underpowered compared with ordinal regression.



p-values from Mann–Whitney test compared to those from ordinal regression (cumulative link model) with simulated data. Dependent variable is 5-point Likert data. Both groups have equal sample sizes (n per group = 8). The blue line is the 1:1 line. The red lines indicate a p -value of 0.05 on each axis.

Kruskal–Wallis test

When there were more than two groups (here $k = 5$, with n per group = 25), p -values from Kruskal–Wallis were more dispersed relative to ordinal regression than were those from Mann–Whitney. Results from Kruskal–Wallis approximated those from ordinal regression relatively well for most cases in the region around $p = 0.05$ and below. In some cases, Kruskal–Wallis was underpowered in this region.



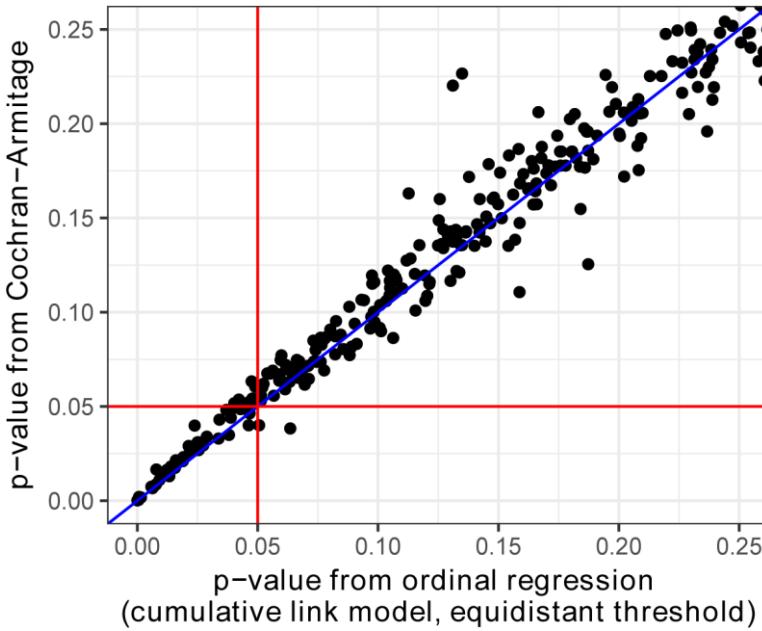
p-values from Kruskal–Wallis test compared to those from ordinal regression (cumulative link model) with simulated data. Dependent variable is 5-point Likert data. All groups have equal sample sizes (n per group = 25). The blue line is the 1:1 line. The red lines indicate a *p*-value of 0.05 on each axis.

Cochran–Armitage and permutation tests

Cochran–Armitage and permutation tests for ordinal data reasonably approximated results from ordinal regression for most cases in the region around $p = 0.05$ and below when the *threshold* = “*equidistant*” option was used for the cumulative link model. The Cochran–Armitage and permutation tests assume equal spacing of ordinal categories by default.

p-values for the methods matched less well when the *threshold* = “*equidistant*” option was not used, or when at least one group had few observations (not shown).

Results from Cochran–Armitage and permutation tests were very similar in the region around $p = 0.05$ and below (not shown).



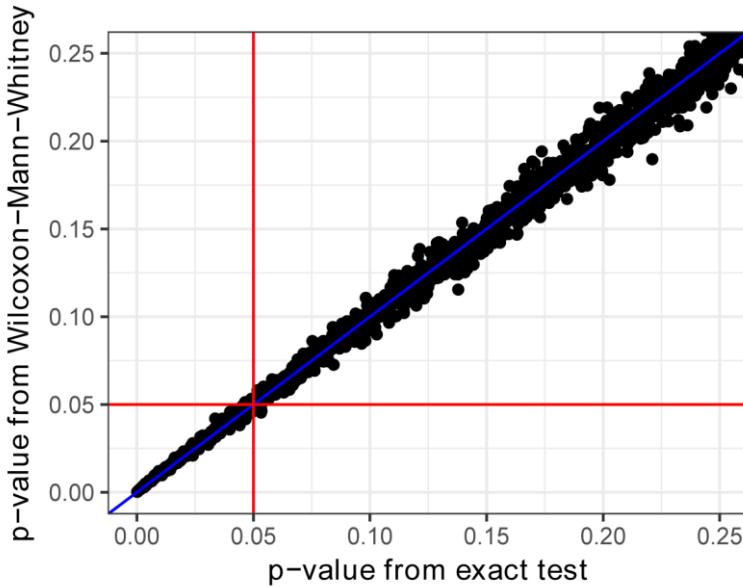
p-values from Cochran–Armitage test compared to those from ordinal regression (cumulative link model) assuming equally spaced categories in the ordinal variable. Dependent variable is 5-point Likert data. Two groups with equal sample sizes (n per group = 25). The blue line is the 1:1 line. The red lines indicate a p -value of 0.05 on each axis.

Optional: Simulated comparisons of traditional nonparametric tests and exact tests and Monte Carlo approaches

These simulations used results from a 5-point Likert item as the dependent variable. Here, the results from the exact test are assumed to be the preferred standard. The exact tests were conducted with the *exactRankTests* package. The results from this package appear to agree with those from the *coin* package for exact tests. Monte Carlo simulations used 10,000 iterations.

Mann–Whitney test

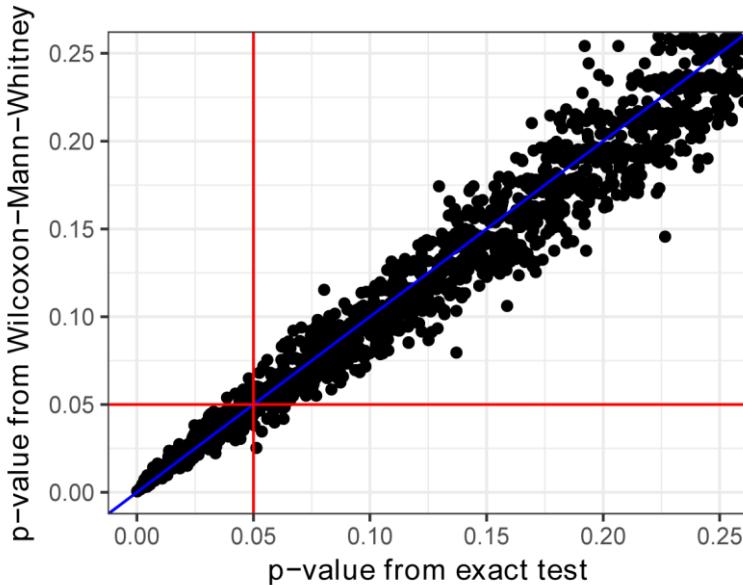
When sample sizes were reasonably large and equal between groups (n per group = 25), p -values from Mann–Whitney were closely related to those from the exact test in the $p = 0.05$ region, with some variability .



p-values from Mann–Whitney test compared to those from exact test with simulated data.
Dependent variable is 5-point Likert data. Both groups have equal sample sizes (n per group = 25).
The blue line is the 1:1 line. The red lines indicate a *p*-value of 0.05 on each axis.

Small sample size

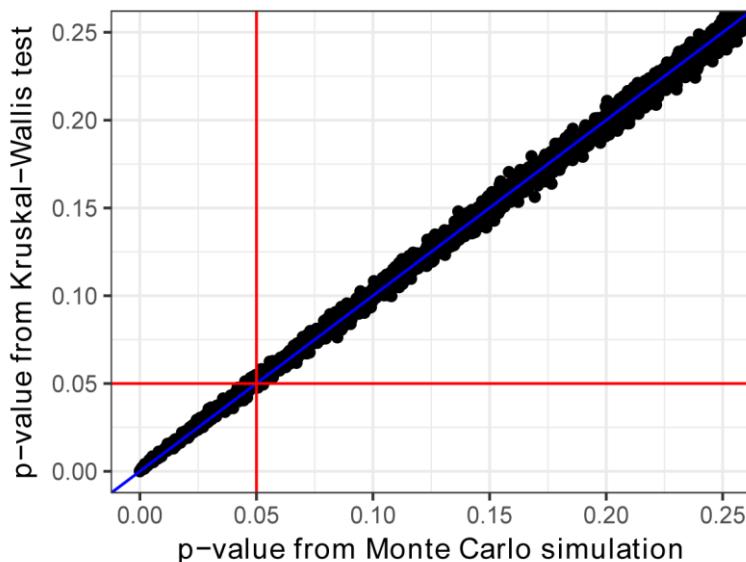
When sample sizes were small (n per group = 8), *p*-values from Mann–Whitney were still related to those from exact test but with more scattering of values.



p-values from Mann–Whitney test compared to those from exact test with simulated data.
Dependent variable is 5-point Likert data. Both groups have equal sample sizes (n per group = 8).
The blue line is the 1:1 line. The red lines indicate a *p*-value of 0.05 on each axis.

Mann–Whitney test and Monte Carlo

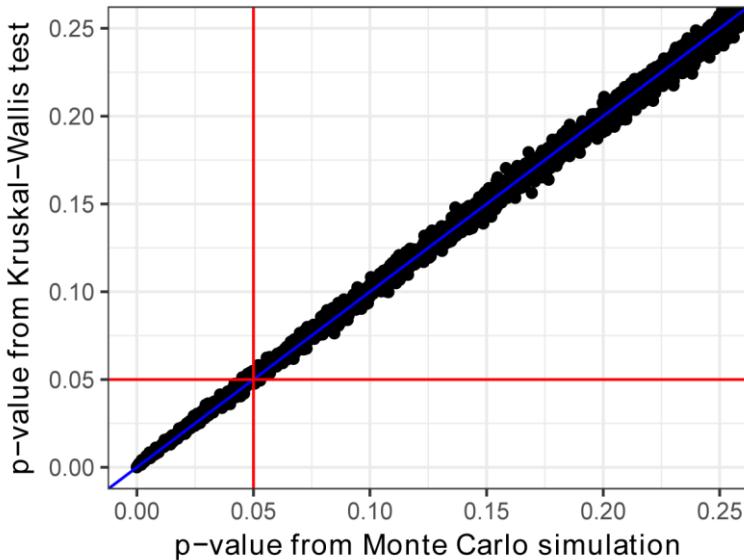
When sample sizes were reasonably large and equal between groups (n per group = 25), p -values from Mann–Whitney were closely related to those from Monte Carlo simulations of the Mann–Whitney test, with some variability.



p -values from Kruskal–Wallis test compared to those from Monte Carlo simulated Kruskal–Wallis with simulated data. Dependent variable is 5-point Likert data. All groups have equal sample sizes (n per group = 25). The blue line is the 1:1 line. The red lines indicate a p -value of 0.05 on each axis.

Kruskal–Wallis test and Monte Carlo

When there were more than two groups (here $k = 3$, with n per group = 25), p -values from Kruskal–Wallis were similar to those from Monte Carlo simulated Kruskal–Wallis, with some variability.



p-values from Kruskal–Wallis test compared to those from Monte Carlo simulated Kruskal–Wallis with simulated data. Dependent variable is 5-point Likert data. All groups have equal sample sizes (n per group = 25). The blue line is the 1:1 line. The red lines indicate a *p*-value of 0.05 on each axis.

Optional reading

"Oh Ordinal data, what do we do with you?" from Dr. Nic. 2013. Learn and Teach Statistics & Operations Research. learnandteachstatistics.wordpress.com/2013/07/08/ordinal/.

"Can Likert Scale Data ever be Continuous?" from Grace-Martin, K. 2008. The Analysis Factor. www.theanalysisfactor.com/can-likert-scale-data-ever-be-continuous/.

Descriptive Statistics for Likert Data

Median

For Likert item data, the primary measure of location we will use is the median. For a set of numbers, the median is the middle value when the data are arranged in numerical order. For example, for the set of values 1, 2, 3, 4, 5, 5, 5, the median value is 4. If there are an even number of values, the two middle values are averaged.

In concept, 50% of observations are less than the median, and 50% are greater than the median (ignoring the median itself in cases of an odd number of observations). You could also find the median of ordinal data consisting just of text descriptors. For example, for the set of values:

“high school”, “associate degree”, “bachelor’s degree”, “master’s degree”, “doctorate”,

the median is “bachelor’s degree”.

Using medians makes sense for ordinal data, whereas using means would in general not be appropriate.

Count of responses and bar plot

In order to visualize the location and variation of Likert data, we will look at the distribution of the responses. The distribution of responses can be enumerated by counting the number of responses for each response level, and can be visualized with a bar plot of these counts. This is similar to a histogram of the responses.

Quartiles and percentiles

As a measure of the variation of data, it is also useful to look at the quartiles of the data set. The first quartile indicates the value below which 25% of the values fall. It is equivalent to the 25th percentile. The second quartile is equivalent to the median. The third quartile indicates the value below which 75% of the values fall. It is equivalent to the 75th percentile.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- ggplot2
- plyr
- boot
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(plyr)){install.packages("plyr")}
if(!require(boot)){install.packages("boot")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Descriptive statistics for one-sample data

Imagine a scenario in which 10 respondents are evaluating a single speaker on a single Likert item. The following example will create a data frame called *Data* consisting of three variables: *Speaker*, *Rater*, and *Likert*. The speaker is the same for all observations, and *Likert* represents the response on a five-point Likert item.

One-sample data

One-sample data refers to a single set of values that is not broken into groups. In this example, we have only one speaker with 10 ratings, hence one set of ratings values.

Variables and functions in this example

A new variable, *Likert.f*, will be created in the data frame. It will have the same values as the *Likert* variable, but we will tell R to treat it as an ordered factor variable, whereas *Likert* is a numeric variable. When summarizing data as a nominal variable, *Likert.f* will be used. But when summarizing data as a numeric variable, *Likert* will be used.

Note that the *str* function reports that *Speaker* is a factor variable, *Rater* and *Likert* are integer variables, and *Likert.f* is an ordered factor variable. The levels of the factor variables are shown with the *level* function.

Recall that *Data\$Likert* tells R to use the *Likert* variable in the data frame *Data*.

```
Input ="
  Speaker      Rater Likert
  'Maggie Simpson' 1     3
  'Maggie Simpson' 2     4
  'Maggie Simpson' 3     5
  'Maggie Simpson' 4     4
  'Maggie Simpson' 5     4
  'Maggie Simpson' 6     4
  'Maggie Simpson' 7     4
  'Maggie Simpson' 8     3
  'Maggie Simpson' 9     2
  'Maggie Simpson' 10    5
")

Data = read.table(textConnection(Input), header=TRUE)

### Create a new variable which is the Likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                      ordered = TRUE,
                      levels = c("1", "2", "3", "4", "5"))

### Double check our data frame

library(psych)

headTail(Data)

  Speaker Rater Likert Likert.f
1 Maggie Simpson 1     3     3
2 Maggie Simpson 2     4     4
3 Maggie Simpson 3     5     5
4 Maggie Simpson 4     4     4
```

```

... <NA> ...
7 Maggie Simpson 7 4 4
8 Maggie Simpson 8 3 3
9 Maggie Simpson 9 2 2
10 Maggie Simpson 10 5 5

```

```
str(Data)
```

```

'data.frame':   10 obs. of  4 variables:
$ Speaker : Factor w/ 1 level "Maggie Simpson": 1 1 1 1 1 1 1 1 1 1
$ Rater   : int  1 2 3 4 5 6 7 8 9 10
$ Likert  : int  3 4 5 4 4 4 4 3 2 5
$ Likert.f: Ord.factor w/ 5 levels "1"<"2"<"3"<"4"<...: 3 4 5 4 4 4 4 3 2 5

```

```
levels(Data$Speaker)
```

```
[1] "Maggie Simpson"
```

```
levels(Data$Likert.f)
```

```
[1] "1" "2" "3" "4" "5"
```

```
summary(Data)
```

```
### Remove unnecessary objects
```

```
rm(Input)
```

Summary treating Likert data as nominal data

The number of responses for each response level of *Likert.f* can be shown with the *summary* function, either on the *Likert.f* variable itself or as part of the whole data frame *Data*.

This same summary is then visualized with a bar plot.

```
summary(Data$Likert.f)
```

```
1 2 3 4 5
0 1 2 5 2
```

```
### The top row is the value of Likert.f,
### and the next row is the counts for each value.
```

```
summary(Data)
```

	Speaker	Rater	Likert	Likert.f
Maggie Simpson:10	Min. : 1.00	Min. : 2.00	1:0	
	1st Qu.: 3.25	1st Qu.: 3.25	2:1	

Median : 5.50	Median : 4.00	3:2
Mean : 5.50	Mean : 3.80	4:5
3rd Qu.: 7.75	3rd Qu.: 4.00	5:2
Max. : 10.00	Max. : 5.00	

xtabs and bar plot

```
XT = xtabs(~ Likert.f,
           data=Data)

XT

Likert.f
1 2 3 4 5
0 1 2 5 2

### Counts for each level of Likert.f

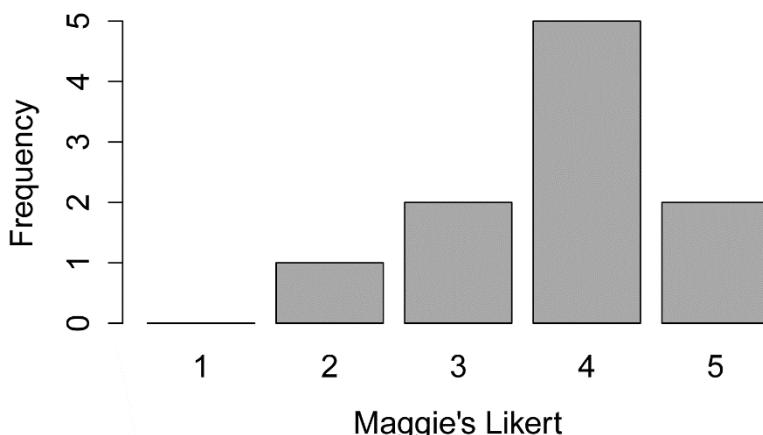
sum(XT)
[1] 10

### Sum of counts for whole table

prop.table(XT)
 1   2   3   4   5 
0.0 0.1 0.2 0.5 0.2

### Proportion of total for each count
```

```
barplot(XT,
        col="dark gray",
        xlab="Maggie's Likert",
        ylab="Frequency")
```



Summary treating Likert data as numeric data

In this code the Likert response data is treated as numeric data. The *summary* function produces the median, as well as the minimum, maximum, first quartile, and third quartile for a variable. The *summary* function can be used for either a single variable or a whole data frame. The *Summarize* function in the *FSA* package produces those statistics as well as the number of observations (*n*).

These statistics are visualized with a box plot. Note that the heavy line in the box is the median, and the ends of the box are the first quartile and third quartile. The extent of the whiskers in the box plot defaults to being the most extreme values, but no more than 1.5 times the length of the box itself. Values beyond the extent of the whiskers will be indicated with circles. Note that in the box plot for this example that the whiskers extend to the most extreme values in the data (2 and 5), and that there are no circles in the plot.

```
summary(Data$Likert)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2.00	3.25	4.00	3.80	4.00	5.00

```
summary(Data)
```

Speaker	Rater	Likert	Likert.f
Maggie Simpson:10	Min. : 1.00	Min. :2.00	1:0
	1st Qu.: 3.25	1st Qu.:3.25	2:1
	Median : 5.50	Median :4.00	3:2
	Mean : 5.50	Mean :3.80	4:5
	3rd Qu.: 7.75	3rd Qu.:4.00	5:2
	Max. :10.00	Max. :5.00	

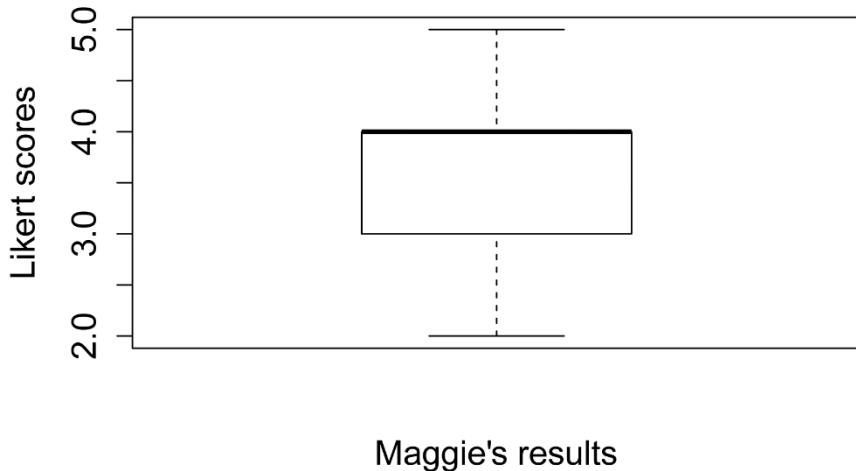
```
library(FSA)
```

```
Summarize(Data$Likert,
           digits=3)
```

n	mean	sd	min	q1	median	q3	max	percZero
10.000	3.800	0.919	2.000	3.250	4.000	4.000	5.000	0.000

Box plot

```
boxplot(Data$Likert,
        ylab="Likert scores",
        xlab="Maggie's results")
```



Descriptive statistics for one-way or multi-way data

For this example, imagine that there are two speakers, each of which is being evaluated on each of three questions on 10-point Likert items. There are six raters, each of whom evaluates each speaker on each of these items.

One-way data

One-way data refers to a data set with a single measured variable, but divided into groups. In this example looking at values of *Likert* for *Spongebob* and for *Patrick* would be an example of one-way data. Note that for one-way data there is a single dependent variable (*Likert*) and a single independent factor variable (*Speaker*).

Looking at *Likert* for the three levels of *Question* (*Information*, *Presentation*, and *Questions*) would also be an example of one-way data.

Multi-way data

Multi-way data refers to looking at a single measured variable divided into groups, where the groups are defined by at least two factor variables. If we look at values of *Likert* for each of the two speakers for each of the three questions, we are looking at two-way data.

Variables and functions in this example

The following example will create a data frame called *Data* consisting of four variables: *Speaker*, *Question*, *Rater*, and *Likert*. An additional variable named *Likert.f* will be created as an ordered factor variable with the same value as *Likert* for each observation.

Note that the *str* function reports that *Speaker*, *Question*, and *Rater* are factor variables, *Likert* is an integer variable, and *Likert.f* is an ordered factor variable. The levels of the factor variables are shown with the *level* function.

Recall that *Data\$Likert* tells R to use the *Likert* variable in the data frame *Data*.

```

Input ="
Speaker Question Rater Likert
Spongebob Information a 5
Spongebob Information b 6
Spongebob Information c 7
Spongebob Information d 8
Spongebob Information e 6
Spongebob Information f 5
Spongebob Presentation a 8
Spongebob Presentation b 7
Spongebob Presentation c 8
Spongebob Presentation d 9
Spongebob Presentation e 10
Spongebob Presentation f 7
Spongebob Questions a 3
Spongebob Questions b 4
Spongebob Questions c 5
Spongebob Questions d 4
Spongebob Questions e 6
Spongebob Questions f 5
Patrick Information a 6
Patrick Information b 7
Patrick Information c 8
Patrick Information d 9
Patrick Information e 7
Patrick Information f 6
Patrick Presentation a 9
Patrick Presentation b 9
Patrick Presentation c 10
Patrick Presentation d 10
Patrick Presentation e 8
Patrick Presentation f 8
Patrick Questions a 5
Patrick Questions b 6
Patrick Questions c 6
Patrick Questions d 7
Patrick Questions e 5
Patrick Questions f 7
")
Data = read.table(textConnection(Input),header=TRUE)

### Order factor levels, otherwise R will alphabetize them

Data$Speaker = factor(Data$Speaker,
levels=unique(Data$Speaker))

Data$Question = factor(Data$Question,
levels=unique(Data$Question))

### Create an ordered factor of Likert data

```

```

Data$Likert.f = factor(Data$Likert,
                       ordered=TRUE,
                       levels = c("1", "2", "3", "4", "5",
                                 "6", "7", "8", "9", "10")
                     )

### Examine data frame

library(psych)

headTail(data)

      Speaker   Question Rater Likert Likert.f
1   Spongebob Information    a      5       5
2   Spongebob Information    b      6       6
3   Spongebob Information    c      7       7
4   Spongebob Information    d      8       8
...     <NA>      <NA>  <NA>   ...     <NA>
33  Patrick   Questions    c      6       6
34  Patrick   Questions    d      7       7
35  Patrick   Questions    e      5       5
36  Patrick   Questions    f      7       7

str(Data)

'data.frame':   36 obs. of  5 variables:
 $ Speaker : Factor w/ 2 levels "Spongebob","Patrick": 1 1 1 1 1 1 1 1 1 ...
 $ Question: Factor w/ 3 levels "Information",...: 1 1 1 1 1 1 2 2 2 ...
 $ Rater   : Factor w/ 6 levels "a","b","c","d",...: 1 2 3 4 5 6 1 2 3 4 ...
 $ Likert  : int  5 6 7 8 6 5 8 7 8 9 ...
 $ Likert.f: Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<...: 5 6 7 8 6 5 8 7 8 9 ...

levels(Data$Speaker)

[1] "Spongebob" "Patrick"

levels(Data$Question)

[1] "Information" "Presentation" "Questions"

levels(Data$Rater)

[1] "a" "b" "c" "d" "e" "f"

levels(Data$Likert.f)

[1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"

```

```
summary(Data)

### Remove unnecessary objects

rm(Input)
```

Summary treating Likert data as nominal data

The number of responses for each response level of *Likert.f* is shown with the *summary* function, but note that these counts are for the entire data frame.

The *xtabs* function can be used to generate counts across groups. This process is known as *cross-tabulation*, or *cross-tabs*. The *prop.table* function, with the *margin=1* option produces the proportions of observations for each row.

This same summary is then visualized with a bar plot.

```
summary(Data)

      Speaker      Question    Rater      Likert      Likert.f
Spongebob:18  Information :12  a:6   Min.   : 3.000   6   :7
Patrick   :18  Presentation:12  b:6   1st Qu.: 5.750   7   :7
              Questions   :12  c:6   Median  : 7.000   5   :6
                               d:6   Mean   : 6.833   8   :6
                               e:6   3rd Qu.: 8.000   9   :4
                               f:6   Max.   :10.000  10  :3
                               (Other):3
```

```
XT = xtabs(~ Speaker + Likert.f, data=Data)
```

```
XT
```

	Likert.f									
Speaker	1	2	3	4	5	6	7	8	9	10
Spongebob	0	0	1	2	4	3	3	3	1	1
Patrick	0	0	0	0	2	4	4	3	3	2

```
prop.table(XT,
           margin = 1)

      Likert.f
Speaker      1      2      3      4      5      6      7      8      9      10
Spongebob 0.0000 0.0000 0.0556 0.1111 0.2222 0.1667 0.1667 0.1667 0.0556 0.0556
Patrick   0.0000 0.0000 0.0000 0.0000 0.1111 0.2222 0.2222 0.1667 0.1667 0.1111
```

```
sum(XT)
```

```
[1] 36
```

```
### Sum of counts in the table
```

```
rowSums(XT)
```

	Spongebob	Patrick
18	18	

```
### Sum of counts for each row
```

```
colSums(XT)
```

1	2	3	4	5	6	7	8	9	10
0	0	1	2	6	7	7	6	4	3

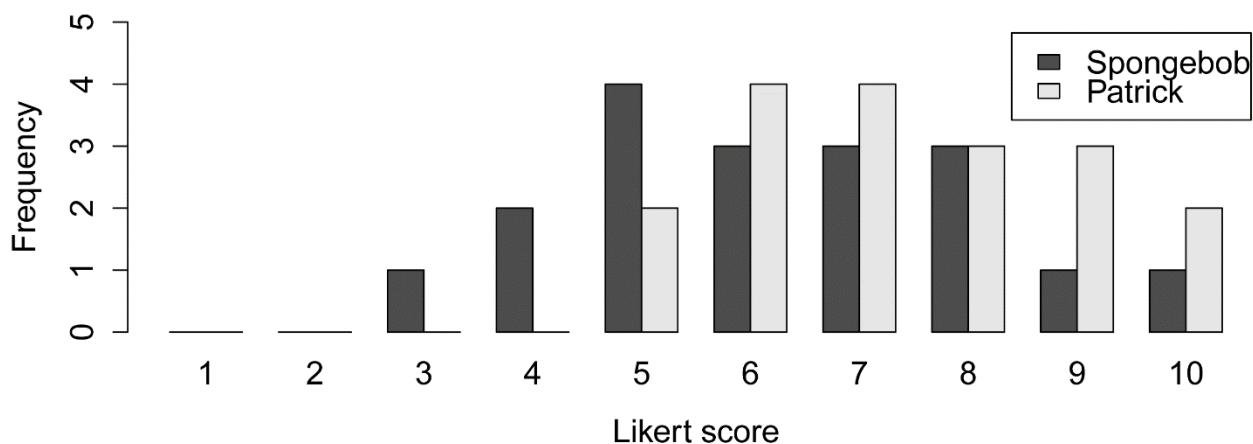
```
### Sum of counts for each column
```

Bar plot by group

```
XT = xtabs(~ Speaker + Likert.f, data=Data)
```

```
XT
```

```
barplot(XT,
        beside=TRUE,
        legend=TRUE,
        ylim=c(0, 5), # adjust to remove legend overlap
        xlab="Likert score",
        ylab="Frequency"
      )
```



xtabs to produce counts for two-way data

```
### Note that the dependent variable (Likert.f)
### is the middle variable in the formula
```

```
XT2 = xtabs(~ Speaker + Likert.f + Question,
```

```

data>Data)

xt2

, , Question = Information

    Likert.f
Speaker   1 2 3 4 5 6 7 8 9 10
Spongebob 0 0 0 0 2 2 1 1 0 0
Patrick   0 0 0 0 0 2 2 1 1 0

, , Question = Presentation

    Likert.f
Speaker   1 2 3 4 5 6 7 8 9 10
Spongebob 0 0 0 0 0 0 2 2 1 1
Patrick   0 0 0 0 0 0 0 2 2 2

, , Question = Questions

    Likert.f
Speaker   1 2 3 4 5 6 7 8 9 10
Spongebob 0 0 1 2 2 1 0 0 0 0

```

Summary treating Likert data as numeric data

In this code the *Likert* response data is treated as numeric data. The *summary* function produces the median, quartiles, and extremes. But note that these statistics are for all values for the whole data frame.

The *Summary* function in the *FSA* package can produce these summary statistics across groups for one-way data or multi-way data.

Histograms are then produced to visualize the distribution of responses across groups. Histograms for both one-way and two-way data can be produced with the *histogram* function in the *lattice* package.

Finally, box plots of the two-way data are produced.

```

summary(Data)

      Speaker       Question     Rater      Likert      Likert.f
Spongebob:18  Information :12  a:6  Min.   : 3.000   6      :7
Patrick   :18  Presentation:12 b:6  1st Qu.: 5.750   7      :7
                  Questions   :12 c:6  Median  : 7.000   5      :6
                               d:6  Mean    : 6.833   8      :6
                               e:6  3rd Qu.: 8.000   9      :4
                               f:6  Max.   :10.000  10     :3
                                         (Other):3

```

```
library(FSA)
```

```
Summarize(Likert ~ Question,
```

```

data=Data,
digits=3)

      Question n  mean    sd min   Q1 median   Q3 max percZero
1 Information 12 6.667 1.231   5 6.00    6.5 7.25    9      0
2 Presentation 12 8.583 1.084   7 8.00    8.5 9.25   10      0
3 Questions   12 5.250 1.215   3 4.75    5.0 6.00    7      0

library(FSA)

Summarize(Likert ~ Speaker + Question,
           data=Data,
           digits=3)

      Speaker     Question n  mean    sd min   Q1 median   Q3 max percZero
1 Spongebob Information 6 6.167 1.169   5 5.25    6.0 6.75    8      0
2 Patrick    Information 6 7.167 1.169   6 6.25    7.0 7.75    9      0
3 Spongebob  Presentation 6 8.167 1.169   7 7.25    8.0 8.75   10      0
4 Patrick    Presentation 6 9.000 0.894   8 8.25    9.0 9.75   10      0
5 Spongebob   Questions  6 4.500 1.049   3 4.00    4.5 5.00    6      0
6 Patrick     Questions  6 6.000 0.894   5 5.25    6.0 6.75    7      0

```

Bar plots by group

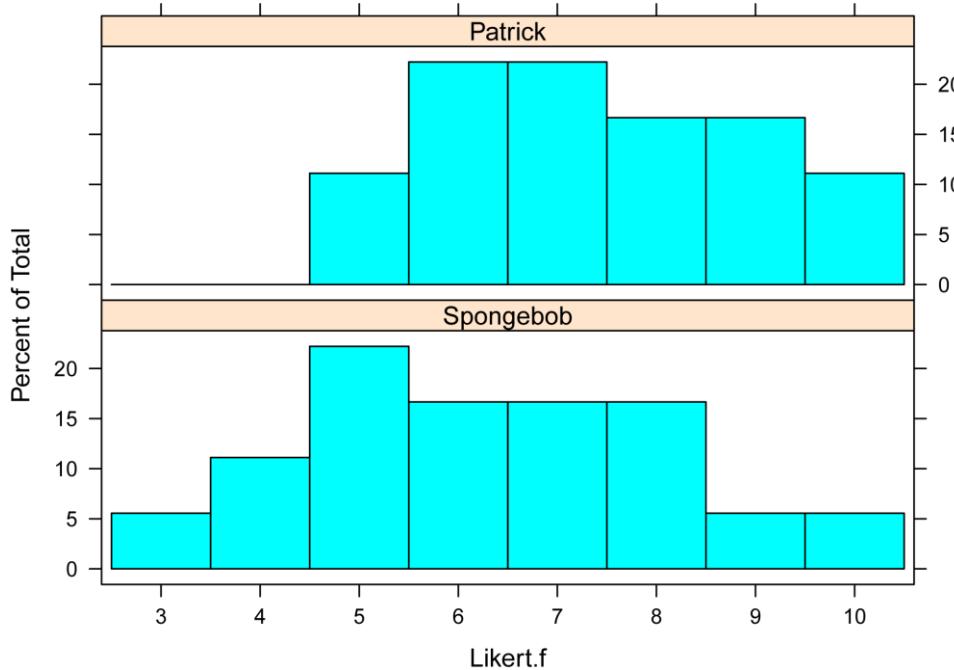
The *histogram* function in the *lattice* package will produce bar plots if the counted variable is ordinal in nature, as *Likert.fis*.

```

library(lattice)

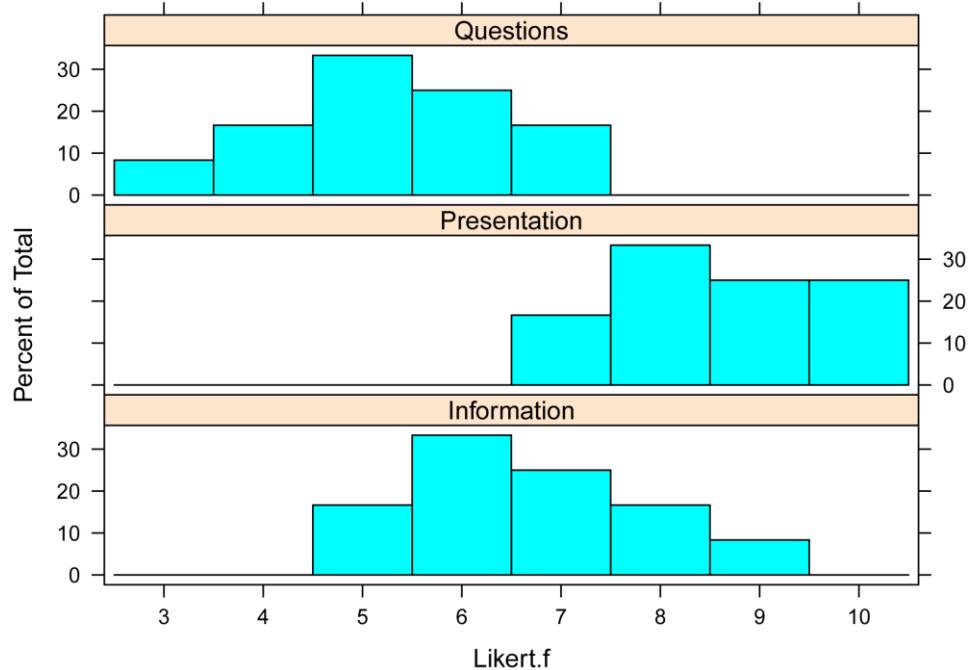
histogram(~ Likert.f | Speaker,
          data=Data,
          layout=c(1,2)      # columns and rows of individual plots
)

```



```
library(lattice)

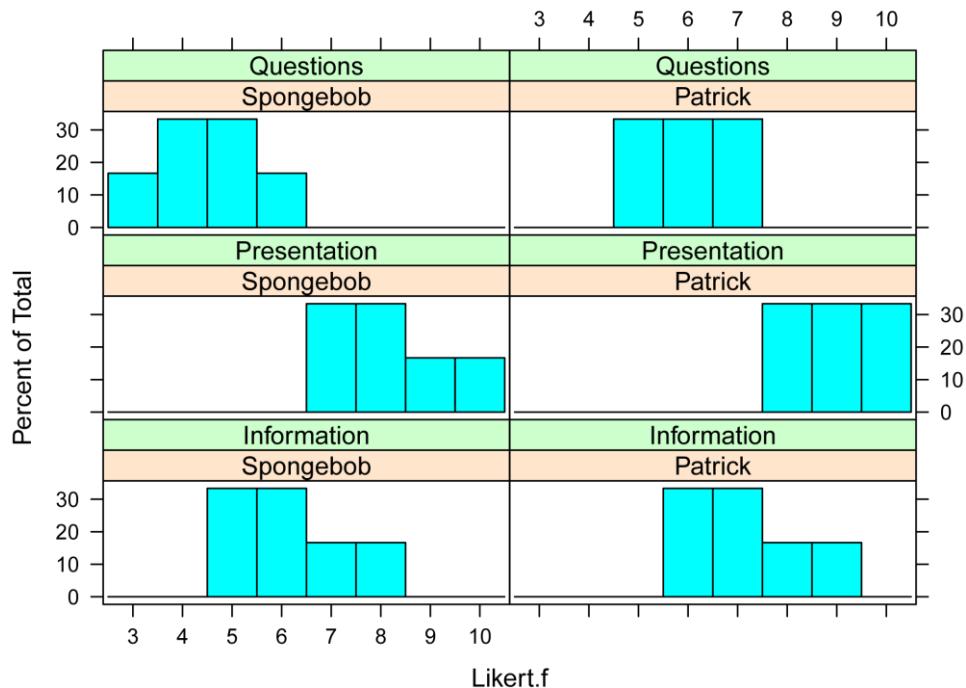
histogram(~ Likert.f | Question,
           data=Data,
           layout=c(1,3)      # columns and rows of individual plots
           )
```



Bar plots for two-way data

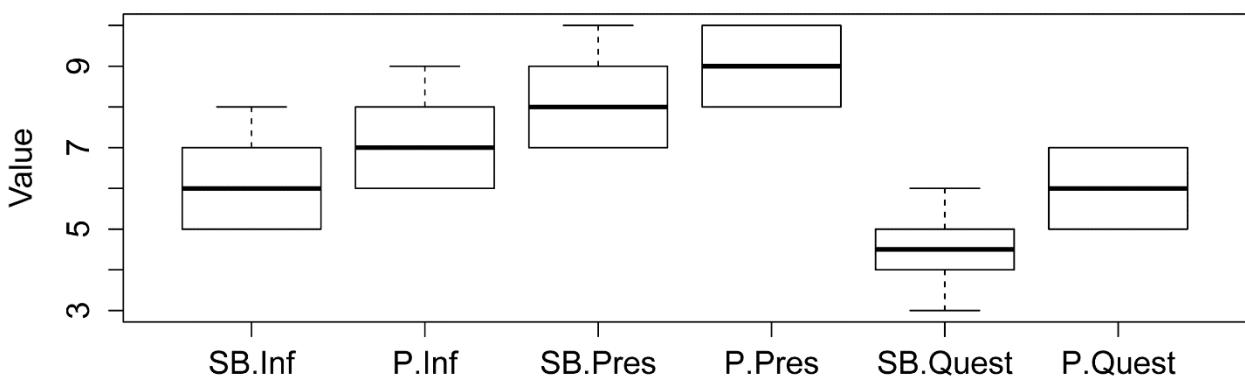
```
library(lattice)

histogram(~ Likert.f | speaker + Question,
          data=Data,
          layout=c(2,3)      # columns and rows of individual plots
)
```

Box plots for two-way data

In general, using the *names* option is not necessary, but here it is useful because the default group names are long.

```
boxplot(Likert ~ Speaker + Question,
        data=Data,
        names=c("SB.Inf", "P.Inf", "SB.Pres", "P.Pres", "SB.Quest", "P.Quest"),
        ylab="Value")
```



```
### Note in this example, SB is Spongebob, P is Patrick,
### Inf is Information, Pres is Presentation,
### and Quest is Question.
```

Interaction plot using medians and quartiles

Interaction plots are useful to visualize two-way data by plotting the value of a measured variable across the interaction of two factor variables.

In this example, the median *Likert* is calculated for each level of the interaction of two factors, and the first and third quartiles are used to indicate the spread of data about each median.

The *Summarize* function in the *FSA* package creates a data frame called *Sum*. The variables *median*, *Q1*, and *Q3* in this data frame will then be used in the plot.

The interaction plot is produced with the *ggplot2* package. This package is very versatile and powerful for creating plots, but the code can be intimidating at first. Note that the code indicates that the data frame to use is *Sum*, and the use of the variables *Speaker*, *median*, *Question*, *Q1*, *Q3* from this data frame. Also note that the y axis label defined as “Median Likert score”.

```
### Create a data frame called Sum with median and quartiles

library(FSA)

Sum = Summarize(Likert ~ Speaker + Question,
                 data=Data,
                 digits=3)

Sum

      Speaker    Question n  mean   sd min   Q1 median   Q3 max percZero
1 Spongebob Information 6 6.167 1.169  5 5.25   6.0 6.75   8     0
2   Patrick Information 6 7.167 1.169  6 6.25   7.0 7.75   9     0
3 Spongebob  Presentation 6 8.167 1.169  7 7.25   8.0 8.75  10     0
4   Patrick  Presentation 6 9.000 0.894  8 8.25   9.0 9.75  10     0
5 Spongebob    Questions 6 4.500 1.049  3 4.00   4.5 5.00   6     0
6   Patrick    Questions 6 6.000 0.894  5 5.25   6.0 6.75   7     0

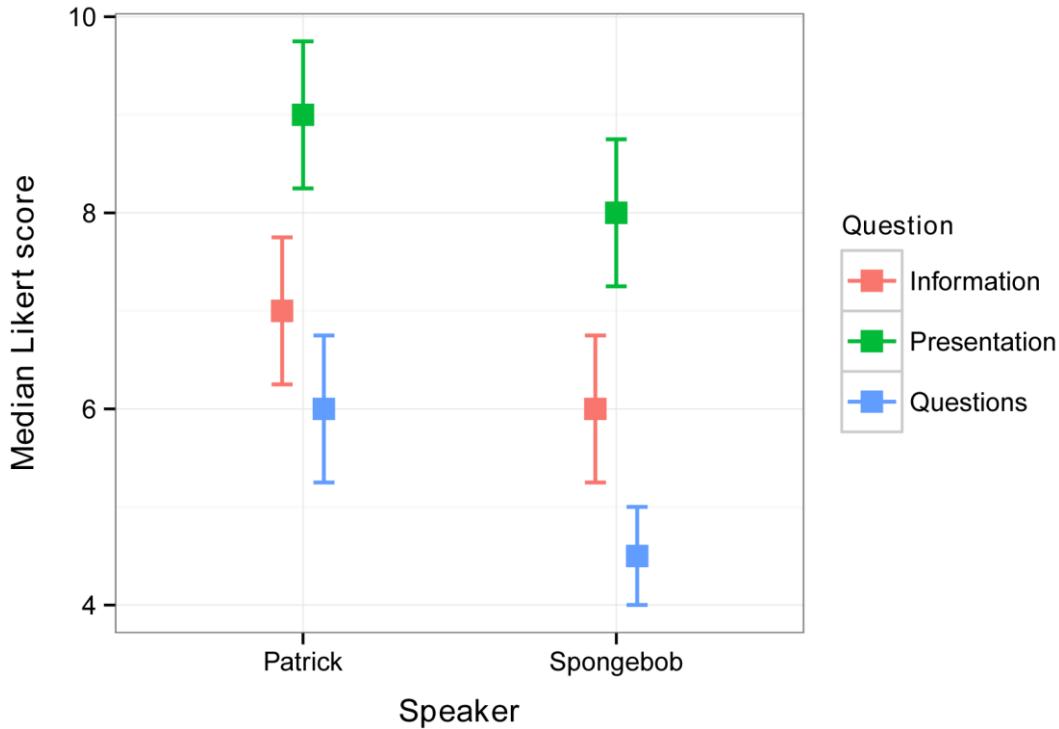
### Produce interaction plot

library(ggplot2)

pd = position_dodge(.2)

ggplot(Sum, aes(x=Speaker,
                 y=median,
                 color=Question)) +
  geom_errorbar(aes(ymin=Q1,
                    ymax=Q3),
                width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position=pd) +
```

```
theme_bw() +
  theme(axis.title = element_text(face = "bold"))+
  ylab("Median Likert score")
```



Interaction plot using medians and confidence intervals

An interaction plot can also use confidence intervals of the medians for its error bars. Confidence intervals are discussed in the *Confidence Intervals* chapter, and confidence intervals for medians are discussed in the *Confidence Intervals for Medians* chapter.

The function *groupwiseMedian* in the *rcompanion* package creates a data frame, called *Sum*, here that includes medians and confidence intervals of the medians for one-way or multi-way data. By default the function uses confidence intervals by the BCa method, but can produce confidence intervals by other methods. Note that bootstrapped confidence intervals may not be reliable for discreet data, such as the ordinal Likert data used in these examples, especially for small samples.

Also, note that the order of speakers has changed from the previous interaction plot.

```
library(rcompanion)

Sum = groupwiseMedian(Likert ~ Speaker + Question,
                      data = Data,
                      conf=0.95,
                      R=5000)

Sum
```

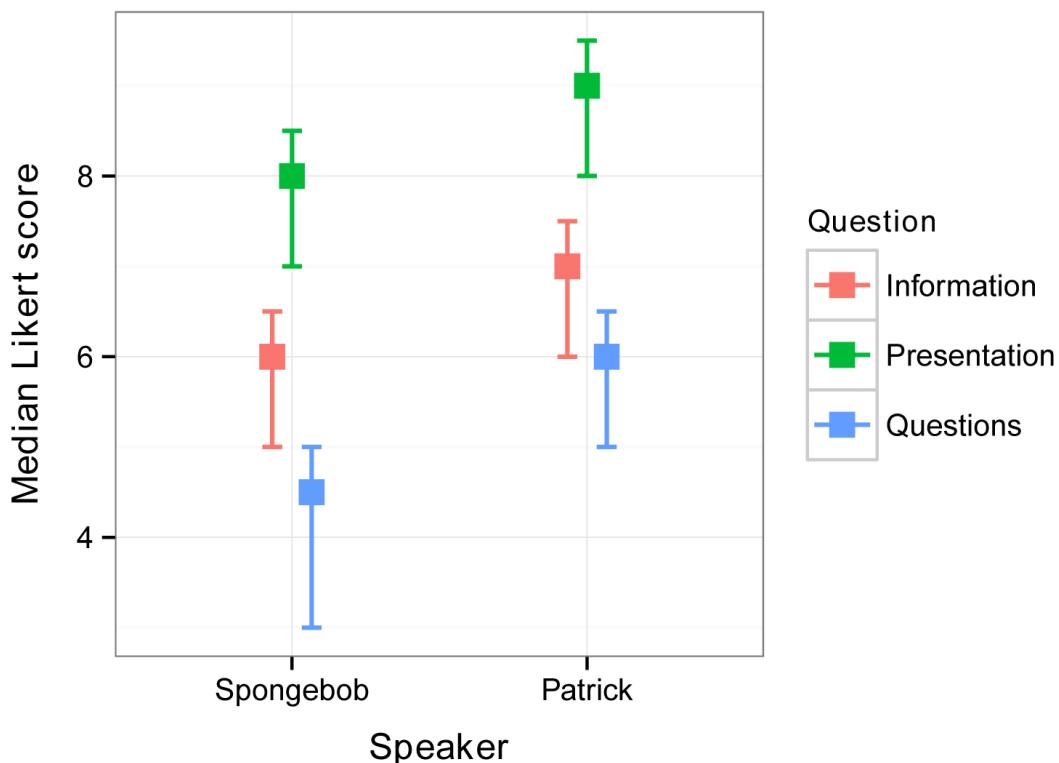
Speaker	Question	n	Median	Conf.level	Bca.lower	Bca.upper
Patrick	Information	10	7.0	0.95	6.2	7.8
	Presentation	10	9.0	0.95	8.2	9.8
	Questions	10	6.0	0.95	5.2	6.8
Spongebob	Information	10	6.0	0.95	5.3	6.7
	Presentation	10	8.0	0.95	7.2	8.8
	Questions	10	4.5	0.95	4.0	5.0

1	Spongebob	Information	6	6.0	0.95	5	6.5
2	Spongebob	Presentation	6	8.0	0.95	7	8.5
3	Spongebob	Questions	6	4.5	0.95	3	5.0
4	Patrick	Information	6	7.0	0.95	6	7.5
5	Patrick	Presentation	6	9.0	0.95	8	9.5
6	Patrick	Questions	6	6.0	0.95	5	6.5

```
library(ggplot2)

pd = position_dodge(.2)

ggplot(Sum, aes(x=Speaker,
                 y=Median,
                 color=Question)) +
  geom_errorbar(aes(ymin=Bca.lower,
                     ymax=Bca.upper),
                width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Median Likert score")
```



Descriptive statistics with opt-out responses

Opt-out responses include options like “don’t know” or “not applicable”. A typical approach to handling opt-out responses in Likert data is to separate them from the numeric or ordered factor responses of the question and report them separately.

For example, if responses to a question were 5, 5, 5, 4, 4, 2, ‘Don’t know’, ‘N/A’, you might report that you had 8 total responses, with 12.5% “Don’t know”, 12.5% “N/A”, and a median response of 4.5.

Analysis for vector data

The *xtabs* function can be used to count the levels of responses with opt-out answers, treating the data as nominal.

The *Summarize* function in the *FSA* will calculate the median for numeric data. Note that the numeric summary includes the number of observations and the number of “valid” observations, which excludes the non-numeric values.

```
Q1 = c("5", "5", "5", "4", "4", "2", "Don't know", "N/A")
```

Counts of responses

```
Q1.f = as.factor(Q1)
```

```
XT = xtabs(~Q1)
```

```
XT
```

Q1	2	4	5	Don't know	N/A
	1	2	3	1	1

```
prop.table(XT)
```

Q1	2	4	5	Don't know	N/A
	0.125	0.250	0.375	0.125	0.125

Numerical summary of responses

```
Q1.n = as.numeric(Q1)
```

```
library(FSA)
```

```
Summarize(Q1.n,
          digits = 2)
```

n	nvalid	mean	sd	min	Q1	median	Q3	max
8.00	6.00	4.17	1.17	2.00	4.00	4.50	5.00	5.00

Analysis for long-form data frame

Similar summaries can be obtained for data in a long-form data frame.

Note that the *read.table* function by default reads in the values of *Likert* as a factor variable, since it contains character values, not as a character variable. Also note that you shouldn't use *NA* for a value of "not applicable" since *NA* is reserved by R for a missing value.

```
Input ="
  Rater Question Likert
  '1'   Q1      5
  '2'   Q1      5
  '3'   Q1      5
  '4'   Q1      4
  '5'   Q1      4
  '6'   Q1      2
  '7'   Q1      DontKnow
  '8'   Q1      NotApp
  '1'   Q2      3
  '2'   Q2      3
  '3'   Q2      3
  '4'   Q2      2
  '5'   Q2      DontKnow
  '6'   Q2      2
  '7'   Q2      DontKnow
  '8'   Q2      DontKnow
  '1'   Q3      1
  '2'   Q3      2
  '3'   Q3      3
  '4'   Q3      2
  '5'   Q3      3
  '6'   Q3      1
  '7'   Q3      2
  '8'   Q3      3
  ")
Data = read.table(textConnection(Input), header=TRUE)
```

Counts of responses

```
XT = xtabs(~ Question + Likert,
           data = Data)

XT
```

		Likert					
Question	1	2	3	4	5	DontKnow	NotApp
Q1	0	1	0	2	3	1	1
Q2	0	2	3	0	0	3	0
Q3	2	3	3	0	0	0	0

```
### Counts for each question
```

```
prop.table(XT,
           margin=1)

  Likert
Question 1 2 3 4 5 DontKnow NotApp
  Q1 0.000 0.125 0.000 0.250 0.375 0.125 0.125
  Q2 0.000 0.250 0.375 0.000 0.000 0.375 0.000
  Q3 0.250 0.375 0.375 0.000 0.000 0.000 0.000

### Proportions for each row
```

Numerical summary of responses

```
Data$Likert.n = as.numeric(as.character(Data$Likert))

library(FSA)

Summarize(Likert.n ~ Question,
          data = Data)

  Question n nvalid      mean        sd min   Q1 median   Q3 max percZero
1       Q1 8     6 4.166667 1.1690452 2 4.00 4.5 5 5 0
2       Q2 8     5 2.600000 0.5477226 2 2.00 3.0 3 3 0
3       Q3 8     8 2.125000 0.8345230 1 1.75 2.0 3 3 0

### Numerical summary
```

Summary for whole data frame

```
XT = xtabs(~ Likert,
           data = Data)

XT

  Likert
    1 2 3 4 5 DontKnow NotApp
    2 6 6 2 3 4 1

prop.table(XT)

  Likert
    1 2 3 4 5 DontKnow NotApp
0.08333333 0.25000000 0.25000000 0.08333333 0.12500000 0.16666667 0.04166667

sum(XT)

[1] 24

### Sum of counts in table
```

```
Summarize(~ Likert.n,
  data = Data,
  digits = 2)
```

n	nvalid	mean	sd	min	Q1	median	Q3	max
24.00	19.00	2.89	1.24	1.00	2.00	3.00	3.50	5.00

Exercises G

1. Considering Maggie Simpson's data,

- a. How many Likert responses of "4" were there?
- b. What percentage of responses was that?
- c. What was the median Likert response?
- d. What was the minimum response?
- e. How would you summarize the bar plot of Maggie's responses?

2. Considering Spongebob and Patrick's workshops,

- a. How many Likert responses of "5" did Spongebob and Patrick get, respectively?
- b. What percentage of total responses for each of them were these?
- c. How would you summarize the bar plot of their responses?
- d. What were the median scores for Patrick for each of Information, Presentation, and Questions?
- e. What do the histograms for the two-way data (Spongebob and Patrick for each of Information, Presentation, and Questions) suggest to you?
- f. Are your impressions borne out by the interaction plots?
- g. Describe these results, including your opinions on the practical implications.

3. Sandy Cheeks and Squidward each delivered presentations, and were each evaluated on each of the two courses (A and B).

Speaker	Course	Likert
---------	--------	--------

Sandy	A	5
Sandy	A	5
Sandy	A	5
Sandy	A	4
Sandy	A	3
Sandy	A	3
Sandy	B	5
Sandy	B	4
Sandy	B	3
Sandy	B	3
Sandy	B	2
Sandy	B	2
Squidward	A	4
Squidward	A	4
Squidward	A	3
Squidward	A	3
Squidward	A	2
Squidward	A	2
Squidward	B	3
Squidward	B	3
Squidward	B	2
Squidward	B	2
Squidward	B	1
Squidward	B	1

- a. Summarize the data as one-way or two-way data, using at least one plot, and at least one numerical summary. Interpret your results, and describe your conclusions. Remember to use statistics that are appropriate for Likert item data. Include your opinion on the practical importance.

Descriptive Statistics with the likert Package

The *likert* package can be used to produce attractive summaries and plots of one-sample or one-way Likert data. The package is somewhat finicky with the form the data it accepts, however. Data must be in “wide” format. Long-format and wide-format data are discussed in the section “Long-format and wide-format data” in the *Types of Variables* chapter. In addition, sample sizes for each level of the factor variable must be equal, but you can use *NA* values for missing observations.

Packages used in this chapter

The packages used in this chapter include:

- psych
- likert

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
```

```
if(!require(likert)){install.packages("likert")}
```

The *likert* package

```
Input ="
  Pooh    Piglet  Tigger
  3       2        4
  5       4        4
  4       2        4
  4       2        4
  4       1        5
  4       2        3
  4       3        5
  4       2        4
  5       2        4
  5       3        3
")

Data = read.table(textConnection(Input), header=TRUE)

### Change Likert scores to factor and specify levels

Data$Pooh = factor(Data$Pooh,
                     levels = c("1", "2", "3", "4", "5"),
                     ordered = TRUE)

Data$Piglet = factor(Data$Piglet,
                      levels = c("1", "2", "3", "4", "5"),
                      ordered = TRUE)

Data$Tigger = factor(Data$Tigger,
                      levels = c("1", "2", "3", "4", "5"),
                      ordered = TRUE)

### Double check the data frame

library(psych)

headTail(Data)

str(Data)

'data.frame':   10 obs. of  3 variables:
 $ Pooh : Ord.factor w/ 5 levels "1<"2<"3<"4<...: 3 5 4 4 4 4 4 4 5 5
 $ Piglet: Ord.factor w/ 5 levels "1<"2<"3<"4<...: 2 4 2 2 1 2 3 2 2 3
 $ Tigger: Ord.factor w/ 5 levels "1<"2<"3<"4<...: 4 4 4 4 5 3 5 4 4 3

summary(Data)
```

```
### Remove unnecessary objects
rm(Input)
```

Summary statistics and plots with the *likert* package

Percent responses in each group

```
library(likert)

likert(Data)

      Item  1  2  3  4  5
1 Pooh    0  0 10 60 30
2 Piglet 10 60 20 10  0
3 Tigger  0  0 20 60 20

### Note: if there are NA's, this summary doesn't tell you!
```

Count responses with summary function in native stats package

```
summary(Data)

Pooh  Piglet Tigger
1:0   1:1    1:0
2:0   2:6    2:0
3:1   3:2    3:2
4:6   4:1    4:6
5:3   5:0    5:2
```

Note: if there are NA's, this summary will tell you

```
library(likert)

Result = likert(Data)

summary(Result)
```

	Item	low	neutral	high	mean	sd
1	Pooh	0	10	90	4.2	0.6324555
3	Tigger	0	20	80	4.0	0.6666667
2	Piglet	70	20	10	2.3	0.8232726

Note: responses are grouped into "low", "neutral", and "high"

Note: if there are NA's, this summary doesn't tell you!

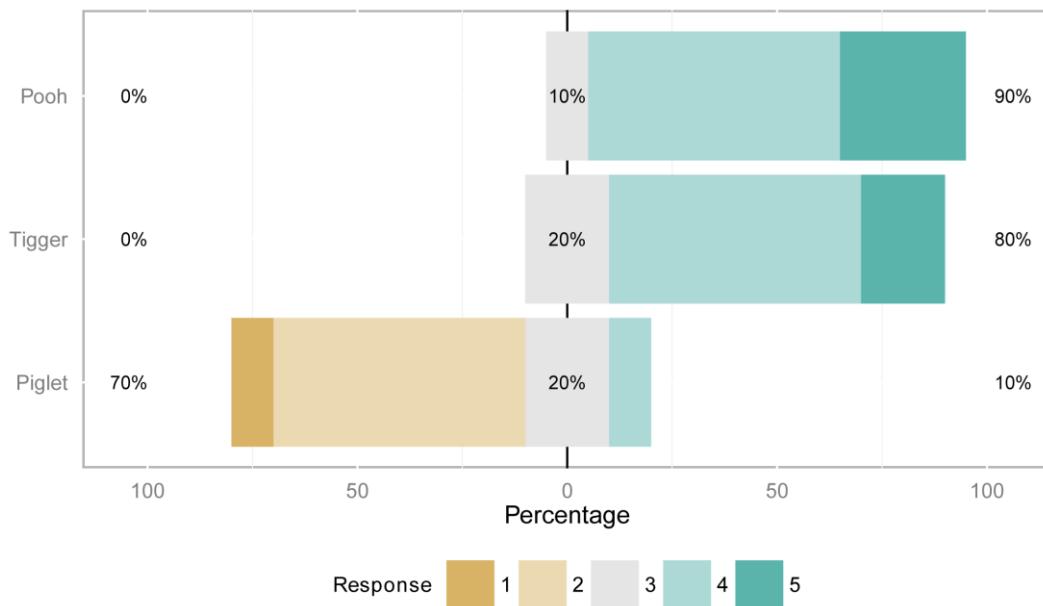
Plots that treat Likert data like factor data

Bar plot

```
library(likert)

Result = likert(Data)

plot(Result,
  type="bar")
```



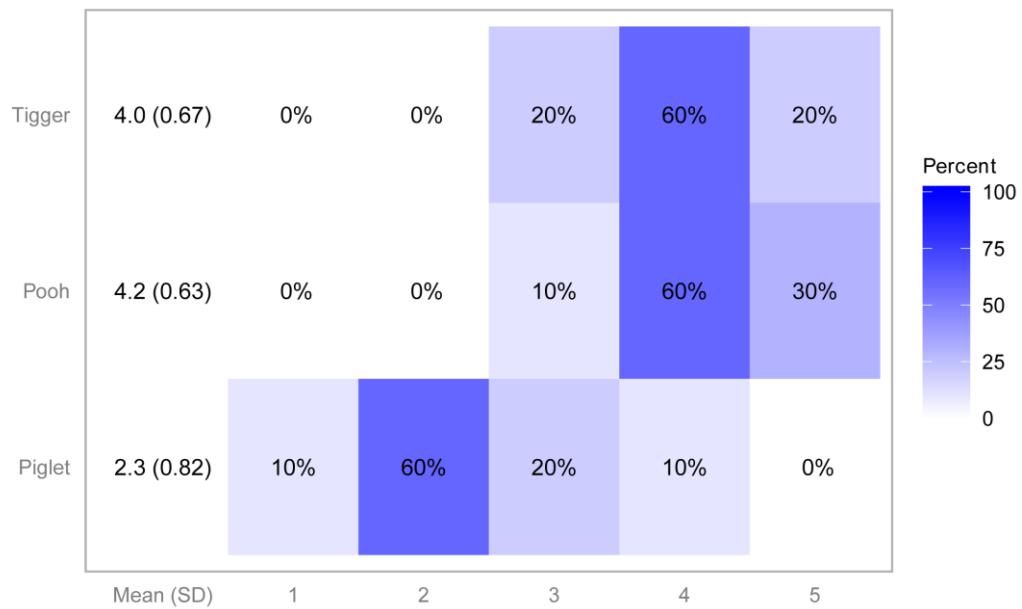
Note: for the percent numbers,
responses are grouped into "low", "neutral", and "high"

Heat plot

```
library(likert)

Result = likert(Data)

plot(Result,
  type="heat",
  low.color = "white",
  high.color = "blue",
  text.color = "black",
  text.size = 4,
  wrap = 50)
```



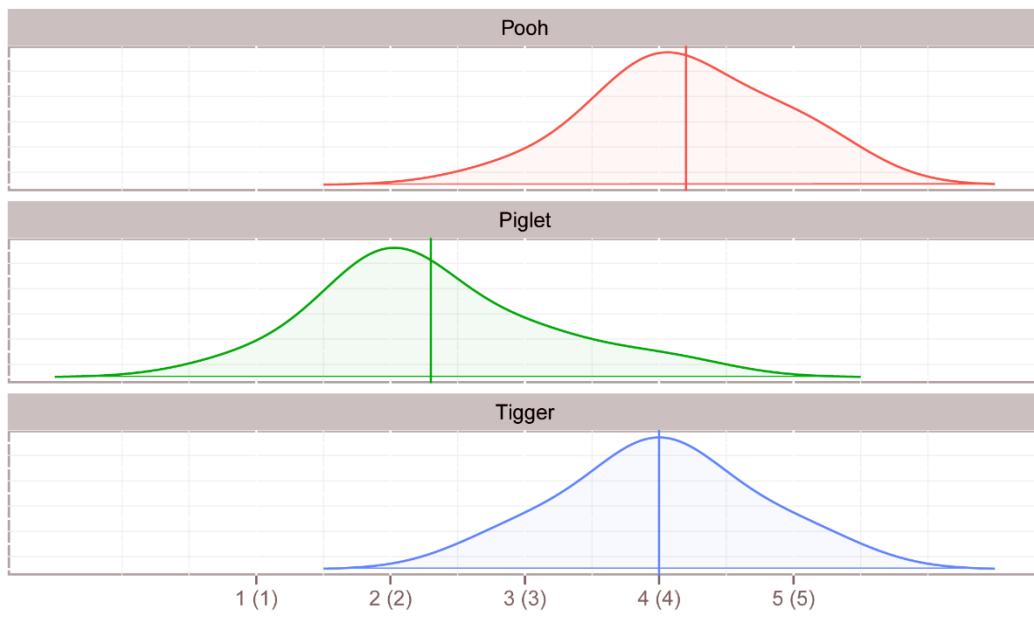
Plots that treat Likert data like numeric data

Density plot

```
library(likert)

Result = likert(Data)

plot(Result,
  type="density",
  facet = TRUE,
  bw = 0.5)
```



Note: Vertical lines are means for each group.

Note: Curves are density plots, which show the distribution of values
similar to a histogram.

References

Jason Bryer, J. and Speerschneider, K. likert: An R package analyzing and visualizing Likert items.
jason.bryer.org/likert/.

Jason Bryer, J. and Speerschneider, K. Package ‘likert’. cran.r-project.org/web/packages/likert/likert.pdf.

Confidence Intervals for Medians

Confidence intervals are discussed in the *Confidence Intervals* chapter.

The methods for determining confidence intervals for medians are distinct from the traditional method for means. There are different methods for calculating confidence intervals for the median, and there are few different methods are presented in this chapter.

When data distributions are normal or uniform in distribution, and the number of observations is large (say, $n \geq 100$), any of the methods should work reasonably well. When distributions are heavily skewed or the number of observations is relatively small (say, $n < 20$), results from some methods can differ notably from others.

For routine use, I recommend the *groupwiseMedian* function in the *rcompanion* package, with either the BCa or the percentile method. These are bootstrap methods.

The BCa (bias corrected, accelerated) is often cited as the best for theoretical reasons. The percentile method is also cited as typically good. However, if you get the “extreme order statistics used as endpoints” warning message, use a different test. For small data sets, the interval from BCa may be wider than for some other methods.

For a description of the bootstrap confidence interval methods, see Carpenter and Bithell (2000) in the “References” section below.

Technical note: Bootstrapped confidence intervals may not be reliable for discrete data, such as the ordinal Likert data used in these examples, especially for small samples. My understanding is that the determination of the confidence intervals assumes a continuous and bell-shaped distribution of the statistic (the median, in this case) from the bootstrapped samples. This concern is not considered in the examples in this book.

Other methods used in the *groupwiseMedian* function include the basic, normal, exact, and wilcox methods.

Other functions that calculate a confidence interval for a median are the *wilcox.test* function, and the *MedianCI* function in the *DescTools* package.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- boot
- DescTools
- plyr
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(boot)){install.packages("boot")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(plyr)){install.packages("plyr")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Medians, quantiles, and confidence intervals for one-sample data

For one-sample data, the median and quantiles can be calculated with the *median* function, the *summary* function, and the *Summarize* function from the *FSA* package.

The function *groupwiseMedian* in the *rcompanion* package produces medians and confidence intervals for medians. It can also calculate these statistics for grouped data (one-way or multi-way).

This example will use some theoretical data for Lisa Simpson, rated on a 10-point Likert item.

```
Input =""
Speaker      Rater  Likert
'Lisa Simpson'  1      8
'Lisa Simpson'  2      10
'Lisa Simpson'  3      9
'Lisa Simpson'  4      10
'Lisa Simpson'  5      6
'Lisa Simpson'  6      5
'Lisa Simpson'  7      3
'Lisa Simpson'  8      7
'Lisa Simpson'  9      8
'Lisa Simpson'  10     5
'Lisa Simpson'  11     10
```

```
'Lisa Simpson' 12      4
'Lisa Simpson' 13      8
'Lisa Simpson' 14      6
'Lisa Simpson' 15      9
'Lisa Simpson' 16      8
'Lisa Simpson' 17      7
'Lisa Simpson' 18      5
'Lisa Simpson' 19      8
'Lisa Simpson' 20      7
'Lisa Simpson' 21      9
'Lisa Simpson' 22      8
'Lisa Simpson' 23      7
'Lisa Simpson' 24      5
'Lisa Simpson' 25      10
'Lisa Simpson' 26      7
")
Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)
```

Produce median with median, summary, and Summarize functions

```
median(Data$Likert)

[1] 7.5

summary(Data)

      Speaker      Rater      Likert
Lisa Simpson:26  Min.   : 1.00  Min.   : 3.000
                  1st Qu.: 7.25  1st Qu.: 6.000
                  Median :13.50  Median : 7.500
                  Mean   :13.50  Mean   : 7.269
                  3rd Qu.:19.75  3rd Qu.: 8.750
                  Max.   :26.00  Max.   :10.000
```

The *Summarize* function in the *FSA* package

Note that the *1* in the formula on the right side of the tilde (~) indicates that the function should treat the data as one-sample data

```
library(FSA)

Summarize(Likert ~ 1,
  data=Data,
  digits=3)

n  nvalid  mean      sd min  Q1 median     Q3 max perczero
26      26 7.269 1.951    3   6     7.5 8.75   10          0
```

groupwiseMedian function to produce medians and confidence intervals

In the *groupwiseMedian* function, the *basic*, *normal*, *percentile*, *bca*, *wilcox*, and *exact* options determine which types of confidence intervals will be calculated. By default, only the *bca* option is set to TRUE, so the other options usually don't need to be included.

Note that results for any statistic derived from an iterative process like bootstrapping may be slightly different.

Note that the *1* in the formula on the right side of the tilde (~) indicates that the function should treat the data as one-sample data.

See *library(rcompanion); ?groupwiseMedian* for more information on this function.

If the function takes too long to complete, you can decrease the *R*= value.

```
library(rcompanion)

groupwiseMedian(Likert ~ 1,
  data      = Data,
  conf      = 0.95,
  R         = 5000,
  percentile = TRUE,
  bca       = FALSE,
  basic     = FALSE,
  normal    = FALSE,
  wilcox   = FALSE,
  digits    = 3)

.id  n Median Conf.level Percentile.lower Percentile.upper
1 <NA> 26    7.5        0.95           6.5             8
```

Medians, quantiles, and confidence intervals for grouped data

The following example revisits the Pooh, Piglet and Tigger data from the *Descriptive Statistics with the likert Package* chapter.

The *Summarize* function in the *FSA* package will produce medians and quantiles for grouped data including one-way and multi-way data.

The function *groupwiseMedian* will produce medians and confidence intervals for grouped data, including one-way and multi-way data.

```
Input ="
Speaker Likert
Pooh      3
Pooh      5
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      5
Pooh      5
Piglet    2
Piglet    4
Piglet    2
Piglet    2
Piglet    1
Piglet    2
Piglet    3
Piglet    2
Piglet    2
Piglet    3
Piglet    4
Tigger    4
Tigger    4
Tigger    4
Tigger    4
Tigger    5
Tigger    3
Tigger    5
Tigger    4
Tigger    4
Tigger    3
")"

Data = read.table(textConnection(Input), header=TRUE)
```

Summarize function in FSA package for grouped data

```
library(FSA)

Summarize(Likert ~ Speaker,
          data=Data,
          digits=3)

  Speaker n  nvalid mean      sd min Q1 median      Q3 max percZero
1  Piglet 10      10  2.3  0.823     1  2       2  2.75     4      0
```

2	Pooh	10	10	4.2	0.632	3	4	4	4.75	5	0
3	Tigger	10	10	4.0	0.667	3	4	4	4.00	5	0

groupwiseMedian function for grouped data

```
library(rcompanion)

groupwiseMedian(Likert ~ Speaker,
                 data      = Data,
                 conf      = 0.95,
                 R         = 5000,
                 percentile = TRUE,
                 bca       = FALSE,
                 digits    = 3)

  Speaker n Median Conf.level Percentile.lower Percentile.upper
1 Piglet 10     2      0.95          2.0          3.0
2 Pooh   10     4      0.95          4.0          5.0
3 Tigger 10     4      0.95          3.5          4.5
```

Optional methods and discussion on confidence intervals for medians

```
Input =""
Speaker      Rater Likert
'Lisa Simpson' 1     8
'Lisa Simpson' 2     10
'Lisa Simpson' 3     9
'Lisa Simpson' 4     10
'Lisa Simpson' 5     6
'Lisa Simpson' 6     5
'Lisa Simpson' 7     3
'Lisa Simpson' 8     7
'Lisa Simpson' 9     8
'Lisa Simpson' 10    5
'Lisa Simpson' 11    10
'Lisa Simpson' 12    4
'Lisa Simpson' 13    8
'Lisa Simpson' 14    6
'Lisa Simpson' 15    9
'Lisa Simpson' 16    8
'Lisa Simpson' 17    7
'Lisa Simpson' 18    5
'Lisa Simpson' 19    8
'Lisa Simpson' 20    7
'Lisa Simpson' 21    9
'Lisa Simpson' 22    8
'Lisa Simpson' 23    7
'Lisa Simpson' 24    5
'Lisa Simpson' 25    10
'Lisa Simpson' 26    7
")
```

```
Data = read.table(textConnection(Input), header=TRUE)
```

Optional: Confidence interval for medians with the wilcox.test function

For one-sample data, the *wilcox.test* function will produce a confidence interval for the median. It will also produce a “(pseudo)median”. For details on the calculations of these statistics, see *?wilcox.test*.

Note that the *conf.int=TRUE* option must be used to produce the confidence interval, and the *conf.level=0.95* option indicates that a 95% confidence interval should be calculated.

```
wilcox.test(Data$Likert,
            alternative="two.sided",
            correct=TRUE,
            conf.int=TRUE,
            conf.level=0.95)
```

```
95 percent confidence interval:
 6.499961 8.000009
```

Optional: Median and confidence interval with the DescTools package

With the *MedianCI* function in the *DescTools* package, the *method=exact* option uses the confidence interval from the *SignTest* function in *DescTools*. The *method=boot* option uses the *basic* method from the *boot* package.

```
library(DescTools)

MedianCI(Data$Likert,
          conf.level = 0.95,
          na.rm = FALSE,
          method = "exact",
          R = 10000)

median lwr.ci upr.ci
 7.5    6.0    8.0
```

Optional: Confidence interval for median by bootstrap

Bootstrapping is a method by which a statistic is calculated by repeated sampling the given data to better estimate the distribution of values in the population from which the sample was taken.

The *boot* package in R can derive various statistics with a bootstrap process. Note that the grammar of the function is somewhat complicated, but here *Data\$Likert* is the variable we wish to get the statistics for, and *R=10000* indicates the number of bootstrap replicates to use. *Mboot* here is defined as the result of the bootstrap. The *boot.ci* function produces four types of confidence intervals for the median. Note also that the displaying the result *Mboot* gives a standard error for the estimated median. See *?boot.ci* for more details on these methods.

If the function takes too long to complete, you can decrease the *R=* value.

Note that results for any statistic derived from an iterative process like bootstrapping may be slightly different if the process is re-run.

```
library(boot)

Mboot = boot(Data$Likert,
              function(x,i) median(x[i]),
              R=10000)

boot.ci(Mboot,
        conf = 0.95,
        type = c("norm", "basic" , "perc", "bca")
      )

Intervals :
Level      Normal          Basic
95%  ( 6.510,  8.535 )  ( 7.000,  8.500 )

Level      Percentile       BCa
95%  ( 6.5,   8.0 )  ( 6.0,   8.0 )

### Other information

Mboot

hist(Mboot$t[,1],
  col = "darkgray")
```

References

Carpenter, J. and J. Bithel. 2000. "Bootstrap confidence intervals: when, which, what? A practical guide for medical statisticians". *Statistics in Medicine* 19:1141–1164.
www.tau.ac.il/~saharon/Boot/10.1.1.133.8405.pdf.

Exercises H

1. Considering Lisa Simpson's data,
 - a. What was her median Likert score?
 - b. Minimum? Maximum? 25th percentile? 75th percentile?
2. For Lisa Simpson's data, what is the 95% confidence interval for the median using the following methods?
 - a. wilcox
 - b. normal

- c. basic
- d. percentile
- e. BCa

3. Considering the Pooh, Piglet, and Tigger data,

- a. For Piglet, what was his median score and 95% confidence interval using the percentile method?

4. Bart Simpson and Milhouse Van Houten were each evaluated on a Likert scale. Determine the median score for each, and determine the 95% confidence interval for these median scores.

- Be sure to state which method for the confidence interval you used.
- How can you interpret the results? Include the practical interpretation. For example, are the differences in scores large enough to be meaningful?

Speaker	Likert
Bart	7
Bart	7
Bart	8
Bart	9
Bart	5
Bart	6
Bart	6
Bart	7
Bart	8
Bart	7
Bart	7
Bart	7
Milhouse	8
Milhouse	8
Milhouse	7
Milhouse	7
Milhouse	9
Milhouse	9
Milhouse	10
Milhouse	6
Milhouse	7
Milhouse	8
Milhouse	7
Milhouse	8

Converting Numeric Data to Categories

There are occasions when it is useful to categorize Likert scores, Likert scales, or continuous data into groups or categories. In general, there are no universal rules for converting numeric data to categories. A few methods are presented here.

Categorizing data by a range of values

One approach is to create categories according to logical cut-off values in the scores or measured values. An example of this is the common grading system in the U.S. in which a 90% grade or better is an "A", 80–89% is "B", etc. It is common in this approach to make the categories with equal spread in values. For example, there is a 10 point spread in a "B" grade and a 10 point spread in a "C" grade. But this equality is not required. Below in the "Categorize data by range of values" example, 5-point Likert data are converted into categories with 4 and 5 being "High", 3 being "Medium", and 1 and 2 being "Low".

This approach relies on the chosen cut-off points being meaningful. For example, for a grade of 70–79 to be considered "sufficient", the evaluation instruments (e.g. the tests, quizzes, and assignments) need to be calibrated so that a grade of 75 really is "sufficient", and not "excellent" or "good", etc. You can imagine a case where a 4 or 5 on a 5-point Likert item is considered "high", but if all respondents scored 4 or 5 on the item, it might not be clear that these values are "high", but may just be the typical response. Likewise, in this case, the decision to group 4 and 5 as "high" needs be compared with a decision to group 4 with 2 and 3 as "medium". This breakdown may be closer to how people interpret a 5-point Likert scale. Either grouping could be meaningful depending on the purpose of the categorization and the interpretation of each level in the Likert item.

Categorizing data by percentiles

A second approach is to use percentiles to categorize data. Remember, the value for the 90th percentile is the score or measurement below which 90% of respondents scored.

The advantage to this approach is that it does not rely on the scoring system being meaningful in its absolute values. That is, students scoring above the 90th percentile are scoring higher than 90% of students. It doesn't matter if the score for the 90th percentile is 90 out of 100 or 50 out of 100. If the groups use equally-spaced breakpoints, for example 0th, 25th, 50th, 75th, and 100th percentiles, there should be approximately an equal number of respondents in each category.

Categorizing data with clustering

A third approach is to use a clustering algorithm to divide data into groups with similar measurements. This is useful when there are multiple measurements for an individual. For example, if students receive scores on reading, math, and analytical thinking, an algorithm can determine if, for example, there is a group of students who do well on all three measurements, and a group of students who do well in math and analysis but who do poorly on reading. Most clustering algorithms assume that measurement data is continuous in nature. Below, the *partitioning around medoids* (PAM) method is used with the *manhattan* metric. This may be relatively more suitable for ordinal data than some other methods

Packages used in this chapter

The packages used in this chapter include:

- psych
- cluster
- fpc

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(cluster)){install.packages("cluster")}
if(!require(fpc)){install.packages("fpc")}
```

Examples for converting numeric data to categories

```
Input ="
Instructor Student Likert
Homer      a        3
Homer      b        4
Homer      c        4
Homer      d        4
Homer      e        4
Homer      f        5
Homer      g        5
Homer      h        5
Homer      i        3
Homer      j        2
Homer      k        3
Homer      l        4
Homer      m        5
Homer      n        5
Homer      o        5
Homer      p        4
Homer      q        4
Homer      r        3
Homer      s        2
Homer      t        5
Homer      u        3
")
Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
```

```
rm(Input)
```

Categorize data by range of values

The following example will categorize responses on a single 5-point Likert item. The algorithm will be a score of 1 or 2 will be called “low”, a score of 3 “medium”, and a score of 4 or 5 “high”.

Categorize data

```
Data$Category[Data$Likert == 1 | Data$Likert == 2] = "Low"
Data$Category[Data$Likert == 3] = "Medium"
Data$Category[Data$Likert == 4 | Data$Likert == 5] = "High"
```

Data

Order factor levels to make output easier to read

```
Data$Category = factor(Data$Category,
                      levels=c("Low", "Medium", "High"))
```

Summarize counts of categories

```
XT = xtabs(~ Category + Instructor,
           data = Data)
```

XT

Instructor	
Category	Homer
Low	2
Medium	5
High	14

Report students in each category

```
Data$Student[Data$Category == "Low"]
```

[1] j s

```
Data$Student[Data$Category == "Medium"]
```

[1] a i k r u

```
Data$Student[Data$Category == "High"]
```

[1] b c d e f g h l m n o p q t

Alternate method with tapply

```
tapply(x      = Data$Student,
       INDEX = Data$Category,
       FUN    = print)

$Low
[1] j s

$Medium
[1] a i k r u

$High
[1] b c d e f g h l m n o p q t
```

Summary table

<u>Category</u>	<u>Range</u>	<u>Count</u>	<u>Students</u>
Low	1 or 2	2	j, s
Medium	3	5	a, i, k, r, u
High	4 or 5	14	b, c, d, e, f, g, h, l, m, n, o, p, q, t

Categorize data by percentile

The following example will categorize responses on a single 5-point Likert item. Respondents scoring below the 33rd percentile will be labeled “Lower third”; those between the 33rd and 67th percentile “Middle third”; and those above the 67th percentile “Upper third”.

Categorize data

```
Percentile_00  = min(Data$Likert)
Percentile_33  = quantile(Data$Likert, 0.33333)
Percentile_67  = quantile(Data$Likert, 0.66667)
Percentile_100 = max(Data$Likert)

RB = rbind(Percentile_00, Percentile_33, Percentile_67, Percentile_100)

dimnames(RB)[[2]] = "value"

RB

          value
Percentile_00 2.0000
Percentile_33 3.6666
Percentile_67 4.3334
Percentile_100 5.0000
```

```
Data$Group[Data$Likert >= Percentile_00 & Data$Likert < Percentile_33] = "Lower_third"
Data$Group[Data$Likert >= Percentile_33 & Data$Likert < Percentile_67] = "Middle_third"
Data$Group[Data$Likert >= Percentile_67 & Data$Likert <= Percentile_100] = "Upper_third"
```

DataOrder factor levels to make output easier to read

```
Data$Group = factor(Data$Group,
  levels=c("Lower_third", "Middle_third", "Upper_third"))
```

Summarize counts of groups

```
XT = xtabs(~ Group + Instructor,
  data = Data)
```

XT

Group	Instructor	
	Homer	
Lower_third	7	
Middle_third	7	
Upper_third	7	

Report students in each group

```
tapply(x      = Data$Student,
  INDEX = Data$Group,
  FUN   = print)
```

\$Lower_third
[1] a i j k r s u

\$Middle_third
[1] b c d e l p q

\$Upper_third
[1] f g h m n o t

Summary table

Group	Range	Count	Students
Lower third	1, 2, or 3	7	a, i, j, k, r, s, u
Middle third	4	7	b, c, d, e, l, p, q
Upper third	5	7	f, g, h, m, n, o, t

Categorize data by clustering

In the following example, each student has a score for *Happy* and for *Tired*. The students will be divided into clusters based on the similarities of their scores across both measures.

```
Input ="  
Instructor Student Happy Tired
```

```
Marge      a      5      5
Marge      b      5      5
Marge      c      2      5
Marge      d      2      5
Marge      e      5      5
Marge      f      5      5
Marge      g      1      5
Marge      h      1      5
Marge      i      1      5
Marge      j      5      5
Marge      k      3      3
Marge      l      3      3
Marge      m      3      3
Marge      n      5      2
Marge      o      5      2
Marge      p      5      1
Marge      q      5      1
Marge      r      5      1
Marge      s      4      1
Marge      t      4      1
")
```

```
Data = read.table(textConnection(Input), header=TRUE)

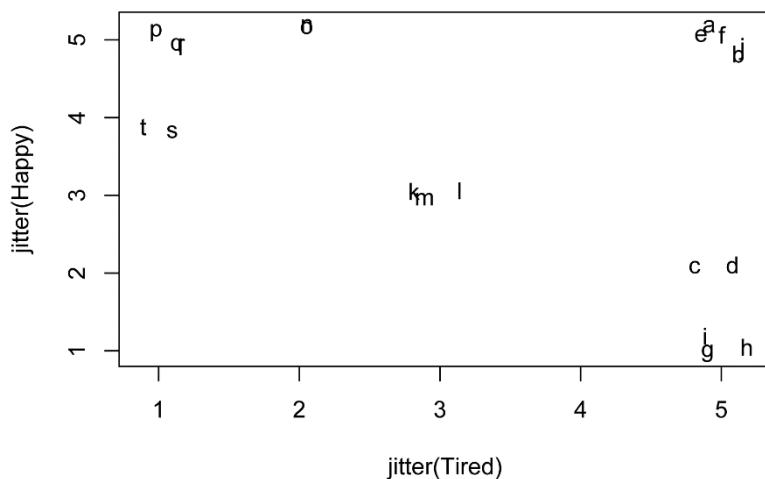
### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)
```

Plot data

In the following plot, each letter represents a *Student* from the data frame. To me, this plot suggests that the data could be reasonably clustered into 4 or perhaps 7 or 8 clusters.

```
plot(jitter(Happy) ~ jitter(Tired),
  data = Data,
  pch=as.character(Data$Student))
```

Use only numeric data

```
Data.num = Data[c("Happy", "Tired")]
```

Determine the optimal number of clusters

The *pamk* function in the *fpc* package can determine the optimum number of clusters for the *partitioning around medoids* (PAM) method. It will also complete the PAM analysis, but we'll do that separately below.

Practical considerations may override the results of the *pamk* function. In the example below, if we include 1 in the possible range of cluster numbers, the function will determine that 1 is the optimum number. Also, if we extend the range to, say, 10, the function will choose 7 as the optimum number, but this may be too many for our purposes.

```
library(fpc)

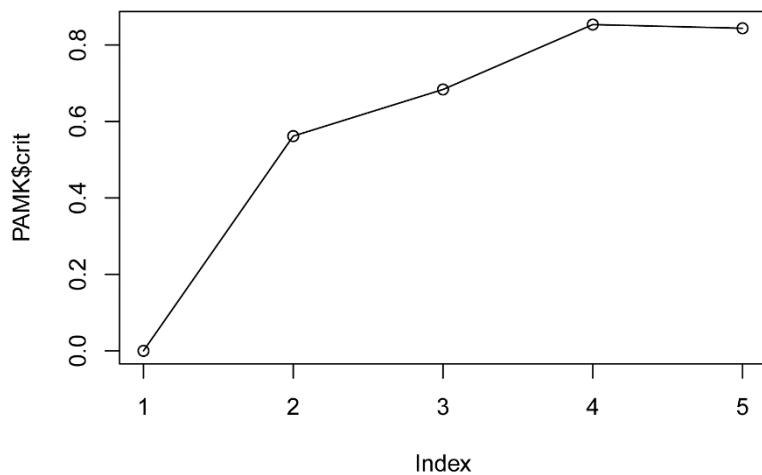
PAMK = pamk(Data.num,
             krange = 2:5,
             metric="manhattan")

PAMK$nc

[1] 4

### This is the optimum number of clusters in the range

plot(PAMK$crit)
lines(PAMK$crit)
```



For the range of cluster numbers shown on the x-axis, the crit value is maximized at 4, suggesting 4 is optimum number of clusters for this range.

Categorize data

We will use the *pam* function in the *cluster* package to divide our data into four clusters.

```
library(cluster)

PAM = pam(x = Data.num,
           k = 4,           ##### Number of clusters to find
           metric="manhattan")

PAM

Medoids:
      ID Happy Tired
[1,] 10     5     5
[2,]  9     1     5
[3,] 13     3     3
[4,] 16     5     1

Clustering vector:
[1] 1 1 2 2 1 1 2 2 1 3 3 3 4 4 4 4 4 4 4

##### Add clusters to data frame

PAMclust = rep("NA", length(Data$Likert))

PAMclust[PAM$clustering == 1] = "Cluster 1"
PAMclust[PAM$clustering == 2] = "Cluster 2"
PAMclust[PAM$clustering == 3] = "Cluster 3"
PAMclust[PAM$clustering == 4] = "Cluster 4"
```

```
Data$Cluster = PAMclust
```

DataOrder factor levels to make output easier to read

```
Data$Cluster = factor(Data$Cluster,
                      levels=c("Cluster 1", "Cluster 2",
                               "Cluster 3", "Cluster 4"))
```

Summarize counts of groups

```
XT = xtabs(~ Cluster + Instructor,
           data = Data)
```

XT

Cluster	Instructor	
	Marge	
Cluster 1	5	
Cluster 2	5	
Cluster 3	3	
Cluster 4	7	

Report students in each group

```
tapply(x      = Data$Student,
       INDEX = Data$Cluster,
       FUN   = print)
```

```
$`Cluster 1`
[1] a b e f j
```

```
$`Cluster 2`
[1] c d g h i
```

```
$`Cluster 3`
[1] k l m
```

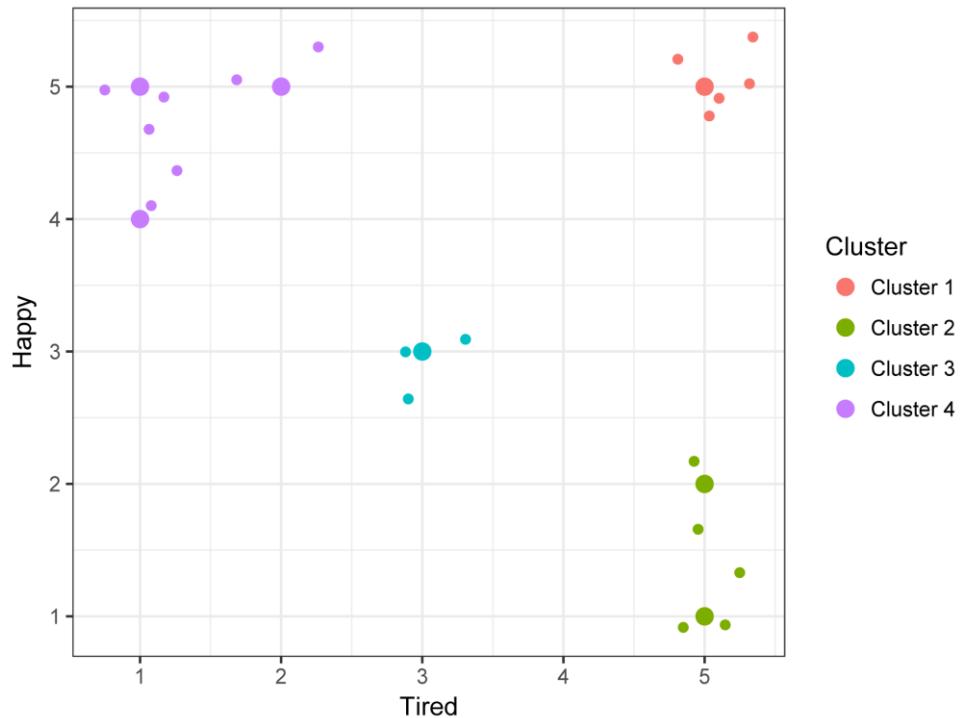
```
$`Cluster 4`
[1] n o p q r s t
```

Summary table

<u>Cluster</u>	<u>Interpretation</u>	<u>Count</u>	<u>Students</u>
Cluster 1	Tired and happy	5	a, b, e, f, j
Cluster 2	Tired and not happy	5	c, d, d, h, i
Cluster 3	Middle of the road	3	k, l, m
Cluster 4	Not tired and happy	7	n, o, p, q, r, s, t

Final plot

```
ggplot(Data,
       aes(x      = Tired,
           y      = Happy,
           color = Cluster)) +
  geom_point(size=3) +
  geom_jitter(width = 0.4, height = 0.4) +
  theme_bw()
```



Traditional Nonparametric Tests

Introduction to Traditional Nonparametric Tests

Packages used in this chapter

The packages used in this chapter include:

- effsize

The following commands will install these packages if they are not already installed:

```
if(!require(effsize)){install.packages("effsize")}
```

Introduction

The traditional nonparametric tests presented in this book are primarily rank-based tests. Instead of using the numeric values of the dependent variable, the dependent variable is converted into relative ranks.

For example, imagine we have the heights of eight students in centimeters.

```
Height = c(110, 132, 137, 139, 140, 142, 142, 145)
```

```
names(Height) = letters[1:8]
```

```
Height
```

a	b	c	d	e	f	g	h
110	132	137	139	140	142	142	145

```
rank(Height)
```

a	b	c	d	e	f	g	h
1.0	2.0	3.0	4.0	5.0	6.5	6.5	8.0

a has the smallest height and so is ranked 1. *b* has the next smallest height and so is ranked 2. And so on. Note that *f* and *g* are tied for spots 6 and 7, and so share a rank of 6.5.

Also note that the value of *a* is quite a bit smaller than the others, but its rank is simply 1. Information about the absolute height values is lost, and only the relative ranking is retained in the ranks. That is, if the value of *a* were changed to 100 or 5 or -10, its rank would remain 1 in this data set.

The advantage of using these rank-based tests is that they don't make many assumptions about the distribution of the data. Instead, their conclusions are based on the relative ranks of values in the groups being tested.

Advantages of nonparametric tests

- Most of the traditional nonparametric tests presented in this section are relatively common, and your audience is relatively likely to be familiar with them.
- They are appropriate for interval/ratio or ordinal dependent variables.
- Their nonparametric nature makes them appropriate for data that don't meet the assumptions of parametric analyses. These include data that are skewed, non-normal, contain outliers, or possibly are censored. (Censored data is data where there is an upper or lower limit to values. For example, if ages under 5 are reported as "under 5".)

Disadvantages of nonparametric tests

- These tests are typically named after their authors, with names like Mann–Whitney, Kruskal–Wallis, and Wilcoxon signed-rank. It may be difficult to remember these names, or to remember which test is used in which situation.
- Most of the traditional nonparametric tests presented here are limited by the types of experimental designs they can address. They are typically limited to a one-way comparison of independent groups (e.g. Kruskal–Wallis), or to unreplicated complete block design for paired samples (e.g. Friedman). The aligned ranks transformation approach, however, allows for more complicated designs.
- Readers are likely to find a lot of contradictory information in different sources about the hypotheses and assumptions of these tests. In particular, authors will often treat the hypotheses of some tests as corresponding to tests of medians, and then list the assumptions of the test as corresponding to these hypotheses. However, if this is not explicitly explained, the result is that different sources list different assumptions that data must meet in order for the test to be valid. This creates unnecessary confusion in the mind of students trying to correctly employ these tests.

Interpretation of nonparametric tests

In general, these tests determine if there is a *systematic* difference among groups. This may be due to a difference in location (e.g. median) or in the shape or spread of the distribution of the data. With the Mann–Whitney and Kruskal–Wallis tests, the difference among groups that is of interest is the probability of an observation from one group being larger than an observation from another group. If this probability is 0.50, this is termed "stochastic equality", and when this probability is far from 0.50, it is sometimes called "stochastic dominance".

Optional technical note: Without additional assumptions about the distribution of the data, the Mann–Whitney and Kruskal–Wallis tests do not test hypotheses about the group medians. Mangiafico (2015) and McDonald (2014) in the “References” section provide an example of a significant Kruskal–Wallis test where the groups have identical medians, but differ in their stochastic dominance.

Effect size statistics

Effect size statistics for traditional nonparametric tests include Cliff’s *delta* and Vargha and Delaney’s *A* for Mann–Whitney, and *epsilon*-squared and Freeman’s “coefficient of determination” (Freeman’s *theta*) (Freeman, 1965) for Kruskal–Wallis. There is also an *r* statistic for Mann–Whitney and the paired signed-rank test. Kendall’s *W* can be used for Friedman’s test.

A couple of accessible resources on effect sizes for these tests are Tomczak and Tomczak (2014) and King and Rosopa (2010).

Some effect size statistics included here determine the degree to which one group has data with higher ranks than other groups. They tend to vary from 0 (groups have data that are stochastically equal) to 1 (one group stochastically dominates). They are related to the probability that a value from one group will be greater than a value from another group.

As rank-based measures, these effect size statistics do not indicate the difference in absolute values between groups. That is, if you were to replace the 5’s in the second example below with 100’s, the value of the effect size statistics would not change, because in either case the 5’s or 100’s are the highest-ranked numbers. For a practical interpretation of results, it is usually important to consider the absolute values of data such as with descriptive statistics.

```
library(effsize)

A = c(1,1,1, 2,2,2, 3,3,3, 4,4,4)
B = c(1,1,1, 2,2,2, 3,3,3, 4,4,4)

cliff.delta(B, A)

Cliff's Delta

delta estimate: 0

### This corresponds to a VDA of 0.5,
### the probability of an observation in B being larger than
### an observation in A.

A = c(1,1,1, 2,2,2, 3,3,3, 4,4,4)
B = c(2,2,2, 3,3,3, 4,4,4, 5,5,5)

cliff.delta(B, A)

Cliff's Delta
```

delta estimate: 0.4375

```
### This corresponds to a VDA of 0.719,
### the probability of an observation in B being larger than
### an observation in A.
```

```
A = c(1,1,1, 2,2,2, 3,3,3, 4,4,4)
B = c(3,3,3, 4,4,4, 5,5,5, 6,6,6)
```

```
cliff.delta(B, A)
```

Cliff's Delta

delta estimate: 0.75

```
### This corresponds to a VDA of 0.875,
### the probability of an observation in B being larger than
### an observation in A.
```

```
A = c(1,1,1, 2,2,2, 3,3,3, 4,4,4)
B = c(5,5,5, 6,6,6, 7,7,7, 8,8,8)
```

```
cliff.delta(B, A)
```

Cliff's Delta

delta estimate: 1

```
### This corresponds to a VDA of 1,
### the probability of an observation in B being larger than
### an observation in A.
```

Optional: Appropriate use of traditional nonparametric tests

Using traditional nonparametric tests with ordinal data

Some authors caution against using traditional nonparametric tests with ordinal dependent variables, since many of them were developed for use with continuous (interval/ratio) data. Some authors have further concerns about situations where there are likely to be many ties in ranks, such as Likert data.

Other authors argue that, since these tests rank-transform data before analysis and have adjustments for tied ranks, that they are appropriate for ordinal data.

Simulations comparing traditionally nonparametric tests to ordinal regression are presented in the “Optional: Simulated comparisons of traditional nonparametric tests and ordinal regression” in the *Introduction to Likert Data* chapter.

Using traditional nonparametric tests with interval/ratio data

These nonparametric tests are commonly used for interval/ratio data when the data fail to meet the assumptions of parametric analysis.

Some authors discourage using common nonparametric tests for interval/ratio data in some circumstances.

- One issue is the interpretation of the results mentioned above. That is, often results are incorrectly interpreted as a difference in medians when they are really describing a stochastic difference in distributions.
- Another problem is the lack of flexibility in designs these test can handle.
- Finally, these tests may lack power relative to their parametric equivalents.

Given these considerations and the fact that that parametric statistics are often relatively robust to minor deviations in their assumptions, some authors argue that it is often better to stick with parametric analyses for interval/ratio data if it's possible to make them work.

References

Freeman, L.C. 1965. *Elementary Applied Statistics: For Students in Behavioral Science*. John Wiley & Sons. New York.

King, B.M., P.J. Rosopa, and E.W. Minium. 2018. Some (Almost) Assumption-Free Tests. In *Statistical Reasoning in the Behavioral Sciences*, 7th ed. Wiley.

"Kruskal-Wallis Test" in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_06.html.

"Kruskal-Wallis Test" in McDonald, J.H. 2014. *Handbook of Biological Statistics*.
www.biostathandbook.com/kruskalwallis.html.

Tomczak, M. and Tomczak, E. 2014. The need to report effect size estimates revisited. An overview of some recommended measures of effect size. Trends in Sports Sciences 1(21):1–25. www.tss.awf.poznan.pl/files/3_Trends_Vol21_2014_no1_20.pdf.

One-sample Wilcoxon Signed-rank Test

One-sample tests are useful to compare a set of values to a given default value. For example, one might ask if a set of five-point Likert scores are significantly different from a "default" or "neutral" score of 3. Another use might be to compare a current set of values to a previously published value.

The one-sample Wilcoxon test is a rank-based test that begins with calculating the difference between the observed values and the default value. Because of this subtraction operation in the calculations,

the data are assumed to be interval. That is, with Likert item data with this test, the data are assumed to be numeric. For purely ordinal data, the one-sample sign test could be used instead.

The null hypothesis for the test is that the data are symmetric about the default value (except that the test is done on ranks after the distances of observations from default value are determined).

A significant result suggests either that the (ranked) data are symmetric about another value or are sufficiently skewed in one direction. In either case, this suggests that the location of the data is different from the chosen default value.

Without further assumptions about the distribution of the data, the test is not a test of the median.

Appropriate data

- One-sample data
- Data are interval or ratio

Hypotheses

- Null hypothesis (simplified): The population from which the data are sampled is symmetric about the default value.
- Alternative hypothesis (simplified, two-sided): The population from which the data are sampled is not symmetric about the default value.

Interpretation

Reporting significant results as e.g. “Likert scores were significantly different from a neutral value of 3” is acceptable.

Notes on name of test

The names used for the one-sample Wilcoxon signed-rank test and similar tests can be confusing. “Sign test” may be used, although properly the sign test is a different test. Both “signed-rank test” and “sign test” are sometimes used to refer to either one-sample or two-sample tests.

The best advice is to use a name specific to the test being used.

Other notes and alternative tests

Some authors recommend this test only in cases where the data are symmetric. It is my understanding that this requirement is only for the test to be considered a test of the median.

For ordinal data or for a test specifically about the median, the sign test can be used.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- rcompanion
- coin

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(coin)){install.packages("coin")}
```

One-sample Wilcoxon signed-rank test example

This example will re-visit the Maggie Simpson data from the *Descriptive Statistics for Likert Data* chapter.

The example answers the question, “Are Maggie’s scores significantly different from a ‘neutral’ score of 3?”

The test will be conducted with the *wilcox.test* function, which produces a *p*-value for the hypothesis, as well a pseudo-median and confidence interval.

```
Input ="
  Speaker      Rater Likert
  'Maggie Simpson'  1      3
  'Maggie Simpson'  2      4
  'Maggie Simpson'  3      5
  'Maggie Simpson'  4      4
  'Maggie Simpson'  5      4
  'Maggie Simpson'  6      4
  'Maggie Simpson'  7      4
  'Maggie Simpson'  8      3
  'Maggie Simpson'  9      2
  'Maggie Simpson' 10      5
  ")
Data = read.table(textConnection(Input),header=TRUE)

### Create a new variable which is the Likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                      ordered = TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)
```

```
### Remove unnecessary objects
rm(Input)
```

Summarize data treating Likert scores as factors

Note that the variable we want to count is *Likert.f*, which is a factor variable. Counts for *Likert.f* are cross tabulated over values of *Speaker*. The *prop.table* function translates a table into proportions. The *margin=1* option indicates that the proportions are calculated for each row.

```
xtabs(~ Speaker + Likert.f,
      data = Data)

          Likert.f
Speaker      2 3 4 5
Maggie Simpson 1 2 5 2

XT = xtabs(~ Speaker + Likert.f,
           data = Data)

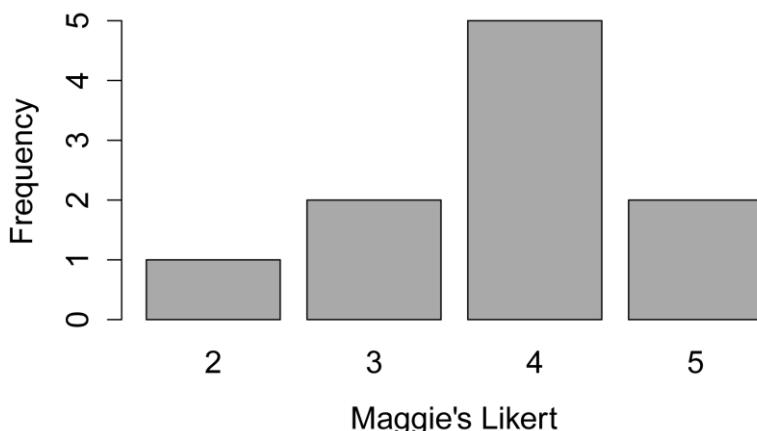
prop.table(XT,
           margin = 1)

          Likert.f
Speaker      2   3   4   5
Maggie Simpson 0.1 0.2 0.5 0.2
```

Bar plot

```
XT = xtabs(~ Likert.f,
           data=Data)

barplot(XT,
        col="dark gray",
        xlab="Maggie's Likert",
        ylab="Frequency")
```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Speaker,
          data=Data,
          digits=3)

  Speaker   n  mean    sd  min   Q1 median   Q3 max percZero
1 Maggie Simpson 10  3.8 0.919   2 3.25      4 4     5       0
```

One-sample Wilcoxon signed-rank test

In the `wilcox.test` function, the `mu` option indicates the value of the default value to compare to. In this example `Data$Likert` is the one-sample set of values on which to conduct the test. For the meaning of other options, see `?wilcox.test`.

```
wilcox.test(Data$Likert,
            mu=3,
            conf.int=TRUE,
            conf.level=0.95)

wilcoxon signed rank test with continuity correction

v = 32.5, p-value = 0.04007

alternative hypothesis: true location is not equal to 3

### Note p-value in the output above

### You will get the "cannot compute exact p-value with ties" error
### You can ignore this, or use the exact=FALSE option.

95 percent confidence interval:
3.000044 4.500083

sample estimates:

(pseudo)median
4.000032

### Note that the output will also produce a pseudo-median value
### and a confidence interval if the conf.int=TRUE option is used.
```

Effect size

I am not aware of any established effect size statistic for the one-sample Wilcoxon signed-rank test. However, using a statistic analogous to the r used in the Mann–Whitney test may make sense.

The following interpretation is based on my personal intuition. It is not intended to be universal.

	<u>Small</u>	<u>medium</u>	<u>large</u>
<i>r</i>	0.10 – < 0.40	0.40 – < 0.60	≥ 0.60

```
library(rcompanion)
wilcoxonOneSampleR(Data$Likert,
mu=3)
```

r
0.681

Exercises I

1. Considering Maggie Simpson's data,

- a. What was her median score?
- b. What were the first and third quartiles for her scores?
- c. According to the one-sample Wilcoxon signed-rank test, are her scores significantly different from a neutral score of 3?
- d. Is the confidence interval output from the test useful in answering the previous question?
- e. Overall, how would you summarize her results? Be sure to address the practical implication of her scores compared with a neutral score of 3.
- f. Do these results reflect what you would expect from looking at the bar plot?

2. Brian Griffin wants to assess the education level of students in his course on creative writing for adults. He wants to know the median education level of his class, and if the education level of his class is different from the typical Bachelor's level.

Brian used the following table to code his data.

Code	Abbreviation	Level
1	< HS	Less than high school
2	HS	High school
3	BA	Bachelor's
4	MA	Master's
5	PhD	Doctorate

The following are his course data.

Instructor	Student	Education
'Brian Griffin'	a	3
'Brian Griffin'	b	2
'Brian Griffin'	c	3
'Brian Griffin'	d	3
'Brian Griffin'	e	3
'Brian Griffin'	f	3
'Brian Griffin'	g	4
'Brian Griffin'	h	5
'Brian Griffin'	i	3
'Brian Griffin'	j	4
'Brian Griffin'	k	3
'Brian Griffin'	l	2

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- a. What was the median education level? (Be sure to report the education level, not just the numeric code!)
- b. What were the first and third quartiles for education level?
- b. According to the one-sample Wilcoxon signed-rank test, are the education levels significantly different from a typical level of Bachelor's?
- e. Is the confidence interval output from the test useful in answering the previous question?
- f. Overall, how would you summarize the results? Be sure to address the practical implications.
- g. Plot Brian's data in a way that helps you visualize the data.
- h. Do the results reflect what you would expect from looking at the plot?

Sign Test for One-sample Data

The one-sample sign test compares the number of observations greater than or less than the default value without accounting for the magnitude of the difference between each observation and the default value. The test is similar in purpose to the one-sample Wilcoxon signed-rank test, but looks specifically at the median value, and is not affected by the distribution of the data.

The test is conducted with the *SIGN.test* function in the *BSDA* package or the *SignTest* function in the *DescTools* package. These functions produce a *p*-value for the hypothesis, as well as the median and confidence interval of the median for the data.

Appropriate data

- One-sample data
- Data are ordinal, interval, or ratio

Hypotheses

- Null hypothesis: The median of the population from which the sample was drawn is equal to the default value.
- Alternative hypothesis (two-sided): The median of the population from which the sample was drawn is not equal to the default value.

Interpretation

Reporting significant results as e.g. "Likert scores were significantly different from a default value of 3" is acceptable. As is e.g. "Median Likert scores were significantly different from a default value of 3"

Packages used in this chapter

The packages used in this chapter include:

- BSDA
- DescTools

The following commands will install these packages if they are not already installed:

```
if(!require(BSDA)){install.packages("BSDA")}
if(!require(DescTools)){install.packages("DescTools")}
```

One-sample sign test example

For appropriate plots and summary statistics, see the *One-sample Wilcoxon Signed-rank Test* chapter.

```
Input ="
  Speaker      Rater    Likert
 'Maggie Simpson'  1        3
 'Maggie Simpson'  2        4
 'Maggie Simpson'  3        5
 'Maggie Simpson'  4        4
 'Maggie Simpson'  5        4
 'Maggie Simpson'  6        4
 'Maggie Simpson'  7        4
 'Maggie Simpson'  8        3
 'Maggie Simpson'  9        2
 'Maggie Simpson' 10        5
")

Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame

library(psych)

headTail(Data)
```

```

str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

```

Sign test with the *BSDA* package

Note that *Data\$Likert* is the one-sample data, and *md=3* indicates the default value to compare to.

```

library(BSDA)

SIGN.test(Data$Likert,
          md = 3)

One-sample Sign-Test

s = 7, p-value = 0.07031

alternative hypothesis: true median is not equal to 3

### Note the p-value in the output above

95 percent confidence interval:
 3.000000 4.675556

sample estimates:
median of x
        4

### Median value and confidence interval

```

Sign test with the *DescTools* package

Note that *Data\$Likert* is the one-sample data, and *mu=3* indicates the default value to compare to.

```

library(DescTools)

SignTest(Data$Likert,
          mu = 3)

One-sample Sign-Test

s = 7, number of differences = 8, p-value = 0.07031

### Note the p-value in the output above

```

```
alternative hypothesis: true median is not equal to 3
```

```
97.9 percent confidence interval:
 3 5
```

```
sample estimates:
median of the differences
 4
```

```
### Median value and confidence interval
```

Two-sample Mann-Whitney U Test

When to use this test

The two-sample Mann-Whitney U test is a rank-based test that compares values for two groups. A significant result suggests that the values for the two groups are different. It is equivalent to a two-sample Wilcoxon rank-sum test.

Without further assumptions about the distribution of the data, the Mann-Whitney test does not address hypotheses about the medians of the groups. Instead, the test addresses if it is likely that an observation in one group is greater than an observation in the other. This is sometimes stated as testing if one sample has stochastic dominance compared with the other.

The test assumes that the observations are independent. That is, it is not appropriate for paired observations or repeated measures data.

The test is performed with the *wilcox.test* function.

Appropriate effect size statistics include Vargha and Delaney's *A*, Cliff's *delta*, among others.

Appropriate data

- Two-sample data. That is, one-way data with two groups only
- Dependent variable is ordinal, interval, or ratio
- Independent variable is a factor with two levels. That is, two groups
- Observations between groups are independent. That is, not paired or repeated measures data
- In order to be a test of medians, the distributions of values for each group need to be of similar shape and spread. Otherwise the test is typically a test of stochastic equality.

Hypotheses

- Null hypothesis: The two groups are sampled from populations with identical distributions. Typically, that the sampled populations exhibit stochastic equality.

- Alternative hypothesis (two-sided): The two groups are sampled from populations with different distributions. Typically, that one sampled population exhibits stochastic dominance.

Interpretation

Significant results can be reported as e.g. “Values for group A were significantly different from those for group B.”

Other notes and alternative tests

The Mann–Whitney U test can be considered equivalent to the Kruskal–Wallis test with only two groups.

Mood’s median test compares the medians of two groups. It is described in its own chapter.

For ordinal data, an alternative is to use cumulative link models, which are described later in this book.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- rcompanion
- coin
- DescTools
- effsize

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(coin)){install.packages("coin")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(effsize)){install.packages("effsize")}
```

Two-sample Mann–Whitney U test example

This example re-visits the Pooh and Piglet data from the *Descriptive Statistics with the likert Package* chapter.

It answers the question, “Are Pooh’s scores significantly different from those of Piglet?”

The Mann–Whitney U test is conducted with the *wilcox.test* function, which produces a *p*-value for the hypothesis. First the data are summarized and examined using bar plots for each group.

```

Input ="
Speaker Likert
Pooh      3
Pooh      5
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      5
Pooh      5
Piglet    2
Piglet    4
Piglet    2
Piglet    2
Piglet    1
Piglet    2
Piglet    3
Piglet    2
Piglet    2
Piglet    3
")
Data = read.table(textConnection(Input),header=TRUE)

### Create a new variable which is the Likert scores as an ordered factor
Data$Likert.f = factor(Data$Likert,
                       ordered = TRUE)

### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

```

Summarize data treating Likert scores as factors

Note that the variable we want to count is *Likert.f*, which is a factor variable. Counts for *Likert.f* are cross tabulated over values of *Speaker*. The *prop.table* function translates a table into proportions. The *margin=1* option indicates that the proportions are calculated for each row.

```

xtabs( ~ Speaker + Likert.f,
      data = Data)

  Likert.f
Speaker 1 2 3 4 5
Piglet 1 6 2 1 0
Pooh    0 0 1 6 3

XT = xtabs( ~ Speaker + Likert.f,
            data = Data)

prop.table(XT,
           margin = 1)

  Likert.f
Speaker 1 2 3 4 5
Piglet 0.1 0.6 0.2 0.1 0.0
Pooh   0.0 0.0 0.1 0.6 0.3

```

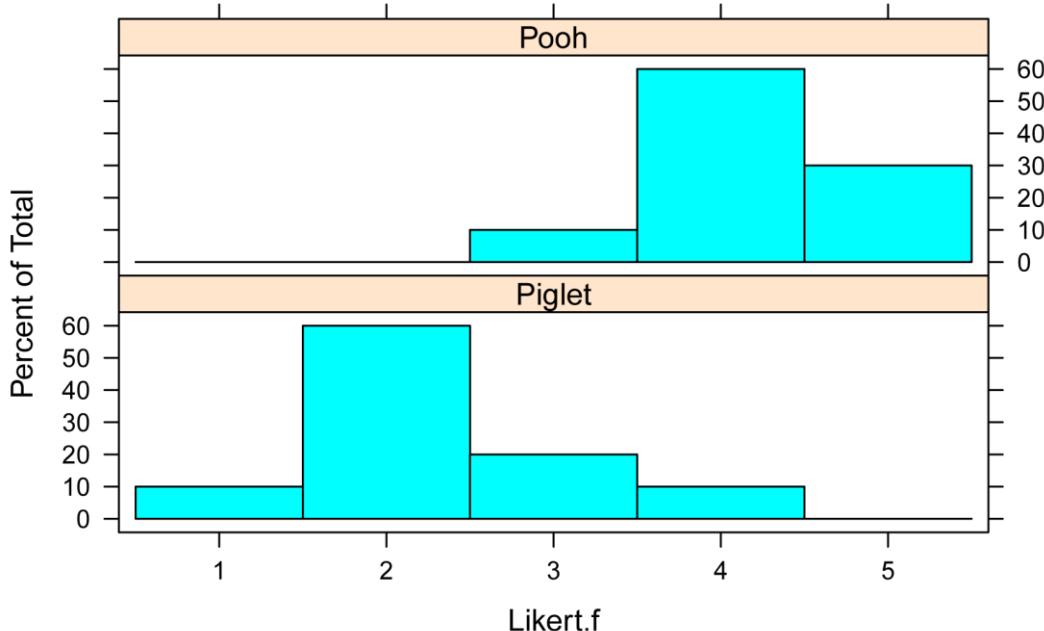
Bar plots of data by group

```

library(lattice)

histogram(~ Likert.f | Speaker,
          data=Data,
          layout=c(1,2)      # columns and rows of individual plots
)

```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Speaker,
           data=Data,
           digits=3)

  Speaker   n  mean      sd  min Q1 median     Q3 max percZero
1  Piglet 10  2.3  0.823    1  2       2  2.75    4      0
2   Pooh 10  4.2  0.632    3  4       4  4.75    5      0
```

Two-sample Mann-Whitney U test example

This example uses the formula notation indicating that *Likert* is the dependent variable and *Speaker* is the independent variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?wilcox.test*.

```
wilcox.test(Likert ~ Speaker,
            data=Data)

Wilcoxon rank sum test with continuity correction
W = 5, p-value = 0.0004713
alternative hypothesis: true location shift is not equal to 0

### You may get a "cannot compute exact p-value with ties" error.
### You can ignore this or use the exact=FALSE option.
```

Effect size

Statistics of effect size for the Mann-Whitney test report the degree to which one group has data with higher ranks than the other group. They are related to the probability that a value from one group will be greater than a value from the other group. Unlike *p*-values, they are not affected by sample size.

Vargha and Delaney's *A* is relatively easy to understand. It reports the probability that a value from one group will be greater than a value from the other group. A value of 0.50 indicates that the two groups are stochastically equal. A value of 1 indicates that the first group shows complete stochastic domination over the other group, and a value of 0 indicates the complete stochastic domination by the second group.

Cliff's *delta* is linearly related to Vargha and Delaney's *A*. It ranges from -1 to 1, with 0 indicating stochastic equality of the two groups. 1 indicates that one group shows complete stochastic domination over the other group, and a value of -1 indicates the complete stochastic domination of the other group. Its absolute value will be numerically equal to Freeman's *theta*.

A common effect size statistic for the Mann-Whitney test is *r*, which is the *Z* value from the test divided by the total number of observations. As written here, *r* varies from 0 to close to 1. In some formulations, it varies from -1 to 1.

Kendall's *tau-b* is sometimes used, and varies from approximately -1 to 1.

Freeman's *theta* and *epsilon*-squared are usually used when there are more than two groups, with the Kruskal-Wallis test, but can also be employed in the case of two groups.

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment, but for behavioral studies, the guidelines proposed by Cohen (1988) are sometimes followed. The following guidelines are based on the literature values and my personal intuition. They should not be considered universal.

Optional technical note: The interpretation values for *r* below are found commonly in published literature and on the internet. I suspect that this interpretation stems from the adoption of Cohen's interpretation of values for Pearson's *r*. This may not be justified, but it turns out that this interpretation for the *r* used here is relatively reasonable. The interpretation for *tau-b*, Freeman's *theta*, and *epsilon*-squared here are based on their values relative to those for *r*, based on simulated data (5-point Likert items, *n* per group between 4 and 25). Plots for some of these simulations are shown below.

Interpretations for Vargha and Delaney's *A* and Cliff's *delta* come from Vargha and Delaney (2000).

	<u>small</u>	<u>medium</u>	<u>large</u>
<i>r</i>	0.10 – < 0.30	0.30 – < 0.50	≥ 0.50
<i>tau-b</i>	0.10 – < 0.30	0.30 – < 0.50	≥ 0.50
Cliff's <i>delta</i>	0.11 – < 0.28	0.28 – < 0.43	≥ 0.43
Vargha and Delaney's <i>A</i>	0.56 – < 0.64 > 0.34 – 0.44	0.64 – < 0.71 > 0.29 – 0.34	≥ 0.71 ≤ 0.29
Freeman's <i>theta</i>	0.11 – < 0.34	0.34 – < 0.58	≥ 0.58
<i>epsilon</i>-squared	0.01 – < 0.08	0.08 – < 0.26	≥ 0.26

Vargha and Delaney's *A*

```
library(effsize)
VD.A(d = Data$Likert,
f = Data$Speaker)
```

Vargha and Delaney A

A estimate: 0.05 (large)

```
library(rcompanion)
vda(Likert ~ Speaker, data=Data)
```

VDA
0.05

```
library(rcompanion)

vda(Likert ~ Speaker, data=Data, ci=TRUE)

  VDA lower.ci upper.ci
1 0.05      0     0.162

### Note: Bootstrapped confidence interval may vary.
```

Cliff's delta

```
library(effsize)

cliff.delta(d = Data$Likert,
            f = Data$Speaker)

Cliff's Delta

delta estimate: -0.9 (large)

95 percent confidence interval:
      lower      upper
-0.9801533 -0.5669338
```

```
library(rcompanion)

cliffDelta(Likert ~ Speaker, data=Data)

Cliff.delta
-0.9
```

```
library(rcompanion)

cliffDelta(Likert ~ Speaker, data=Data, ci=TRUE)

  cliff.delta lower.ci upper.ci
1           -0.9       -1     -0.67
```

Note: Bootstrapped confidence interval may vary.

r

```
library(rcompanion)

wilcoxonR(x = Data$Likert,
          g = Data$Speaker)
```

r
0.791

```
library(rcompanion)

wilcoxonR(x = Data$Likert,
          g = Data$Speaker,
          ci = TRUE)

      r lower.ci upper.ci
1 0.791    0.602    0.897

### Note: Bootstrapped confidence interval may vary.
```

tau-b

```
library(DescTools)

KendallTauB(x = Data$Likert,
            y = as.numeric(Data$Speaker))

[1] 0.7397954
```

```
library(DescTools)

KendallTauB(x = Data$Likert,
            y = as.numeric(Data$Speaker),
            conf.level = 0.95)

      tau_b    lwr.ci    upr.ci
0.7397954 0.6074611 0.8721298
```

Freeman's theta

```
library(rcompanion)

freemanTheta(x = Data$Likert,
             g = Data$Speaker)

Freeman.theta
0.9
```

```
library(rcompanion)

freemanTheta(x = Data$Likert,
             g = Data$Speaker,
             ci = TRUE)
```

```

Freeman.theta lower.ci upper.ci
1           0.9     0.688      1

### Note: Bootstrapped confidence interval may vary.

```

epsilon-squared

```

library(rcompanion)

epsilonSquared(x = Data$Likert,
               g = Data$Speaker)

epsilon.squared
0.658

library(rcompanion)

epsilonSquared(x = Data$Likert,
               g = Data$Speaker,
               ci = TRUE)

epsilon.squared lower.ci upper.ci
1           0.658     0.383     0.842

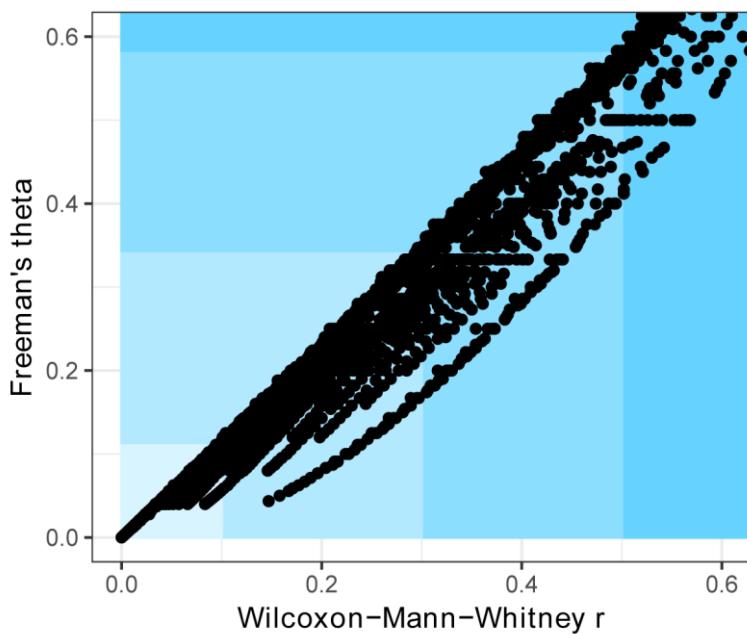
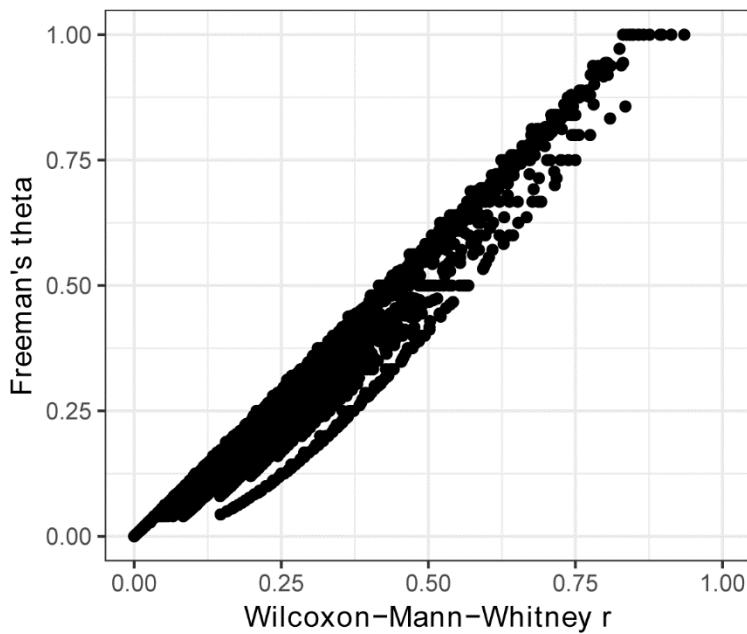
### Note: Bootstrapped confidence interval may vary.

```

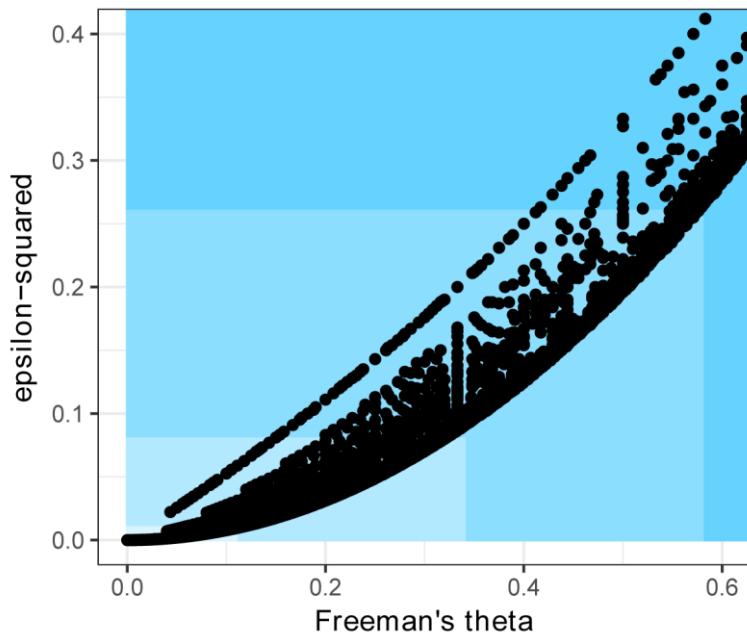
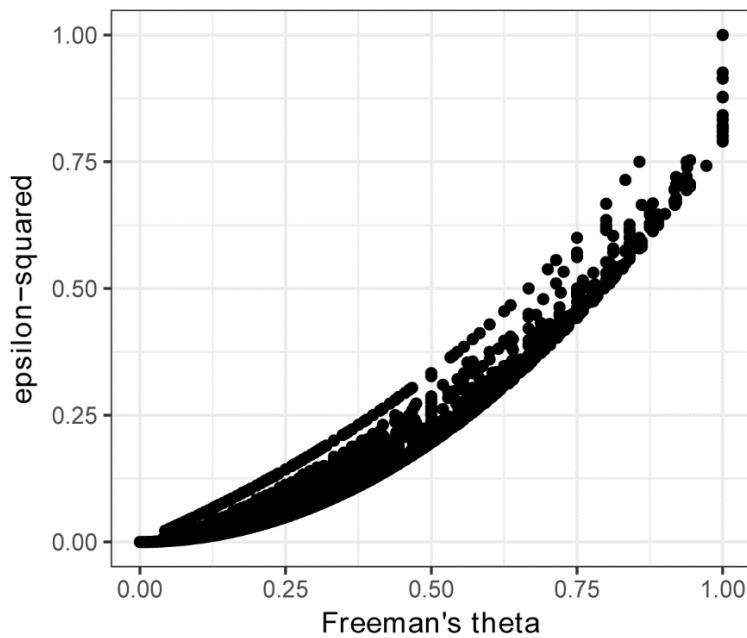
Optional: Comparison among effect size statistics

The follow plots show the relationship among effect size statistics discussed in this chapter. Data were 5-point Likert item responses, with n per group between 4 and 25.

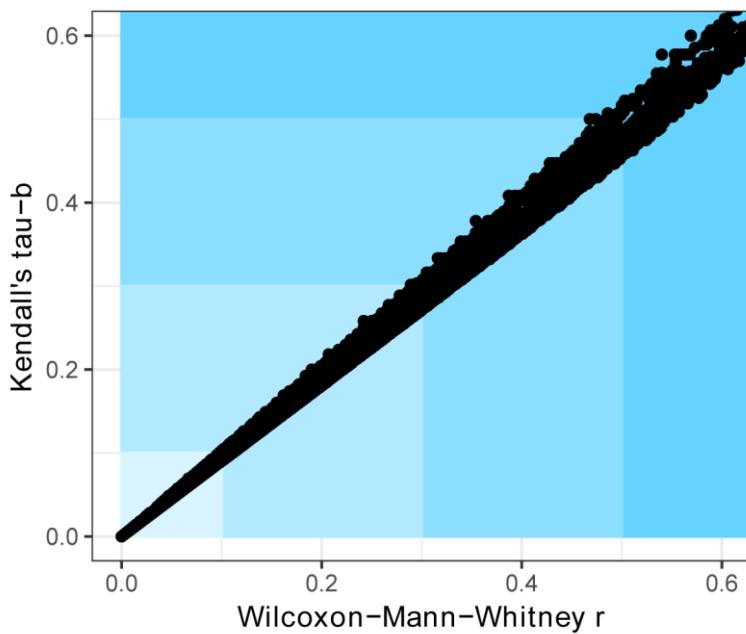
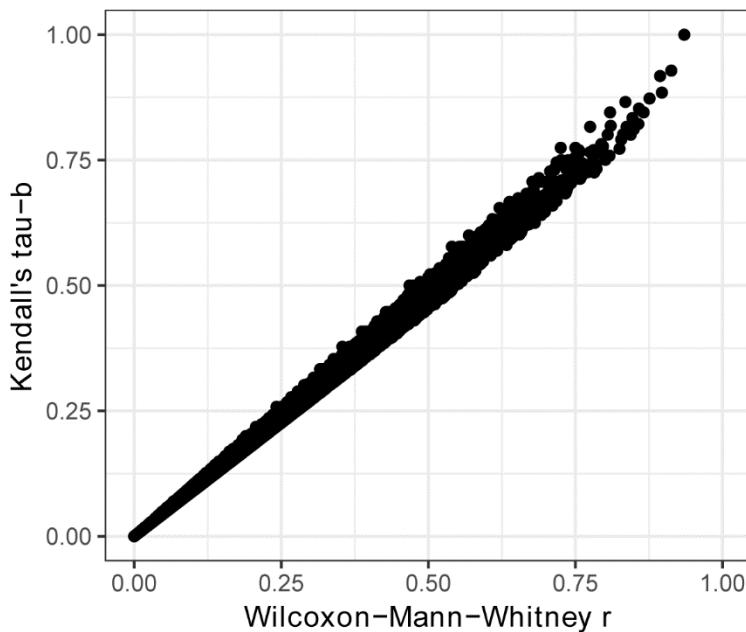
Freeman's *theta* was mostly linearly related to r , with variation depending on sample size and data values. In the second figure below, the colors indicate interpretation of less-than-small, small, medium, and large as the blue becomes darker.



The relationship of *epsilon*-squared and Freeman's *theta* was curvilinear, with variation depending on sample size and data values. In the second figure below, the colors indicate interpretation of less-than-small, small, medium, and large as the blue becomes darker



Kendall's τ -b was relatively closely linearly related to r , up to a value of about 0.88. In second figure below, the colors indicate interpretation of less-than-small, small, medium, and large as the blue becomes darker.



References

- Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.
- Vargha, A. and H.D. Delaney. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. 2000. *Journal of Educational and Behavioral Statistics* 25(2):101-132.

Exercises J

1. Considering Pooh and Piglet's data,
 - a. What was the median score for each instructor?
 - b. What were the first and third quartiles for each instructor's scores?
 - c. According to the Mann-Whitney test, is there a difference in scores between the instructors?
 - d. What was the value of Vargha and Delaney's *A* for the effect size for these data?
 - e. How do you interpret this value? (What does it mean? And is the standard interpretation in terms of "small", "medium", or "large"?)
 - f. How would you summarize the results of the descriptive statistics and tests? Include practical considerations of any differences.

2. Brian and Stewie Griffin want to assess the education level of students in their courses on creative writing for adults. They want to know the median education level for each class, and if the education level of the classes were different between instructors.

They used the following table to code his data.

Code	Abbreviation	Level
1	< HS	Less than high school
2	HS	High school
3	BA	Bachelor's
4	MA	Master's
5	PhD	Doctorate

The following are the course data.

Instructor	Student	Education
'Brian Griffin'	a	3
'Brian Griffin'	b	2
'Brian Griffin'	c	3
'Brian Griffin'	d	3
'Brian Griffin'	e	3
'Brian Griffin'	f	3
'Brian Griffin'	g	4
'Brian Griffin'	h	5
'Brian Griffin'	i	3
'Brian Griffin'	j	4
'Brian Griffin'	k	3
'Brian Griffin'	l	2

```
'Stewie Griffin' m      4
'Stewie Griffin' n      5
'Stewie Griffin' o      4
'Stewie Griffin' p      4
'Stewie Griffin' q      4
'Stewie Griffin' r      4
'Stewie Griffin' s      3
'Stewie Griffin' t      5
'Stewie Griffin' u      4
'Stewie Griffin' v      4
'Stewie Griffin' w      3
'Stewie Griffin' x      2
```

For each of the following, answer the question, and ***show the output from the analyses you used to answer the question.***

- a. What was the median education level for each instructor? (Be sure to report the education level, not just the numeric code!)
- b. What were the first and third quartiles for education level for each instructor?
- c. According to the Mann–Whitney test, is there a difference in scores between the instructors?
- d. What was the value of Vargha and Delaney's A for the effect size for these data?
- e. How do you interpret this value? (What does it mean? And is the standard interpretation in terms of "small", "medium", or "large"?)
- f. Plot Brian and Stewie's data in a way that helps you visualize the data. Do the results reflect what you would expect from looking at the plot?
- g. How would you summarize the results of the descriptive statistics and tests? Include your practical interpretation.

Mood's Median Test for Two-sample Data

Mood's median test compares the medians of two or more groups. The test can be conducted with the *mood.medtest* function in the *RVAideMemoire* package or with the *median_test* function in the *coin* package.

Appropriate data

- One-way data with two or more groups
- Dependent variable is ordinal, interval, or ratio
- Independent variable is a factor with levels indicating groups

- Observations between groups are independent. That is, not paired or repeated measures data

Hypotheses

- Null hypothesis: The medians of the populations from which the groups were sampled are equal.
- Alternative hypothesis (two-sided): The medians of the populations from which the groups were sampled are not equal.

Interpretation

Significant results can be reported as “The median value of group A was significantly different from group B.”

Packages used in this chapter

The packages used in this chapter include:

- RVAideMemoire
- coin

The following commands will install these packages if they are not already installed:

```
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(coin)){install.packages("coin")}
```

Example using the *RVAideMemoire* package

This example uses the formula notation indicating that *Likert* is the dependent variable and *Speaker* is the independent variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?mood.medtest*.

For appropriate plots and summary statistics, see the *Two-sample Mann–Whitney U Test* chapter.

```
Input =""
Speaker Likert
Pooh      3
Pooh      5
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      5
Pooh      5
Piglet    2
Piglet    4
Piglet    2
Piglet    2
```

```

Piglet    1
Piglet    2
Piglet    3
Piglet    2
Piglet    2
Piglet    3
")
)

Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Mood's Median Test

```

library(RVAideMemoire)

mood.medtest(Likert ~ Speaker,
             data = Data,
             exact = FALSE)

Mood's median test

X-squared = 9.8, df = 1, p-value = 0.001745

### Median test by Monte Carlo simulation

library(coin)

median_test(Likert ~ Speaker,
            data = Data,
            distribution = approximate(B = 10000))

Approximative Two-Sample Brown-Mood Median Test

Z = -3.4871, p-value = 0.0011

```

Two-sample Paired Signed-rank Test

When to use this test

The two-sample signed-rank test for paired data is used to compare values for two groups where each observation in one group is paired with one observation in the other group.

The test is useful to compare scores on a pre-test vs. scores on a post-test, or scores or ratings from two speakers, two different presentations, or two groups of audiences when there is a reason to pair observations, such as being done by the same rater.

A discussion of paired data can be found in the *Independent and Paired Values* chapter of this book.

Because the first step in the calculations is the subtraction of the paired values, one from the other, the data must be at least ordinal in nature.

The test is equivalent to using a one-sample signed-rank test on the difference of the paired values.

In base R, the test is performed with the *wilcox.test* function with the *paired=TRUE* option. However, the *wilcoxonsign_test* in the *coin* package has the advantage of using the Pratt method to handle zero differences, which may be preferable in some cases.

Appropriate data

- Two-sample paired data. That is, one-way data with two groups only, where the observations are paired between groups.
- Dependent variable is interval, or ratio
- Independent variable is a factor with two levels. That is, two groups
- For the test to be a test of the median of the differences, the distribution of differences in paired samples needs to be symmetric

Hypotheses

- Null hypothesis: The population of the differences of paired values is symmetric around zero.
- Alternate hypothesis: (two-sided): The population of the differences of paired values is not symmetric around zero.

Interpretation

Significant results can be reported as e.g. "Values for group A were significantly different from those of group B."

Other notes and alternative tests

Some authors recommend this test only in cases where the distribution of the differences is symmetric. It is my understanding that this requirement is only for the test to be considered a test of the median of the differences.

If the median is the statistic of interest, the two-sample sign test for paired data can be used.

Packages used in this chapter

The packages used in this chapter include:

- psych
- coin
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(coin)){install.packages("coin")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Two-sample paired signed-rank test example

For this example, imagine we want to compare scores for Pooh between Time 1 and Time 2. Here, we've recorded the identity of the student raters, and Pooh's score for each rater. This allows us to focus on the changes for each rater between Time 1 and Time 2. This makes for a more powerful test than would the Mann–Whitney U test in cases like this where one rater might tend to rate high and another rater might tend to rate low, but there is an overall trend in how raters change their scores between Time 1 and Time 2.

Note in this example we needed to record the identity of the student rater so that a rater's score from Time 1 can be paired with their score from Time 2. If we cannot pair data in this way—for example, if we did not record the identity of the raters—the data would have to be treated as unpaired, independent samples, for example like those in the *Two-sample Mann–Whitney U Test* chapter.

Also note that the data is arranged in long form. In this form, for this test, the data must be ordered so that the first observation where *Time* = 1 is paired to the first observation where *Time* = 2, and so on.

```
Input =""
Speaker  Time  Student  Likert
Pooh     1      a        1
Pooh     1      b        4
Pooh     1      c        3
Pooh     1      d        3
Pooh     1      e        3
Pooh     1      f        3
Pooh     1      g        4
Pooh     1      h        3
Pooh     1      i        3
Pooh     1      j        3
Pooh     2      a        4
Pooh     2      b        5
Pooh     2      c        4
Pooh     2      d        5
Pooh     2      e        4
Pooh     2      f        5
```

```

Pooh      2      g      3
Pooh      2      h      4
Pooh      2      i      3
Pooh      2      j      4
")

Data = read.table(textConnection(Input),header=TRUE)

### Order data by Time and Student if not already ordered

Data = Data[order(Data$Time, Data$Student),]

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Number of observations per group

It is helpful to check the data to be sure there is one observation per *student* per *time*.

```

xtabs(~ Student + Time,
      data = Data)

      Time
Student 1 2
      a 1 1
      b 1 1
      c 1 1
      d 1 1
      e 1 1
      f 1 1
      g 1 1
      h 1 1
      i 1 1
      j 1 1

```

Plot the paired data

Scatter plot with one-to-one line

Paired data can be visualized with a scatter plot of the paired cases. In the plot below, points that fall above and to the left of the blue line indicate cases for which the value for Time 2 was greater than for Time 1.

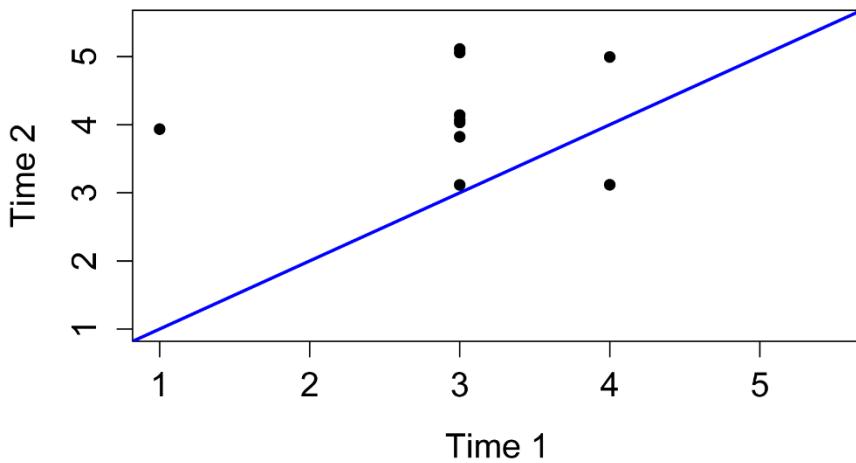
Note that the points in the plot are jittered slightly so that points which would fall directly on top of one another can be seen.

First, two new variables, *Time.1* and *Time.2*, are created by extracting the values of *Likert* for observations with the *Time* variable equal to 1 or 2, respectively, and then the plot is produced.

Note that for this to work correctly, the data must be ordered so that the first observation where *Time* = 1 is paired to the first observation where *Time* = 2, and so on.

```
Time.1 = Data$Likert[Data$Time==1]
Time.2 = Data$Likert[Data$Time==2]

plot(Time.1, jitter(Time.2),      # jitter offsets points so you can see them all
      pch = 16,                  # shape of points
      cex = 1.0,                  # size of points
      xlim=c(1, 5.5),            # limits of x axis
      ylim=c(1, 5.5),            # limits of y axis
      xlab="Time 1",
      ylab="Time 2"
)
abline(0,1, col="blue", lwd=2) # Line with intercept of 0 and slope of 1
```

Bar plot of differences

Paired data can also be visualized with a bar chart of differences. In the plot below, bars with a value greater than zero indicate cases for which values for Time 2 are greater than for Time 1.

New variables are first created for *Time.1*, *Time.2*, and their *Difference*. And then the plot is produced.

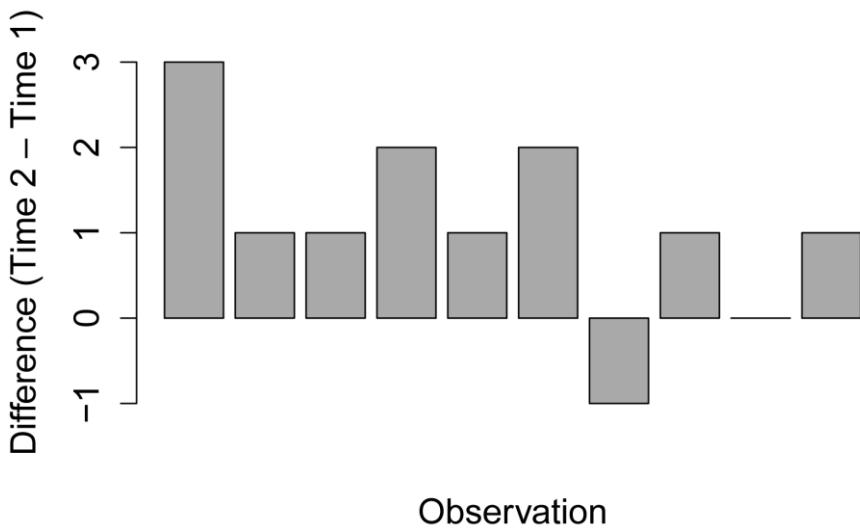
Note that for this to work correctly, the data must be ordered so that the first observation where *Time* = 1 is paired to the first observation where *Time* = 2, and so on.

```
Time.1 = Data$Likert[Data$Time==1]
Time.2 = Data$Likert[Data$Time==2]

Difference = Time.2 - Time.1

barplot(Difference,
        col="dark gray",
        xlab="Observation",
        ylab="Difference (Time 2 - Time 1)")
```

variable to plot
color of bars
x-axis label
y-axis label



Bar plot of differences

A bar plot of differences in paired data can be used to examine the distribution of the differences.

Here, new variables are created: *Time.1*, *Time.2*, *Difference*, and *Diff.f*, which has the same values as *Difference* but as a factor variable. The *xtabs* function is used to create a count of values of *Diff.f*. The *barplot* function then uses these counts.

Note that for this to work correctly, the data must be ordered so that the first observation where *Time* = 1 is paired to the first observation where *Time* = 2, and so on.

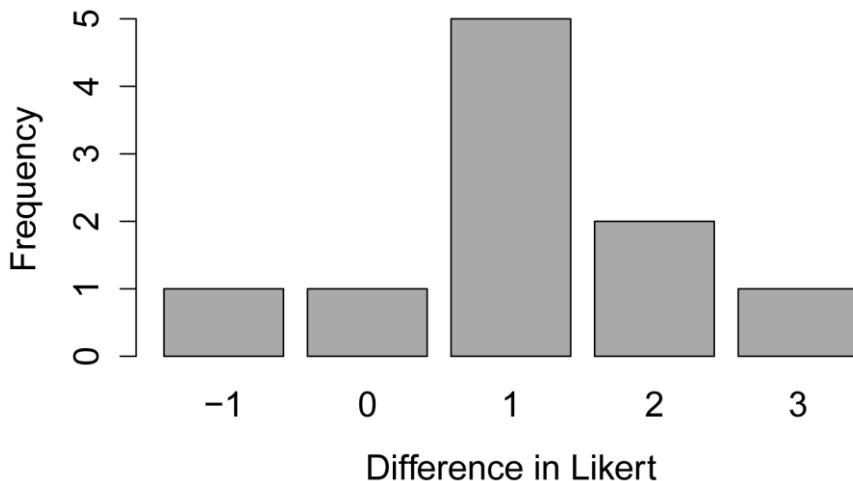
```
Time.1 = Data$Likert[Data$Time==1]
Time.2 = Data$Likert[Data$Time==2]

Difference = Time.2 - Time.1

Diff.f = factor(Difference)

XT = xtabs(~ Diff.f)
```

```
barplot(xT,
        col="dark gray",
        xlab="Difference in Likert",
        ylab="Frequency")
```



Descriptive statistics

It is helpful to look at medians for each group and the median difference between groups in order to determine the practical importance of the differences.

```
library(FSA)

Summarize(Likert ~ Time,
           data = Data)

  Time   n  mean        sd  min  Q1 median    Q3 max
1     1 10  3.0 0.8164966  1  3      3 3.00  4
2     2 10  4.1 0.7378648  3  4      4 4.75  5
```

Note that for the following to work correctly, the data must be ordered so that the first observation where *Time* = 1 is paired to the first observation where *Time* = 2, and so on.

```
Time.1 = Data$Likert[Data$Time==1]

Time.2 = Data$Likert[Data$Time==2]

Difference = Time.2 - Time.1

median(Difference)

[1] 1
```

Two-sample paired signed-rank test

Note that if data are in long format, the data must be ordered so that the first observation of Time 1 is paired to the first observation of Time 2, and so on, because the *wilcox.test* function will take the observations in order.

This example uses the formula notation indicating that *Likert* is the dependent variable and *Time* is the independent variable. The *data=* option indicates the data frame that contains the variables, and *paired=TRUE* indicates that the test for paired data should be used. For the meaning of other options, see *?wilcox.test*.

```
wilcox.test(Likert ~ Time,
            data = Data,
            paired = TRUE,
            conf.int = TRUE,
            conf.level = 0.95)

Wilcoxon signed rank test with continuity correction
V = 3.5, p-value = 0.02355
alternative hypothesis: true location shift is not equal to 0

### Note the p-value given in the above results

95 percent confidence interval:
-2.000051e+00 -1.458002e-05

### Confidence interval for the median or the location of differences

### You may get a "cannot compute exact p-value with ties" error.
### You can ignore this or use the exact=FALSE option.
```

Coin package

By default the *wilcoxonsign_test* function uses the Pratt method to handle zero differences.

```
library(coin)

wilcoxonsign_test(Likert ~ Time | Student,
                  data = Data)

Asymptotic Wilcoxon-Pratt Signed-Rank Test

Z = -2.3522, p-value = 0.01866

alternative hypothesis: true mu is not equal to 0

Time.1 = Data$Likert[Data$Time==1]
Time.2 = Data$Likert[Data$Time==2]
wilcoxonsign_test(Time.1 ~ Time.2)
```

Asymptotic Wilcoxon-Pratt Signed-Rank Test

Z = -2.3522, p-value = 0.01866

alternative hypothesis: true mu is not equal to 0

Effect size

I am not aware of any established effect size statistic for the Wilcoxon signed-rank test. However, using a statistic analogous to the r used in the Mann-Whitney test may make sense.

As written here, r varies from 0 to 1. In some formulations, it varies from -1 to 1.

The following interpretation is based on my personal intuition. It is not intended to be universal.

	<u>small</u>	<u>medium</u>	<u>large</u>
r	0.10 – < 0.40	0.40 – < 0.60	≥ 0.60

```
library(rcompanion)

wilcoxonPairedR(x = Data$Likert,
                 g = Data$Time)

      r
0.744
```

Exercises K

1. Considering Pooh's data for Time 1 and Time 2,

- What do the plots suggest about the relative value of the scores for Time 1 and Time 2? That is, do they suggest that scores increased, decreased, or stayed the same between Time 1 and Time 2?
- Is the distribution of the differences between paired samples relatively symmetrical?
- Does the two-sample paired signed-rank test indicate that there is a significant difference between Time 1 and Time 2?
- Practically speaking, what do you conclude? If significant, is the difference between Time 1 and Time 2 of practical importance?

2. Lois Griffin gave proficiency scores to her students in her course on piano playing for adults. She gave a score for each student for their left hand playing and right hand playing. She wants to know if

students in her class are more proficient in the right hand, left hand, or if there is no difference in hands.

Instructor	Student	Hand	Score
'Lois Griffin'	a	left	8
'Lois Griffin'	a	right	9
'Lois Griffin'	b	left	6
'Lois Griffin'	b	right	5
'Lois Griffin'	c	left	7
'Lois Griffin'	c	right	9
'Lois Griffin'	d	left	6
'Lois Griffin'	d	right	7
'Lois Griffin'	e	left	7
'Lois Griffin'	e	right	7
'Lois Griffin'	f	left	9
'Lois Griffin'	f	right	9
'Lois Griffin'	g	left	4
'Lois Griffin'	g	right	6
'Lois Griffin'	h	left	5
'Lois Griffin'	h	right	8
'Lois Griffin'	i	left	5
'Lois Griffin'	i	right	6
'Lois Griffin'	j	left	7
'Lois Griffin'	j	right	8

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- Is the distribution of the differences between paired samples relatively symmetrical?
- Does the two-sample paired signed-rank test indicate that there is a difference between hands? If so, which hand received higher scores?
- What can you conclude about the results of the plots, summary statistics, effect size, and statistical test? Practically speaking, what do you conclude? If significant, is the difference between hands of practical importance?
- What if Lois wanted to change the design of the experiment so that she could determine if each student were more proficient in one hand or the other? That is, is student *a* more proficient in left hand or right? Is student *b* more proficient in left hand or right? How should she change what data she's collecting to determine this?

Sign Test for Two-sample Paired Data

The two-sample sign test assesses the number of observations in one group that are greater than paired observations in the other group without accounting for the magnitude of the difference. The

test is similar in purpose to the two-sample Wilcoxon signed-rank test, but looks specifically at the median value of differences (if the values are numeric), and is not affected by the distribution of the data.

The *SIGN.test* function in the *BSDA* package requires the data to be separated into two variables, each of which is ordered so that the first observation of each are paired, and so on. Information on options for the function can be viewed with *?SIGN.test*. The *SignTest* function in the *DescTools* package is similar.

For appropriate plots and summary statistics, see the *Two-sample Paired Signed-rank Test* chapter.

Appropriate data

- Two-sample paired data. That is, one-way data with two groups only, where the observations are paired between groups.
- Dependent variable is ordinal, interval, or ratio.
- Independent variable is a factor with two levels. That is, two groups.

Hypotheses

- Null hypothesis: For numeric data, the median of the paired differences in the population from which the sample was drawn is equal to zero.
- Alternative hypothesis (two-sided): For numeric data, the median of the paired differences in the population from which the sample was drawn is not equal to zero.

Interpretation

Significant results can be reported as “There was a significant difference in values between group A and group B.”

Packages used in this chapter

The packages used in this chapter include:

- psych
- BSDA
- DescTools

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(BSDA)){install.packages("BSDA")}
if(!require(DescTools)){install.packages("DescTools")}
```

Sign test for paired two-sample data example

```
Input =""
Speaker Time Student Likert
Pooh     1      a      1
Pooh     1      b      4
Pooh     1      c      3
```

```

Pooh      1      d      3
Pooh      1      e      3
Pooh      1      f      3
Pooh      1      g      4
Pooh      1      h      3
Pooh      1      i      3
Pooh      1      j      3
Pooh      2      a      4
Pooh      2      b      5
Pooh      2      c      4
Pooh      2      d      5
Pooh      2      e      4
Pooh      2      f      5
Pooh      2      g      3
Pooh      2      h      4
Pooh      2      i      3
Pooh      2      j      4
")
Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame

library(psych)

headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects

rm(Input)

```

Two-sample sign test with BSDA package

```

Time.1 = Data$Likert [Data$Time == 1]
Time.2 = Data$Likert [Data$Time == 2]

library(BSDA)

SIGN.test(x = Time.1,
          y = Time.2,
          alternative = "two.sided",
          conf.level = 0.95)

Dependent-samples Sign-Test

S = 1, p-value = 0.03906

### p-value reported above

```

```
95 percent confidence interval:
-2.0000000 -0.3244444
```

```
sample estimates:
median of x-y
-1
```

```
### median of differences and confidence interval of differences
```

Two-sample sign test with DescTools package

```
Time.1 = Data$Likert [Data$Time == 1]
Time.2 = Data$Likert [Data$Time == 2]

library(DescTools)

SignTest(x = Time.1,
          y = Time.2)

Dependent-samples Sign-Test

S = 1, number of differences = 9, p-value = 0.03906

### p-value reported above

alternative hypothesis: true median difference is not equal to 0

97.9 percent confidence interval:
-2 0

sample estimates:
median of the differences
-1

### median of differences and confidence interval of differences
```

Kruskal-Wallis Test

The Kruskal-Wallis test is a rank-based test that is similar to the Mann-Whitney U test, but can be applied to one-way data with more than two groups.

Without further assumptions about the distribution of the data, the Kruskal-Wallis test does not address hypotheses about the medians of the groups. Instead, the test addresses if it is likely that an observation in one group is greater than an observation in the other. This is sometimes stated as testing if one sample has stochastic dominance compared with the other.

The test assumes that the observations are independent. That is, it is not appropriate for paired observations or repeated measures data.

It is performed with the *kruskal.test* function.

Appropriate effect size statistics include maximum Vargha and Delaney's *A*, maximum Cliff's *delta*, Freeman's *theta*, and *epsilon*-squared.

Post-hoc tests

The outcome of the Kruskal-Wallis test tells you if there are differences among the groups, but doesn't tell you *which* groups are different from other groups. In order to determine which groups are different from others, post-hoc testing can be conducted. Probably the most common post-hoc test for the Kruskal-Wallis test is the Dunn test, here conducted with the *dunnTest* function in the *FSA* package.

Appropriate data

- One-way data
- Dependent variable is ordinal, interval, or ratio
- Independent variable is a factor with two or more levels. That is, two or more groups
- Observations between groups are independent. That is, not paired or repeated measures data
- In order to be a test of medians, the distributions of values for each group need to be of similar shape and spread. Otherwise the test is typically a test of stochastic equality.

Hypotheses

- Null hypothesis: The groups are sampled from populations with identical distributions. Typically, that the sampled populations exhibit stochastic equality.
- Alternative hypothesis (two-sided): The groups are sampled from populations with different distributions. Typically, that one sampled population exhibits stochastic dominance.

Interpretation

Significant results can be reported as "There was a significant difference in values among groups."

Post-hoc analysis allows you to say "There was a significant difference in values between groups A and B.", and so on.

Other notes and alternative tests

Mood's median test compares the medians of groups.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- multcompView

- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Kruskal-Wallis test example

This example re-visits the Pooh, Piglet, and Tigger data from the *Descriptive Statistics with the likert Package* chapter.

It answers the question, “Are the scores significantly different among the three speakers?”

The Kruskal-Wallis test is conducted with the *kruskal.test* function, which produces a *p*-value for the hypothesis. First the data are summarized and examined using bar plots for each group.

```
Input =""
Speaker Likert
Pooh    3
Pooh    5
Pooh    4
Pooh    4
Pooh    4
Pooh    4
Pooh    4
Pooh    4
Pooh    5
Pooh    5
Pooh    5
Piglet   2
Piglet   4
Piglet   2
Piglet   2
Piglet   1
Piglet   2
Piglet   3
Piglet   2
Piglet   2
Piglet   3
Piglet   3
Tigger   4
Tigger   4
Tigger   4
Tigger   5
Tigger   3
Tigger   5
Tigger   4
Tigger   4
Tigger   3
```

```

")
Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them

Data$Speaker = factor(Data$Speaker,
                      levels=unique(Data$Speaker))

### Create a new variable which is the Likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                       ordered = TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Summarize data treating Likert scores as factors

```

xtabs(~ Speaker + Likert.f,
      data = Data)

          Likert.f
Speaker 1 2 3 4 5
Pooh    0 0 1 6 3
Piglet  1 6 2 1 0
Tigger  0 0 2 6 2

XT = xtabs(~ Speaker + Likert.f,
           data = Data)

prop.table(XT,
           margin = 1)

          Likert.f
Speaker   1   2   3   4   5
Pooh     0.0 0.0 0.1 0.6 0.3
Piglet   0.1 0.6 0.2 0.1 0.0

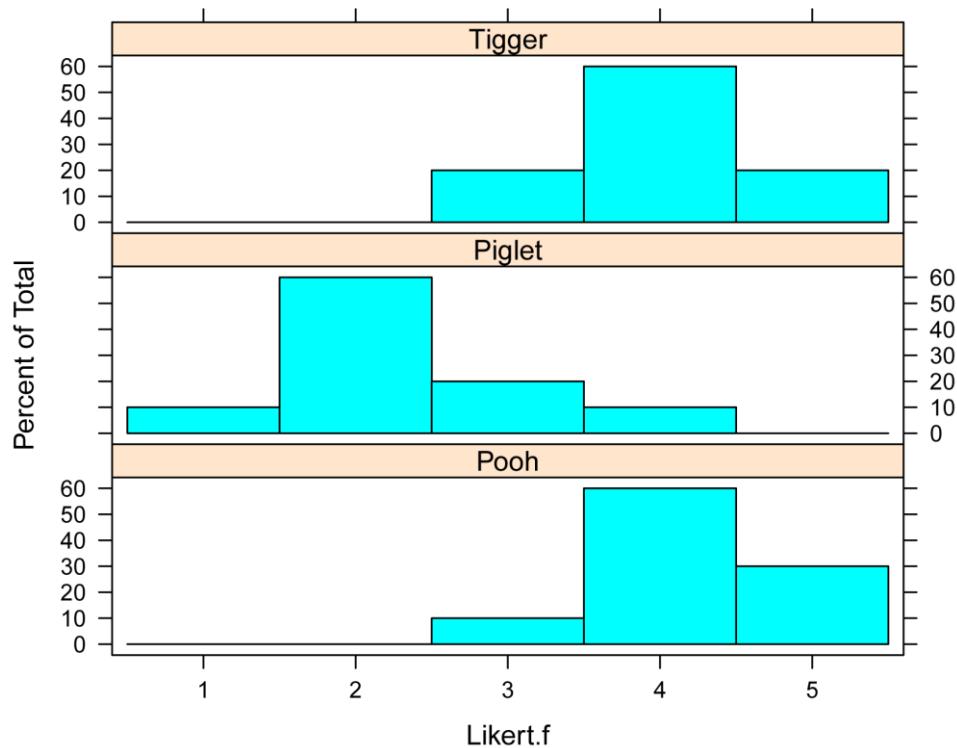
```

```
Tigger 0.0 0.0 0.2 0.6 0.2
```

Bar plots of data by group

```
library(lattice)

histogram(~ Likert.f | Speaker,
  data=Data,
  layout=c(1,3)      # columns and rows of individual plots
)
```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Speaker,
  data=Data,
  digits=3)
```

Speaker	n	mean	sd	min	Q1	median	Q3	max	percZero
Pooh	10	4.2	0.632	3	4		4.75	5	0
Piglet	10	2.3	0.823	1	2		2.75	4	0
Tigger	10	4.0	0.667	3	4		4.00	5	0

Kruskal-Wallis test example

This example uses the formula notation indicating that *Likert* is the dependent variable and *Speaker* is the independent variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?kruskal.test*.

```
kruskal.test(Likert ~ Speaker,
             data = Data)
```

Kruskal-Wallis rank sum test

Kruskal-Wallis chi-squared = 16.842, df = 2, p-value = 0.0002202

Effect size

Statistics of effect size for the Kruskal-Wallis test provide the degree to which one group has data with higher ranks than another group. They are related to the probability that a value from one group will be greater than a value from another group. Unlike *p*-values, they are not affected by sample size.

Appropriate effect size statistics for the Kruskal-Wallis test include Freeman's *theta* and *epsilon*-squared. *epsilon*-squared is probably the most common. For Freeman's *theta*, an effect size of 1 indicates that the measurements for each group are entirely greater or entirely less than some other group, and an effect size of 0 indicates that there is no effect; that is, that the groups are absolutely stochastically equal.

Another option is to use the maximum Cliff's *delta* or Vargha and Delaney's *A* (VDA) from pairwise comparisons of all groups. VDA is the probability that an observation from one group is greater than an observation from the other group. Because of this interpretation, VDA is an effect size statistic that is relatively easy to understand.

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment. The following guidelines are based on my personal intuition or published values. They should not be considered universal.

Technical note: The values for the interpretations for Freeman's *theta* to *epsilon*-squared below were derived by keeping the interpretation for *epsilon*-squared constant and equal to that for the Mann-Whitney test. Interpretation values for Freeman's *theta* were determined through comparing Freeman's *theta* to *epsilon*-squared for simulated data (5-point Likert items, *n* per group between 4 and 25).

Interpretations for Vargha and Delaney's *A* and Cliff's *delta* come from Vargha and Delaney (2000).

	<u>small</u>	<u>medium</u>	<u>large</u>
<i>epsilon</i>-squared	0.01 – < 0.08	0.08 – < 0.26	≥ 0.26
Freeman's <i>theta</i>, <i>k</i> = 2	0.11 – < 0.34	0.34 – < 0.58	≥ 0.58
Freeman's <i>theta</i>, <i>k</i> = 3	0.05 – < 0.26	0.26 – < 0.46	≥ 0.46
Freeman's <i>theta</i>, <i>k</i> = 5	0.05 – < 0.21	0.21 – < 0.40	≥ 0.40

Freeman's theta, k = 7	0.05	- < 0.20	0.20	- < 0.38	≥ 0.38
Freeman's theta, k = 7	0.05	- < 0.20	0.20	- < 0.38	≥ 0.38
Maximum Cliff's delta	0.11	- < 0.28	0.28	- < 0.43	≥ 0.43
Maximum Vargha and Delaney's A	0.56	- < 0.64	0.64	- < 0.71	≥ 0.71
	> 0.34	- 0.44	> 0.29	- 0.34	≤ 0.29

epsilon-squared

```
library(rcompanion)

epsilonSquared(x = Data$Likert,
               g = Data$Speaker)

epsilon.squared
0.581
```

Freeman's theta

```
library(rcompanion)

freemanTheta(x = Data$Likert,
              g = Data$Speaker)

Freeman.theta
0.64
```

Maximum Vargha and Delaney's A or Cliff's delta

Here, the *multiVDA* function is used to calculate Vargha and Delaney's A (VDA), Cliff's delta (CD), and *r* between all pairs of groups. The function identifies the comparison with the most extreme VDA statistic (0.95 for *Pooh - Piglet*). That is, it identifies the most disparate groups.

```
source("http://rcompanion.org/r_script/multiVDA.r")

library(rcompanion)

library(coin)

multivDA(x = Data$Likert,
          g = Data$Speaker)

$pairs
      Comparison   VDA      CD       r  VDA.m  CD.m    r.m
1 Pooh - Piglet = 0 0.95  0.90  0.791  0.95  0.90  0.791
2 Pooh - Tigger = 0 0.58  0.16  0.154  0.58  0.16  0.154
3 Piglet - Tigger = 0 0.07 -0.86 -0.756  0.93  0.86  0.756

$comparison
      Comparison
"Pooh - Piglet = 0"
```

```
$statistic
```

```
VDA
```

```
0.95
```

```
$statistic.m
```

```
VDA.m
```

```
0.95
```

Post-hoc test: Dunn test for multiple comparisons of groups

If the Kruskal–Wallis test is significant, a post-hoc analysis can be performed to determine which groups differ from each other group.

Probably the most popular post-hoc test for the Kruskal–Wallis test is the Dunn test. The Dunn test can be conducted with the *dunnTest* function in the *FSA* package.

Because the post-hoc test will produce multiple *p*-values, adjustments to the *p*-values can be made to avoid inflating the possibility of making a type-I error. There are a variety of methods for controlling the familywise error rate or for controlling the false discovery rate. See *?p.adjust* for details on these methods.

When there are many *p*-values to evaluate, it is useful to condense a table of *p*-values to a compact letter display format. In the output, groups are separated by letters. Groups sharing the same letter are not significantly different. Compact letter displays are a clear and succinct way to present results of multiple comparisons.

```
### Order groups by median

Data$Speaker = factor(Data$Speaker,
                      levels=c("Pooh", "Tigger", "Piglet"))

levels(Data$Speaker)

### Dunn test

library(FSA)

DT = dunnTest(Likert ~ Speaker,
               data=Data,
               method="bh")      # Adjusts p-values for multiple comparisons;
                      # See ?dunnTest for options

DT

Dunn (1964) Kruskal-wallis multiple comparison
p-values adjusted with the Benjamini-Hochberg method.

      Comparison      Z     P.unadj      P.adj
1 Pooh - Tigger 0.4813074 0.6302980448 0.6302980448
2 Pooh - Piglet 3.7702412 0.0001630898 0.0004892695
3 Tigger - Piglet 3.2889338 0.0010056766 0.0015085149
```

```

### Compact letter display

PT = DT$res

PT

library(rcompanion)

cldList(P.adj ~ Comparison,
       data = PT,
       threshold = 0.05)

  Group Letter MonoLetter
1 Pooh      a      a
2 Tigger    a      a
3 Piglet    b      b

Groups sharing a letter not significantly different (alpha = 0.05).

```

Post-hoc test: pairwise Mann-Whitney U-tests for multiple comparisons

I don't recommend using pairwise Mann-Whitney U-tests for post-hoc testing for the Kruskal-Wallis test, but the following example shows how this can be done.

The *pairwise.wilcox.test* function produces a table of *p*-values comparing each pair of groups.

To prevent the inflation of type I error rates, adjustments to the *p*-values can be made using the *p.adjust.method* option. Here the *fdr* method is used. See *?p.adjust* for details on available *p*-value adjustment methods.

When there are many *p*-values to evaluate, it is useful to condense a table of *p*-values to a compact letter display format. This can be accomplished with a combination of the *fullPTable* function in the *rcompanion* package and the *multcompLetters* function in the *multcompView* package.

In a compact letter display, groups sharing the same letter are not significantly different.

Here the *fdr* *p*-value adjustment method is used. See *?p.adjust* for details on available methods.

The code creates a matrix of *p*-values called *PT*, then converts this to a fuller matrix called *PT1*. *PT1* is then passed to the *multcompLetters* function to be converted to a compact letter display.

Note that the *p*-value results of the pairwise Mann-Whitney U-tests differ somewhat from those of the Dunn test.

```

### Order groups by median

Data$Speaker = factor(Data$Speaker,
                      levels=c("Pooh", "Tigger", "Piglet"))

```

Data

```
### Pairwise Mann-Whitney
```

```
PT = pairwise.wilcox.test(Data$Likert,
                           Data$Speaker,
                           p.adjust.method="fdr")
                           # Adjusts p-values for multiple comparisons;
                           # See ?p.adjust for options
```

```
PT
```

Pairwise comparisons using wilcoxon rank sum test

Pooh	Tigger
Tigger	0.5174 -
Piglet	0.0012 0.0012

P value adjustment method: fdr

```
### Note that the values in the table are p-values comparing each
### pair of groups.
```

```
### Convert PT to a full table and call it PT1
```

```
PT = PT$p.value    ### Extract p-value table
```

```
library(rcompanion)
```

```
PT1 = fullPTable(PT)
```

```
PT1
```

	Pooh	Tigger	Piglet
Pooh	1.000000000	0.517377650	0.001241095
Tigger	0.517377650	1.000000000	0.001241095
Piglet	0.001241095	0.001241095	1.000000000

```
### Produce compact letter display
```

```
library(multcompView)
```

```
multcompLetters(PT1,
                compare="<",
                threshold=0.05, # p-value to use as significance threshold
                Letters=letters,
                reversed = FALSE)
```

Pooh	Tigger	Piglet
"a"	"a"	"b"

```
### Values sharing a letter are not significantly different
```

Plot of medians and confidence intervals

The following code uses the *groupwiseMedian* function to produce a data frame of medians for each speaker along with the 95% confidence intervals for each median with the percentile method. These medians are then plotted, with their confidence intervals shown as error bars. The grouping letters from the multiple comparisons (Dunn test or pairwise Mann-Whitney U-tests) are added.

Note that bootstrapped confidence intervals may not be reliable for discrete data, such as the ordinal Likert data used in these examples, especially for small samples.

```
library(rcompanion)

Sum = groupwiseMedian(Likert ~ Speaker,
                      data      = Data,
                      conf      = 0.95,
                      R         = 5000,
                      percentile = TRUE,
                      bca       = FALSE,
                      digits    = 3)

Sum

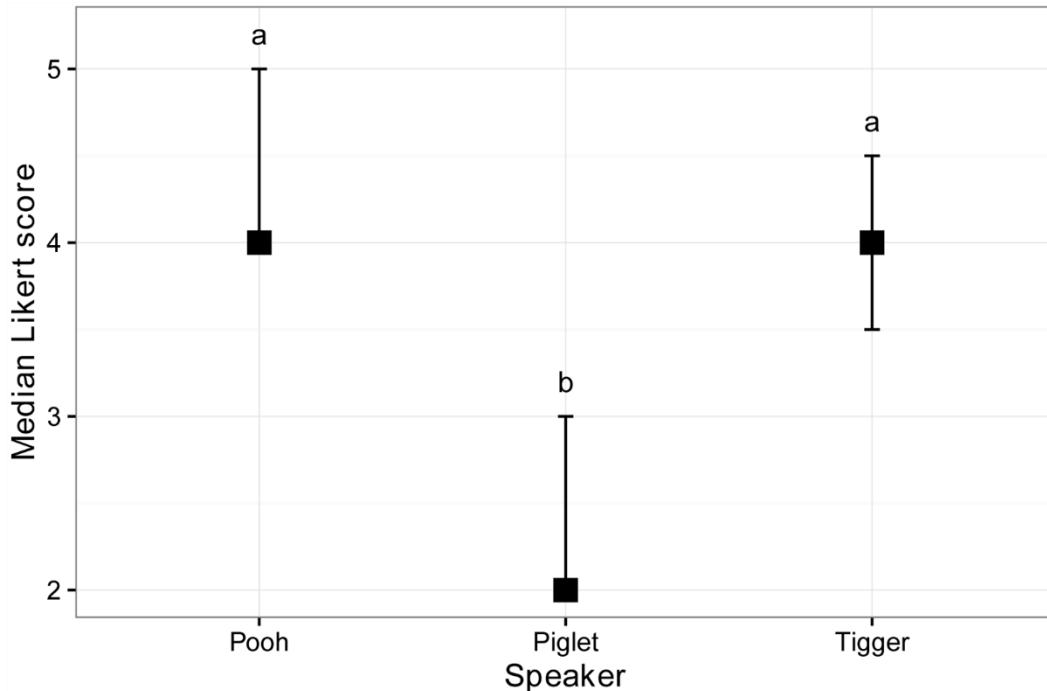
  Speaker n Median Conf.level Percentile.lower Percentile.upper
1   Pooh 10     4      0.95          4.0          5.0
2 Piglet 10     2      0.95          2.0          3.0
3 Tigger 10     4      0.95          3.5          4.5

X     = 1:3
Y     = Sum$Percentile.upper + 0.2
Label = c("a", "b", "a")

library(ggplot2)

ggplot(Sum,           ### The data frame to use.
       aes(x = Speaker,
           y = Median)) +
  geom_errorbar(aes(ymin = Percentile.lower,
                    ymax = Percentile.upper),
                width = 0.05,
                size  = 0.5) +
  geom_point(shape = 15,
             size  = 4) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Median Likert score") +
```

```
annotate("text",
  x = X,
  y = Y,
  label = Label)
```



Plot of median Likert score versus Speaker. Error bars indicate the 95% confidence intervals for the median with the percentile method.

References

- Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.
- Vargha, A. and H.D. Delaney. A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong. 2000. Journal of Educational and Behavioral Statistics 25(2):101–132.

Exercises L

1. Considering Pooh, Piglet, and Tigger's data,
 - a. What was the median score for each instructor?
 - b. According to the Kruskal-Wallis test, is there a statistical difference in scores among the instructors?

- c. What is the value of maximum Vargha and Delaney's *A* for these data?
- d. How do you interpret this value? (What does it mean? And is the standard interpretation in terms of "small", "medium", or "large")?
- e. Looking at the post-hoc analysis, which speakers' scores are statistically different from which others? Who had the statistically highest scores?
- f. How would you summarize the results of the descriptive statistics and tests? Include practical considerations of any differences.

2. Brian, Stewie, and Meg want to assess the education level of students in their courses on creative writing for adults. They want to know the median education level for each class, and if the education level of the classes were different among instructors.

They used the following table to code his data.

Code	Abbreviation	Level
1	< HS	Less than high school
2	HS	High school
3	BA	Bachelor's
4	MA	Master's
5	PhD	Doctorate

The following are the course data.

Instructor	Student	Education
'Brian Griffin'	a	3
'Brian Griffin'	b	2
'Brian Griffin'	c	3
'Brian Griffin'	d	3
'Brian Griffin'	e	3
'Brian Griffin'	f	3
'Brian Griffin'	g	4
'Brian Griffin'	h	5
'Brian Griffin'	i	3
'Brian Griffin'	j	4
'Brian Griffin'	k	3
'Brian Griffin'	l	2
'Stewie Griffin'	m	4
'Stewie Griffin'	n	5
'Stewie Griffin'	o	4
'Stewie Griffin'	p	4
'Stewie Griffin'	q	4
'Stewie Griffin'	r	4
'Stewie Griffin'	s	3
'Stewie Griffin'	t	5
'Stewie Griffin'	u	4
'Stewie Griffin'	v	4

'Stewie Griffin'	w	3
'Stewie Griffin'	x	2
'Meg Griffin'	y	3
'Meg Griffin'	z	4
'Meg Griffin'	aa	3
'Meg Griffin'	ab	3
'Meg Griffin'	ac	3
'Meg Griffin'	ad	2
'Meg Griffin'	ae	3
'Meg Griffin'	af	4
'Meg Griffin'	ag	2
'Meg Griffin'	ah	3
'Meg Griffin'	ai	2
'Meg Griffin'	aj	1

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- a. What was the median education level for each instructor's class? (Be sure to report the education level, not just the numeric code!)
- b. According to the Kruskal-Wallis test, is there a difference in the education level of students among the instructors?
- c. What is the value of maximum Vargha and Delaney's A for these data?
- d. How do you interpret this value? (What does it mean? And is the standard interpretation in terms of "small", "medium", or "large"?)
- e. Looking at the post-hoc analysis, which classes education levels are statistically different from which others? Who had the statistically highest education level?
- f. Plot Brian, Stewie, and Meg's data in a way that helps you visualize the data. Do the results reflect what you would expect from looking at the plot?
- g. How would you summarize the results of the descriptive statistics and tests? What do you conclude practically?

Mood's Median Test

Mood's median test compares the medians of two or more groups. The test can be conducted with the *mood.medtest* function in the *RVAideMemoire* package or with the *median_test* function in the *coin* package.

Post-hoc tests

The outcome of Mood's median test tells you if there are differences among the groups, but doesn't tell you *which* groups are different from other groups. In order to determine which groups are different from others, post-hoc testing can be conducted. The function *pairwiseMedianTest* in the *rcompanion* package can perform the post-hoc tests.

Appropriate data

- One-way data with two or more groups
- Dependent variable is ordinal, interval, or ratio
- Independent variable is a factor with levels indicating groups
- Observations between groups are independent. That is, not paired or repeated measures data

Hypotheses

- Null hypothesis: The medians of the populations from which the groups were sampled are equal.
- Alternative hypothesis (two-sided): The medians of the populations from which the groups were sampled are not all equal.

Interpretation

Significant results can be reported as "There was a significant difference in the median values among groups."

Post-hoc analysis allows you to say "The median for group A was higher than the median for group B", and so on.

Packages used in this chapter

The packages used in this chapter include:

- RVAideMemoire
- coin
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(coin)){install.packages("coin")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Example using the *RVAideMemoire* package

This example uses the formula notation indicating that *Likert* is the dependent variable and *Speaker* is the independent variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?mood.medtest*.

A significant *p*-value for Mood's median test indicates that not all medians among groups are equal.

For appropriate plots and summary statistics, see the *Kruskal-Wallis Test* chapter.

```
Input ="  
Speaker Likert  
Pooh    3  
Pooh    5  
Pooh    4  
Pooh    4  
Pooh    4  
Pooh    4  
Pooh    4  
Pooh    4  
Pooh    5  
Pooh    5  
Piglet   2  
Piglet   4  
Piglet   2  
Piglet   2  
Piglet   1  
Piglet   2  
Piglet   3  
Piglet   2  
Piglet   2  
Piglet   3  
Piglet   3  
Tigger   4  
Tigger   4  
Tigger   4  
Tigger   5  
Tigger   3  
Tigger   5  
Tigger   4  
Tigger   4  
Tigger   3  
")  
  
Data = read.table(textConnection(Input),header=TRUE)  
  
### Check the data frame  
  
library(psych)  
headTail(Data)  
str(Data)  
summary(Data)  
  
### Remove unnecessary objects  
rm(Input)
```

Mood's median test

```
library(RVAideMemoire)

mood.medtest(Likert ~ Speaker,
             data = Data,
             exact = FALSE)

Mood's median test

X-squared = 3.36, df = 2, p-value = 0.1864
```

An interesting thing happened with the result here. The test counts how many observations in each group are greater than the global median for all groups together, in this case 4. It then tests if there is a significant difference in this proportion among groups. For this data set, however, both Pooh and Tigger have a majority of observations equal to the global median. Because they are *equal* to the global median, they are not *greater than* the global median, and so aren't much different than Piglet's scores on this count. The result in this case is a non-significant *p*-value.

But the test would come out differently if we were counting observation *less than* the global median, because Pooh and Tigger have few of these, and Piglet has relatively many.

This is a quirk with Mood's median test, and isn't common in statistical tests.

One solution would be to re-code the function to count observations less than the global median.

But it is easier to simply invert the scale we are using. This is really an arbitrary change, but for this test, it can make a difference. Imagine if our original scale interpreted 5 to be the best, and 1 to be the worst. When we designed the survey tool, we could just as easily have made 1 the best and 5 the worst. And then instead of ranking "good" with a 4, the respondents would have marked it 2, and so on. By the way the calculations are done, this arbitrary change in scale will change the results of Mood's median test.

For a 5-point scale, we do this inversion by simply by making a new variable equal to 6 minus the original score.

With Mood's median test, I recommend making this kind of inversion in cases where many values are equal to the global median. Then use whichever result has a lower *p*-value.

```
Data$Likert.inv = 6 - Data$Likert

library(psych)

headTail(Data)

  Speaker Likert Likert.inv
1      Pooh      3          3
2      Pooh      5          1
3      Pooh      4          2
```

	Pooh		2
4	4		
...	<NA>
27	Tigger	5	1
28	Tigger	4	2
29	Tigger	4	2
30	Tigger	3	3

```
library(RVAideMemoire)

mood.medtest(Likert.inv ~ Speaker,
             data = Data,
             exact = FALSE)

Mood's median test

X-squared = 15.833, df = 2, p-value = 0.0003646

### Median test by Monte Carlo simulation

library(coin)

median_test(Likert.inv ~ Speaker,
            data = Data,
            distribution = approximate(B = 10000))

Approximative K-Sample Brown-Mood Median Test

chi-squared = 15.306, p-value = 1e-04
```

Post-hoc test: pairwiseMedianTest function

If Mood's median test is significant, a post-hoc analysis can be performed to determine which groups differ from each other group.

For this we will use the *pairwiseMedianTest* function in the *rcompanion* package, which conducts Mood's median test on all pairs of groups from one-way data.

Because the post-hoc test will produce multiple *p*-values, adjustments to the *p*-values can be made to avoid inflating the possibility of making a type-I error. There are a variety of methods for controlling the familywise error rate or for controlling the false discovery rate. See *?p.adjust* for details on these methods.

```
### Order groups by median

Data$Speaker = factor(Data$Speaker,
                      levels=c("Pooh", "Tigger", "Piglet"))

### Pairwise median tests

library(rcompanion)
```

```

PT = pairwiseMedianTest(Likert ~ Speaker,
                        data   = Data,
                        exact   = NULL,
                        method  = "fdr")
                        # Adjusts p-values for multiple comparisons;
                        # See ?p.adjust for options

PT

      Comparison  p.value p.adjust
1 Pooh - Tigger = 0      1 1.000000
2 Pooh - Piglet = 0 0.001093 0.003279
3 Tigger - Piglet = 0 0.005477 0.008216

### Compact letter display

library(rcompanion)

cldList(p.adjust ~ Comparison,
        data = PT,
        threshold = 0.05)

  Group Letter MonoLetter
1 Pooh     a         a
2 Tigger   a         a
3 Piglet   b         b

Groups sharing a letter are not significantly different (alpha = 0.05).

```

Post-hoc test: pairwiseMedianMatrix function

```

### Order groups by median

Data$Speaker = factor(Data$Speaker,
                      levels=c("Pooh", "Tigger", "Piglet"))

### Pairwise median tests

library(rcompanion)

PT = pairwiseMedianMatrix(Likert ~ Speaker,
                        data   = Data,
                        exact   = NULL,
                        method  = "fdr")
                        # Adjusts p-values for multiple comparisons;
                        # See ?p.adjust for options

PT

library(multcompView)

```

```
multcompLetters(PT$Adjusted,
  compare="<",
  threshold=0.05,
  Letters=letters)
```

Pooh	Tigger	Piglet
"a"	"a"	"b"

Friedman Test

The Friedman test determines if there are differences among groups for two-way data structured in a specific way, namely in an *unreplicated complete block design*. In this design, one variable serves as the *treatment* or *group* variable, and another variable serves as the *blocking* variable. It is the differences among treatments or groups that we are interested in. We aren't necessarily interested in differences among blocks, but we want our statistics to take into account differences in the blocks. In the unreplicated complete block design, each block has one and only one observation of each treatment.

For an example of this structure, look at the Belcher family data below. *Rater* is considered the blocking variable, and each rater has one observation for each *Instructor*. The test will determine if there are differences among values for *Instructor*, taking into account any consistent effect of a *Rater*. For example if *Rater a* rated consistently low and *Rater g* rated consistently high, the Friedman test can account for this statistically.

In other cases, the blocking variable might be the class where the ratings were done or the school where the ratings were done. If you were testing differences among curricula or other teaching treatments with different instructors, different instructors might be used as blocks.

If the distribution of the differences in scores between each pair of groups are all symmetrical, or if the distribution of values for each group is similar in shape and spread, the Friedman test determines if there is a difference in medians among groups. If not, the test determines if there is a systematic difference in the values among the groups.

Some people critique the Friedman test for having low power in detecting differences among groups. It has been suggested, however, that Friedman test may be powerful when there are five or more groups.

Post-hoc tests

The outcome of the Friedman test tells you if there are differences among the groups, but doesn't tell you *which* groups are different from other groups. In order to determine which groups are different from others, post-hoc testing can be conducted.

For a post-hoc analysis, the *pairwiseSignTest* function can be used. It performs a two-sample paired sign test on each pair of groups.

Appropriate data

- Two-way data arranged in an unreplicated complete block design
- Dependent variable is ordinal, interval, or ratio
- Treatment or group independent variable is a factor with two or more levels. That is, two or more groups
- Blocking variable is a factor with two or more levels
- Blocks are independent of each other and have no interaction with treatments
- In order to be a test of medians, the distribution of the differences in scores between each pair of groups are all symmetrical, or the distributions of values for each group have similar shape and spread. Otherwise the test is a test of distributions.

Hypotheses

If the distribution of the differences in scores between each pair of groups are all symmetrical, or the distributions of values for each group have similar shape and spread:

- Null hypothesis: The medians of values for each group are equal.
- Alternative hypothesis (two-sided): The medians of values for each group are not equal.

If the above conditions are not met:

- Null hypothesis: The distributions of values for each group are equal.
- Alternative hypothesis (two-sided): There is systematic difference in the distribution of values for the groups.

Interpretation

If the distribution of the differences in scores between each pair of groups are all symmetrical, or the distributions of values for each group have similar shape and spread:

Significant results can be reported as “There was a significant difference in median values across groups.”

Post-hoc analysis allows you to say “The median for group A was higher than the median for group B”, and so on.

If the above conditions are not met:

Significant results can be reported as “There was a significant difference in values among groups.”

Other notes and alternative tests

The Quade test is used for the same kinds of data and hypotheses, but can be more powerful in some cases. It has been suggested that Friedman test may be preferable when there are a larger number of groups (five or more), while the Quade is preferable for fewer groups. The Quade test is described in the next chapter.

Another alternative is to use cumulative link models for ordinal data, which are described later in this book.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- BSDA
- multcompView
- PMCMR
- rcompanion
- DescTools

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(BSDA)){install.packages("BSDA")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(PMCMR)){install.packages("PMCMR")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(DescTools)){install.packages("DescTools")}
```

Friedman test example

```
Input ="
Instructor      Rater Likert
'Bob Belcher'    a     4
'Bob Belcher'    b     5
'Bob Belcher'    c     4
'Bob Belcher'    d     6
'Bob Belcher'    e     6
'Bob Belcher'    f     6
'Bob Belcher'    g    10
'Bob Belcher'    h     6
'Linda Belcher'   a     8
'Linda Belcher'   b     6
'Linda Belcher'   c     8
'Linda Belcher'   d     8
'Linda Belcher'   e     8
'Linda Belcher'   f     7
'Linda Belcher'   g    10
'Linda Belcher'   h     9
'Tina Belcher'    a     7
'Tina Belcher'    b     5
'Tina Belcher'    c     7
'Tina Belcher'    d     8
'Tina Belcher'    e     8
'Tina Belcher'    f     9
'Tina Belcher'    g    10
'Tina Belcher'    h     9
'Gene Belcher'    a     6
'Gene Belcher'    b     4
'Gene Belcher'    c     5
```

```

'Gene Belcher'      d      5
'Gene Belcher'      e      6
'Gene Belcher'      f      6
'Gene Belcher'      g      5
'Gene Belcher'      h      5
'Louise Belcher'    a      8
'Louise Belcher'    b      7
'Louise Belcher'    c      8
'Louise Belcher'    d      8
'Louise Belcher'    e      9
'Louise Belcher'    f      9
'Louise Belcher'    g      8
'Louise Belcher'    h     10
")

Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                           levels=unique(Data$Instructor))

### Create a new variable which is the Likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                       ordered=TRUE)

### Check the data frame

library(psych)

headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects

rm(Input)

```

Summarize data treating Likert scores as factors

```

xtabs(~ Instructor + Likert.f,
      data = Data)

                    Likert.f
Instructor        4 5 6 7 8 9 10
  Bob Belcher     2 1 4 0 0 0  1
  Linda Belcher   0 0 1 1 4 1  1
  Tina Belcher    0 1 0 2 2 2  1
  Gene Belcher    1 4 3 0 0 0  0

```

```
Louise Belcher 0 0 0 1 4 2 1
```

```
XT = xtabs(~ Instructor + Likert.f,
           data = Data)

prop.table(XT,
           margin = 1)

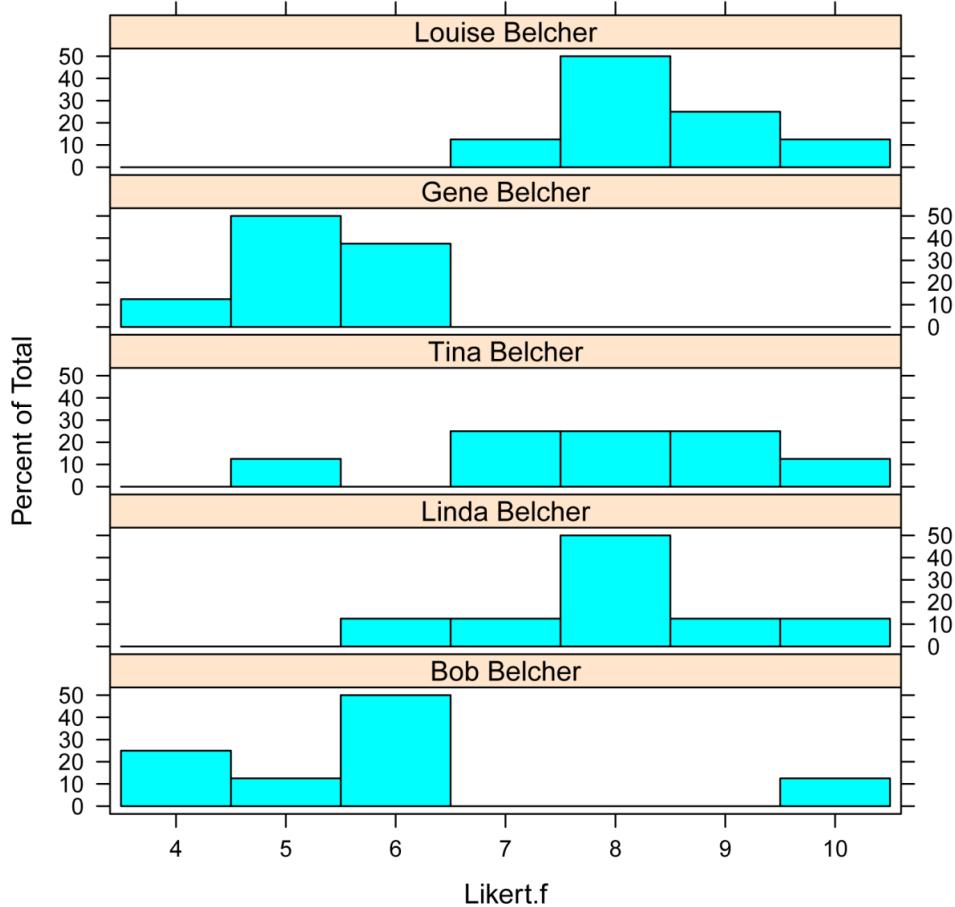
    Likert.f
Instructor      4      5      6      7      8      9      10
  Bob Belcher  0.250  0.125  0.500  0.000  0.000  0.000  0.125
  Linda Belcher 0.000  0.000  0.125  0.125  0.500  0.125  0.125
  Tina Belcher  0.000  0.125  0.000  0.250  0.250  0.250  0.125
  Gene Belcher  0.125  0.500  0.375  0.000  0.000  0.000  0.000
  Louise Belcher 0.000  0.000  0.000  0.125  0.500  0.250  0.125
```

Bar plots by group

Note that the bar plots don't show the effect of the blocking variable.

```
library(lattice)

histogram(~ Likert.f | Instructor,
          data=Data,
          layout=c(1,5)      # columns and rows of individual plots
)
```



Bar plots of differences between groups

We can make a bar plot of the differences in values for two groups, just as we did for the sign test previously. Values for Bob and Louisa were chosen based on their bar plots having very different shapes.

The resulting plot shows a distribution that is clearly not symmetrical.

Note that the data must be ordered by the blocking variable so that the first observation for *Louisa* will be paired with the first observation for *Bob*, and so on.

Also note that we had to specify the *levels* in the *factor* function defining *Diff.f*. This is so that the values with zero counts will be displayed on the plot.

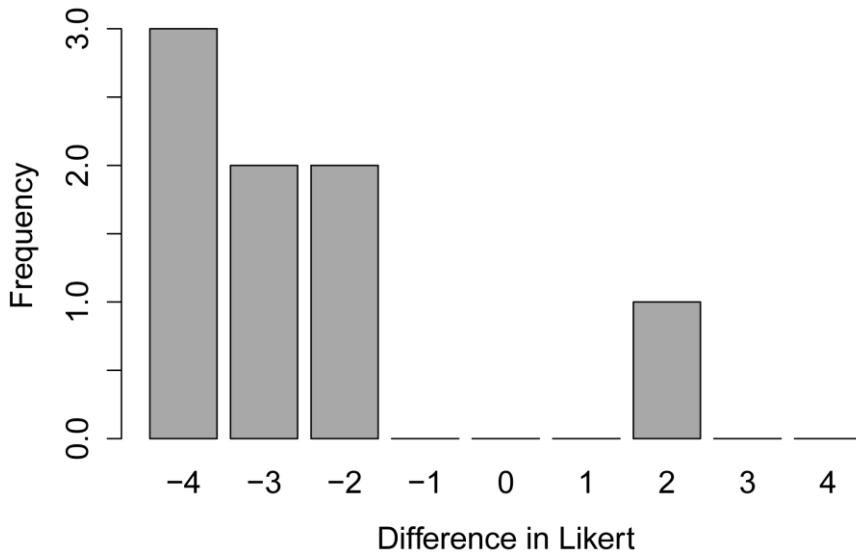
```

Bob = Data$Likert [Data$Instructor == "Bob Belcher"]
Louisa = Data$Likert [Data$Instructor == "Louise Belcher"]

Difference = Bob - Louisa

diff.f = factor(Difference,
                 ordered = TRUE,
                 levels = c("-4", "-3", "-2", "-1", "0", "1", "2", "3", "4"))
    
```

```
)
x = xtabs(~ diff.f)
barplot(x,
        col="dark gray",
        xlab="Difference in Likert",
        ylab="Frequency")
```



pairwiseDifferences to produce bar plots of differences between all groups

The *pairwiseDifferences* function in the *rcompanion* package will create a new data frame of differences for all pairs of differences. The *plotit=TRUE* option will produce bar plots of the counts for each pair. Otherwise the *lattice* package can be used to show these plots in one large trellis plot.

Note that the data must be ordered by the blocking variable so that the first observation for *Bob* will be paired with the first observation for *Linda*, and so on.

```
library(rcompanion)

Data.diff=pairwiseDifferences(Likert ~ Instructor,
                               data      = Data,
                               factorize = TRUE,
                               plotit    = TRUE)
```

```
library(psych)
```

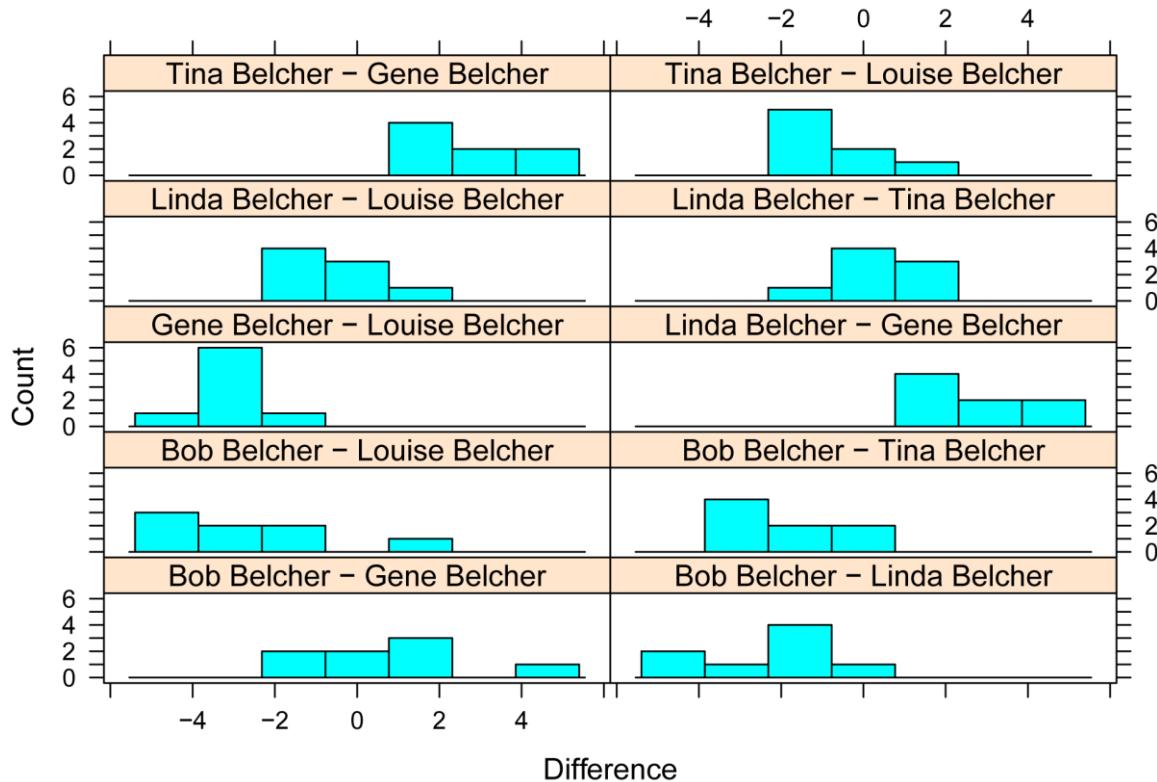
```
headTail(Data.diff)
```

	Comparison	Difference	Difference.f
1	Bob Belcher - Linda Belcher	-4	-4
2	Bob Belcher - Linda Belcher	-1	-1
3	Bob Belcher - Linda Belcher	-4	-4
4	Bob Belcher - Linda Belcher	-2	-2
...	<NA>	...	<NA>

77	Gene Belcher - Louise Belcher	-3	-3
78	Gene Belcher - Louise Belcher	-3	-3
79	Gene Belcher - Louise Belcher	-3	-3
80	Gene Belcher - Louise Belcher	-5	-5

```
library(lattice)
```

```
histogram(~ Difference | Comparison,
  data=Data.diff,
  type = "count",
  layout=c(2,5)      # columns and rows of individual plots
)
```



Summarize data treating Likert scores as numeric

```
library(FSA)
```

```
Summarize(Likert ~ Instructor,
  data=Data,
  digits=3)
```

	Instructor	n	mean	sd	min	Q1	median	Q3	max	percZero
1	Bob Belcher	8	5.875	1.885	4	4.75	6	6.00	10	0
2	Linda Belcher	8	8.000	1.195	6	7.75	8	8.25	10	0
3	Tina Belcher	8	7.875	1.553	5	7.00	8	9.00	10	0
4	Gene Belcher	8	5.250	0.707	4	5.00	5	6.00	6	0
5	Louise Belcher	8	8.375	0.916	7	8.00	8	9.00	10	0

Friedman test example

This example uses the formula notation indicating that *Likert* is the dependent variable, *Instructor* is the independent variable, and *Rater* is the blocking variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?friedman.test*

```
friedman.test(Likert ~ Instructor | Rater,
               data = Data)
```

Friedman rank sum test

Friedman chi-squared = 23.139, df = 4, p-value = 0.0001188

Effect size

Kendall's *W*, or Kendall's coefficient of concordance, can be used as an effect size statistic for Friedman's test.

The following interpretations are based on personal intuition. They are not intended to be universal.

		<u>small</u>	<u>medium</u>	<u>large</u>
Kendall's <i>W</i>	k = 3	< 0.10	0.10 – < 0.30	≥ 0.30
	k = 5	< 0.10	0.10 – < 0.25	≥ 0.25
	k = 7	< 0.10	0.10 – < 0.20	≥ 0.20
	k = 9	< 0.10	0.10 – < 0.20	≥ 0.20

```
XT = xtabs(Likert ~ Instructor + Rater,
            data = Data)
```

XT

Instructor	a	b	c	d	e	f	g	h
Bob Belcher	4	5	4	6	6	6	10	6
Linda Belcher	8	6	8	8	8	7	10	9
Tina Belcher	7	5	7	8	8	9	10	9
Gene Belcher	6	4	5	5	6	6	5	5
Louise Belcher	8	7	8	8	9	9	8	10

For the *KendallW* function, groups must be in rows, and raters must be in columns.

```
library(DescTools)
```

```
KendallW(XT,
          correct=TRUE,
          test=TRUE)
```

Kendall's coefficient of concordance wt

```
Kendall chi-squared = 23.139, df = 4, subjects = 5, raters = 8,
p-value = 0.0001188
```

```
sample estimates:
wt
0.7230903
```

In the output, check that the correct number of groups and raters is listed under "subjects" and "raters", respectively.

Post-hoc test: pairwise sign test for multiple comparisons of groups

Post-hoc testing can be conducted with the functions *pairwiseSignTest* and *pairwiseSignMatrix*. These functions conduct a paired two-sample sign test on each pair of groups, and output the results as either a table or a matrix, respectively. The matrix output can be converted to a compact letter display using the *multcompLetters* function in the *multcompView* package.

To prevent the inflation of type I error rates, adjustments to the *p*-values can be made using the *p.adjust.method* option. See *?p.adjust* for details on available *p*-value adjustment methods.

It has been suggested that the sign test may lack power in detecting differences in paired data sets. But is useful because it has few assumptions about the distributions of the data to compare, and is the test analogous to the Friedman test with two groups.

Table format and compact letter display

Note that the data must be ordered by the blocking variable so that the first observation for *Bob* will be paired with the first observation for *Linda*, and so on.

```
### Order groups by median

Data$Instructor = factor(Data$Instructor,
                         levels = c("Linda Belcher", "Louise Belcher",
                                    "Tina Belcher", "Bob Belcher",
                                    "Gene Belcher"))

### Pairwise sign tests

library(rcompanion)

PT = pairwiseSignTest(Likert ~ Instructor,
                      data   = Data,
                      method = "fdr")
# Adjusts p-values for multiple comparisons;
# See ?p.adjust for options

PT
```

	Comparison	S	p.value	p.adjust
1	Linda Belcher - Louise Belcher	= 0 1	0.375	0.46880
2	Linda Belcher - Tina Belcher	= 0 3	0.625	0.68750
3	Linda Belcher - Bob Belcher	= 0 7	0.01563	0.03908

```

4 Linda Belcher - Gene Belcher = 0 8 0.007812 0.02604
5 Louise Belcher - Tina Belcher = 0 5 0.2187 0.31240
6 Louise Belcher - Bob Belcher = 0 7 0.07031 0.11720
7 Louise Belcher - Gene Belcher = 0 8 0.007812 0.02604
8 Tina Belcher - Bob Belcher = 0 6 0.03125 0.06250
9 Tina Belcher - Gene Belcher = 0 8 0.007812 0.02604
10 Bob Belcher - Gene Belcher = 0 4 0.6875 0.68750

```

```
### Compact letter display
```

```
library(rcompanion)

cldList(p.adjust ~ Comparison,
        data = PT,
        threshold = 0.05)
```

	Group	Letter	MonoLetter
1	LindaBelcher	a	a
2	LouiseBelcher	ab	ab
3	TinaBelcher	ab	ab
4	BobBelcher	bc	bc
5	GeneBelcher	c	c

Groups sharing a letter are not significantly different (alpha = 0.05).

Matrix format and compact letter display

Note that the data must be ordered by the blocking variable so that the first observation for *Bob* will be paired with the first observation for *Linda*, and so on.

```
### Order groups by median
```

```
Data$Instructor = factor(Data$Instructor,
                         levels = c("Linda Belcher", "Louise Belcher",
                                   "Tina Belcher", "Bob Belcher",
                                   "Gene Belcher"))
```

```
### Pairwise sign tests
```

```
library(rcompanion)
```

```
PM = pairwiseSignMatrix(Likert ~ Instructor,
                        data = Data,
                        method = "fdr")
# Adjusts p-values for multiple comparisons;
# See ?p.adjust for options
```

```
PM
```

	Linda Belcher	Louise Belcher	Tina Belcher	Bob Belcher	Gene Belcher
Linda Belcher	1.00000	0.46880	0.68750	0.03908	0.02604

Louise Belcher	0.46880	1.00000	0.31240	0.11720	0.02604
Tina Belcher	0.68750	0.31240	1.00000	0.06250	0.02604
Bob Belcher	0.03908	0.11720	0.06250	1.00000	0.68750
Gene Belcher	0.02604	0.02604	0.02604	0.68750	1.00000

```
library(multcompView)

multcompLetters(PM$Adjusted,
                compare="<",
                threshold=0.05, # p-value to use as significance threshold
                Letters=letters,
                reversed = FALSE)

Linda Belcher Louise Belcher Tina Belcher Bob Belcher Gene Belcher
      "a"           "ab"       "ab"       "bc"       "c"

### Groups sharing a letter are not significantly different.
```

Post-hoc Conover test

A different post-hoc test for the Friedman test can be conducted with the *posthoc.friedman.conover.test* function in the *PMCMR* package. The output can be converted to a compact letter display using the *multcompLetters* function in the *multcompView* package.

```
### Order groups by median

Data$Instructor = factor(Data$Instructor,
                          levels = c("Linda Belcher", "Louise Belcher",
                                    "Tina Belcher", "Bob Belcher",
                                    "Gene Belcher"))

### Conover test

library(PMCMR)

PT = posthoc.friedman.conover.test(y      = Data$Likert,
                                    groups = Data$Instructor,
                                    blocks = Data$Rater,
                                    p.adjust.method="fdr")
# Adjusts p-values for multiple comparisons;
# See ?p.adjust for options
```

PT

Pairwise comparisons using Conover's test for a two-way
balanced complete block design

	Linda Belcher	Louise Belcher	Tina Belcher	Bob Belcher
Louise Belcher	0.27303	-	-	-
Tina Belcher	0.31821	0.05154	-	-
Bob Belcher	2.8e-05	1.9e-06	0.00037	-
Gene Belcher	1.2e-06	1.1e-07	8.8e-06	0.17328

```

P value adjustment method: fdr

### Compact letter display

PT0 = as.matrix(PT$p.value)

library(rcompanion)

PT1 = fullPTable(PT0)

library(multcompView)

multcompLetters(PT1,
  compare="<",
  threshold=0.05,
  Letters=letters,
  reversed = FALSE)

Linda Belcher Louise Belcher Tina Belcher Bob Belcher Gene Belcher
      "a"           "a"       "a"       "b"       "b"

```

Quade Test

The Quade test is used for similar data and hypotheses as the Friedman test, namely for unreplicated complete block designs.

While the Friedman test is a generalization of the paired sign test, the Quade test is a generalization of the two-sample signed-rank test.

Some authors indicate that the Quade test is appropriate for ordinal data, while others suggest that it is appropriate only for interval or ratio data. It is included in this section as an alternative to the Friedman test.

Post-hoc tests

The outcome of the Quade test tells you if there are differences among the groups, but doesn't tell you *which* groups are different from other groups. In order to determine which groups are different from others, post-hoc testing can be conducted with the *pairwise.wilcox.test* function.

Appropriate data

- Two-way data arranged in an unreplicated complete block design
- Dependent variable is ordinal, interval, or ratio, although some authors say data must be interval or ratio only.

- Treatment or group independent variable is a factor with two or more levels. That is, two or more groups
- Blocking variable is a factor with two or more levels
- Blocks are independent of each other and have no interaction with treatments
- In order to be a test of medians, the distributions of values for each group should have similar shape and spread. If not, the test determines if there is a systematic difference in the values among the groups.

Hypotheses

If the distribution of the differences in scores between each pair of groups are all symmetrical, or the distributions of values for each group have similar shape and spread:

- Null hypothesis: The medians of values for each group are equal.
- Alternative hypothesis (two-sided): The medians of values for each group are not equal.

If the above conditions are not met:

- Null hypothesis: The distributions of values for each group are equal.
- Alternative hypothesis (two-sided): There is systematic difference in the distribution of values for the groups.

Interpretation

If the distribution of the differences in scores between each pair of groups are all symmetrical, or the distributions of values for each group have similar shape and spread:

Significant results can be reported as “There was a significant difference in median values across groups.”

Post-hoc analysis allows you to say “The median for group A was higher than the median for group B”, and so on.

If the above conditions are not met:

Significant results can be reported as “There was a significant difference in values among groups.”

Other notes and alternative tests

The Friedman test is used for the same kinds of data and hypotheses, The Friedman test is described in the previous chapter.

Another alternative is to use cumulative link models for ordinal data, which are described later in this book.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- multcompView
- PMCMR

- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(PMCMR)){install.packages("PMCMR")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Quade test example

Instructor	Rater	Likert
'Bob Belcher'	a	4
'Bob Belcher'	b	5
'Bob Belcher'	c	4
'Bob Belcher'	d	6
'Bob Belcher'	e	6
'Bob Belcher'	f	6
'Bob Belcher'	g	10
'Bob Belcher'	h	6
'Linda Belcher'	a	8
'Linda Belcher'	b	6
'Linda Belcher'	c	8
'Linda Belcher'	d	8
'Linda Belcher'	e	8
'Linda Belcher'	f	7
'Linda Belcher'	g	10
'Linda Belcher'	h	9
'Tina Belcher'	a	7
'Tina Belcher'	b	5
'Tina Belcher'	c	7
'Tina Belcher'	d	8
'Tina Belcher'	e	8
'Tina Belcher'	f	9
'Tina Belcher'	g	10
'Tina Belcher'	h	9
'Gene Belcher'	a	6
'Gene Belcher'	b	4
'Gene Belcher'	c	5
'Gene Belcher'	d	5
'Gene Belcher'	e	6
'Gene Belcher'	f	6
'Gene Belcher'	g	5
'Gene Belcher'	h	5
'Louise Belcher'	a	8
'Louise Belcher'	b	7
'Louise Belcher'	c	8
'Louise Belcher'	d	8
'Louise Belcher'	e	9
'Louise Belcher'	f	9

```

'Louise Belcher'      g      8
'Louise Belcher'      h     10
")

Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                           levels=unique(Data$Instructor))

### Create a new variable which is the likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                           ordered=TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Summarize data treating Likert scores as factors

```

xtabs( ~ Instructor + Likert.f,
       data = Data)

          Likert.f
Instructor    4 5 6 7 8 9 10
  Bob Belcher  2 1 4 0 0 0  1
  Linda Belcher 0 0 1 1 4 1  1
  Tina Belcher 0 1 0 2 2 2  1
  Gene Belcher 1 4 3 0 0 0  0
  Louise Belcher 0 0 0 1 4 2  1

XT = xtabs( ~ Instructor + Likert.f,
            data = Data)

prop.table(XT,
           margin = 1)

          Likert.f
Instructor    4      5      6      7      8      9      10

```

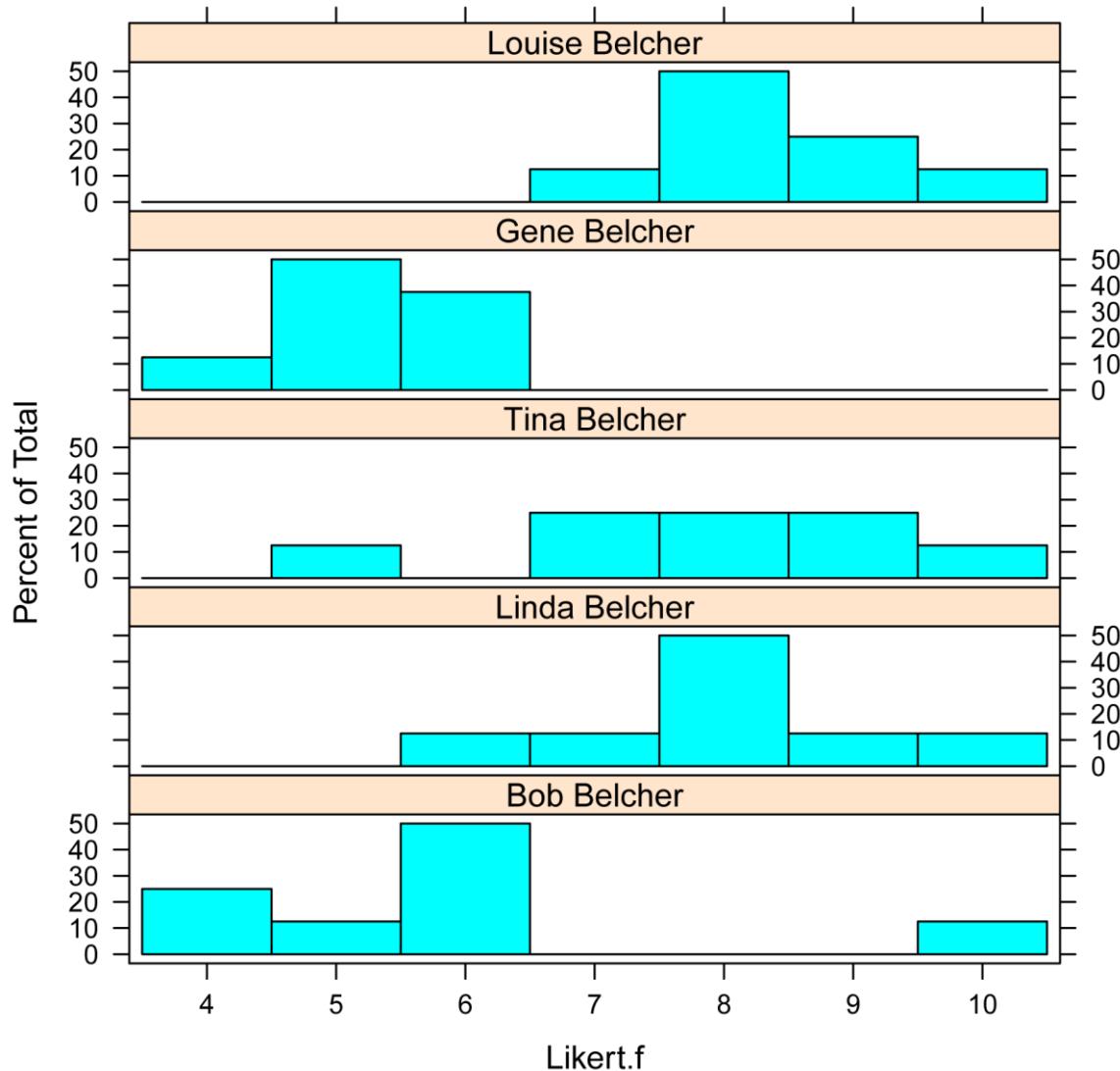
Bob Belcher	0.250	0.125	0.500	0.000	0.000	0.000	0.125
Linda Belcher	0.000	0.000	0.125	0.125	0.500	0.125	0.125
Tina Belcher	0.000	0.125	0.000	0.250	0.250	0.250	0.125
Gene Belcher	0.125	0.500	0.375	0.000	0.000	0.000	0.000
Louise Belcher	0.000	0.000	0.000	0.125	0.500	0.250	0.125

Bar plots by group

Note that the bar plots don't show the effect of the blocking variable.

```
library(lattice)
```

```
histogram(~ Likert.f | Instructor,
          data=Data,
          layout=c(1,5)      # columns and rows of individual plots
        )
```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Instructor,
          data=Data,
          digits=3)

  Instructor n  mean    sd min   Q1 median   Q3 max percZero
1 Bob Belcher 8 5.875 1.885  4 4.75     6 6.00  10      0
2 Linda Belcher 8 8.000 1.195  6 7.75     8 8.25  10      0
3 Tina Belcher 8 7.875 1.553  5 7.00     8 9.00  10      0
4 Gene Belcher 8 5.250 0.707  4 5.00     5 6.00  6       0
5 Louise Belcher 8 8.375 0.916  7 8.00     8 9.00  10      0
```

Quade test example

This example uses the formula notation indicating that *Likert* is the dependent variable, *Instructor* is the independent variable, and *Rater* is the blocking variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?quade.test*.

```
quade.test(Likert ~ Instructor | Rater,
           data = Data)

Quade test

Quade F = 8.0253, num df = 4, denom df = 28, p-value = 0.0001924
```

Post-hoc test: pairwise two-sample paired rank-sum test for multiple comparisons

To prevent the inflation of type I error rates, adjustments to the *p*-values can be made using the *p.adjust.method* option. See *?p.adjust* for details on available *p*-value adjustment methods.

Note that the data must be ordered by the blocking variable so that the first observation for *Bob* will be paired with the first observation for *Linda*, and so on.

```
### Order groups by median

Data$Instructor = factor(Data$Instructor,
                          levels=c("Linda Belcher", "Louise Belcher",
                                   "Tina Belcher", "Bob Belcher",
                                   "Gene Belcher"))

Data

### Pairwise Mann-Whitney

PT = pairwise.wilcox.test(Data$Likert,
                           Data$Instructor,
                           p.adjust.method="fdr",
                           paired=TRUE)$p.value
                           # Adjusts p-values for multiple comparisons;
```

```
# See ?p.adjust for options
```

```
PT
```

```
### Convert PT to a full table and call it PT1
```

```
library(rcompanion)
```

```
PT1 = fullPTable(PT)
```

```
PT1
```

	Linda Belcher	Louise Belcher	Tina Belcher	Bob Belcher	Gene Belcher
Linda Belcher	1.00000000	0.61269120	0.85010674	0.05090808	0.04637053
Louise Belcher	0.61269120	1.00000000	0.47198578	0.05090808	0.04637053
Tina Belcher	0.85010674	0.47198578	1.00000000	0.05090808	0.04637053
Bob Belcher	0.05090808	0.05090808	0.05090808	1.00000000	0.65530421
Gene Belcher	0.04637053	0.04637053	0.04637053	0.65530421	1.00000000

```
# Produce compact letter display
```

```
library(multcompView)
```

```
multcompLetters(PT1,
  compare="<",
  threshold=0.05, # p-value to use as significance threshold
  Letters=letters,
  reversed = FALSE)
```

Linda Belcher	Louise Belcher	Tina Belcher	Bob Belcher	Gene Belcher
"a"	"a"	"a"	"ab"	"b"

```
### values sharing a letter are not significantly different
```

Post-hoc test: Quade post-hoc test

A different post-hoc test for the Quade test can be conducted with the *posthoc.quade.test* function in the *PMCMR* package. The output can be converted to a compact letter display using the *multcompLetters* function in the *multcompView* package.

```
### Order groups by median
```

```
Data$Instructor = factor(Data$Instructor,
  levels=c("Linda Belcher", "Louise Belcher",
  "Tina Belcher", "Bob Belcher",
  "Gene Belcher"))
```

```
Data
```

```
### Post-hoc test
```

```

library(PMCMR)

PT = posthoc.quade.test(y      = Data$Likert,
                        groups = Data$Instructor,
                        blocks = Data$Rater,
                        p.adjust.method="fdr")
                         # Adjusts p-values for multiple comparisons;
                         # See ?p.adjust for options

PT

Pairwise comparisons using posthoc-Quade test with TDist approximation

Linda Belcher Louise Belcher Tina Belcher Bob Belcher
Louise Belcher 0.00902   -        -        -
Tina Belcher   0.04533   0.50547   -        -
Bob Belcher    0.36572   0.00108   0.00539   -
Gene Belcher   0.00544   0.77620   0.38132   0.00099

P value adjustment method: fdr

### Compact letter display

PT0 = as.matrix(PT$p.value)

library(rcompanion)

PT1 = fullPTable(PT0)

library(multcompView)
multcompLetters(PT1,
                compare="<",
                threshold=0.05,
                Letters=letters,
                reversed = FALSE)

Linda Belcher Louise Belcher Tina Belcher Bob Belcher Gene Belcher
      "a"           "b"          "b"       "a"           "b"

```

Scheirer-Ray-Hare Test

The Scheirer-Ray-Hare test is a nonparametric test used for a two-way factorial design. It appears to be not well documented nor well-regarded. In my experience, the Scheirer-Ray-Hare test is less likely to find the interaction effect significant than would an ordinary least squares analysis of variance. Because of this, I recommend not using this test, but instead using aligned ranks transformation ANOVA as long as the dependent variable can be treated as interval.

It has been suggested that the observations should be balanced and that each cell in the interaction should have at least five observations.

Appropriate data

- Two-way data arranged in a factorial design
- Dependent variable is ordinal, interval, or ratio
- There are two treatment or group independent variables. Each is a factor with two or more levels
- Observations are independent. That is, they are not paired or repeated measures

Post-hoc tests

Appropriate post-hoc tests might be Dunn test or pairwise Mann-Whitney tests for each significant factor or interaction

Packages used in this chapter

The packages used in this chapter include:

- rcompanion
- FSA

The following commands will install these packages if they are not already installed:

```
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(FSA)){install.packages("FSA")}
```

Scheirer-Ray-Hare test examples***Midichlorians example***

```
### Assemble the data

Location = c(rep("Olympia" , 6), rep("Ventura", 6),
            rep("Northampton", 6), rep("Burlington", 6))

Tribe = c(rep(c("Jedi", "Sith"), 12))

Midichlorians = c(10, 4, 12, 5, 15, 4, 15, 9, 15, 11, 18, 12,
                 8, 13, 8, 15, 10, 17, 22, 22, 20, 22, 20, 25)

Data = data.frame(Tribe, Location, Midichlorians)

str(Data)

### Scheirer-Ray-Hare test

library(rcompanion)

scheirerRayHare(Midichlorians ~ Tribe + Location,
                data = Data)

DV: Midichlorians
Observations: 24
D: 0.9917391
MS total: 50

          Df Sum Sq      H p.value
Tribe       1   8.17  0.1647 0.68487
Location     3 746.58 15.0560 0.00177
Tribe:Location 3 315.58  6.3642 0.09517
Residuals    16  70.17
```

```

### Post-hoc test

### Order groups by median

Data$Location = factor(Data$Location,
                       levels=c("Burlington", "Ventura", "Northampton", "Olympia"))

levels(Data$Location)

### Dunn test

library(FSA)

DT = dunnTest(Midichlorians ~ Location,
              data=Data,
              method="bh")      # Adjusts p-values for multiple comparisons;
                     # See ?dunnTest for options

DT

### Compact letter display

PT = DT$res

PT

library(rcompanion)

cldList(P.adj ~ Comparison,
        data = PT,
        threshold = 0.05)

      Group Letter MonoLetter
1  Burlington     a       a
2 Northampton     b       b
3    Olympia     b       b
4    Ventura     b       b

Groups sharing a letter not significantly different (alpha = 0.05).

```

Example from Sokal and Rohlf

```

### Assemble the data

value = c(709,679,699,657,594,677,592,538,476,508,505,539)
Sex   = c(rep("Male",3), rep("Female",3), rep("Male",3), rep("Female",3))
Fat   = c(rep("Fresh", 6), rep("Rancid", 6))

Sokal = data.frame(value, Sex, Fat)

```

```

str(Sokal)

### Scheirer-Ray-Hare test

library(rcompanion)

scheirerRayHare(Value ~ Sex + Fat,
                 data=Sokal)

DV: Value
Observations: 12
D: 1
MS total: 13

      Df  Sum Sq   H p.value
Sex       1  8.333 0.6410 0.42334
Fat       1 108.000 8.3077 0.00395
Sex:Fat   1  5.333 0.4103 0.52184
Residuals 8  21.333

```

Example from Real Statistics Using Excel

This example from www.real-statistics.com/two-way-anova/scheirer-ray-hare-test/.

```

### Assemble the data

wheat = c(123,156,112,100,168,135,130,176,120,155,156,180,147,146,193)
Corn = c(128,150,174,116,109,175,132,120,187,184,186,138,178,176,190)
Soy = c(166,178,187,153,195,140,145,159,131,126,185,206,188,165,188)
Rice = c(151,125,117,155,158,167,183,142,167,168,175,173,154,191,169)
Yield = c(wheat, Corn, Soy, Rice)

Fert = rep(c(rep("Blend X",5), rep("Blend Y",5), rep("Blend Z",5)),4)
Crop = c(rep("Wheat",15), rep("Corn",15), rep("Soy",15), rep("Rice",15))

Real.stats = data.frame(Yield, Fert, Crop)

str(Real.stats)

### Scheirer-Ray-Hare test

library(rcompanion)

scheirerRayHare(Yield ~ Fert + Crop,
                 data=Real.stats)

DV: Yield
Observations: 60
D: 0.9997221
MS total: 305

```

```

Df Sum Sq      H p.value
Fert      2 4004.4 13.1329 0.001407
Crop      3 1339.0  4.3915 0.222175
Fert:Crop 6 2893.6  9.4900 0.147839
Residuals 48 9752.9

### Post-hoc test

### Order groups by median

Real.stats$Fert = factor(Real.stats$Fert,
                         levels=c("Blend Z", "Blend Y", "Blend X"))

levels(Real.stats$Fert)

### Dunn test

library(FSA)

DT = dunnTest(Yield ~ Fert,
               data=Real.stats,
               method="bh")      # Adjusts p-values for multiple comparisons;
                      # See ?dunnTest for options

DT

### Compact letter display

PT = DT$res

PT

library(rcompanion)

cldList(P.adj ~ Comparison,
        data = PT,
        threshold = 0.05)

  Group Letter MonoLetter
1 BlendX   a       a
2 BlendY   a       a
3 BlendZ   b       b

Groups sharing a letter not significantly different (alpha = 0.05).

```

References

Sokal, R.R. and F.J. Rohlf. 1995. *Biometry*, 3rd ed. W.H. Freeman. New York.

Aligned Ranks Transformation ANOVA

Introduction

Aligned ranks transformation ANOVA (ART anova) is a nonparametric approach that allows for multiple independent variables, interactions, and repeated measures.

My understanding is that, since the aligning process requires subtracting values, the dependent variable needs to be interval in nature. That is, strictly ordinal data would be treated as numeric in the process.

The package *ARTool* makes using this approach in R relatively easy.

A few notes on using *ARTool*:

- All independent variables must be nominal
- All interactions of fixed independent variables need to be included in the model
- Post-hoc comparisons can be conducted for main effects
- For interactions effects, post-hoc comparisons can be conducted for two-way models. For more complex models, difference-in-difference post-hoc comparisons can be conducted.
- For fixed-effects models, *eta*-squared can be calculated as an effect size

Packages used in this chapter

The packages used in this chapter include:

- ARTool
- emmeans
- rcompanion
- ggplot2
- psych

The following commands will install these packages if they are not already installed:

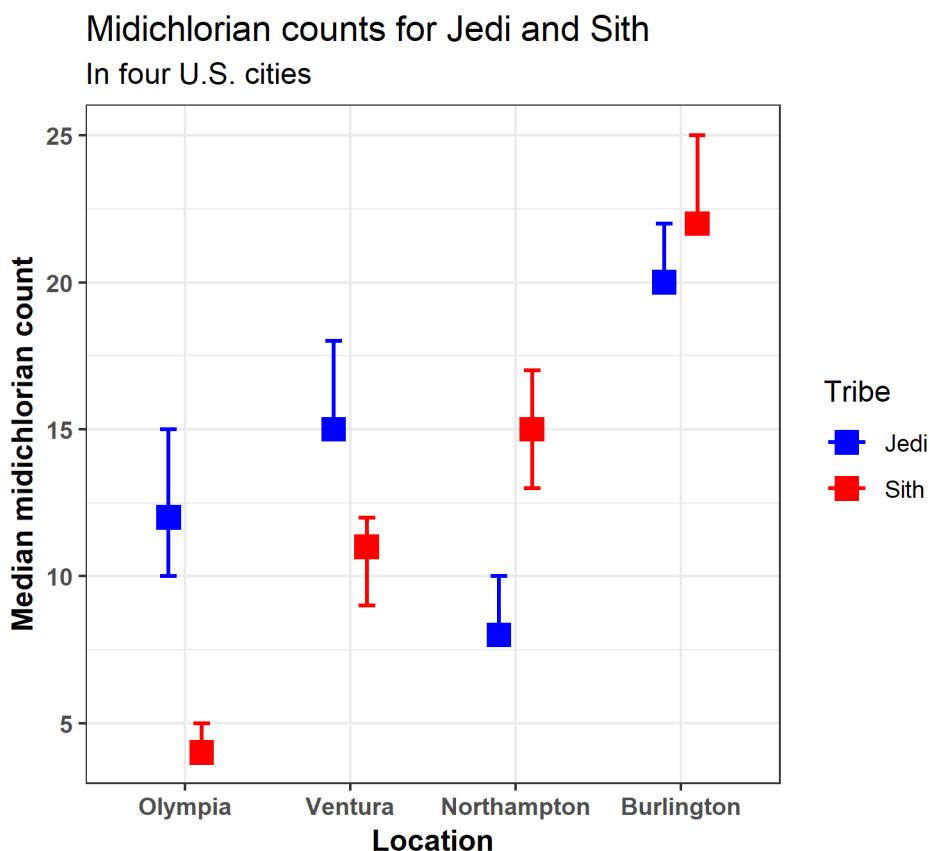
```
if(!require(ARTool)){install.packages("ARTool")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(rcompanion)){install.packages("rcompanion ")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(psych)){install.packages("psych")}
```

Aligned Ranks Transformation ANOVA examples

Midichlorians example

This example reproduces the data used in the *Scheirer–Ray–Hare Test* chapter. Note that the aligned ranks anova finds a significant interaction, where the Scheirer–Ray–Hare test failed to detect this. Also note that the results are similar to those from a standard anova in the *Estimated Marginal Means for Multiple Comparisons*.

Note that code for producing the plot is found at the end of the chapter.



```
### Assemble the data
```

```
Location = c(rep("Olympia", 6), rep("Ventura", 6),
            rep("Northampton", 6), rep("Burlington", 6))

Tribe = c(rep(c("Jedi", "Sith"), 12))

Midichlorians = c(10, 4, 12, 5, 15, 4, 15, 9, 15, 11, 18, 12,
                 8, 13, 8, 15, 10, 17, 22, 22, 20, 22, 20, 25)

Data = data.frame(Tribe, Location, Midichlorians)
```

```

str(Data)

### Aligned ranks anova

library(ARTool)

model = art(Midichlorians ~ Tribe + Location + Tribe:Location,
            data = Data)

### Check the success of the procedure

model

Aligned Rank Transform of Factorial Model

Call:
art(formula = Midichlorians ~ Tribe + Location + Tribe:Location,
     data = Data)

Column sums of aligned responses (should all be ~0):
  Tribe          Location Tribe:Location
    0              0             0

### Conduct ANOVA

anova(model)

Analysis of Variance of Aligned Rank Transformed Data

Table Type: Anova Table (Type III tests)
Model: No Repeated Measures (lm)
Response: art(Midichlorians)

  Df Df.res F value   Pr(>F)
1 Tribe      1    16 3.0606  0.099364 .
2 Location   3    16 34.6201 3.1598e-07 ***
3 Tribe:Location 3    16 29.9354 8.4929e-07 ***

```

Post-hoc comparisons for main effects

It is my understanding that post-hoc comparisons for main effects can be handled by the *emmeans* package in the usual way, except that the *artlm* function must be used to first fit a model that can be passed to *emmeans*.

Estimate values in the *emmeans* output should be ignored.

```

model.lm = artlm(model, "Location")

library(emmeans)

marginal = emmeans(model.lm,

```

~ Location)

```
pairs(marginal,
      adjust = "tukey")

contrast           estimate   SE df t.ratio p.value
Burlington - Northampton    10.83 1.78 16  6.075  0.0001
Burlington - Olympia        17.83 1.78 16 10.000 <.0001
Burlington - Ventura       7.33 1.78 16  4.112  0.0041
Northampton - Olympia       7.00 1.78 16  3.925  0.0060
Northampton - Ventura      -3.50 1.78 16 -1.963  0.2426
Olympia - Ventura          -10.50 1.78 16 -5.888  0.0001
```

Results are averaged over the levels of: Tribe
P value adjustment: tukey method for comparing a family of 4 estimates

```
cld(marginal,
     alpha=0.05,
     Letters=letters,
     adjust="tukey")

Location    emmean   SE df lower.CL upper.CL .group
Olympia      3.67 1.26 16    0.131     7.2    a
Northampton  10.67 1.26 16    7.131    14.2    b
Ventura      14.17 1.26 16   10.631    17.7    b
Burlington   21.50 1.26 16   17.964    25.0    c
```

Results are averaged over the levels of: Tribe
Confidence level used: 0.95
Conf-level adjustment: sidak method for 4 estimates
P value adjustment: tukey method for comparing a family of 4 estimates
significance level used: alpha = 0.05

Post-hoc comparisons for interactions in a two-way model

It is my understanding that for a two-way interaction, the following code can be used for post-hoc comparisons, but that this approach may not be generalizable to other designs. Note that the *cld* function cannot be used here.

Estimate values in the *emmeans* output should be ignored.

```
model.int = artlm(model, "Tribe:Location")
marginal = emmeans(model.int, ~ Tribe:Location)
contrast(marginal, method="pairwise", adjust="none")
### For Tukey-adjusted p-values, use adjust="tukey"

### Here, results truncated to comparisons within each Location
contrast           estimate   SE df t.ratio p.value
```

Jedi, Burlington - Sith, Burlington	-6.500	2.68	16	-2.428	0.0273
Jedi, Northampton - Sith, Northampton	-16.833	2.68	16	-6.288	<.0001
Jedi, Olympia - Sith, Olympia	15.000	2.68	16	5.603	<.0001
Jedi, Ventura - Sith, Ventura	9.667	2.68	16	3.611	0.0023

Post-hoc comparisons for other interactions

It is my understanding that for interactions in general, the following code for difference-in-difference post-hoc comparisons can be used. These sorts of comparisons may not be as useful as other types of comparisons.

Estimate values in the *emmeans* output should be ignored.

```
model.diff = aovlrm(model, "Tribe:Location")
marginal = emmeans(model.diff, ~Tribe:Location)
contrast(marginal, method="pairwise", interaction=TRUE)

Tribe_pairwise Location_pairwise estimate SE df t.ratio p.value
Jedi - Sith    Burlington - Northampton 10.33 3.79 16 2.729 0.0148
Jedi - Sith    Burlington - Olympia   -21.50 3.79 16 -5.679 <.0001
Jedi - Sith    Burlington - Ventura  -16.17 3.79 16 -4.270 0.0006
Jedi - Sith    Northampton - Olympia -31.83 3.79 16 -8.408 <.0001
Jedi - Sith    Northampton - Ventura -26.50 3.79 16 -7.000 <.0001
Jedi - Sith    Olympia - Ventura     5.33 3.79 16 1.409 0.1781
```

Here, the comparison with Olympia and Ventura doesn't indicate that the values for the two cities are similar, but rather that the *difference* between Jedi and Sith within each of the two cities is similar. This is consistent with the plot.

Partial eta-squared

Partial *eta*-squared can be calculated as an effect size statistic for aligned ranks transformation anova.

Interpretation of eta-squared

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment, but for behavioral studies, the guidelines proposed by Cohen (1988) are sometimes followed. They should not be considered universal.

	<u>Small</u>	<u>Medium</u>	<u>Large</u>
eta-squared	0.01 - < 0.06	0.06 - < 0.14	≥ 0.14

Source: Cohen (1988).

```
Result = anova(model)
Result$part.eta.sq = with(Result, `Sum Sq`/(`Sum Sq` + `Sum Sq.res`))
```

Result

Analysis of Variance of Aligned Rank Transformed Data

	Df	Df.res	F value	Pr(>F)	part.eta.sq.
1 Tribe	1	16	3.0606	0.099364	0.16057 .
2 Location	3	16	34.6201	3.1598e-07	0.86651 ***
3 Tribe:Location	3	16	29.9354	8.4929e-07	0.84878 ***

One-way example

The following example addresses the data from the *Kruskal–Wallis Test* chapter. Results are relatively similar to results from the Kruskal–Wallis and Dunn tests, and to those from ordinal regression. Here, the *p*-value for the global test by ART anova is lower than that from the Kruskal–Wallis test.

```
Input ="
Speaker Likert
Pooh    3
Pooh    5
Pooh    4
Pooh    4
Pooh    4
Pooh    4
Pooh    4
Pooh    4
Pooh    5
Pooh    5
Pooh    5
Piglet   2
Piglet   4
Piglet   2
Piglet   2
Piglet   1
Piglet   2
Piglet   3
Piglet   2
Piglet   2
Piglet   3
Piglet   3
Tigger   4
Tigger   4
Tigger   4
Tigger   4
Tigger   5
Tigger   3
Tigger   5
Tigger   4
Tigger   4
Tigger   3
")
Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them
```

```

Data$Speaker = factor(Data$Speaker,
                      levels=unique(Data$Speaker))

### Create a new variable which is the likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                       ordered = TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

### Aligned ranks anova

library(ARTool)

model = art(Likert ~ Speaker,
            data = Data)

anova(model)

Analysis of Variance of Aligned Rank Transformed Data

Table Type: Anova Table (Type III tests)
Model: No Repeated Measures (lm)
Response: art(Likert)

      Df Df.res F value     Pr(>F)
1 Speaker  2      27 18.702 8.0005e-06 ***

### Post-hoc comparisons

model.lm = artlm(model, "Speaker")

library(emmeans)

marginal = emmeans(model.lm,
                    ~ Speaker)

```

```

pairs(marginal,
      adjust = "tukey")

contrast      estimate   SE df t.ratio p.value
Pooh - Piglet     14.1 2.51 27  5.619 <.0001
Pooh - Tigger      1.8 2.51 27  0.717  0.7555
Piglet - Tigger    -12.3 2.51 27 -4.901  0.0001

P value adjustment: tukey method for comparing a family of 3 estimates

cld(marginal,
     alpha=0.05,
     Letters=letters,
     adjust="tukey")

Speaker emmean   SE df lower.CL upper.CL .group
Piglet      6.7 1.77 27     2.18     11.2   a
Tigger     19.0 1.77 27    14.48     23.5   b
Pooh       20.8 1.77 27    16.28     25.3   b

Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05

### Partial eta squared

Result = anova(model)

Result$part.eta.sq = with(Result, `Sum Sq`/(`Sum Sq` + `Sum Sq.res`))

Result

Analysis of Variance of Aligned Rank Transformed Data

  Df Df.res F value    Pr(>F) part.eta.sq
1 Speaker  2     27 18.702 8.0005e-06    0.58077 ***

```

Repeated measures example

The following example addresses the data from the *Friedman Test* chapter. Results are relatively similar to results from the Friedman and Conover tests, and to those from ordinal regression. Here, the *p*-value for the global test by ART anova is lower than that from the Friedman test.

```

Input =""
Instructor      Rater Likert
'Bob Belcher'    a     4
'Bob Belcher'    b     5
'Bob Belcher'    c     4
'Bob Belcher'    d     6

```

```
'Bob Belcher'      e    6
'Bob Belcher'      f    6
'Bob Belcher'      g   10
'Bob Belcher'      h    6
'Linda Belcher'    a    8
'Linda Belcher'    b    6
'Linda Belcher'    c    8
'Linda Belcher'    d    8
'Linda Belcher'    e    8
'Linda Belcher'    f    7
'Linda Belcher'    g   10
'Linda Belcher'    h    9
'Tina Belcher'     a    7
'Tina Belcher'     b    5
'Tina Belcher'     c    7
'Tina Belcher'     d    8
'Tina Belcher'     e    8
'Tina Belcher'     f    9
'Tina Belcher'     g   10
'Tina Belcher'     h    9
'Gene Belcher'     a    6
'Gene Belcher'     b    4
'Gene Belcher'     c    5
'Gene Belcher'     d    5
'Gene Belcher'     e    6
'Gene Belcher'     f    6
'Gene Belcher'     g    5
'Gene Belcher'     h    5
'Louise Belcher'   a    8
'Louise Belcher'   b    7
'Louise Belcher'   c    8
'Louise Belcher'   d    8
'Louise Belcher'   e    9
'Louise Belcher'   f    9
'Louise Belcher'   g    8
'Louise Belcher'   h   10
")
```

```
Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))

### Create a new variable which is the likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                      ordered=TRUE)

### Check the data frame

library(psych)
```

```

headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

### Aligned ranks anova
library(ARTool)

model = art(Likert ~ Instructor + (1|Rater),
            data = Data)

anova(model)

Analysis of Variance of Aligned Rank Transformed Data

Table Type: Analysis of Deviance Table (Type III Wald F tests with Kenward-Roger
df)
Model: Mixed Effects (lmer)
Response: art(Likert)

      F  DF DF.res      Pr(>F)
1 Instructor 16.052  4      28 6.0942e-07 ***

### Post-hoc comparisons

model.lm = artlm(model, "Instructor")

library(emmeans)

marginal = emmeans(model.lm,
                     ~ Speaker)

pairs(marginal,
      adjust = "tukey")

contrast          estimate    SE df t.ratio p.value
Bob Belcher - Linda Belcher   -13.562 3.23 28 -4.202  0.0021
Bob Belcher - Tina Belcher    -12.750 3.23 28 -3.950  0.0040
Bob Belcher - Gene Belcher     4.312 3.23 28  1.336  0.6717
Bob Belcher - Louise Belcher  -16.125 3.23 28 -4.996  0.0003
Linda Belcher - Tina Belcher   0.812 3.23 28  0.252  0.9991
Linda Belcher - Gene Belcher  17.875 3.23 28  5.538  0.0001
Linda Belcher - Louise Belcher -2.562 3.23 28 -0.794  0.9302
Tina Belcher - Gene Belcher   17.062 3.23 28  5.287  0.0001
Tina Belcher - Louise Belcher  -3.375 3.23 28 -1.046  0.8319
Gene Belcher - Louise Belcher -20.438 3.23 28 -6.332 <.0001

```

P value adjustment: tukey method for comparing a family of 5 estimates

```
cld(marginal,
  alpha=0.05,
  Letters=letters,
  adjust="tukey")

Instructor      emmean    SE   df lower.CL upper.CL .group
Gene Belcher    8.56  2.98 20.7    0.135    17.0     a
Bob Belcher     12.88  2.98 20.7    4.447    21.3     a
Tina Belcher    25.62  2.98 20.7   17.197    34.1     b
Linda Belcher   26.44  2.98 20.7   18.010    34.9     b
Louise Belcher  29.00  2.98 20.7   20.572    37.4     b
```

Degrees-of-freedom method: kenward-roger

Confidence level used: 0.95

Conf-level adjustment: sidak method for 5 estimates

P value adjustment: tukey method for comparing a family of 5 estimates

significance level used: alpha = 0.05

Optional: Plot of medians and confidence intervals for midichlorians data

```
library(rcompanion)

Sum = groupwiseMedian(Midichlorians ~ Tribe + Location,
                      data>Data,
                      bca=FALSE, percentile=TRUE)

Sum

  Tribe    Location n Median Conf.level Percentile.lower Percentile.upper
1  Jedi    Burlington 3    20      0.95            20          22
2  Jedi    Northampton 3     8      0.95            8           10
3  Jedi      Olympia 3    12      0.95            10          15
4  Jedi      Ventura 3    15      0.95            15          18
5  Sith    Burlington 3    22      0.95            22          25
6  Sith    Northampton 3    15      0.95            13          17
7  Sith      Olympia 3     4      0.95            4            5
8  Sith      Ventura 3    11      0.95            9           12
```

Order the levels for printing

```
Sum$Location = factor(Sum$Location,
                      levels=c("Olympia", "Ventura", "Northampton", "Burlington"))

Sum$Tribe = factor(Sum$Tribe,
                   levels=c("Jedi", "Sith"))
```

Plot

```

library(ggplot2)

pd = position_dodge(0.4)      ### How much to jitter the points on the plot

png(filename = "Rplot01.png",
    width  = 5,
    height = 5,
    units   = "in",
    res     = 300)

ggplot(sum,
       aes(x      = Location,
            y      = Median,
            color = Tribe)) +
  geom_point(shape  = 15,
             size   = 4,
             position = pd) +
  geom_errorbar(aes(ymin  = Percentile.lower,
                     ymax   = Percentile.upper),
                width = 0.2,
                size   = 0.7,
                position = pd) +
  theme_bw() +
  theme(axis.title  = element_text(face = "bold"),
        axis.text    = element_text(face = "bold"),
        plot.caption = element_text(hjust = 0)) +
  ylab("Median midichlorian count") +
  ggtitle ("Midichlorian counts for Jedi and Sith",
            subtitle = "In four U.S. cities") +
  labs(caption = paste0("\nMidichlorian counts for two tribes across ",
                        "four locations. Boxes indicate \n",
                        "the median. ",
                        "Error bars indicate the 95% confidence ",
                        "interval ",
                        "of the median."),
       hjust=0.5) +
  scale_color_manual(values = c("blue", "red"))

dev.off()

```

References

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.

Kay, M. 2019. *Contrast tests with ART*. cran.r-project.org/web/packages/ARTool/vignettes/art-contrasts.html.

Kay, M. 2019. *Effect Sizes with ART*. <https://cran.r-project.org/web/packages/ARTool/vignettes/art-effect-size.html>.

Kay, M. 2019. *Package 'ARTool'*. cran.r-project.org/web/packages/ARTool/ARTool.pdf.

Wobbrock, J. O., Findlater, L., Gergle, D., & Higgins, J. J. (2011). The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Conference on Human Factors in Computing Systems* (pp. 143–146). faculty.washington.edu/wobbrock/pubs/chi-11.06.pdf.

Wobbrock, J. O., Findlater, L., Gergle, D., Higgins, J. J., & Kay, M. (2018). ARTTool: Align-and-rank data for a nonparametric ANOVA. University of Washington. Retrieved from depts.washington.edu/madlab/proj/art/index.html.

Nonparametric Regression and Local Regression

There are different techniques that are considered to be forms of nonparametric regression. Kendall-Theil regression fits a linear model between one x variable and one y variable using a completely nonparametric approach. Quantile regression is a very flexible approach that can find a linear relationship between a dependent variable and one or more independent variables. Local regression fits a smooth curve to the dependent variable, and can accommodate multiple independent variables. Generalized additive models are a powerful and flexible approach.

Packages used in this chapter

The packages used in this chapter include:

- psych
- mblm
- quantreg
- rcompanion
- mgcv
- lmtest

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(mblm)){install.packages("mblm")}
if(!require(quantreg)){install.packages("quantreg")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(mgcv)){install.packages("mgcv")}
if(!require(lmtest)){install.packages("lmtest")}
```

Nonparametric correlation

Nonparametric correlation is discussed in the chapter *Correlation and Linear Regression*.

Nonparametric regression examples

Data for the examples in this chapter are borrowed from the *Correlation and Linear Regression* chapter. In this hypothetical example, students were surveyed for their weight, daily caloric intake, daily sodium intake, and a score on an assessment of knowledge gain

```
Input = "
Instructor      Grade    weight   Calories  Sodium  Score
'Brendon Small'  6        43     2069     1287    77
'Brendon Small'  6        41     1990     1164    76
'Brendon Small'  6        40     1975     1177    76
'Brendon Small'  6        44     2116     1262    84
'Brendon Small'  6        45     2161     1271    86
'Brendon Small'  6        44     2091     1222    87
'Brendon Small'  6        48     2236     1377    90
'Brendon Small'  6        47     2198     1288    78
'Brendon Small'  6        46     2190     1284    89
'Jason Penopolis' 7        45     2134     1262    76
'Jason Penopolis' 7        45     2128     1281    80
'Jason Penopolis' 7        46     2190     1305    84
'Jason Penopolis' 7        43     2070     1199    68
'Jason Penopolis' 7        48     2266     1368    85
'Jason Penopolis' 7        47     2216     1340    76
'Jason Penopolis' 7        47     2203     1273    69
'Jason Penopolis' 7        43     2040     1277    86
'Jason Penopolis' 7        48     2248     1329    81
'Melissa Robins'  8        48     2265     1361    67
'Melissa Robins'  8        46     2184     1268    68
'Melissa Robins'  8        53     2441     1380    66
'Melissa Robins'  8        48     2234     1386    65
'Melissa Robins'  8        52     2403     1408    70
'Melissa Robins'  8        53     2438     1380    83
'Melissa Robins'  8        52     2360     1378    74
'Melissa Robins'  8        51     2344     1413    65
'Melissa Robins'  8        51     2351     1400    68
'Paula Small'     9        52     2390     1412    78
'Paula Small'     9        54     2470     1422    62
'Paula Small'     9        49     2280     1382    61
'Paula Small'     9        50     2308     1410    72
'Paula Small'     9        55     2505     1410    80
'Paula Small'     9        52     2409     1382    60
'Paula Small'     9        53     2431     1422    70
'Paula Small'     9        56     2523     1388    79
'Paula Small'     9        50     2315     1404    71
'Coach McGuirk'  10       52     2406     1420    68
'Coach McGuirk'  10       58     2699     1405    65
'Coach McGuirk'  10       57     2571     1400    64
'Coach McGuirk'  10       52     2394     1420    69
'Coach McGuirk'  10       55     2518     1379    70
```

```
'Coach McGuirk'    10      52      2379      1393      61
'Coach McGuirk'    10      59      2636      1417      70
'Coach McGuirk'    10      54      2465      1414      59
'Coach McGuirk'    10      54      2479      1383      61
")

Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)
```

Kendall-Theil Sen Siegel nonparametric linear regression

Kendall-Theil regression is a completely nonparametric approach to linear regression where there is one independent and one dependent variable. It is robust to outliers in the dependent variable. It simply computes all the lines between each pair of points, and uses the median of the slopes of these lines. This method is sometimes called Theil-Sen. A modified, and preferred, method is named after Siegel.

The method yields a slope and intercept for the fit line, and a *p*-value for the slope can be determined as well. Typically, no measure analogous to *r-squared* is reported.

The *mblm* function in the *mblm* package uses the Siegel method by default. The Theil-Sen procedure can be chosen with the *repeated=FALSE* option. See *library(mblm); ?mblm* for more details.

```
library(mblm)

model.k = mblm(Calories ~ Sodium,
                data>Data)

summary(model.k)
```

Coefficients:

	Estimate	MAD	V	Pr(> v)
(Intercept)	-208.5875	608.4540	230	0.000861 ***
Sodium	1.8562	0.4381	1035	5.68e-14 ***

Values under Estimate are used to determine the fit line.

MAD is the median absolute deviation, a robust measure of variability

Plot with statistics

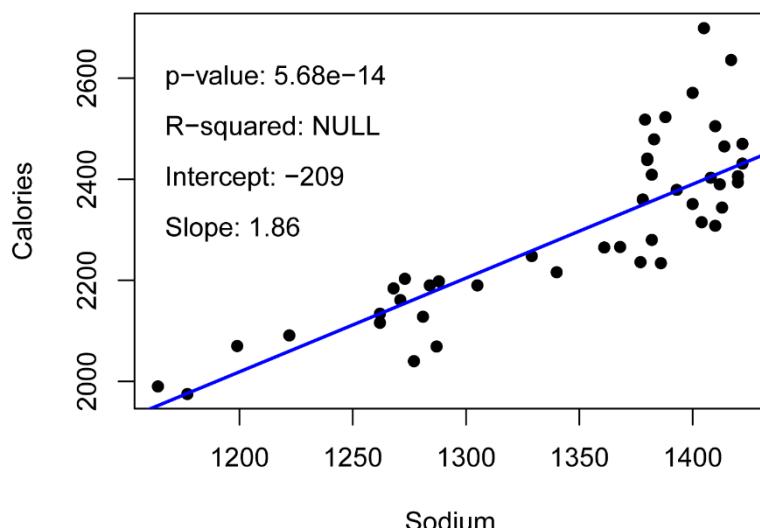
```
plot(Calories ~ Sodium,
      data = Data,
      pch = 16)

abline(model.k,
       col="blue",
       lwd=2)

Pvalue    = as.numeric(summary(model.k)$coefficients[2,4])
Intercept = as.numeric(summary(model.k)$coefficients[1,1])
Slope     = as.numeric(summary(model.k)$coefficients[2,1])
R2        = NULL

t1        = paste0("p-value: ", signif(Pvalue, digits=3))
t2        = paste0("R-squared: ", "NULL")
t3        = paste0("Intercept: ", signif(Intercept, digits=3))
t4        = paste0("Slope: ", signif(Slope, digits=3))

text(1160, 2600, labels = t1, pos=4)
text(1160, 2500, labels = t2, pos=4)
text(1160, 2400, labels = t3, pos=4)
text(1160, 2300, labels = t4, pos=4)
```



Quantile regression

While traditional linear regression models the conditional mean of the dependent variable, quantile regression models the conditional median or other quantile. Medians are most common, but for example, if the factors predicting the highest values of the dependent variable are to be investigated, a 95th percentile could be used. Likewise, models for several quantiles, e.g. 25th, 50th, 75th percentiles, could be investigated simultaneously.

Quantile regression makes no assumptions about the distribution of the underlying data, and is robust to outliers in the dependent variable. It does assume the dependent variable is continuous. However, there are functions for other types of dependent variables in the *qtools* package. The model assumes that the terms are linearly related. Quantile regression is sometimes considered “semiparametric”.

Quantile regression is very flexible in the number and types of independent variables that can be added to the model. The example, here, however, confines itself to a simple case with one independent variable and one dependent variable.

This example models the median of dependent variable, which is indicated with the *tau = 0.5* option.

A *p*-value for the model can be found by using the *anova* function with the fit model and the null model. A pseudo *R-squared* value can be found with the *nagelkerke* function in the *rcompanion* package.

```
library(quantreg)

model.q = rq(Calories ~ Sodium,
             data = Data,
             tau = 0.5)

summary(model.q)

tau: [1] 0.5

Coefficients:
            coefficients lower bd   upper bd
(Intercept) -84.12409   -226.58102  134.91738
Sodium       1.76642      1.59035   1.89615

### values under Coefficients are used to determine the fit line.
### bd appears to be a confidence interval for the coefficients

model.null = rq(Calories ~ 1,
                data = Data,
                tau = 0.5)

anova(model.q, model.null)

Quantile Regression Analysis of Deviance Table

  Df Resid Df F value    Pr(>F)
  1    1      43  187.82 < 2.2e-16 ***
```

```
### p-value for model overall

library(rcompanion)
nagelkerke(model.q)

$Pseudo.R.squared.for.model.vs.null
                           Pseudo.R.squared
McFadden                      0.115071
Cox and Snell (ML)            0.783920
Nagelkerke (Cragg and Uhler)  0.783921
```

Plot with statistics

```
plot(Calories ~ Sodium,
      data = Data,
      pch = 16)

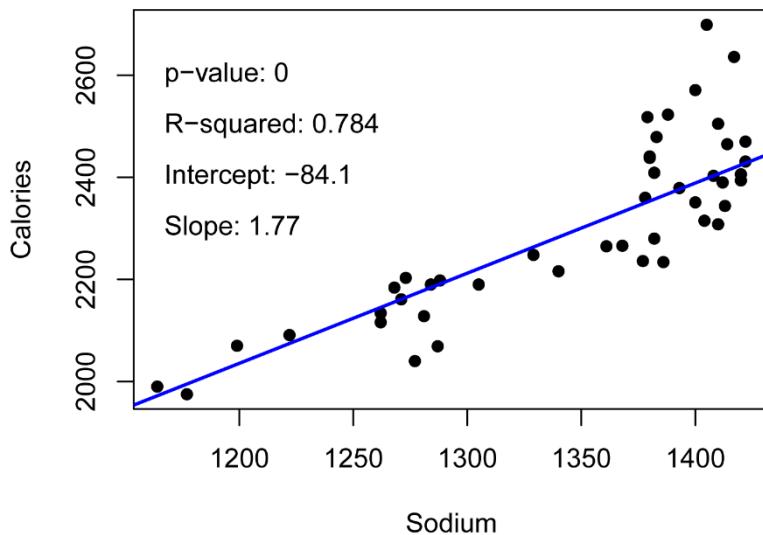
abline(model,
       col="blue",
       lwd=2)

library(rcompanion)

Pvalue = anova(model.q, model.null)[[1]][1,4]
Intercept = as.numeric(summary(model.q)$coefficients[1,1])
Slope     = as.numeric(summary(model.q)$coefficients[2,1])
R2       = nagelkerke(model.q)[[2]][3,1]

t1      = paste0("p-value: ", signif(Pvalue, digits=3))
t2      = paste0("R-squared: ", signif(R2, digits=3))
t3      = paste0("Intercept: ", signif(coefficients(model)[1], digits=3))
t4      = paste0("Slope: ", signif(coefficients(model)[2], digits=3))

text(1160, 2600, labels = t1, pos=4)
text(1160, 2500, labels = t2, pos=4)
text(1160, 2400, labels = t3, pos=4)
text(1160, 2300, labels = t4, pos=4)
```



Local regression

There are several techniques for local regression. The idea is to fit a curve to data by averaging, or otherwise summarizing, data points that are next to one another. The amount of “wiggleness” of the curve can be adjusted.

Local regression is useful for investigating the behavior of the response variable in more detail than would be possible with a simple linear model.

The function *loess* in the native *stats* package can be used for one continuous dependent variable and up to four independent variables. The process is essentially nonparametric, and is robust to outliers in the dependent variable. Usually no *p*-value or *r-squared* are reported. Integer variables have to be coerced to numeric variables.

The plot below shows a basically linear response, but also shows an increase in *Calories* at the upper end of *Sodium*.

```
Data$Sodium = as.numeric(Data$Sodium)

model.1 = loess(calories ~ Sodium,
                 data = Data,
                 span = 0.75,          ### higher numbers for smoother fits
                 degree=2,             ### use polynomials of order 2
                 family="gaussian")   ### the default, use least squares to fit

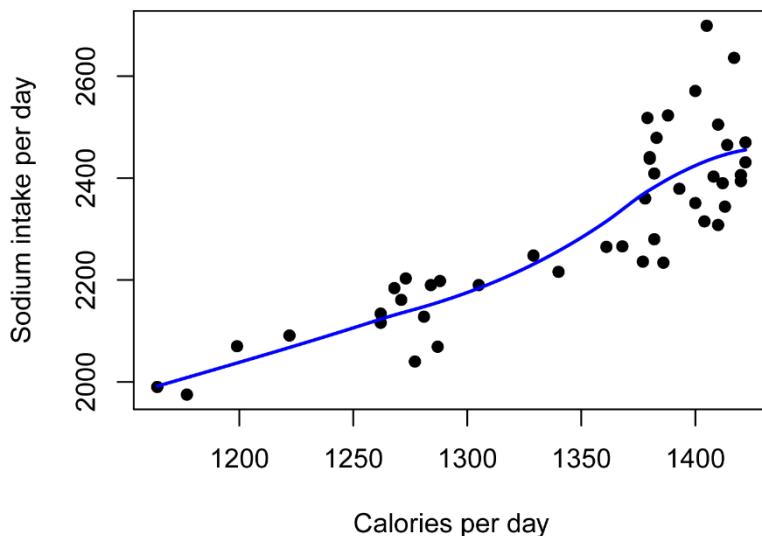
summary(model.1)

Number of Observations: 45
Equivalent Number of Parameters: 4.19
Residual Standard Error: 91.97
```

Plot

```
library(rcompanion)

plotPredy(data = Data,
          x = Sodium,
          y = Calories,
          model = model.1,
          xlab = "Calories per day",
          ylab = "Sodium intake per day")
```

**Generalized additive models**

Generalized additive models are very flexible, allowing for a variety of types of independent variables and of dependent variables. A smoother function is often used to create a “wiggly” model analogous to that used in local regression. The *gam* function in the *mgcv* package uses smooth functions plus a conventional parametric component, and so would probably be classified as a semiparametric approach. The *summary* function reports an *R-squared* value, and *p*-values for the terms. The *anova* function can be used for one model, or to compare two models.

```
library(mgcv)

model.g = gam(Calories ~ s(Sodium),
               data = Data,
               family=gaussian())

summary(model.g)

Parametric coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) 2304.87     13.62   169.2   <2e-16 ***
```

Approximate significance of smooth terms:

	edf	Ref.df	F	p-value
s(Sodium)	1.347	1.613	66.65	4.09e-15 ***

R-sq.(adj) = 0.718 Deviance explained = 72.6%
GCV = 8811.5 Scale est. = 8352 n = 45

```
model.null = gam(Calories ~ 1,
                  data = Data,
                  family=gaussian())
```

```
anova(model.g,
      model.null)
```

Analysis of Deviance Table

Model 1: Calories ~ s(Sodium)
Model 2: Calories ~ 1

	Resid. Df	Resid. Dev	Df	Deviance
1	42.387	356242		
2	44.000	1301377	-1.6132	-945135

```
library(lmtest)
```

```
lrtest(model.g,
       model.null)
```

Likelihood ratio test

Model 1: Calories ~ s(Sodium)
Model 2: Calories ~ 1

	#Df	LogLik	Df	Chisq	Pr(>Chisq)
1	3.3466	-265.83			
2	2.0000	-294.98	-1.3466	58.301	2.25e-14 ***

Plot with statistics

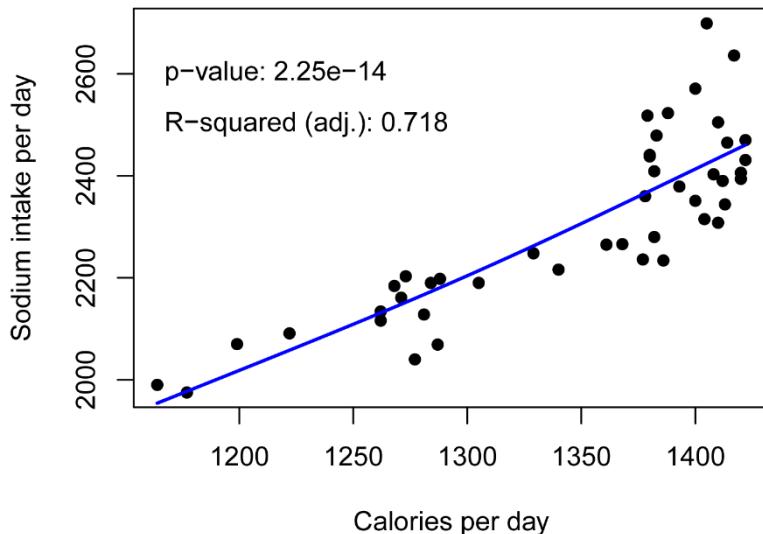
```
library(rcompanion)
```

```
plotPredy(data = Data,
          x = Sodium,
          y = Calories,
          model = model.g,
          xlab = "Calories per day",
          ylab = "Sodium intake per day")
```

```
Pvalue = 2.25e-14
R2 = 0.718
```

```
t1      = paste0("p-value: ", signif(Pvalue, digits=3))
t2      = paste0("R-squared (adj.): ", signif(R2, digits=3))

text(1160, 2600, labels = t1, pos=4)
text(1160, 2500, labels = t2, pos=4)
```



```
### Note that the fit line is slightly curved.
```

Nonparametric Regression for Time Series

There are specific nonparametric techniques that are commonly used for time series data. The Mann-Kendall trend test is commonly used to determine if a trend exists, and can handle seasonal patterns within the data. The slope of the trend is often determined with Sen's slope, which can also handle seasonality in the data. Finally, Pettitt's test is used to determine if there is a point at which the values in the data change.

It is my understanding that the tests in this chapter do not take into consideration any autocorrelation in the data. Autocorrelation would be for example if an observation with a high value would tend to be followed by another observation with a high value. For the stage data in this chapter, since the stage of a tidal river is affected by the phase of the moon and by the direction and strength of the wind, it would be reasonable to assume that the average stage for one day would be correlated to the previous day. Due to this limitation in these tests, there are two conditions under which they are usually used. 1) When the observations are far enough apart in time that autocorrelation is unlikely to be important. For example, in the stage data in this chapter, monthly averages are used. 2) When the values are likely to be influenced mostly by non-autocorrelated factors. For data where autocorrelation is likely to be important, other models, such as autoregressive integrated moving average (ARIMA), could be used.

Packages used in this chapter

The packages used in this chapter include:

- mice
- Kendall
- trend

The following commands will install these packages if they are not already installed:

```
if(!require(mice)){install.packages("mice")}
if(!require(Kendall)){install.packages("Kendall")}
if(!require(trend)){install.packages("trend")}
```

Nonparametric regression examples

The data used in this chapter is a times series of stage measurements of the tidal Cohansey River in Greenwich, NJ. Stage is the height of the river, in this case given in feet, with an arbitrary 0 datum. The data are from U.S. Geology Survey site 01413038, and are monthly averages.

The *MannKendall* and *SeasonalMannKendall* functions can handle missing data. However, the *sens.slope*, *sea.sens.slope*, *decompose*, and *stl* functions, at the time of writing, cannot handle missing data.

Because the data set has a few missing data points, we will impute those values with the *mice* function in the *mice* package.

The function *ts* converts data into a time series object. The user needs to be careful using this function, since it does not check for missing lines in the data. For example, if there is no observation for February, the function will simply take the March the value for February and continue. However, one could add an observation for February with a value of *NA*.

```
Greenwich = read.table("http://rcompanion.org/documents/Greenwich.csv",
                      header=TRUE, sep=",")
```

```
Greenwich[125:136,]
```

	Agency	Station	Parameter	TS_ID	Year	Month	Stage
125	USGS	1413038		65	97255	2010	11 0.107
126	USGS	1413038		65	97255	2010	12 -0.297
127	USGS	1413038		65	97255	2011	1 -0.350
128	USGS	1413038		65	97255	2011	2 NA
129	USGS	1413038		65	97255	2011	3 NA
130	USGS	1413038		65	97255	2011	4 NA
131	USGS	1413038		65	97255	2011	5 NA
132	USGS	1413038		65	97255	2011	6 NA
133	USGS	1413038		65	97255	2011	7 NA
134	USGS	1413038		65	97255	2011	8 0.347
135	USGS	1413038		65	97255	2011	9 0.738

```
136    USGS 1413038      65 97255 2011      10  0.499
```

```
library(mice)
```

```
Mousey = mice(Greenwich)
```

```
Greenwich = complete(Mousey)
```

```
Greenwich[125:136,]
```

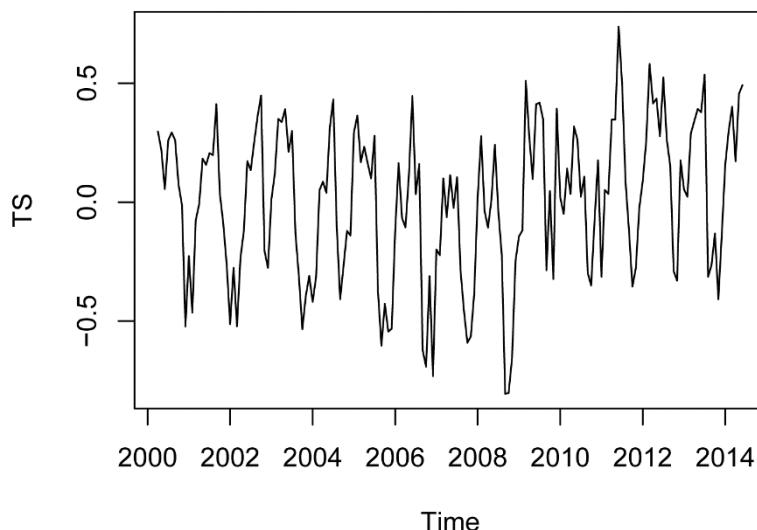
	Agency	Station	Parameter	TS_ID	Year	Month	Stage
125	USGS	1413038		65	97255	2010	11 0.107
126	USGS	1413038		65	97255	2010	12 -0.297
127	USGS	1413038		65	97255	2011	1 -0.350
128	USGS	1413038		65	97255	2011	2 -0.076
129	USGS	1413038		65	97255	2011	3 0.176
130	USGS	1413038		65	97255	2011	4 -0.314
131	USGS	1413038		65	97255	2011	5 0.051
132	USGS	1413038		65	97255	2011	6 0.035
133	USGS	1413038		65	97255	2011	7 0.347
134	USGS	1413038		65	97255	2011	8 0.347
135	USGS	1413038		65	97255	2011	9 0.738
136	USGS	1413038		65	97255	2011	10 0.499

```
TS = ts(Greenwich$Stage,
        frequency=12,
        start=c(2000,4))
```

```
TS
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2000				0.297	0.219	0.055	0.260	0.293	0.261	0.074	-0.013	-0.523
2001	-0.227	-0.465	-0.076	-0.008	0.183	0.157	0.206	0.198	0.412	0.037	-0.089	-0.263
2002	-0.513	-0.276	-0.522	-0.235	-0.120	0.172	0.135	0.253	0.362	0.448	-0.205	-0.276
2003	0.011	0.120	0.350	0.337	0.391	0.212	0.299	-0.132	-0.313	-0.534	-0.395	-0.310
2004	-0.419	-0.314	0.051	0.086	0.039	0.314	0.432	-0.099	-0.408	-0.267	-0.121	-0.140
2005	0.293	0.364	0.168	0.233	0.164	0.100	0.279	-0.375	-0.603	-0.427	-0.544	-0.532
2006	-0.122	0.164	-0.067	-0.106	0.115	0.446	0.034	0.160	-0.623	-0.692	-0.311	-0.731
2007	-0.199	-0.223	0.100	-0.063	0.113	-0.024	0.105	-0.284	-0.456	-0.591	-0.565	-0.381
2008	0.020	0.277	-0.037	-0.107	0.011	0.241	-0.038	-0.222	-0.806	-0.802	-0.654	-0.247
2009	-0.144	-0.119	0.510	0.274	0.097	0.412	0.418	0.350	-0.286	0.047	-0.323	0.393
2010	0.021	-0.049	0.141	0.035	0.318	0.263	0.023	0.107	-0.297	-0.350	0.051	0.289
2011	-0.063	0.164	0.314	0.037	0.347	0.738	0.499	0.087	-0.119	-0.355	-0.277	-0.025
2012	0.088	0.254	0.581	0.415	0.435	0.277	0.524	0.263	0.153	-0.290	-0.330	0.176
2013	0.054	0.023	0.289	0.341	0.392	0.378	0.536	-0.314	-0.265	-0.132	-0.408	-0.134
2014	0.158	0.300	0.401	0.172	0.456	0.492						

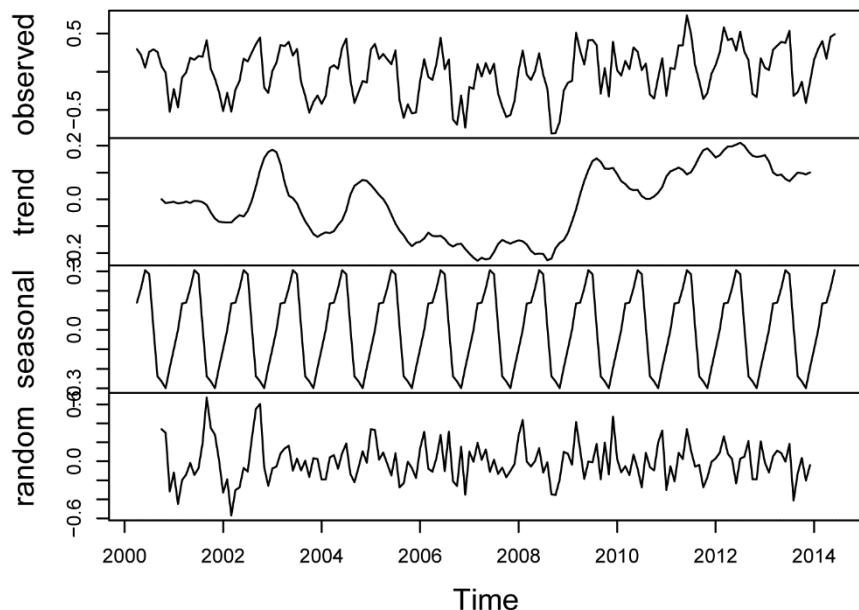
```
plot(TS)
```



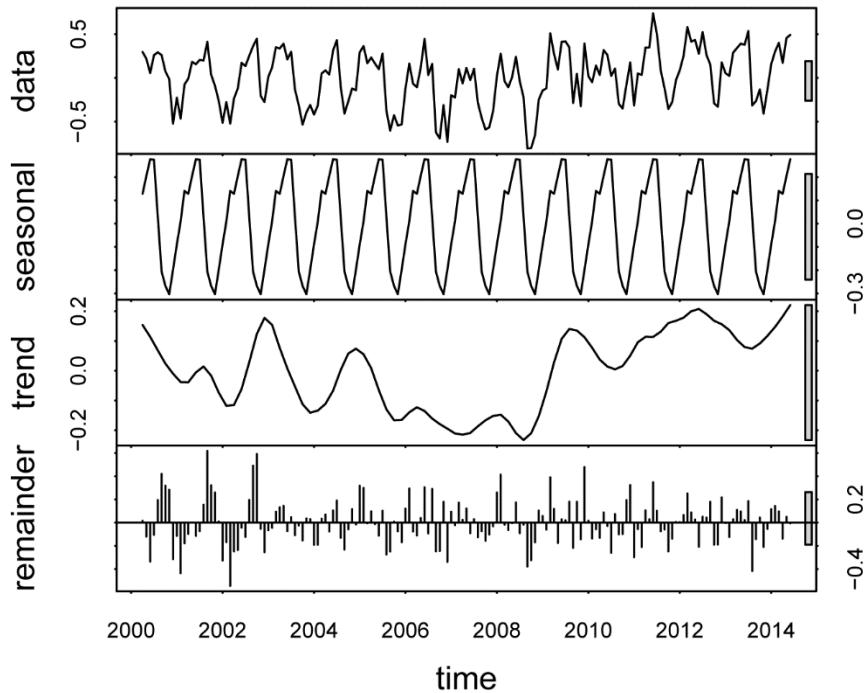
Decomposing time series objects

It is helpful to decompose time series data into seasonal and trend components. The *decompose* function in the native *stats* package uses “classical seasonal decomposition by moving averages”, and the *stl* function in the native *stats* package uses “seasonal decomposition of time series by loess”.

```
plot(decompose(TS))
```



```
plot(stl(TS,  
s.window="periodic"))
```



Mann-Kendall trend test

The Mann-Kendall trend test is completely nonparametric. The *MannKendall* function in the *Kendall* package can be used with a time series object. The *SeasonalMannKendall* function performs the test while taking into account the seasonality of the data.

```
library(Kendall)
MK = MannKendall(TS)
summary(MK)

Score = 1371 , Var(Score) = 560382.3
denominator = 14524
tau = 0.0944, 2-sided pvalue =0.067233

### tau is a measure of the strength of the trend
### p-value for trend is also shown

library(Kendall)
SMK = SeasonalMannKendall(TS)
summary(SMK)

Score = 217 , Var(Score) = 4227
denominator = 1133.499
tau = 0.191, 2-sided pvalue =0.00084484
```

```
### tau is a measure of the strength of the trend
### p-value for trend is also shown
```

Sen's slope for time series data

The *sens.slope* function in the *trend* package is used with a time series object. The *sea.sens.slope* function performs the test while taking into account the seasonality of the data.

```
library(trend)

sens.slope(TS)

  Sen's slope and intercept

slope: 0.001
95 percent confidence intervall for slope
0.002 -1e-04

intercept: -0.0475
nr. of observations: 171
```

The *sea.sens.slope* function appears to not tolerate incomplete time series. For these data, we can remove the data for the incomplete years (2000 and 2014).

```
GW = Greenwich[10:165,]

TS2 = ts(GW$Stage,
          frequency=12,
          start=c(2001,1))

TS2

  Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
2001 -0.227 -0.465 -0.076 -0.008  0.183  0.157  0.206  0.198  0.412  0.037 -0.089 -0.263
2002 -0.513 -0.276 -0.522 -0.235 -0.120  0.172  0.135  0.253  0.362  0.448 -0.205 -0.276
2003  0.011  0.120  0.350  0.337  0.391  0.212  0.299 -0.132 -0.313 -0.534 -0.395 -0.310
2004 -0.419 -0.314  0.051  0.086  0.039  0.314  0.432 -0.099 -0.408 -0.267 -0.121 -0.140
2005  0.293  0.364  0.168  0.233  0.164  0.100  0.279 -0.375 -0.603 -0.427 -0.544 -0.532
2006 -0.122  0.164 -0.067 -0.106  0.115  0.446  0.034  0.160 -0.623 -0.692 -0.311 -0.731
2007 -0.199 -0.223  0.100 -0.063  0.113 -0.024  0.105 -0.284 -0.456 -0.591 -0.565 -0.381
2008  0.020  0.277 -0.037 -0.107  0.011  0.241 -0.038 -0.222 -0.806 -0.802 -0.654 -0.247
2009 -0.144 -0.119  0.510  0.274  0.097  0.412  0.418  0.350 -0.286  0.047 -0.323  0.393
2010  0.021 -0.049  0.141  0.035  0.318  0.263  0.023  0.107 -0.297 -0.350 -0.565 -0.314
2011 -0.330  0.233  0.164  0.035  0.347  0.738  0.499  0.087 -0.119 -0.355 -0.277 -0.025
2012  0.088  0.254  0.581  0.415  0.435  0.277  0.524  0.263  0.153 -0.290 -0.330  0.176
2013  0.054  0.023  0.289  0.341  0.392  0.378  0.536 -0.314 -0.265 -0.132 -0.408 -0.134
```

```
library(trend)

sea.sens.slope(TS2)

[1] 0.0185
```

Pettitt's test for change in value

The *pettitt.test* function in the *trend* package identifies a point at which the values in the data change. The results here suggest that the stage values were higher after May 2009 than before.

```
library(trend)

pettitt.test(TS)

Pettitt's test for single change-point detection

K = 2453, p-value = 0.001526
alternative hypothesis: true change point is present in the series

sample estimates:
probable change point at tau
107

Greenwich[107,]

   Agency Station Parameter TS_ID Year Month Stage
107    USGS 1413038       65 97255 2009      5 -0.119
```

Introduction to Permutation Tests

Permutation tests are increasingly common tests to perform certain types of statistical analyses. They do not rely on assumptions about the distribution of the data, as some other tests do. They are therefore considered to be nonparametric tests. It is my understanding, however, that for certain tests—for example those testing a difference in means—that there are assumptions about the underlying data.

Permutation tests work by resampling the observed data many times in order to determine a *p*-value for the test. Recall that the *p*-value is defined as the probability of getting data as extreme as the observed data when the null hypothesis is true. If the data are shuffled many times in accordance with the null hypothesis being true, the number of cases with data as extreme as the observed data could be counted, and a *p*-value calculated.

The advantages of permutation tests are

- the lack of assumptions about the distribution of the underlying data,
- their flexibility in the kinds of data they can handle (nominal, ordinal, interval/ratio),
- and their being relatively straightforward to conduct and interpret.

The disadvantages of permutation tests are

- the limited complexity of designs they can handle
- the unfamiliarity with them for many readers.

The coin package

Permutation tests in this book will use the *coin* package, with either of two functions, *independence_test* and *symmetry_test*. This book will use permutation tests with ordinal dependent variables, but the *coin* package is able to handle nominal, ordinal, and interval/ratio data.

A few notes on using permutation tests:

- If the dependent variable is to be treated as an ordinal variable, it must be coded as an ordered factor variable in R. It does not need to have numerals for levels. For example it could have levels *doctorate* > *masters* > *bachelors* > *associates* > *high.school*. But also it could have the levels *5* > *4* > *3* > *2* > *1*.
- The general interpretation for significant results of these models isn't that there is a difference among medians, but that there is a significant effect of the independent variable on the dependent variable, or that there is a significant difference among groups.
- Post-hoc tests for factors or groups can be conducted with pairwise tests of groups. The appropriate functions in the *rcompanion* package are *pairwisePermutationTest*, *pairwisePermutationMatrix*, *pairwisePermutationSymmetry*, and *pairwisePermutationSymmetryMatrix*.
- Permutation tests for data arranged in contingency tables are presented in the *Association Tests for Ordinal Tables* chapter.

Packages used in this chapter

The packages used in this chapter include:

- coin

The following commands will install these packages if they are not already installed:

```
if(!require(coin)){install.packages("coin")}
```

Permutation test example

The following example uses the left hand and right hand data from the *Independent and Paired Values* chapter. For this example, we are interested in comparing the length of left hands and rights from 16 individuals. First we will compare the left hands to right hands as independent samples (analogous to a Mann–Whitney test or t-test), then as paired values for each individual (analogous to a paired rank sum test or paired t-test).

```
Input = "
Individual Hand Length
A Left 17.5
B Left 18.4
C Left 16.2
D Left 14.5
E Left 13.5
F Left 18.9
G Left 19.5
H Left 21.1
I Left 17.8
J Left 16.8
K Left 18.4
L Left 17.3
M Left 18.9
N Left 16.4
O Left 17.5
P Left 15.0
A Right 17.6
B Right 18.5
C Right 15.9
D Right 14.9
E Right 13.7
F Right 18.9
G Right 19.5
H Right 21.5
I Right 18.5
J Right 17.1
K Right 18.9
L Right 17.5
M Right 19.5
N Right 16.5
O Right 17.4
```

```
P          Right   15.6
")
```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
### Check the data frame
```

```
Data
```

```
str(Data)
```

```
summary(Data)
```

```
### Remove unnecessary objects
```

```
rm(Input)
```

```
### Summarize data
```

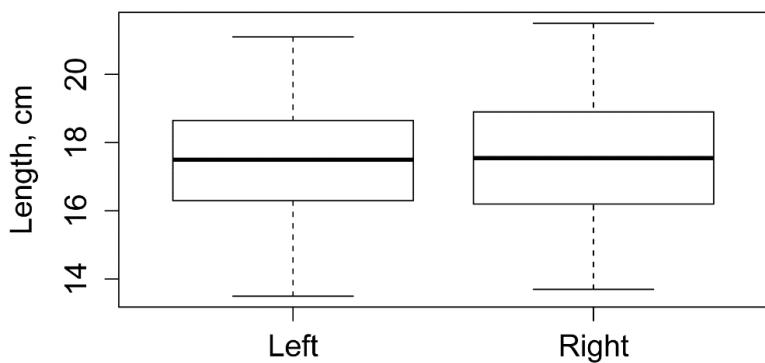
```
library(FSA)
```

```
Summarize(Length ~ Hand,
           data=Data,
           digits=3)
```

	Hand	n	invalid	mean	sd	min	Q1	median	Q3	max	percZero
1	Left	16	16	17.356	1.948	13.5	16.35	17.50	18.52	21.1	0
2	Right	16	16	17.594	1.972	13.7	16.35	17.55	18.90	21.5	0

Box plot

```
boxplot(Length ~ Hand,
        data=Data,
        ylab="Length, cm")
```



Scatter plot with one-to-one line

In the plot below, each point represents a pair of paired values. Points that fall above and to the left of the blue line indicate cases for which the value for Right was greater than for Left.

```

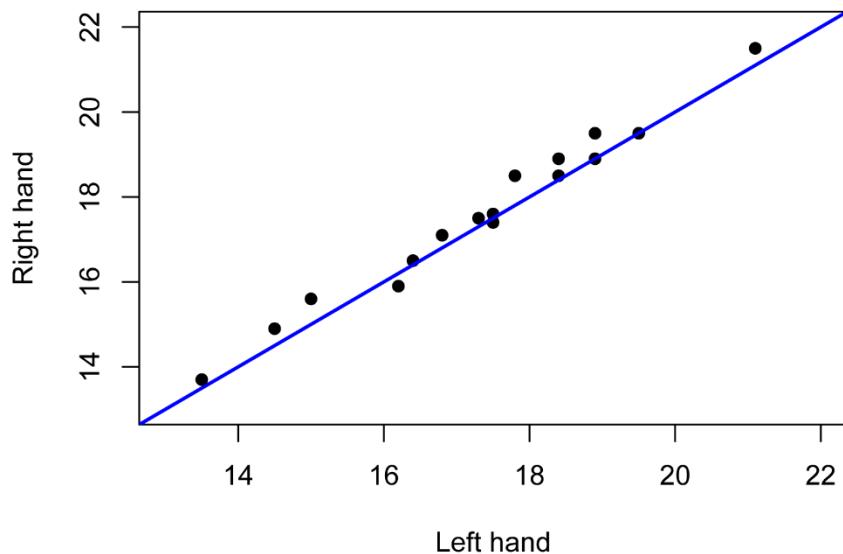
Left = Data$Length[Data$Hand=="Left"]

Right = Data$Length[Data$Hand=="Right"]

plot(Left, Right,
      pch = 16,                      # shape of points
      cex = 1.0,                      # size of points
      xlim=c(13, 22),                # limits of x axis
      ylim=c(13, 22),                # limits of y axis
      xlab="Left hand",
      ylab="Right hand")

abline(0,1, col="blue", lwd=2) # line with intercept of 0 and slope of 1

```



Permutation test of independence

This test treats the two groups (left hand and right hand) as independent samples, and tests if there is a difference in values between the two groups. The box plot above reflects the approach of this test.

```

library(coin)

independence_test(Length ~ Hand,
                  data = Data)

Asymptotic General Independence Test

z = -0.34768, p-value = 0.7281

```

Permutation test of symmetry

This test treats the two groups (left hand and right hand) as having paired or repeated data, paired within *Individual*. That is, the test looks at the difference between left hand and right hand for each individual. The scatter plot above reflects the approach of this test.

Note the use of the *symmetry_test* function.

```
library(coin)

symmetry_test(Length ~ Hand | Individual,
              data = Data)

Asymptotic General Symmetry Test

z = -2.6348, p-value = 0.008418
```

References

For more information on permutation tests and the *coin* package, see:

Hothorn, T., K. Hornik, M.A. van de Wiel, and A. Zeileis. 2015. *Implementing a Class of Permutation Tests: The coin Package*. cran.r-project.org/web/packages/coin/vignettes/coin_implementation.pdf.

```
library(coin); help(package="coin")
```

One-way Permutation Test of Independence for Ordinal Data

When to use this test

A permutation test of independence can be used for one-way data with an ordinal dependent variable. It will determine if there is a difference in the response variable among groups. There can be two or more groups.

This test covers the designs used with Mann–Whitney, Kruskal–Wallis, two-sample t-test, one-way anova, one-way anova with blocks, and ordinal regression equivalents.

The test assumes that the observations are independent. That is, it is not appropriate for paired observations or repeated measures data. It does not make assumptions about the distribution of values.

The test is performed with the *independence_test* function in the *coin* package.

Post-hoc testing can be conducted with pairwise permutation tests across groups.

It is important that the dependent variable be specified as an ordered factor variable in R.

Appropriate data

- One-way data. That is, one measurement variable in two or more groups
- Dependent variable is ordinal, and specified in R as an ordered factor variable
- Independent variable is a factor with two or more levels. That is, two or more groups
- Observations between groups are independent. That is, not paired or repeated measures data

Hypotheses

- Null hypothesis: The values of the dependent variable among groups are equal.
- Alternative hypothesis (two-sided): The values of the dependent variable among groups are not equal.

Interpretation

- Reporting significant results for the omnibus test as “Significant differences were found among values for groups.” is acceptable. Alternatively, “A significant effect for Independent Variable on Dependent Variable was found.”
- Reporting significant results for mean separation post-hoc tests as “Value of Dependent Variable for group A was different than that for group B.” is acceptable.

Other notes and alternative tests

Ordinal regression is an alternative.

The traditional nonparametric tests Mann–Whitney U or Kruskal–Wallis are alternatives in cases where there is not a blocking variable.

Packages used in this chapter

The packages used in this chapter include:

- psych
- lattice
- FSA
- coin
- rcompanion
- multcompView
- ggplot2

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(coin)){install.packages("coin")}
```

```
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(ggplot2)){install.packages("ggplot2")}
```

One-way ordinal permutation test example

This example re-visits the Pooh, Piglet, and Tigger data from the *Descriptive Statistics with the likert Package* chapter.

It answers the question, “Are the scores significantly different among the three speakers?”

Note that a new variable, *Likert.f*, is created for the Likert scores as an ordered factor variable. It is necessary that the dependent variable passed to *independence_test* be an ordered factor variable for it to be treated as an ordinal variable.

```
Input =""
Speaker Likert
Pooh     3
Pooh     5
Pooh     4
Pooh     4
Pooh     4
Pooh     4
Pooh     4
Pooh     4
Pooh     5
Pooh     5
Piglet   2
Piglet   4
Piglet   2
Piglet   2
Piglet   1
Piglet   2
Piglet   3
Piglet   2
Piglet   2
Piglet   3
Tigger   4
Tigger   4
Tigger   4
Tigger   5
Tigger   3
Tigger   5
Tigger   4
Tigger   4
Tigger   3
")
Data = read.table(textConnection(Input),header=TRUE)
### Order levels of the factor; otherwise R will alphabetize them
```

```

Data$Speaker = factor(Data$Speaker,
                      levels=unique(Data$Speaker))

### Create a new variable which is the Likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                       ordered = TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Summarize data treating Likert scores as factors

Note that the variable we want to count is *Likert.f*, which is a factor variable. Counts for *Likert.f* are cross tabulated over values of *Speaker*. The *prop.table* function translates a table into proportions. The *margin=1* option indicates that the proportions are calculated for each row.

```

xtabs( ~ Speaker + Likert.f,
      data = Data)

  Likert.f
Speaker 1 2 3 4 5
  Pooh 0 0 1 6 3
  Piglet 1 6 2 1 0
  Tigger 0 0 2 6 2

XT = xtabs( ~ Speaker + Likert.f,
            data = Data)

prop.table(XT,
           margin = 1)

  Likert.f
Speaker 1 2 3 4 5
  Pooh 0.0 0.0 0.1 0.6 0.3
  Piglet 0.1 0.6 0.2 0.1 0.0
  Tigger 0.0 0.0 0.2 0.6 0.2

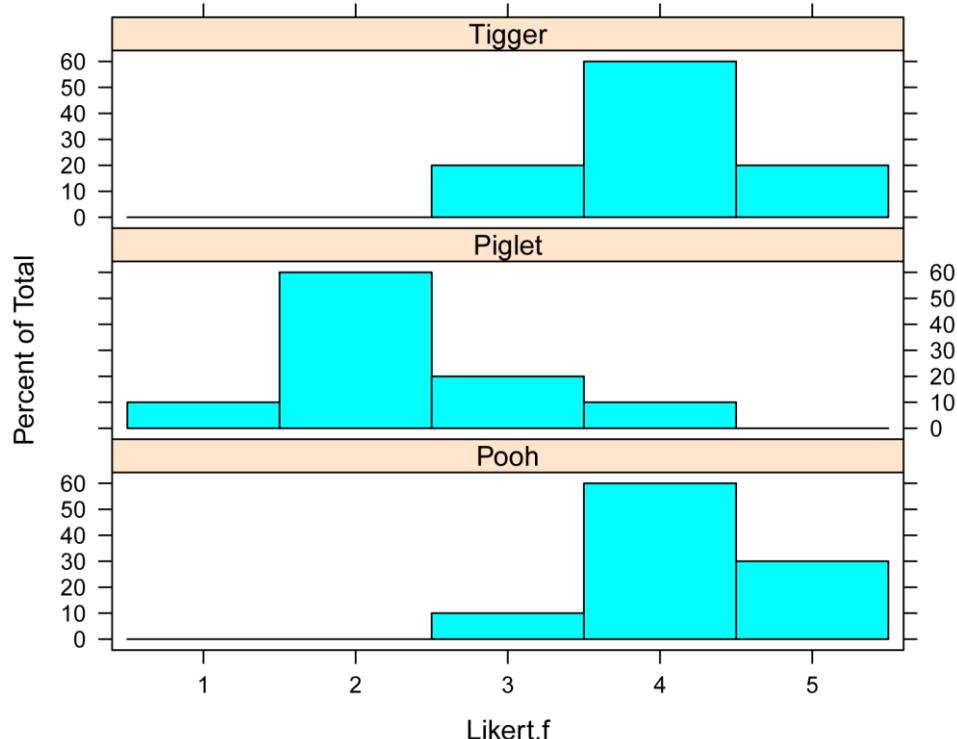
```

Bar plots by group

Note that the variable we want to count is *Likert.f*, which is a factor variable. Counts for *Likert.f* are presented for values of *Speaker*.

```
library(lattice)

histogram(~ Likert.f | Speaker,
           data=Data,
           layout=c(1,3))      # columns and rows of individual plots
```

**Summarize data treating Likert scores as numeric**

It may be useful to look at the minimum, first quartile, median, third quartile, and maximum for *Likert* for each group.

```
library(FSA)

Summarize(Likert ~ Speaker,
           data=Data,
           digits=3)
```

Speaker	n	mean	sd	min	Q1	median	Q3	max	percZero
Pooh	10	4.2	0.632	3	4	4	4.75	5	0
Piglet	10	2.3	0.823	1	2	2	2.75	4	0
Tigger	10	4.0	0.667	3	4	4	4.00	5	0

One-way ordinal permutation test

Note that the dependent variable is an ordered factor variable, *Likert.f*. *Speaker* is the independent variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *library(coin); ?independence_test*. There is the option of adding a blocking variable to the formula with e.g. / *Block*.

```
library(coin)

independence_test(Likert.f ~ Speaker,
                  data = Data)

Asymptotic General Independence Test

maxT = 4.2006, p-value = 7.685e-05
```

Post-hoc test: pairwise permutation tests

If the independence test is significant, a post-hoc analysis can be performed to determine which groups differ from which other groups.

The *pairwisePermutationTest* and *pairwisePermutationMatrix* functions in the *rcompanion* package conduct permutation tests across groups in a pairwise manner. See *library(rcompanion); ?pairwisePermutationTest* for further details.

Because the post-hoc test will produce multiple *p*-values, adjustments to the *p*-values can be made to avoid inflating the possibility of making a type-I error. Here, the method of adjustment is indicated with the *method* option. There are a variety of methods for controlling the familywise error rate or for controlling the false discovery rate. See *?p.adjust* for details on these methods.

Before conducting the pairwise tests, we will re-order the levels of the grouping variable by the median of each group. This makes interpretation of the pairwise comparisons and compact letter display easier. In the output, groups sharing a same letter are not significantly different.

Table output and compact letter display

```
### Order groups by median

Data$Speaker = factor(Data$Speaker,
                      levels=c("Pooh", "Tigger", "Piglet"))

### Pairwise permutation tests

library(rcompanion)

PT = pairwisePermutationTest(Likert.f ~ Speaker,
                             data = Data,
                             method = "fdr")

PT
```

Comparison	Stat	p.value	p.adjust
Pooh vs Tigger	4.2006	7.685e-05	0.0001
Pooh vs Piglet	4.2006	7.685e-05	0.0001
Tigger vs Piglet	4.2006	7.685e-05	0.0001

```

1 Pooh - Tigger = 0  0.698    0.4852 0.485200
2 Pooh - Piglet = 0  3.515    0.000439 0.001238
3 Tigger - Piglet = 0 -3.344   0.0008254 0.001238

```

```
### Compact letter display
```

```
library(rcompanion)
```

```
cldList(p.adjust ~ Comparison,
        data = PT,
        threshold = 0.05)
```

	Group	Letter	MonoLetter
1	Pooh	a	a
2	Tigger	a	a
3	Piglet	b	b

Groups sharing a letter are not significantly different (alpha = 0.05).

Matrix output and compact letter display

The code creates a matrix of p -values called PM , which is then passed to the *multcompLetters* function to be converted to a compact letter display.

Here the *fdr* p -value adjustment method is used. See *?p.adjust* for details on available methods.

```
### Order groups by median
```

```
Data$Speaker = factor(Data$Speaker,
                      levels=c("Pooh", "Tigger", "Piglet"))
```

```
### Conduct pairwise permutation tests
```

```
library(rcompanion)
```

```
PM = pairwisePermutationMatrix(Likert.f ~ Speaker,
                               data = Data,
                               method = "fdr")$Adjusted
```

```
PM
```

```
### Produce compact letter display
```

```
library(multcompView)
```

```
multcompLetters(PM,
                compare="<",
                threshold=0.05, # p-value to use as significance threshold
                Letters=letters,
                reversed = FALSE)
```

Pooh Tigger Piglet

"a" "a" "b"

Groups sharing a letter are not significantly different (alpha = 0.05).

Plot of medians and confidence intervals

The following code uses the *groupwiseMedian* function to produce a data frame of medians for each speaker along with the 95% confidence intervals for each median with the percentile method. See *library(rcompanion); ?groupwiseMedian* for further options.

These medians are then plotted, with their confidence intervals shown as error bars. The grouping letters from the multiple comparisons are added.

```
### Order groups by original order

Data$Speaker = factor(Data$Speaker,
                      levels=c("Pooh", "Piglet", "Tigger"))

### Create data frame of medians and confidence intervals

library(rcompanion)

Sum = groupwiseMedian(Likert ~ Speaker,
                      data      = Data,
                      conf      = 0.95,
                      R         = 5000,
                      percentile = TRUE,
                      bca       = FALSE,
                      digits    = 3)

Sum

### Prepare labels

x = 1:3

Y = Sum$Percentile.upper + 0.2

Label = c("a", "b", "a")

### Plot

library(ggplot2)

ggplot(Sum,           ### The data frame to use.
       aes(x = Speaker,
           y = Median)) +
  geom_errorbar(aes(ymin = Percentile.lower,
                    ymax = Percentile.upper),
                width = 0.05,
                size  = 0.5) +
```

```

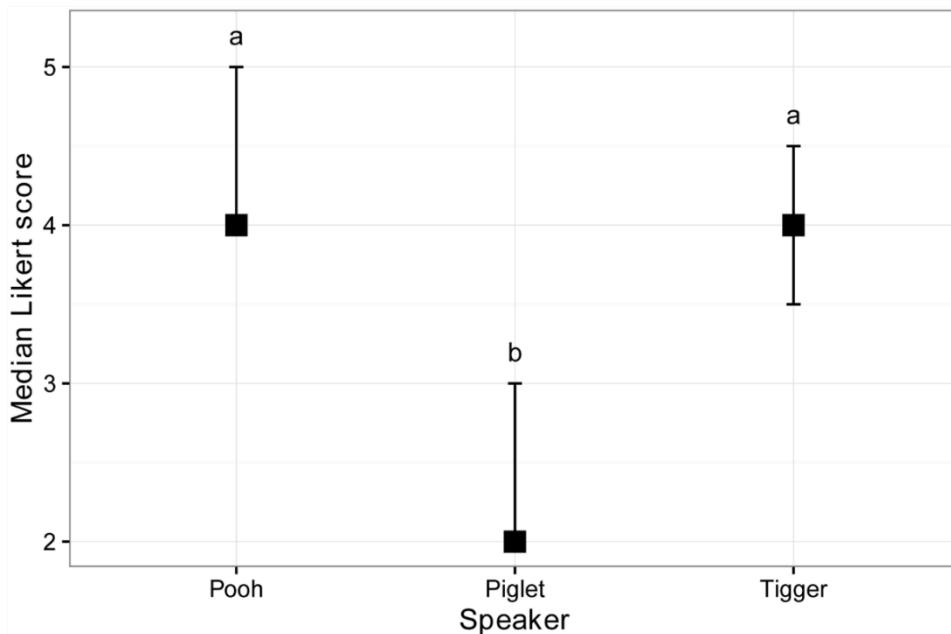
geom_point(shape = 15,
           size  = 4) +
theme_bw() +
theme(axis.title  = element_text(face  = "bold")) +

ylab("Median Likert score") +  

annotate("text",
       x = X,
       y = Y,
       label = Label)

```



Plot of median Likert score versus Speaker. Error bars indicate the 95% confidence intervals for the median with the percentile method.

Comparison of methods for independence tests for ordinal data

<u>Data</u>	<u>Test</u>	<u>p-value</u>	<u>Mean separation</u>
Pooh Piglet	Permutation	0.0004	
Pooh Piglet	Ordinal regression	0.00003	
Pooh Piglet	Mann–Whitney	0.0005	
Pooh Tigger Piglet	Permutation	0.00008	a a b
Pooh Tigger Piglet	Ordinal regression	0.000008	a a b
Pooh Tigger Piglet	Kruskal–Wallis	0.0002	a a b

One-way Permutation Test of Symmetry for Paired Ordinal Data

When to use this test

A permutation test of symmetry can be used for one-way data with an ordinal dependent variable where observations are paired within a blocking variable. It will determine if there is a difference in the response variable among groups when controlling for the effect of the blocking variable. There can be two or more groups.

The *coin* package can accommodate designs used with Friedman, Quade, paired t-test, repeated measures one-way anova, and ordinal regression equivalents.

The test does not make assumptions about the distribution of values.

The test is performed with the *symmetry_test* function in the *coin* package.

Post-hoc testing can be conducted with pairwise permutation tests across groups.

It is important that the dependent variable be specified as an ordered factor variable in R, if it is to be treated as an ordinal variable.

Appropriate data

- One-way data plus a blocking variable. That is, one measurement variable in two or more groups, where observations are paired within levels of a blocking variable
- Dependent variable is ordinal, and specified in R as an ordered factor variable
- Independent variable is a factor with two or more levels. That is, two or more groups. The blocking variable is also a factor variable
- The data is arranged in a complete block design, with one or more observations per cell

Hypotheses

- Null hypothesis: The response of the dependent variable among groups are equal.
- Alternative hypothesis (two-sided): The response of the dependent variable among groups are not equal.

Interpretation

- Reporting significant results for the omnibus test as “Significant differences were found in the response among groups.” is acceptable. Alternatively, “A significant effect for Independent Variable on Dependent Variable was found when controlling for the effect of Blocking Variable.”
- Reporting significant results for mean separation post-hoc tests as “Response of Dependent Variable for group A was different than that for group B.” is acceptable.

Other notes and alternative tests

Ordinal regression is an alternative.

The traditional nonparametric tests Friedman, Quade, Paired rank-sum test, or Sign test may be alternatives depending on the design of the experiment.

Packages used in this chapter

The packages used in this chapter include:

- psych
- lattice
- FSA
- coin
- rcompanion
- multcompView
- ggplot2

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(coin)){install.packages("coin")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(ggplot2)){install.packages("ggplot2")}
```

One-way ordinal permutation test of symmetry example

This example re-visits the Belcher data from the *Friedman Test* chapter. Note that each instructor is rated by each of eight raters. Because of this, we want to stratify the responses by *Rater*.

It answers the question, “Are the scores significantly different among the five speakers when controlling for the effect of the different raters?”

Note that a new variable, *Likert.f*, is created for the Likert scores as an ordered factor variable. It is necessary that the dependent variable passed to *symmetry_test* be an ordered factor variable for it to be treated as an ordinal variable.

```
Input ="
Instructor      Rater  Likert
'Bob Belcher'    a      4
'Bob Belcher'    b      5
'Bob Belcher'    c      4
'Bob Belcher'    d      6
'Bob Belcher'    e      6
'Bob Belcher'    f      6
'Bob Belcher'    g     10
```

```
'Bob Belcher'      h     6
'Linda Belcher'    a     8
'Linda Belcher'    b     6
'Linda Belcher'    c     8
'Linda Belcher'    d     8
'Linda Belcher'    e     8
'Linda Belcher'    f     7
'Linda Belcher'    g    10
'Linda Belcher'    h     9
'Tina Belcher'     a     7
'Tina Belcher'     b     5
'Tina Belcher'     c     7
'Tina Belcher'     d     8
'Tina Belcher'     e     8
'Tina Belcher'     f     9
'Tina Belcher'     g    10
'Tina Belcher'     h     9
'Gene Belcher'     a     6
'Gene Belcher'     b     4
'Gene Belcher'     c     5
'Gene Belcher'     d     5
'Gene Belcher'     e     6
'Gene Belcher'     f     6
'Gene Belcher'     g     5
'Gene Belcher'     h     5
'Louise Belcher'   a     8
'Louise Belcher'   b     7
'Louise Belcher'   c     8
'Louise Belcher'   d     8
'Louise Belcher'   e     9
'Louise Belcher'   f     9
'Louise Belcher'   g     8
'Louise Belcher'   h    10
")
Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them
Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))

### Create a new variable which is the likert scores as an ordered factor
Data$Likert.f = factor(Data$Likert,
                      ordered=TRUE)

### Check the data frame
library(psych)
headTail(Data)
```

```

str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

```

Summarize data treating Likert scores as factors

Note that the variable we want to count is *Likert.f*, which is a factor variable. Counts for *Likert.f* are cross tabulated over values of *Instructor*. The *prop.table* function translates a table into proportions. The *margin=1* option indicates that the proportions are calculated for each row.

```

xtabs( ~ Instructor + Likert.f,
      data = Data)

          Likert.f
Instructor    4 5 6 7 8 9 10
  Bob Belcher  2 1 4 0 0 0  1
  Linda Belcher 0 0 1 1 4 1  1
  Tina Belcher 0 1 0 2 2 2  1
  Gene Belcher 1 4 3 0 0 0  0
  Louise Belcher 0 0 0 1 4 2  1

XT = xtabs( ~ Instructor + Likert.f,
            data = Data)

prop.table(XT,
           margin = 1)

          Likert.f
Instructor    4     5     6     7     8     9     10
  Bob Belcher  0.250 0.125 0.500 0.000 0.000 0.000 0.125
  Linda Belcher 0.000 0.000 0.125 0.125 0.500 0.125 0.125
  Tina Belcher 0.000 0.125 0.000 0.250 0.250 0.250 0.125
  Gene Belcher 0.125 0.500 0.375 0.000 0.000 0.000 0.000
  Louise Belcher 0.000 0.000 0.000 0.125 0.500 0.250 0.125

```

Bar plots by group

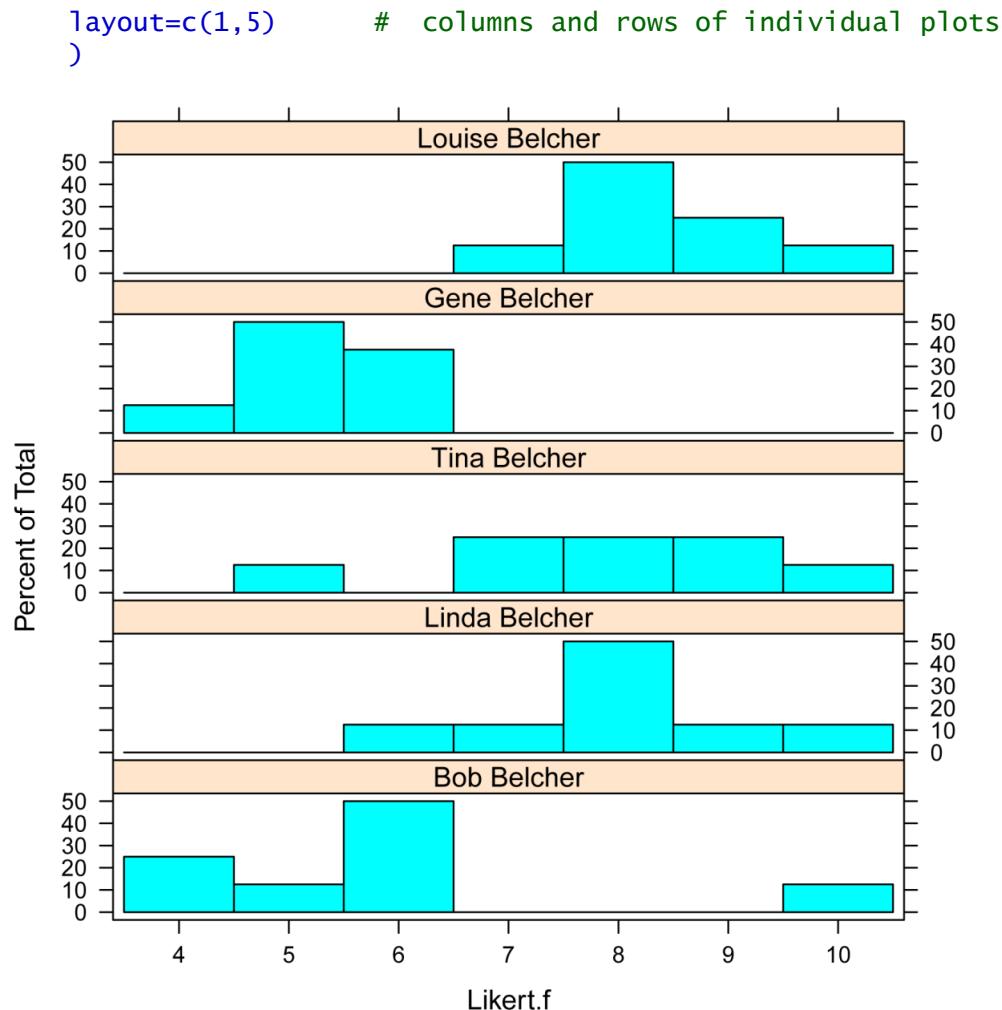
Note that the variable we want to count is *Likert.f*, which is a factor variable. Counts for *Likert.f* are presented for values of *Speaker*. Also note that the bar plots don't show the effect of the blocking variable.

```

library(lattice)

histogram(~ Likert.f | Instructor,
          data=Data,

```



Summarize data treating Likert scores as numeric

It may be useful to look at the minimum, first quartile, median, third quartile, and maximum for *Likert* for each group.

```
library(FSA)

Summarize(Likert ~ Instructor,
          data=Data,
          digits=3)
```

Instructor	n	mean	sd	min	Q1	median	Q3	max	perczero
1 Bob Belcher	8	5.875	1.885	4	4.75	6	6.00	10	0
2 Linda Belcher	8	8.000	1.195	6	7.75	8	8.25	10	0
3 Tina Belcher	8	7.875	1.553	5	7.00	8	9.00	10	0
4 Gene Belcher	8	5.250	0.707	4	5.00	5	6.00	6	0
5 Louise Belcher	8	8.375	0.916	7	8.00	8	9.00	10	0

Permutation symmetry test

Note that the dependent variable is an ordered factor variable, *Likert.f*. *Instructor* is the independent variable, and *Rater* is the blocking variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *library(coin); ?symmetry_test*.

```
library(coin)

symmetry_test(Likert.f ~ Instructor | Rater,
  data = Data)

Asymptotic General Symmetry Test

maxT = 3.4036, p-value = 0.003416
```

Post-hoc test: pairwise permutation tests

If the symmetry test is significant, a post-hoc analysis can be performed to determine which groups differ from which other groups.

The *pairwisePermutationSymmetry* and *pairwisePermutationSymmetryMatrix* functions in the *rcompanion* package conduct permutation tests across groups in a pairwise manner. See *library(rcompanion); ?pairwisePermutationSymmetry* for further details.

Because the post-hoc test will produce multiple *p*-values, adjustments to the *p*-values can be made to avoid inflating the possibility of making a type-I error. Here, the method of adjustment is indicated with the *method* option. There are a variety of methods for controlling the familywise error rate or for controlling the false discovery rate. See *?p.adjust* for details on these methods.

Before conducting the pairwise tests, we will re-order the levels of the grouping variable by the median of each group. This makes interpretation of the pairwise comparisons and compact letter display easier.

Table output and compact letter display

```
### Order groups by median

Data$Instructor = factor(Data$Instructor,
  levels = c("Linda Belcher", "Louise Belcher",
  "Tina Belcher", "Bob Belcher",
  "Gene Belcher"))

### Pairwise permutation tests

library(rcompanion)

PT = pairwisePermutationSymmetry(Likert.f ~ Instructor | Rater,
  data = Data,
  method = "fdr")

PT
```

	Comparison	w	p.value	p.adjust
1	Linda Belcher - Louise Belcher = 0	-0.9045	0.3657	0.42680
2	Linda Belcher - Tina Belcher = 0	0.378	0.7055	0.70550
3	Linda Belcher - Bob Belcher = 0	-2.38	0.01729	0.03458
4	Linda Belcher - Gene Belcher = 0	2.593	0.009522	0.03458
5	Louise Belcher - Tina Belcher = 0	-1.155	0.2482	0.35460
6	Louise Belcher - Bob Belcher = 0	-2.265	0.02354	0.03923
7	Louise Belcher - Gene Belcher = 0	-2.744	0.006068	0.03458
8	Tina Belcher - Bob Belcher = 0	-2.412	0.01586	0.03458
9	Tina Belcher - Gene Belcher = 0	2.528	0.01147	0.03458
10	Bob Belcher - Gene Belcher = 0	0.8704	0.3841	0.42680

```
### Compact letter display
```

```
library(rcompanion)

cldList(p.adjust ~ Comparison,
        data = PT,
        threshold = 0.05)
```

	Group	Letter	MonoLetter
1	LindaBelcher	a	a
2	LouiseBelcher	a	a
3	TinaBelcher	a	a
4	BobBelcher	b	b
5	GeneBelcher	b	b

Groups sharing a letter are not significantly different (alpha = 0.05).

Matrix output and compact letter display

A compact letter display condenses a table of *p*-values into a simpler format. In the output, groups sharing a same letter are not significantly different. Compact letter displays are a clear and succinct way to present results of multiple comparisons.

Here the *fdr* *p*-value adjustment method is used. See *?p.adjust* for details on available methods.

The code creates a matrix of *p*-values called *PM* which is then passed to the *multcompLetters* function to be converted to a compact letter display.

```
### Order groups by median

Data$Instructor = factor(Data$Instructor,
                           levels = c("Linda Belcher", "Louise Belcher",
                                     "Tina Belcher", "Bob Belcher",
                                     "Gene Belcher"))

### Pairwise permutation tests
```

```

library(rcompanion)

PM = pairwisePermutationSymmetryMatrix(Likert.f ~ Instructor | Rater,
                                         data = Data,
                                         method = "fdr")

PM

# Produce compact letter display

library(multcompView)

multcompLetters(PM$Adjusted,
                 compare = "<",
                 threshold = 0.05, # p-value to use as significance threshold
                 Letters = letters,
                 reversed = FALSE)

Linda Belcher Louise Belcher Tina Belcher Bob Belcher Gene Belcher
      "a"           "a"       "a"       "b"       "b"

### Groups sharing a letter are not significantly different (alpha = 0.05).

```

Plot of medians and confidence intervals

The following code uses the *groupwiseMedian* function to produce a data frame of medians for each speaker along with the 95% confidence intervals for each median with the percentile method. See *library(rcompanion); ?groupwiseMedian* for further options.

These medians are then plotted, with their confidence intervals shown as error bars. The grouping letters from the multiple comparisons are added.

```

### Order groups by original order

Data$Instructor = factor(Data$Instructor,
                           levels = c("Bob Belcher", "Linda Belcher", "Tina Belcher",
                                     "Gene Belcher", "Louise Belcher"))

### Create data frame of medians and confidence intervals

library(rcompanion)

Sum = groupwiseMedian(Likert ~ Instructor,
                      data = Data,
                      conf = 0.95,
                      R = 5000,
                      percentile = TRUE,
                      bca = FALSE,
                      digits = 3)

Sum

```

```
### Prepare Labels

X = 1:5

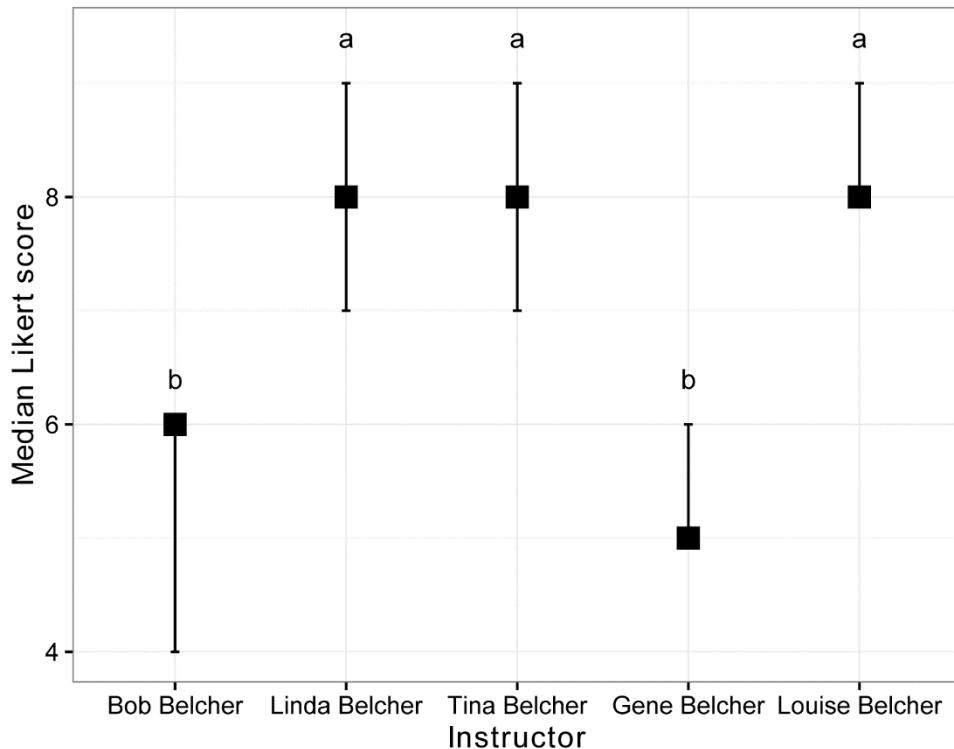
Y = Sum$Percentile.upper + 0.4

Label = c("b", "a", "a", "b", "a")

### Plot

library(ggplot2)

ggplot(Sum,           ### The data frame to use.
       aes(x = Instructor,
           y = Median)) +
  geom_errorbar(aes(ymax = Percentile.upper,
                    ymin = Percentile.lower),
                width = 0.05,
                size = 0.5) +
  geom_point(shape = 15,
             size = 4) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Median Likert score") +
  annotate("text",
          x = X,
          y = Y,
          label = Label)
```



Plot of median Likert score versus Instructor. Error bars indicate the 95% confidence intervals for the median with the percentile method.

Comparison of methods for symmetry tests for ordinal data

<u>Data</u>	<u>Test</u>	<u>p-value</u>	<u>Mean separation</u>
Pooh Time	Permutation symmetry	0.19	
Pooh Time	Ordinal regression	0.002	
Pooh Time	Two-sample paired rank-sum	0.02	
Bob Belcher et al.	Permutation symmetry	0.003	a a a b b
Bob Belcher et al.	Ordinal regression	0.00000009	a a a b b
Bob Belcher et al.	Friedman (Conover)	0.0001	a a a b b
Bob Belcher et al.	Quade (pairwise)	0.0002	a a a ab b

Permutation Tests for Medians and Percentiles

Permutation tests can be used to compare medians or percentiles among groups. This is useful, for example, to compare the 25th percentile or 75th percentile among groups.

The examples presented here use the *percentileTest* function in the *rcompanion* package, which can compare only two groups. This approach may be more powerful than using Mood's median test for medians. For more complicated models, quantile regression could be used.

The *percentileTest* function can also be used for means, variance, interquartile range, or proportion of observations less than a threshold value.

In general, there are not assumptions about the distributions of the data in the groups. Medians and percentiles are meaningful for ordinal or discrete data. If the data within groups is not symmetrical, means or variances may not be useful.

Pairwise tests

Pairwise tests across multiple groups can be conducted with the *pairwisePercentileTest* function.

Confidence intervals for percentiles

Confidence intervals for percentiles across groups can be determined with the *groupwisePercentile* function.

Packages used in this chapter

The packages used in this chapter include:

- FSA
- ggplot2
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(FSA)){install.packages("FSA")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Examples

The examples in this chapter will revisit the income data from the *Choosing a Statistical Test* chapter, with the addition of data for a third town.

```
### Import and modify the data
```

```
Two Towns = read.table("http://rcompanion.org/documents/Two Towns.csv",
header=TRUE, sep=",")
```

```

Town.A = TwoTowns$Income[TwoTowns$Town=="Town.A"]
Town.B = TwoTowns$Income[TwoTowns$Town=="Town.B"]
Town.C = TwoTowns$Income[TwoTowns$Town=="Town.B"] + 20000

Income = c(Town.A, Town.B, Town.C)
Town = c(rep("Town.A", 101), rep("Town.B", 101), rep("Town.C", 101))
ThreeTowns = data.frame(Town, Income)

str(ThreeTowns)

### Statistics by groups

library(FSA)

Summarize(Income ~ Town, data = ThreeTowns, digits = 3)

  Town   n     mean       sd    min    Q1 median      Q3    max
1 Town.A 101 48146.43 10851.67 23557 40971  48007  56421 77774
2 Town.B 101 115275.22 163878.16 29050 34142  47223 108216 880027
3 Town.C 101 135275.22 163878.16 49050 54142  67223 128216 900027

```

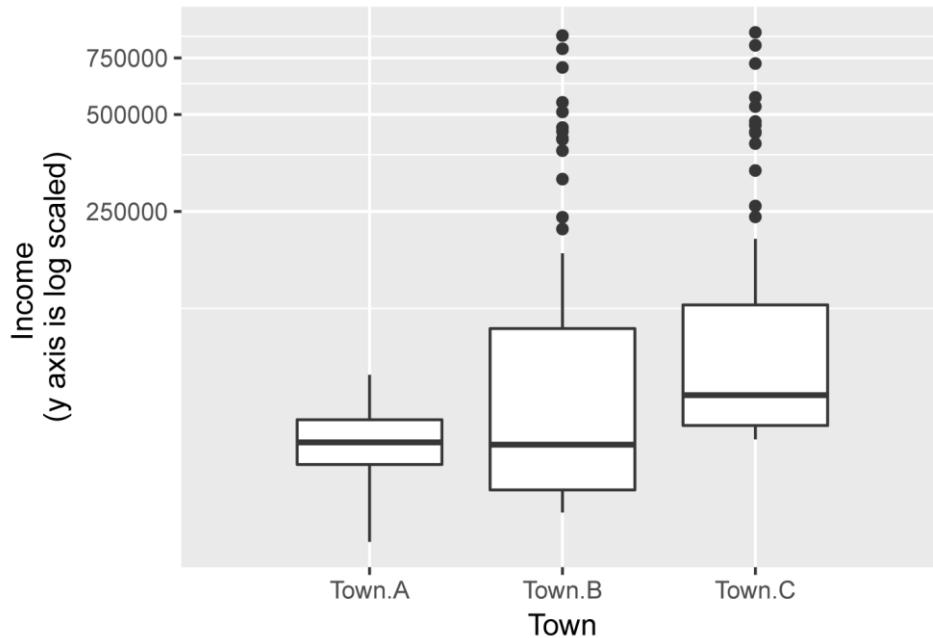
Box plots

```

library(ggplot2)

ggplot(ThreeTowns,
       aes(x = Town, y = Income)) +
  geom_boxplot() +
  coord_trans(y = "log10") +
  ylab("Income \n(y axis is log scaled)") +
  xlab("Town")

```



Confidence intervals for percentiles

Note that the percentiles below match Q_1 , *median*, and Q_3 above, respectively, except that they are rounded to three significant figures. The confidence limits are determined by bootstrap, so your values may differ from the values here.

```
library(rcompanion)
```

```
groupwisePercentile(Income ~ Town,
                     data = ThreeTowns,
                     tau = 0.25,
                     R = 1000)
```

Town	n	tau	Percentile	Conf.level	Bca.lower	Bca.upper
1 Town.A	101	0.25	41000	0.95	36000	43600
2 Town.B	101	0.25	34100	0.95	31400	36700
3 Town.C	101	0.25	54100	0.95	51400	56700

```
groupwisePercentile(Income ~ Town,
                     data = ThreeTowns,
                     tau = 0.50,
                     R = 1000)
```

Town	n	tau	Percentile	Conf.level	Bca.lower	Bca.upper
1 Town.A	101	0.5	48000	0.95	45600	50100
2 Town.B	101	0.5	47200	0.95	40700	59800
3 Town.C	101	0.5	67200	0.95	60800	79800

```
groupwisePercentile(Income ~ Town,
                     data = ThreeTowns,
```

```
tau = 0.75,
R   = 1000)
```

	Town	n	tau	Percentile	Conf.level	Bca.lower	Bca.upper
1	Town.A	101	0.75	56400	0.95	52400	59400
2	Town.B	101	0.75	108000	0.95	70900	144000
3	Town.C	101	0.75	128000	0.95	91500	164000

Test for percentiles between two groups

The *percentileTest* function compares only two groups. It uses the first two levels of the grouping variable, treating it as a factor. The *p*-value is determined by permutation test, so your result may differ from the value here. If the test takes too long, the *r* value can be decreased, but a greater *r* value will result in a more precise *p*-value.

```
percentileTest(Income ~ Town,
               data = ThreeTowns,
               test = "percentile",
               tau = 0.25,
               r    = 5000)

$Test
      Formula      Data     Test  tau
1 Income ~ Town ThreeTowns percentile 0.25

$Summary
      n   mean     sd   min   p25 median   p75   max   iqr
1 Town.A 101 48150 10850 23560 40970 48010 56420 77770 15450
2 Town.B 101 115300 163900 29050 34140 47220 108200 880000 74070

$Result
      p.value
1 p-value 0.0162

percentileTest(Income ~ Town,
               data = ThreeTowns,
               test = "median",
               r    = 5000)

$Test
      Formula      Data     Test
1 Income ~ Town ThreeTowns median

$Summary
      n   mean     sd   min   p25 median   p75   max   iqr
1 Town.A 101 48150 10850 23560 40970 48010 56420 77770 15450
2 Town.B 101 115300 163900 29050 34140 47220 108200 880000 74070

$Result
      p.value
1 p-value 0.7458
```

```

percentileTest(Income ~ Town,
               data = ThreeTowns,
               test = "percentile",
               tau = 0.75,
               r = 5000)

$Test
      Formula      Data     Test  tau
1 Income ~ Town ThreeTowns percentile 0.75

$Summary
      n   mean    sd   min   p25 median   p75   max   iqr
1 Town.A 101 48150 10850 23560 40970 48010 56420 77770 15450
2 Town.B 101 115300 163900 29050 34140 47220 108200 880000 74070

$Result
      p.value
1 p-value      0

```

Test for percentiles among multiple groups

The *pairwisePercentileTest* function will compare percentiles among multiple groups. The *p*-values are determined by permutation test, so your results may differ from the values here. If the test takes too long, the *r* value can be decreased, but a greater *r* value will result in a more precise *p*-value.

```

PT25 = pairwisePercentileTest(Income ~ Town,
                             data = ThreeTowns,
                             test = "percentile",
                             tau = 0.25,
                             r = 5000)

```

```

PT25
      Comparison p.value p.adjust
1 Town.A - Town.B = 0 0.0166 0.0166
2 Town.A - Town.C = 0 0 0.0000
3 Town.B - Town.C = 0 0 0.0000

```

```
cldList(p.adjust ~ Comparison, data = PT25)
```

	Group	Letter	MonoLetter
1	Town.A	a	a
2	Town.B	b	b
3	Town.C	c	c

```

PT50 = pairwisePercentileTest(Income ~ Town,
                             data = ThreeTowns,
                             test = "median",
                             r = 5000)

```

PT50

```
Comparison p.value p.adjust
1 Town.A - Town.B = 0 0.7538 0.753800
2 Town.A - Town.C = 0 0 0.000000
3 Town.B - Town.C = 0 0.00155 0.002325
```

```
cldList(p.adjust ~ Comparison, data = PT50)
```

	Group	Letter	MonoLetter
1	Town.A	a	a
2	Town.B	a	a
3	Town.C	b	b

```
PT75 = pairwisePercentileTest(Income ~ Town,
                               data = ThreeTowns,
                               test = "percentile",
                               tau = 0.75,
                               r = 5000)
```

PT75

```
Comparison p.value p.adjust
1 Town.A - Town.B = 0 0 0.000
2 Town.A - Town.C = 0 0 0.000
3 Town.B - Town.C = 0 0.456 0.456
```

```
cldList(p.adjust ~ Comparison, data = PT75)
```

	Group	Letter	MonoLetter
1	Town.A	a	a
2	Town.B	b	b
3	Town.C	b	b

Summary table of results

	----- Grouping letters * -----		
	25 th percentile	Median	75 th percentile
Town			
A	a	a	a
B	b	a	b
C	c	b	b

* Groups sharing a letter in each column are not significantly different.

Tests for Ordinal Data in Tables

Association Tests for Ordinal Tables

The linear-by-linear test can be used to test the association among variables in a contingency table with ordered categories (Agresti, 2007). This test or a test with a similar function is sometimes called “ordinal chi-square” test.

In Agresti, the method used is called the *linear-by-linear association model*. In R, the test can be performed by permutation test with the *coin* package.

An association test can also be performed on a contingency table with one ordered nominal variable and one non-ordered nominal variable. The Cochran–Armitage test is a special case of this when the non-ordered variable has only two variables.

Most of the examples in this chapter use two-dimensional tables, although the *coin* package can handle three-dimensional tables. For three-dimensional analyses, it may be easier to use data in the long format, as is shown in the final example in this chapter.

Packages used in this chapter

The packages used in this chapter include:

- *coin*
- *rcompanion*

The following commands will install these packages if they are not already installed:

```
if(!require(coin)){install.packages("coin")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Linear-by-linear test for ordered contingency tables

The *lbl_test* function in the *coin* package will automatically treat the variables as ordered, with the levels in the table ordered from smallest to largest. By default, the levels are equally spaced, but the *scores* option can be used to specify the distance between the levels of each variable.

The null hypothesis for the linear-by-linear test is that there is no association among the variables in the table. A significant *p*-value suggests that there is an association. This is similar to a chi-square test, except that the categories are ordered in nature.

Example of linear-by-linear test 1

For this hypothetical example, farmers were surveyed about how often they use some best management practice. Responses are organized according to the size of the operation. Both variables in the contingency table are ordered categories.

Note the placement of the initial quote mark in the *Input* function.

```

Input =(
"Adopt      Always  Sometimes  Never
Size
Hobbies    0        1        5
Mom-and-pop 2        3        4
Small      4        4        4
Medium     3        2        0
Large       2        0        0
")

Tabla = as.table(read.ftable(textConnection(Input)))

Tabla

sum (Tabla)

prop.table(Tabla,
           margin = NULL)  ### proportion in the table

      Size      Always  Sometimes  Never
      Hobbies    0.00000000 0.02941176 0.14705882
      Mom-and-pop 0.05882353 0.08823529 0.11764706
      Small      0.11764706 0.11764706 0.11764706
      Medium     0.08823529 0.05882353 0.00000000
      Large       0.05882353 0.00000000 0.00000000

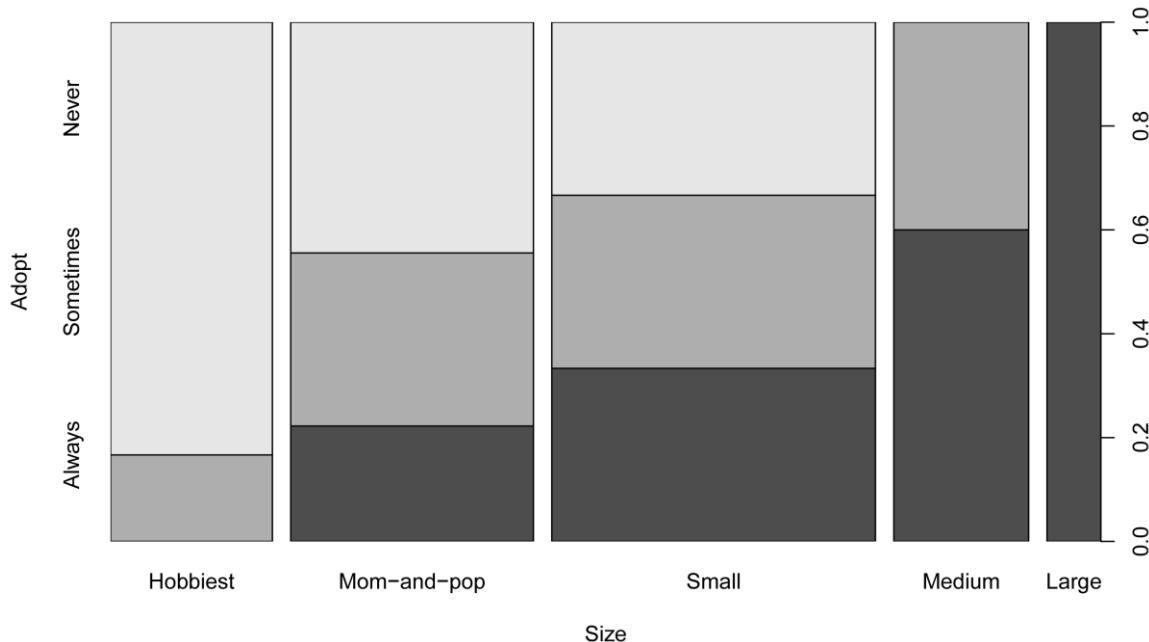
```

```

library(coin)

spineplot(Tabla)

```



Spine plot for each Size showing the proportion of Always (dark gray), Sometimes (medium gray), and Never (light gray).

```

library(coin)

LXL = lbl_test(Tabla)

LXL

Asymptotic Linear-by-Linear Association Test

data: Adopt (ordered) by Size (Hobbiest < Mom-and-pop < Small < Medium < Large)
z = -3.3276, p-value = 0.0008761

statistic(LXL)^2

11.07262

```

Compare to chi-square test without ordered categories

```

chisq = chisq_test(Tabla)

Chisq

Asymptotic Pearson Chi-Squared Test

data: Adopt by Size (Hobbiest, Mom-and-pop, Small, Medium, Large)
chi-squared = 13.495, df = 8, p-value = 0.09593

```

Example of linear-by-linear test 2

The following example revisits the data from *Converting Numeric Data to Categories* chapter. It tests if there is an association between *Tired* and *Happy*. Both variables in the contingency table are ordered categories, with the levels of each being 1, 2, 3, 4, and 5.

```

Input =(
 "Tired 1 2 3 5
 Happy
 1 0 0 0 3
 2 0 0 0 2
 3 0 0 3 0
 4 2 0 0 0
 5 3 2 0 5
")

Tabla = as.table(read.ftable(textConnection(Input)))

Tabla

sum (Tabla)

```

```

prop.table(Tabla,
           margin = NULL)    ### proportion in the table

      Tired
Happy   1   2   3   5
  1 0.00 0.00 0.00 0.15
  2 0.00 0.00 0.00 0.10
  3 0.00 0.00 0.15 0.00
  4 0.10 0.00 0.00 0.00
  5 0.15 0.10 0.00 0.25

library(coin)

LXL = lbl_test(Tabla)

LXL

Asymptotic Linear-by-Linear Association Test

data: Tired (ordered) by Happy (1 < 2 < 3 < 4 < 5)
z = -1.8878, p-value = 0.05906

```

Extended Cochran-Armitage test

The *chisq.test* function in the *coin* package can be used to conduct a test of association for a contingency table with one ordered nominal variable and one non-ordered nominal variable. The Cochran-Armitage test is a special case of this where the non-ordered variable has only two categories.

The *scores* option is used to indicate which variable should be treated as ordered, and the spacing of the levels of this variable.

Example of extended Cochran-Armitage test 1

For this hypothetical example, students were surveyed about how often they eat breakfast and how they travel to school. *Breakfast* is treated as ordered, and *Travel* is treated as nominal.

Note the placement of the initial quote mark in the *Input* function.

```

Input =(
"Breakfast  Never  Rarely  Sometimes  Often  Always
Travel
Walk       6       9       6       5       2
Bus        2       5       8       5       3
Drive      2       4       6       8       8
")

```

```

Tabla = as.table(read.ftable(textConnection(Input)))

```

```
Tabla
```

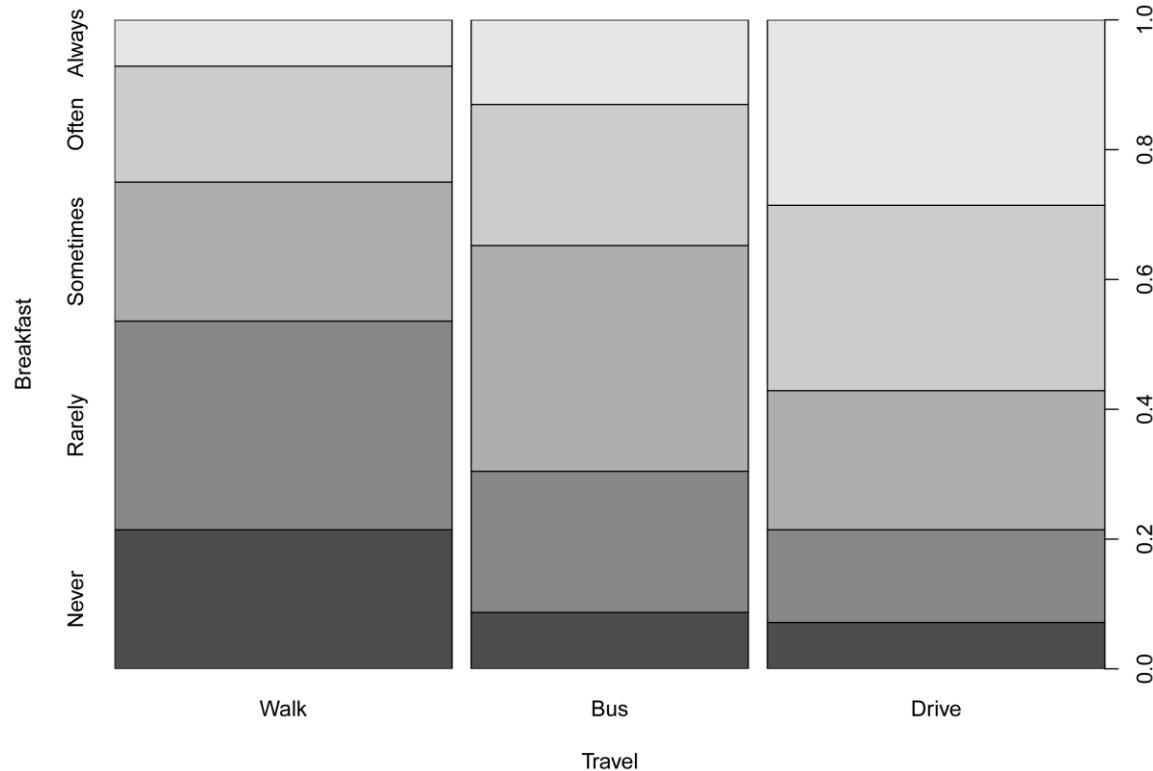
```
sum (Tabla)
```

```
prop.table(Tabla,
           margin = 1)    ### proportion in each row
```

		Breakfast				
Travel		Never	Rarely	Sometimes	Often	Always
Walk	0.21428571	0.32142857	0.21428571	0.17857143	0.07142857	
Bus	0.08695652	0.21739130	0.34782609	0.21739130	0.13043478	
Drive	0.07142857	0.14285714	0.21428571	0.28571429	0.28571429	

```
library(coin)
```

```
spineplot(Tabla)
```



```
library(coin)
```

```
Test = chisq_test(Tabla,
                   scores = list("Breakfast" = c(-2, -1, 0, 1, 2)))
```

```
Test
```

Asymptotic Generalized Pearson Chi-Squared Test

```

data: Breakfast (ordered) by Travel (Walk, Bus, Drive)
chi-squared = 8.6739, df = 2, p-value = 0.01308

statistic(Test)^2
[1] 75.23709

```

Post-hoc analysis

For a contingency table with one ordered variable and one non-ordered variable, it makes sense to analyze the component tables with pairwise comparisons of the levels of the non-ordered variable.

Results of the compact letter display will be easier to interpret if the table is ordered so that the first row, in this case, ranks highest or lowest in the ordered variable, and so on.

```

library(rcompanion)

PT = pairwiseOrdinalIndependence(Tabla,
                                compare="row")

```

```

PT

Comparison p.value p.adjust
1 Walk : Bus 0.12800 0.1570
2 Walk : Drive 0.00462 0.0139
3 Bus : Drive 0.15700 0.1570

```

```

library(rcompanion)

cldList(p.value ~ Comparison,
        data      = PT,
        threshold = 0.05)

Group Letter MonoLetter
1 Walk     a         a
2 Bus      ab        ab
3 Drive    b         b

```

Example of extended Cochran-Armitage test 2

This example revisits the example from Kruskal-Wallis chapter. *Likert* is treated as ordered, and *Speaker* is treated as non-ordered.

Note the placement of the initial quote mark in the *Input* function.

```

Input =(
"Likert 1 2 3 4 5
Speaker
Pooh   0 0 1 6 3

```

```

Piglet 1 6 2 1 0
Tigger 0 0 2 6 2
")

Tabla = as.table(read.ftable(textConnection(Input)))

Tabla
sum(Tabla)

prop.table(Tabla,
           margin = 1)    ### proportion in each row

  Likert
Speaker   1   2   3   4   5
  Pooh    0.0 0.0 0.1 0.6 0.3
  Piglet  0.1 0.6 0.2 0.1 0.0
  Tigger  0.0 0.0 0.2 0.6 0.2

library(coin)

Test = chisq_test(Tabla,
                  scores = list("Likert" = c(1, 2, 3, 4, 5)))

Test

  Asymptotic Generalized Pearson Chi-Squared Test

  data: Likert (ordered) by Speaker (Pooh, Piglet, Tigger)
  chi-squared = 18.423, df = 2, p-value = 9.991e-05

library(rcompanion)

PT = pairwiseOrdinalIndependence(Tabla,
                                  compare="row")

PT

  Comparison  p.value p.adjust
1 Pooh : Piglet 0.000310 0.000902
2 Pooh : Tigger 0.474000 0.474000
3 Piglet : Tigger 0.000601 0.000902

library(rcompanion)

cldList(p.value ~ Comparison,
        data      = PT,
        threshold = 0.05)

Group Letter MonoLetter

```

1 Pooh	a	a
2 Piglet	b	b
3 Tigger	a	a

Long-form data and three-dimensional contingency tables

The tests in this chapter can also be performed on data in long format rather than in table format. Each row of data can represent a single observation, or one variable can hold counts of each category, as the *Count* variable does below.

For the *independence_test* function in the *coin* package to handle ordered and non-ordered variables properly, the user need only specify which variables are to be considered ordered.

The *independence_test* function can also handle a stratification variable. That is, a two-dimensional table within each level of the stratification variable.

```
Input ="
Crop      Size      Adopt      Count
Nursery   Hobbiest  Always     0
Nursery   Hobbiest  Sometimes  1
Nursery   Hobbiest  Never      3
Nursery   Mom-and-pop Always    1
Nursery   Mom-and-pop Sometimes 1
Nursery   Mom-and-pop Never     2
Nursery   Small     Always     2
Nursery   Small     Sometimes  3
Nursery   Small     Never      2
Nursery   Medium    Always     2
Nursery   Medium    Sometimes  1
Nursery   Medium    Never      0
Nursery   Large     Always     1
Nursery   Large     Sometimes  0
Nursery   Large     Never      0
Vegetable Hobbiest  Always     0
Vegetable Hobbiest  Sometimes  0
Vegetable Hobbiest  Never      2
Vegetable Mom-and-pop Always    1
Vegetable Mom-and-pop Sometimes 2
Vegetable Mom-and-pop Never     2
Vegetable Small     Always     2
Vegetable Small     Sometimes  1
Vegetable Small     Never      2
Vegetable Medium    Always     1
Vegetable Medium    Sometimes  1
Vegetable Medium    Never      0
Vegetable Large    Always     1
Vegetable Large    Sometimes  0
Vegetable Large    Never      0
")
```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```

### Tell R which variables are ordered

Data$Size = factor(Data$Size,
                    ordered = TRUE,
                    levels=unique(Data$Size))

Data$Adopt = factor(Data$Adopt,
                     ordered = TRUE,
                     levels=unique(Data$Adopt))

### Make sure the ordered levels are in the proper order

str(Data$Adopt)

ord.factor w/ 3 levels "Always"<"Sometimes"<...: 1 2 3 1 2 3 1 2 3 1 ...

levels(Data$Adopt)

[1] "Always"    "Sometimes" "Never"

str(Data$Size)

ord.factor w/ 5 levels "Hobbiest"<"Mom-and-pop"<...: 1 1 1 2 2 2 3 3 3 4 ...

levels(Data$Size)

[1] "Hobbiest"   "Mom-and-pop" "Small"       "Medium"      "Large"

```

Two-dimensional table example

```

XT = xtabs(Count ~ Size + Adopt,
           data = Data)

XT

          Adopt
Size      Always Sometimes Never
Hobbiest      0        1      5
Mom-and-pop   2        3      4
Small         4        4      4
Medium        3        2      0
Large         2        0      0

```

```

spineplot(XT)

library(coin)

independence_test(Adopt ~ Size,

```

```
data = Data,
weights = ~ Count)
```

Asymptotic General Independence Test

```
data: Adopt (ordered) by Size (Hobbiest < Mom-and-pop < Small < Medium < Large)
z = -3.3276, p-value = 0.0008761
```

Three-dimensional table example

The following example adds *Crop* as a stratification variable.

```
ftable(xtabs(Count ~ Crop + Size + Adopt,
            data = Data))
```

		Adopt	Always	Sometimes	Never
Crop	Size				
Nursery	Hobbiest	0	1	3	
	Mom-and-pop	1	1	2	
	Small	2	3	2	
	Medium	2	1	0	
	Large	1	0	0	
Vegetable	Hobbiest	0	0	2	
	Mom-and-pop	1	2	2	
	Small	2	1	2	
	Medium	1	1	0	
	Large	1	0	0	

```
library(coin)
```

```
independence_test(Adopt ~ Size | Crop,
                  data = Data,
                  weights = ~ Count)
```

Asymptotic General Independence Test

```
data: Adopt (ordered) by Size (Hobbiest < Mom-and-pop < Small < Medium < Large)
stratified by Crop
z = -3.281, p-value = 0.001034
```

Spine plot for two-dimensional tables

```
XT1 = xtabs(Count ~ Size + Adopt,
            data = Data,
            subset = Data$Crop=="Nursery")
```

```
XT1
```

```
spineplot(XT1,
          main = "Nursery")
```

```

XT2 = xtabs(Count ~ Size + Adopt,
            data = Data,
            subset = Data$Crop=="Vegetable")

XT2

spineplot(XT2,
          main = "Vegetable")

```

References

Agresti, A. 2007. An Introduction to Categorical Data Analysis 2nd Edition. Wiley-Interscience.

Measures of Association for Ordinal Tables

Measures of association for one ordinal variable and one nominal variable

The statistics Freeman's *theta* and *epsilon*-squared are used to gauge the strength of the association between one ordinal variable and one nominal variable. Both of these statistics range from 0 to 1, with 0 indicating no association and 1 indicating perfect association.

As effect sizes, these statistics are not affected by the sample size per se. *epsilon*-squared is usually used as the effect size for a Kruskal–Wallis test, whereas Freeman's *theta* is most often used as an effect size for data arranged in a table, such as for a Cochran–Armitage test. However, it is my understanding that neither statistic assumes or prohibits one variable being designated as the dependent variable

Measures of association for two ordinal variables

Measures of association for ordinal variables include Somers' *D* (or *delta*), Kendall's *tau*-*b*, and Goodman and Kruskal's *gamma*.

Yule's *Q* is equivalent in magnitude to Goodman and Kruskal's *gamma* for a 2 x 2 table, but can be either positive or negative depending on the order of the cells in the table.

Polychoric and tetrachoric correlation

Polychoric correlation is used to measure the degree of correlation between two ordinal variables with the assumption that each ordinal variable is a discrete summary of an underlying (latent) normally distributed continuous variable. For example, if an ordinal variable *Height* were measured as *very short*, *short*, *average*, *tall*, *very tall*, one could assume that these categories represent actual height

measurements that are continuous and normally distributed. A similar assumption might be made for Likert items, for example on an *agree-disagree* spectrum.

Tetrachoric correlation is a special case of polychoric correlation when both variables are dichotomous.

Appropriate data

- One ordinal variable and one nominal variable, or two ordinal variables. Usually expressed as a contingency table.
- Experimental units aren't paired.

Hypotheses

- There are no hypotheses tested directly with these statistics.

Other notes and alternative tests

- Cramér's *V* and *phi* are used for tables with two nominal variables. Cohen's *w* is a variant. Goodman and Kruskal's *lambda* is also used.
- Tetrachoric and polychoric correlation are used for two ordinal variables when there is an assumption that the ordinal variables represent latent continuous variables underlying the ordinal variables.
- Biserial and polyserial correlation are used for one continuous variable and one ordinal (or dichotomous) variable, when there is an assumption that the ordinal variable represents a latent continuous variable.

Packages used in this chapter

The packages used in this chapter include:

- rcompanion
- psych
- DescTools

The following commands will install these packages if they are not already installed:

```
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(psych)){install.packages("psych")}
if(!require(DescTools)){install.packages("DescTools")}
```

Examples for Freeman's *theta* and *epsilon-squared*

The hypothetical *Breakfast* example from the previous chapter includes *Breakfast* as an ordinal variable and *Travel* as a nominal variable.

```
Input =(
"Breakfast  Never  Rarely  Sometimes  Often  Always
Travel
Walk       6       9       6       5       2
Bus        2       5       8       5       3
Drive      2       4       6       8       8
```

```
")
```

```
Tabla = as.table(read.ftable(textConnection(Input)))
```

```
Tabla
```

Freeman's theta

```
library(rcompanion)

freemanTheta(Tabla,
             group = "row")

Freeman.theta
0.312
```

Epsilon-squared

```
library(rcompanion)

epsilonSquared(Tabla,
               group = "row")

epsilon.squared
0.11
```

The hypothetical example with Pooh, Piglet, and Tigger includes *Likert* as an ordinal variable and *Speaker* as a nominal variable.

```
Input =(
"Likert  1 2 3 4 5
Speaker
Pooh     0 0 1 6 3
Piglet   1 6 2 1 0
Tigger   0 0 2 6 2
")

Tabla = as.table(read.ftable(textConnection(Input)))

Tabla
```

Freeman's theta

```
library(rcompanion)

freemanTheta(Tabla,
             group = "row")

Freeman.theta
0.64
```

Epsilon-squared

```
library(rcompanion)
epsilonSquared(Tabla,
               group = "row")

epsilon.squared
0.581
```

Additional examples for Freeman's *theta* and *epsilon-squared****Perfect association***

```
Input =(
"Ordinal" 1 2 3
Category
A          10 0 0
B          0 10 0
C          0 0 10
")

Tabla = as.table(read.ftable(textConnection(Input)))

Tabla
```

```
library(rcompanion)

freemanTheta(Tabla)

Freeman.theta
1
```

```
library(rcompanion)

epsilonSquared(Tabla)

epsilon.squared
1
```

Zero association

```
Input =(
"Ordinal" 1 2 3
Category
A          5 5 5
B          10 10 10
C          15 15 15
```

```

")
Tabla = as.table(read.ftable(textConnection(Input)))

Tabla

library(rcompanion)
freemanTheta(Tabla)

Freeman.theta
0

library(rcompanion)
epsilonSquared(Tabla)

epsilon.squared
0

```

Examples for Somers' *D*, Kendall's *tau-b*, and Goodman and Kruskal's *gamma*

First example

This example includes two ordinal variables, *Adopt* and *Size*.

```

Input =(
"Adopt      Always  Sometimes  Never
Size
Hobbies      0        1        5
Mom-and-pop  2        3        4
Small        4        4        4
Medium       3        2        0
Large         2        0        0
")

Tabla = as.table(read.ftable(textConnection(Input)))

Tabla

library(DescTools)

SomersDelta(Tabla,
            direction = "column",
            conf.level = 0.95)

      somers    lwr.ci    upr.ci
-0.4665127 -0.6452336 -0.2877918

```

```
### Somers' D for (Column | Row), with confidence interval
```

```
library(DescTools)
```

```
KendallTauB(Tabla,
conf.level = 0.95)
```

	tau_b	lwr.ci	ups.ci
	-0.4960301	-0.6967707	-0.2952895

```
### Kendall's tau-b with confidence interval
```

```
library(DescTools)
```

```
GoodmanKruskalGamma(Tabla,
conf.level = 0.95)
```

	gamma	lwr.ci	ups.ci
	-0.6778523	-0.9199026	-0.4358021

```
### Goodman and Kruskal's gamma with confidence interval
```

Second example

This example includes two ordinal variables, *Tired* and *Happy*.

```
Input =(
"Tired 1 2 3 5
Happy
1 0 0 0 3
2 0 0 0 2
3 0 0 3 0
4 2 0 0 0
5 3 2 0 5
")
Tabla = as.table(read.ftable(textConnection(Input)))
Tabla
```

```
library(DescTools)
```

```
SomersDelta(Tabla,
direction = "row",
conf.level = 0.95)
```

	somers	lwr.ci	ups.ci
	-0.32061069	-0.68212474	0.04090337

```
### Somers' D for (Row | Column), with confidence interval
```

```

library(DescTools)

KendallTauB(Tabla,
            conf.level = 0.95)

  tau_b      lwr.ci     ups.ci
-0.31351142 -0.67251606  0.04549322

### Kendall's tau-b with confidence interval

library(DescTools)

GoodmanKruskalGamma(Tabla,
                     conf.level = 0.95)

  gamma      lwr.ci     ups.ci
-0.42000000 -0.87863057  0.038630

### Goodman and Kruskal's gamma with confidence interval

```

Examples for polychoric correlation

First example

This example includes two ordinal variables, *Adopt* and *Size*. A single correlation coefficient is produced.

```

Input =(
"Adopt      Always  Sometimes  Never
Size
Hobbiest      0        1        5
Mom-and-pop   2        3        4
Small         4        4        4
Medium        3        2        0
Large         2        0        0
")
)

Tabla = as.table(read.ftable(textConnection(Input)))

Tabla

library(psych)

polychoric(Tabla,
           correct=FALSE)

$rho
[1] -0.678344

```

Second example

This example includes two ordinal variables, *Tired* and *Happy*.

```
Input =(
  "Tired  1 2 3 5
  Happy
    1  0 0 0 3
    2  0 0 0 2
    3  0 0 3 0
    4  2 0 0 0
    5  3 2 0 5
  ")
  Tabla = as.table(read.ftable(textConnection(Input)))
  Tabla

library(psych)
polychoric(Tabla,
  correct=FALSE)

$rho
[1] -0.495164
```

Third example

The following example revisits the Belcher family data. Here, we want to know if the scores among instructors are correlated. This makes sense since each rater rated each instructor.

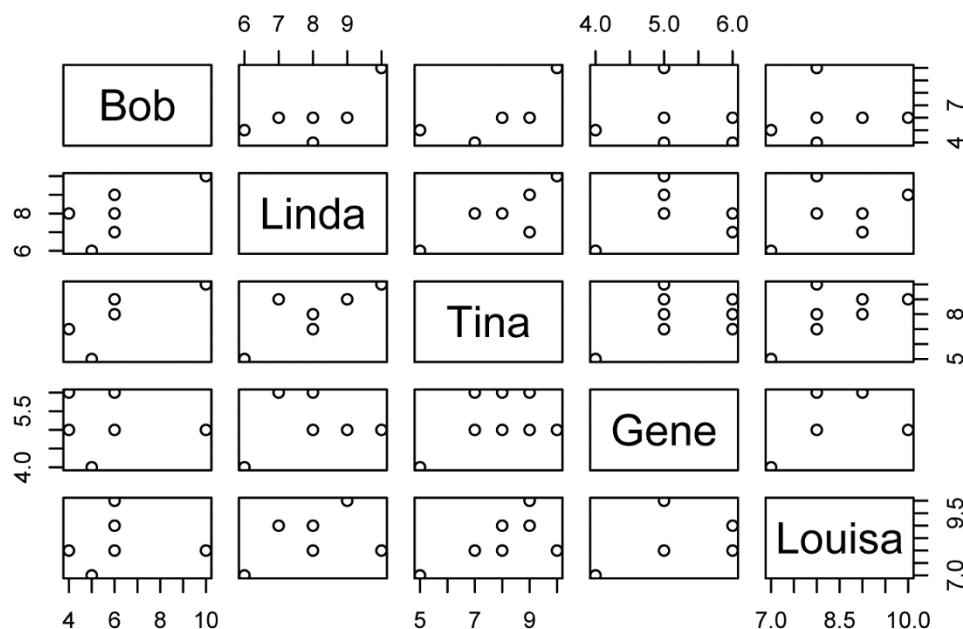
```
Input =""
Rater Bob Linda Tina Gene Louisa
a     4     8     7     6     8
b     5     6     5     4     7
c     4     8     7     5     8
d     6     8     8     5     8
e     6     8     8     6     9
f     6     7     9     6     9
g    10    10    10     5     8
h     6     9     9     5    10
  ")
  Data = read.table(textConnection(Input),header=TRUE)
  Data.num = Data[c("Bob", "Linda", "Tina", "Gene", "Louisa")]
  library(psych)
```

```
polychoric(Data.num,
            correct=FALSE)
```

Polychoric correlations

	Bob	Linda	Tina	Gene	Louis
Bob	1.00				
Linda	0.59	1.00			
Tina	0.88	0.76	1.00		
Gene	-0.03	0.16	0.37	1.00	
Louisa	0.35	0.47	0.70	0.64	1.00

```
pairs(data = Data.num,
      ~ Bob + Linda + Tina + Gene + Louisa)
```



Introduction to Linear Models

Linear models are used for a wide variety of statistical analyses. The basic concept is that a dependent variable can be predicted from a set of independent variables that are related in a linear fashion. This framework leads to models that are very flexible, and can include a variety of continuous or categorical independent variables.

All linear models make some assumptions about the underlying data. For one thing, the data should accord with the fact the model is composed of a linear combination of the effects. Non-linearity would suggest that the model effects be modified or that a different kind of model be used. In addition, each type of linear model usually makes some assumptions about the distribution of the data. For the reported statistics to be valid, it is essential to understand and check these assumptions for the specific type of model being used.

Types of linear models

In general, the type of model to be used is determined by the nature of the dependent variable.

General linear models

Readers may be familiar with linear regression, multiple linear regression, or analysis of variance (ANOVA).

These models can be considered part of larger category of linear models called *general linear model*. The dependent variable is continuous interval/ratio, and there are assumptions about the distribution of the data. These models are usually called *parametric* models or tests, although technically other types of models which assume properties about the distribution of the data are also parametric in nature.

Generalized linear models

Models for other types of dependent variables can be developed in a *generalized linear model* framework. This approach is similar to general linear model approach, except that there are different assumptions about the distribution of the data.

For example, ordinal dependent variables can be modeled with cumulative link models. Binary (yes/no) dependent variables can be modeled with logistic regression. Dependent variables of discrete counted quantities can be modeled with Poisson regression and similar techniques. Percent and proportion data can be modeled with beta regression.

Other linear models

Not all linear models are included in the *general linear model* and *generalized linear model* categories. For example, common quantile regression is a type of linear model not included in these categories.

Fitting models

For any type of linear model, some method is used to find the value for the parameters for the model which best fit the data. For simple models like linear regression or a simple analysis of variance, these parameters estimates could be found by hand calculations. Luckily, more complex models can be fit by

computer algorithm. General linear models are typically fit by ordinary least squares (OLS), whereas generalized linear models are typically fit by maximum likelihood estimation (MLE). OLS is actually a specific case of MLE that is valid only when the conditional distribution of the data are normal. These approaches to estimation are generally reliable, but the reader should be aware that there are cases where MLE may fail.

Formulae for specifying models in R

Most packages for specifying types of models in R use a similar grammar in the model formula.

The formula

$$y \sim x_1 + x_2 + x_1:x_2$$

Specifies y as the dependent variables, and x_1, x_2 , and the interaction of x_1 and x_2 as the independent variables.

The formula

$$y \sim x_1 | g$$

Specifies g as a stratification variable or as a conditional variable, that is x_1 given g .

The symbol 1 specifies an intercept for the model, so that

$$y \sim x_1 - 1$$

indicates that the intercept should not be included in the model, and

$$y \sim 1$$

indicates that only the intercept is to be included in the right side of this model.

The King article in the “References” section has some more detail on model specification syntax in R.

Data in data frames

Usually the variables in the model will be taken from a data frame. For example

$$y \sim x$$

will look for y and x in the global environment, whereas

$$y \sim x, \text{ data} = \text{Data}$$

will look for y and x in the data frame called *Data*.

Random effects

The syntax varies somewhat for random effects in models across packages, but

```
y ~ x, random = ~1|Subject
```

or

```
y ~ x +(1|Subject)
```

Specifies that y is the dependent variable, x is an independent variable, and *Subject* is an independent variable treated as random variable, specifically with an intercept fit for each level of *Subject*.

More on specifying random effects is discussed in the chapter *Repeated Measures ANOVA*.

Extracting model information from R

Each package varies on the methods used to extract information about the model, but some are relatively common across several packages. For other types of model objects, there may be methods to extract similar information with different functions.

Assuming that *model* has been defined as a model object by an appropriate function,

```
summary(model)
```

produces a summary of the model with estimates of the coefficients, and sometimes other useful information like a *p*-value for the model, or an *r-squared* or pseudo *R-squared* for the model.

```
library(car)
```

```
Anova(model)
```

for some models, will produce an analysis of variance table, or an analysis of deviance table.

```
plot(model)
```

will produce plots of the model, usually diagnostic plots.

```
anova(model1, model2)
```

will compare two models by an analysis of variance.

```
predict(model)
```

will report predicted values in the dependent variable from the model for each observation that went in to the model.

```
residuals(model)
```

will report residual values from the model.

```
str(model)
```

reports the structure of the information stored in the model object.

References

King, W.B. Model Formulae. ww2.coastal.edu/kingw/statistics/R-tutorials/formulae.html.

Using Random Effects in Models

This book will not investigate the concept of random effects in models in any substantial depth. The goal of this chapter is to empower the reader to include random effects in models in cases of paired data or repeated measures.

Random effects in models for paired and repeated measures

As an example, if we are measuring the left hand and right of several individuals, the measurements are paired within each individual. That is, we want to statistically match the left hand of *Individual A* to the right hand of *Individual A*, since we suppose that someone with a large left hand will have a large right hand. Therefore, the variable *Individual* will be included in the model as a random variable. Each *Individual* could be thought of has a block including a measurement for the left hand and a measurement for the right hand.

As a second example, imagine we are measuring the scores of instructors multiple times, and we can match each rater's score at each time to the same rater's score at the other times. So, we want to statistically match the score of *Rater A* at *Time 1* to the score of *Rater A* at *Time 2*, since we suppose that someone who scores an instructor high at one time might score an instructor high at another time. Therefore, the variable *Rater* will be included in the model as a random variable. Each *Rater* could be thought of as a block including a measurement for each of the times.

The concept of random variables

Getting a firm grasp of the concept of random variables is not an easy task, so it is okay to not fret too much about completely understanding the concept at this point. It is also helpful to keep in mind that whether a variable should be considered a fixed or random variable will depend on the interpretation of the person doing the analysis.

In the previous chapter, the variable *Speaker* was used as a *fixed effect* in the model. Conceptually, the idea is that we are interested in the effect of each of the specific levels of that variable. That is, we care specifically about Pooh and Piglet and their scores.

In many cases of blocking variables, though, we don't necessarily care about the values for those specific blocks.

For example, if we conducted a study in two different schools focusing on two different curricula, we may want to know about the effect of the curricula, but more or less chose the schools at random. We don't really care if Springfield Elementary had a higher score than Shelbyville Elementary. The two schools represent any two random schools, not those two specific schools. But we definitely want to include the *School* effect in the model, because we suppose that one school could do better with both curricula than would the other school.

Another example of this is when we have instructors rated by student raters. If we are studying the performance of the instructors, we don't necessarily care about how Nelson or Milhouse or Ralph as individuals rate instructors. They are representing students chosen at random. We just need to include the effect of the variable *Rater* in the model to statistically account for the fact that each rater might tend to rate all instructors lower or higher than other raters.

In these examples, *School* and *Rater* could be included in their respective models as random effects.

Mixed effects models

When a model includes both fixed effects and random effects, it is called a *mixed effects* model.

Optional technical note: Random effects in more complex models

For more complex models, specifying random effects can become difficult. Random effects can be crossed with one another or can be nested within one another. Also, correlation structures for the random effects can be specified. However, approaching this complexity is beyond the scope of this book.

The examples in this book treat random variables as simple intercept-only variables, as simple blocks that are not nested within other variables.

Packages used in this chapter

The packages used in this chapter include:

- FSA
- psych
- lme4
- lmerTest
- nlme
- car

The following commands will install these packages if they are not already installed:

```
if(!require(FSA)){install.packages("FSA")}
if(!require(psych)){install.packages("psych")}
if(!require(lme4)){install.packages("lme4")}
if(!require(lmerTest)){install.packages("lmerTest")}
if(!require(nlme)){install.packages("nlme")}
if(!require(car)){install.packages("car")}
```

An example of a mixed model

For this example we will revisit the hand size example from the *Independent and Paired Values* chapter. The variable hand *Length* will be treated as an interval/ratio variable. We will return to using cumulative link models for Likert data in subsequent chapters.

Note that the model includes the blocking variable, *Individual*, and so data do not need to be in a certain order to match the paired observations.

Also note that mixed models may make certain assumptions about the distributions of the data. For simplicity, this example will not check if those assumption are met.

By default, an analysis of variance for a mixed model doesn't test the significance of the random effects in the model. However, the effect of random terms can be tested by comparing the model to a model including only the fixed effects and excluding the random effects, or with the *rand* function from the *lmerTest* package if the *lme4* package is used to specify the model.

As a technical note, the *lmerTest* package has options to use Satterthwaite or Kenward–Roger degrees of freedom, and options for type-III or type-II tests in the analysis of variance, if the *lme4* package is used to specify the model.

```
Input = "
Individual  Hand    Length
A           Left    17.5
B           Left    18.4
C           Left    16.2
D           Left    14.5
E           Left    13.5
F           Left    18.9
G           Left    19.5
H           Left    21.1
I           Left    17.8
J           Left    16.8
K           Left    18.4
L           Left    17.3
M           Left    18.9
N           Left    16.4
O           Left    17.5
P           Left    15.0
A           Right   17.6
B           Right   18.5
C           Right   15.9
D           Right   14.9
E           Right   13.7
F           Right   18.9
G           Right   19.5
H           Right   21.5
I           Right   18.5
J           Right   17.1
```

```

K      Right  18.9
L      Right  17.5
M      Right  19.5
N      Right  16.5
O      Right  17.4
P      Right  15.6
")

```

```

Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Mixed model with lmer

One way to construct a mixed effects model for interval/ratio data is with the *lmer* function in the *lme4* package. The *lmerTest* package is used to produce an analysis of variance with *p*-values for model effects.

Notice the grammar in the *lmer* function that defines the model: the term (*1/Individual*) is added to the model to indicate that *Individual* is the random term.

As a technical note, the *1* indicates that an intercept is to be fitted for each level of the random variable.

As another technical note, *REML* stands for *restricted maximum likelihood*. It is a method of fitting the model, and is often considered better than fitting with a conventional *ML* (*maximum likelihood*) method.

Define model and conduct analysis of variance

```

library(lme4)

library(lmerTest)

model = lmer(Length ~ Hand + (1|Individual),
            data=Data,
            REML=TRUE)

anova(model)

```

Analysis of Variance Table of type III with Satterthwaite approximation for degrees of freedom

	Sum Sq	Mean Sq	NumDF	DenDF	F.value	Pr(>F)
Hand	0.45125	0.45125	1	15	11.497	0.004034 **

Test the random effects in the model

The *rand* function from the *lmerTest* package will test the random effects in the model.

```
rand(model1)
```

Analysis of Random effects Table

	Chi.sq	Chi.DF	p.value
Individual	58.4	1	2e-14 ***

Mixed model with nlme

Another way to construct a mixed effects model for interval/ratio data is with the *lme* function in the *nlme* package.

Notice the grammar in the *lme* function that defines the model: the option *random=~1|Individual* is added to the model to indicate that *Individual* is the random term.

As a technical note, the *1* indicates that an intercept is to be fitted for each level of the random variable.

As another technical note, *REML* stands for *restricted maximum likelihood*. It is a method of fitting the model, and is often considered better than fitting with a conventional *ML* (*maximum likelihood*) method.

Define model and conduct analysis of deviance

```
library(nlme)

model = lme(Length ~ Hand, random=~1|Individual,
            data=Data,
            method="REML")
```

```
library(car)
```

```
Anova(model)
```

Analysis of Deviance Table (Type II tests)

	Chisq	Df	Pr(>Chisq)
Hand	11.497	1	0.0006972 ***

Test the random effects in the model

The random effects in the model can be tested by comparing the model to a model fitted with just the fixed effects and excluding the random effects. Because there are not random effects in this second model, the *gls* function in the *nlme* package is used to fit this model.

We will use a similar method for cumulative link models.

```
model.fixed = gls(Length ~ Hand,
                  data=Data,
                  method="REML")

anova(model,
      model.fixed)

Model df      AIC      BIC    LogLik  Test L.Ratio p-value
model       1 4  80.62586  86.23065 -36.31293
model.fixed 2  3 137.06356 141.26715 -65.53178 1 vs 2 58.4377 <.0001
```

What are Estimated Marginal Means?

Estimated marginal means are means for groups that are adjusted for means of other factors in the model.

Imagine a case where you are measuring the height of 7th-grade students in two classrooms, and want to see if there is a difference between the two classrooms. You are also recording the sex of the students, and at this age girls tend to be taller than boys. Say classroom *A* happens to have far more girls than boys. If you were to look at the mean height in the classrooms, you might find that classroom *A* had a higher mean, but this may not be an effect of the different classrooms, but because of the difference in the counts of boys and girls in each. In this case, reporting estimated marginal means for the classrooms may give a more representative result. Reporting estimated marginal means for studies where there are not equal observations for each combination of treatments is sometimes recommended. We say the design of these studies is *unbalanced*.

The following example details this hypothetical example. Looking at the means from the *Summarize* function in *FSA*, we might think there is a meaningful difference between the classrooms, with a mean height of 153.5 cm vs. 155.0 cm. But looking at the estimated marginal means (*emmeans*), which are adjusted for the difference in boys and girls in each classroom, this difference disappears. Each classroom has an estimated marginal mean of 153.5 cm, indicating the mean of classroom *B* was inflated due to the higher proportion of girls.

Note that the following example uses a linear model with the *lm* function. Here, *Height* is being treated as an interval/ratio variable.

This kind of analysis makes certain assumptions about the distribution of the data, but for simplicity, this example will ignore the need to determine that the data meet these assumptions.

Packages used in this chapter

The packages used in this chapter include:

- FSA
- psych
- emmeans
- car

The following commands will install these packages if they are not already installed:

```
if(!require(FSA)){install.packages("FSA")}
if(!require(psych)){install.packages("psych")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(car)){install.packages("car")}
```

Estimated marginal means example

```
Input ="
Classroom Sex Height
A Male 151
A Male 150
A Male 152
A Male 149
A Female 155
A Female 156
A Female 157
A Female 158
B Male 151
B Male 150
B Female 155
B Female 156
B Female 157
B Female 158
B Female 156
B Female 157
")
Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)
```

```
### Remove unnecessary objects
rm(Input)
```

Arithmetic means

```
library(FSA)
Summarize(Height ~ Classroom,
           data=Data,
           digits=3)

  Classroom n nvalid  mean      sd min   Q1 median     Q3 max percZero
1          A  8       8 153.5 3.423 149 150.8 153.5 156.2 158       0
2          B  8       8 155.0 2.928 150 154.0 156.0 157.0 158       0
```

Estimated marginal means

```
model = lm(Height ~ Classroom + Sex + Classroom:Sex,
           data = Data)

library(emmeans)
emmeans(model,
        ~ Classroom)

  Classroom emmean       SE df lower.CL upper.CL
  A          153.5 0.4082483 12 152.6105 154.3895
  B          153.5 0.4714045 12 152.4729 154.5271
```

Note that an analysis of variance also would have told us that there is a difference between levels of *Sex*, but not between levels of *Classroom*.

```
library(car)
Anova(model)

  Anova Table (Type II tests)

  Sum Sq Df F value    Pr(>F)
Classroom      0  1   0.0      1
Sex            126  1   94.5 4.857e-07 ***
Classroom:Sex  0  1   0.0      1
Residuals     16 12
```

Estimated Marginal Means for Multiple Comparisons

Comparisons of values across groups in linear models, cumulative link models, and other models can be conducted easily with the *emmeans* package. Importantly, it can make comparisons among interactions of factors.

E.M. means stands for *estimated marginal means*. Estimated marginal means are means for treatment levels that are adjusted for means of other factors in the model. For a more complete explanation, see the *What are Estimated Marginal Means?* chapter.

p-value adjustments for multiple comparisons

Adjustment of p-values for multiple comparisons is indicated with the *adjust*= option in the *pairs*, *cld*, and *summary* functions. Options are "tukey", "scheffe", "sidak", "bonferroni", "dunnett", "mvt", and "none". For further information on these options, see the *summary.emmGrid* function section in:

Lenth, R.V., J. Love, and M. Hervé. 2017. Package ‘emmeans: Estimated Marginal Means, aka Least-Squares Means’. cran.r-project.org/web/packages/emmeans/emmeans.pdf.

or use

```
library(emmeans)
?summary.emmGrid
```

Confidence intervals for marginal means

To adjust the confidence intervals for the EM means, use e.g. *summary(marginal, level=0.99)* or e.g. *cld(marginal, level=0.99)*.

Ignore values of emmeans for clm and clmm models

Typically you should ignore the values of the estimated marginal means themselves (*emmeans*) when using them with *clm* and *clmm* models. With default settings, the values of the estimated marginal means and the values of differences among the estimated marginal means are not easy to interpret. See “*clm*” in *help(models, package="emmeans")* for details and for other options.

Packages used in this chapter

The packages used in this chapter include:

- psych
- ordinal
- car
- RVAideMemoire
- emmeans
- multcompView

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(ordinal)){install.packages("ordinal")}
if(!require(car)){install.packages("car")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(multcompView)){install.packages("multcompview")}
```

Multiple comparisons with *emmeans*

This example uses data set and model from the *One-way Ordinal regression with CLM* chapter.

```
Input ="
Speaker Likert
Pooh      3
Pooh      5
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      4
Pooh      5
Pooh      5
Piglet    2
Piglet    4
Piglet    2
Piglet    2
Piglet    1
Piglet    2
Piglet    3
Piglet    2
Piglet    2
Piglet    3
Piglet    3
Tigger    4
Tigger    4
Tigger    4
Tigger    4
Tigger    5
Tigger    3
Tigger    5
Tigger    4
Tigger    4
Tigger    3
")
Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them
Data$Speaker = factor(Data$Speaker,
```

```

levels=unique(Data$Speaker))

### Create a new variable which is the Likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                      ordered = TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Define model and conduct analysis of deviance

```

library(ordinal)

model = clm(Likert.f ~ Speaker,
            data = Data)

library(car)

library(RVAideMemoire)

Anova(model,
      type = "II")

Analysis of Deviance Table (Type II tests)

  LR Chisq Df Pr(>Chisq)
Speaker  23.395  2  8.315e-06 ***

```

Group separations by emmeans in table format

In the *emmeans* function, *model* specifies the model object that was previously fitted. Note the specialized formula where *pairwise* indicates that all pairwise comparisons should be conducted, and *Speaker* indicates the variable whose levels will be compared.

The output here compares the levels of the grouping variable. For example, a significant *p*-value in the *Pooh - Piglet* line suggests that the value of the dependent variable (*Likert.f*) is different for Pooh compared with Piglet.

Remember to ignore the *estimate*, *SE*, and *emmean* values when using *emmeans* with a clm or clmm model object, unless specific options in *emmeans* are selected.

```
library(emmeans)

marginal = emmeans(model,
~ Speaker)

pairs(marginal,
adjust="tukey")

contrast      estimate      SE df z.ratio p.value
Pooh - Piglet 4.943822 1.3764706 NA 3.5916658 0.0010
Pooh - Tigger  0.633731 0.9055691 NA 0.6998152 0.7636
Piglet - Tigger -4.310091 1.3173294 NA -3.2718403 0.0031

P value adjustment: tukey method for comparing a family of 3 estimates

### Remember to ignore "estimate" and "SE" of differences with CLM,
### as well as "emmeans" in the output not shown here
```

Compact letter display

Here we'll use the *emmeans* output called *marginal* created above, and pass this object to the *cld* function to create a compact letter display.

In the output, groups sharing a letter in the *.group* column are not significantly different, at the *alpha* level specified.

```
library(multcompView)

cld(marginal,
alpha=0.05,
Letters=letters,      ### Use lower-case letters for .group
adjust="tukey")       ### Tukey-adjusted comparisons

Speaker   emmean      SE df asymp.LCL asymp.UCL .group
Piglet    -1.783599 0.7755958 NA -3.6355182 0.06832068 a
Tigger     2.526492 0.8413200 NA  0.5176407 4.53534421 b
Pooh      3.160223 0.8912968 NA  1.0320404 5.28840661 b

Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05

### Groups sharing a letter in .group are not significantly different

### Remember to ignore "emmean", "SE", "LCL", and "UCL" for CLM
### unless certain options are used.
```

Optional: Interaction plot of estimated marginal means with mean separation letters

It is often desirable to plot estimated marginal means from an analysis with either their confidence intervals or standard errors. This can be conducted as a one-way plot or an interaction plot. The *emmeans* and *ggplot2* packages make it relatively easy to extract the EM means and the group separation letters and use them for plotting.

Input data and specify linear model

```
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(ggplot2)){install.packages("ggplot2")}

Location = c(rep("Olympia" , 6), rep("Ventura", 6),
             rep("Northampton", 6), rep("Burlington", 6))

Tribe = c(rep(c("Jedi", "Sith"), 12))

Midichlorians = c(10, 4, 12, 5, 15, 4, 15, 9, 15, 11, 18, 12,
                  8, 13, 8, 15, 10, 17, 22, 22, 20, 22, 20, 25)

Data = data.frame(Tribe, Location, Midichlorians)

str(Data)

model = lm(Midichlorians ~ Tribe + Location + Tribe:Location,
           data = Data)

library(car)
Anova(model)

Anova Table (Type II tests)

Response: Midichlorians
          Sum Sq Df F value    Pr(>F)
Tribe      8.17  1 3.0154   0.1017
Location  591.00  3 72.7385 1.535e-09 ***
Tribe:Location 198.83  3 24.4718 3.218e-06 ***
Residuals   43.33 16
```

One-way estimated marginal means and plot

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
                    ~ Location)
```

```

CLD = cld(marginal,
           alpha=0.05,
           Letters=letters,
           adjust="tukey")

CLD

  Location      emmean       SE df lower.CL upper.CL .group
  Olympia     8.333333 0.6718548 16  6.449596 10.21707    a
  Northampton 11.833333 0.6718548 16  9.949596 13.71707    b
  Ventura     13.333333 0.6718548 16 11.449596 15.21707    b
  Burlington  21.833333 0.6718548 16 19.949596 23.71707    c

### Order the levels for printing

CLD$Location = factor(CLD$Location,
                       levels=c("Olympia", "Ventura", "Northampton", "Burlington"))

### Remove spaces in .group

CLD$.group=gsub(" ", "", CLD$.group)

### Plot

library(ggplot2)

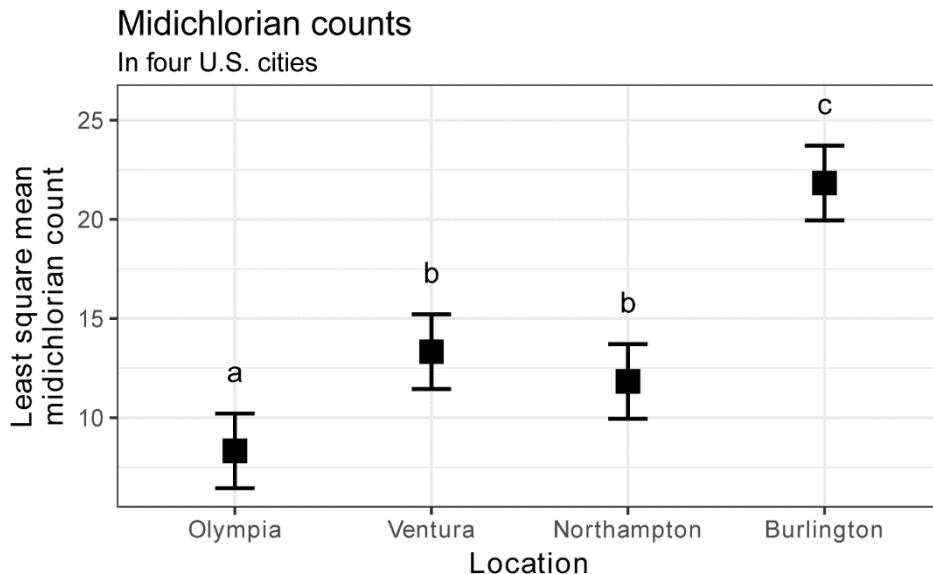
ggplot(CLD,
       aes(x      = Location,
           y      = emmean,
           label = .group)) +
  geom_point(shape  = 15,
             size   = 4) +
  geom_errorbar(aes(ymin  = lower.CL,
                    ymax   = upper.CL),
                width = 0.2,
                size  = 0.7) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold"),
        axis.text   = element_text(face = "bold"),
        plot.caption = element_text(hjust = 0)) +
  ylab("Estimated marginal mean\nmidichlorian count") +
  ggtitle ("Midichlorian counts",
            subtitle = "In four U.S. cities") +
  labs(caption = paste0("\nMidichlorian counts for four locations. ",
                        "Boxes indicate the EM mean. \n",
                        "Error bars indicate the 95% "))


```

```

"confidence interval of the EM mean. \n",
"Means sharing a letter are not ",
"significantly different (Tukey-adjusted \n",
"comparisons)."),
hjust=0.5) +
geom_text(nudge_x = c(0, 0, 0, 0),
           nudge_y = c(4, 4, 4, 4),
           color    = "black")

```



Midichlorian counts for four locations. Boxes indicate the LS mean.
Error bars indicate the 95% confidence interval of the LS mean.
Means sharing a letter are not significantly different (Tukey-adjusted comparisons).

Interaction plot of estimated marginal means

```

library(multcompView)
library(emmeans)

marginal = emmeans(model,
                    ~ Tribe:Location)

CLD = cld(marginal,
           alpha=0.05,
           Letters=letters,
           adjust="tukey")

CLD

```

Tribe	Location	emmean	SE	df	lower.CL	upper.CL	group
Sith	Olympia	4.333333	0.9501462	16	1.354477	7.31219	a
Jedi	Northampton	8.666667	0.9501462	16	5.687810	11.64552	ab
Sith	Ventura	10.666667	0.9501462	16	7.687810	13.64552	bc

Jedi	Olympia	12.333333	0.9501462	16	9.354477	15.31219	bcd
Sith	Northampton	15.000000	0.9501462	16	12.021143	17.97886	cd
Jedi	Ventura	16.000000	0.9501462	16	13.021143	18.97886	d
Jedi	Burlington	20.666667	0.9501462	16	17.687810	23.64552	e
Sith	Burlington	23.000000	0.9501462	16	20.021143	25.97886	e

```
### Order the levels for printing
```

```
CLD$Location = factor(CLD$Location,
                       levels=c("Olympia", "Ventura", "Northampton", "Burlington"))
```

```
CLD$Tribe = factor(CLD$Tribe,
                     levels=c("Jedi", "Sith"))
```

```
### Remove spaces in .group
```

```
CLD$.group=gsub(" ", "", CLD$.group)
```

```
CLD
```

```
### Plot
```

```
library(ggplot2)
```

```
pd = position_dodge(0.4)      ### How much to jitter the points on the plot
```

```
ggplot(CLD,
       aes(x      = Location,
           y      = emmean,
           color = Tribe,
           label = .group)) +
```

```
geom_point(shape  = 15,
           size   = 4,
           position = pd) +
```

```
geom_errorbar(aes(ymin  = lower.CL,
                   ymax   = upper.CL),
                  width = 0.2,
                  size  = 0.7,
                  position = pd) +
```

```
theme_bw() +
  theme(axis.title  = element_text(face = "bold"),
        axis.text    = element_text(face = "bold"),
        plot.caption = element_text(hjust = 0)) +
```

```
ylab("Estimated marginal mean\nmidichlorian count") +
  ggtitle ("Midichlorian counts for Jedi and Sith",
            subtitle = "In four U.S. cities") +
```

```
  labs(caption = paste0("\nMidichlorian counts for two tribes across ",
```

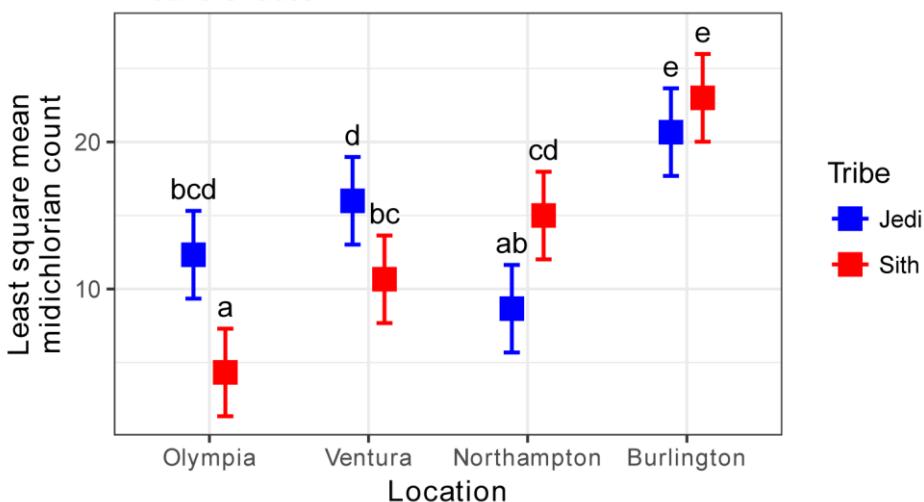
```

"four locations. Boxes indicate \n",
"the EM mean. ",
"Error bars indicate the 95% confidence ",
"interval ",
"of the LS \n",
"mean. Means sharing a letter are ",
"not significantly different \n",
"(Tukey-adjusted comparisons)."),
hjust=0.5) +
geom_text(nudge_x = c(0.1, -0.1, 0.1, -0.1, 0.1, -0.1, -0.1, 0.1),
nudge_y = c(4.5, 4.5, 4.5, 4.5, 4.5, 4.5, 4.5, 4.5),
color   = "black") +
scale_color_manual(values = c("blue", "red"))

```

Midichlorian counts for Jedi and Sith

In four U.S. cities



Midichlorian counts for two tribes across four locations. Boxes indicate the LS mean. Error bars indicate the 95% confidence interval of the LS mean. Means sharing a letter are not significantly different (Tukey-adjusted comparisons).

Factorial ANOVA: Main Effects, Interaction Effects, and Interaction Plots

Two-way or multi-way data often come from experiments with a *factorial design*. A factorial design has at least two factor variables for its independent variables, and multiple observation for every combination of these factors.

The weight gain example below show factorial data. In this example, there are three observations for each combination of *Diet* and *Country*.

With this kind of data, we are usually interested in testing the effect of each factor variable (*main effects*) and then the effect of their combination (*interaction effect*).

For two-way data, an interaction plot shows the mean or median value for the response variable for each combination of the independent variables. This type of plot, especially if it includes error bars to indicate the variability of data within each group, gives us some understanding of the effect of the main factors and their interaction.

When main effects or interaction effects are statistically significant, post-hoc testing can be conducted to determine which groups differ significantly from other groups. With a factorial experiment, there are a few guidelines for determining when to do post-hoc testing. The following guidelines are presented for two-way data for simplicity.

- *When neither the main effects nor the interaction effect is statistically significant*, no post-hoc mean-separation testing should be conducted.
- *When one or more of the main effects are statistically significant and the interaction effect is not*, post-hoc mean-separation testing should be conducted on significant main effects only.
(This is shown in the first weight gain example below)
- *When the interaction effect is statistically significant*, post-hoc mean-separation testing should be conducted on the interaction effect only. This is the case even when the main effects are also statistically significant.
(This is shown in the second weight gain example below)

The final guideline above is not always followed. There are times when people will present the mean-separation tests for significant main effects even when the interaction effect is significant. In general, though, if there is a significant interaction, the mean-separation tests for interaction will better explain the results of the analysis, and the mean-separation tests for the main effects will be of less interest.

Packages used in this chapter

The packages used in this chapter include:

- psych
- car
- multcompView
- emmeans
- FSA
- ggplot2
- phia

The following commands will install these packages if they are not already installed:

```
if(!require(car)){install.packages("car")}
if(!require(psych)){install.packages("psych")}
if(!require(multcompView)){install.packages("multcompview")}
```

```
if(!require(emmeans)){install.packages("emmeans")}
if(!require(FSA)){install.packages("FSA")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(phia)){install.packages("phia")}
```

Two-way ANOVA example with interaction effect

Imagine for this example an experiment in which people were put on one of three diets to encourage weight gain. The amount of weight gained will be the dependent variable, and will be considered an interval/ratio variable. For independent variables, there are three different diets and the country in which subjects live.

The model will be fit with the *lm* function, whose syntax is similar to that of the *clm* function.

This kind of analysis makes certain assumptions about the distribution of the data, but for simplicity, this example will ignore the need to determine that the data meet these assumptions.

```
Input ="
Diet      Country   weight_change
A          USA        0.120
A          USA        0.125
A          USA        0.112
A          UK         0.052
A          UK         0.055
A          UK         0.044
B          USA        0.096
B          USA        0.100
B          USA        0.089
B          UK         0.025
B          UK         0.029
B          UK         0.019
C          USA        0.149
C          USA        0.150
C          USA        0.142
C          UK         0.077
C          UK         0.080
C          UK         0.066
")

Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them

Data$Country = factor(Data$Country,
                      levels=unique(Data$Country))

### Check the data frame

library(psych)

headTail(Data)
```

```
str(Data)
summary(Data)

### Remove unnecessary objects

rm(Input)
```

Simple interaction plot

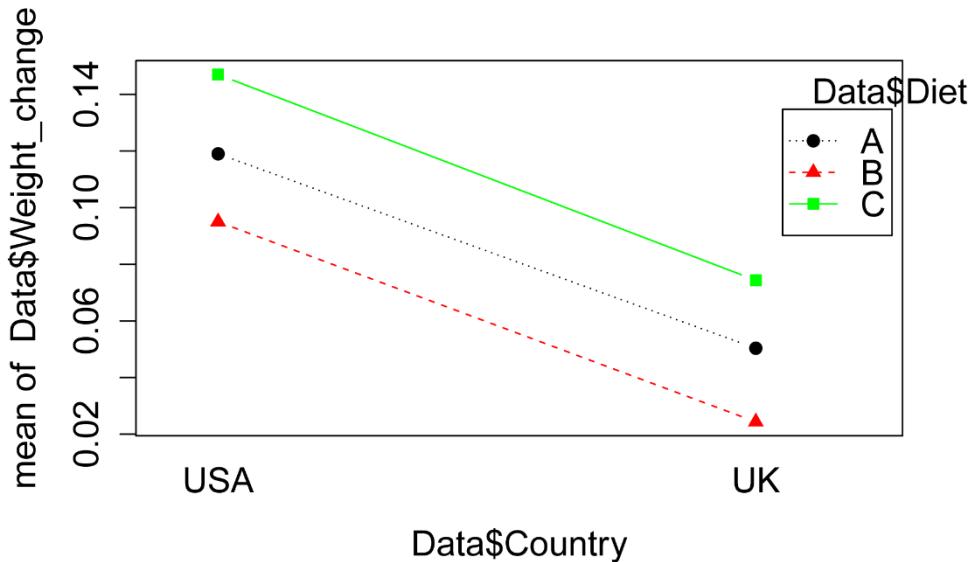
The *interaction.plot* function creates a simple interaction plot for two-way data. The options shown indicate which variables will be used for the x-axis, trace variable, and response variable. The *fun=mean* option indicates that the mean for each group will be plotted. For the meaning of other options, see *?interaction.plot*.

This style of interaction plot does not show the variability of each group mean, so it is difficult to use this style of plot to determine if there are significant differences among groups.

The plot shows that mean weight gain for each diet was lower for the UK compared with USA. And that this difference was relatively constant for each diet, as is evidenced by the lines on the plot being parallel. This suggests that there is no large or significant interaction effect. That is, the difference among diets is consistent across countries. And vice-versa, the difference in countries is consistent across diets.

A couple of other styles of interaction plot are shown at the end of this chapter.

```
interaction.plot(x.factor      = Data$Country,
                  trace.factor = Data$Diet,
                  response     = Data$Weight_change,
                  fun = mean,
                  type="b",
                  col=c("black","red","green"),    ### Colors for levels of trace var.
                  pch=c(19, 17, 15),             ### Symbols for levels of trace var.
                  fixed=TRUE,                   ### Order by factor order in data
                  leg.bty = "o")
```



Specify the linear model and conduct an analysis of variance

A linear model is specified with the *lm* function. *Weight_change* is the dependent variable. *Country* and *Diet* are the independent variables, and including *Country:Diet* in the formula adds the interaction term for *Country* and *Diet* to the model.

The ANOVA table indicates that the main effects are significant, but that the interaction effect is not.

```
model = lm(Weight_change ~ Country + Diet + Country:Diet,
           data = Data)

library(car)

Anova(model,
      type = "II")

Anova Table (Type II tests)

          Sum Sq Df F value    Pr(>F)
Country     0.022472  1 657.7171 7.523e-12 ***
Diet        0.007804  2 114.2049 1.547e-08 ***
Country:Diet 0.000012  2   0.1756   0.8411
Residuals   0.000410 12
```

Post-hoc testing with emmeans

Because the main effects were significant, we will want to perform post-hoc mean separation tests for each main effect factor variable.

For this, we will use the *emmeans* package. The linear model under consideration is called *model*, created the *lm* function above. The formula in the *emmeans* function indicates that comparisons

should be conducted for the variable *Country* in the first call, and for the variable *Diet* in the second call.

```
library(emmeans)

marginal = emmeans(model,
~ Country)

pairs(marginal,
adjust="tukey")

contrast estimate      SE df t.ratio p.value
USA - UK 0.07066667 0.002755466 12  25.646 <.0001


library(emmeans)

marginal = emmeans(model,
~ Diet)

pairs(marginal,
adjust="tukey")

contrast estimate      SE df t.ratio p.value
A - B     0.025 0.003374743 12   7.408 <.0001
A - C    -0.026 0.003374743 12  -7.704 <.0001
B - C    -0.051 0.003374743 12 -15.112 <.0001
```

Extended example with additional country

For the following example, the hypothetical data have been amended to include a third country, New Zealand.

```
Input =""
Diet   Country  weight_change
A      USA      0.120
A      USA      0.125
A      USA      0.112
A      UK       0.052
A      UK       0.055
A      UK       0.044
A      NZ       0.080
A      NZ       0.090
A      NZ       0.075
B      USA      0.096
B      USA      0.100
B      USA      0.089
B      UK       0.025
B      UK       0.029
B      UK       0.019
B      NZ       0.055
B      NZ       0.065
B      NZ       0.050
C      USA      0.149
```

```
C      USA    0.150
C      USA    0.142
C      UK     0.077
C      UK     0.080
C      UK     0.066
C      NZ     0.055
C      NZ     0.065
C      NZ     0.050
C      NZ     0.054
")
```

```
Data = read.table(textConnection(Input), header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them

Data$Country = factor(Data$Country,
                      levels=unique(Data$Country))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

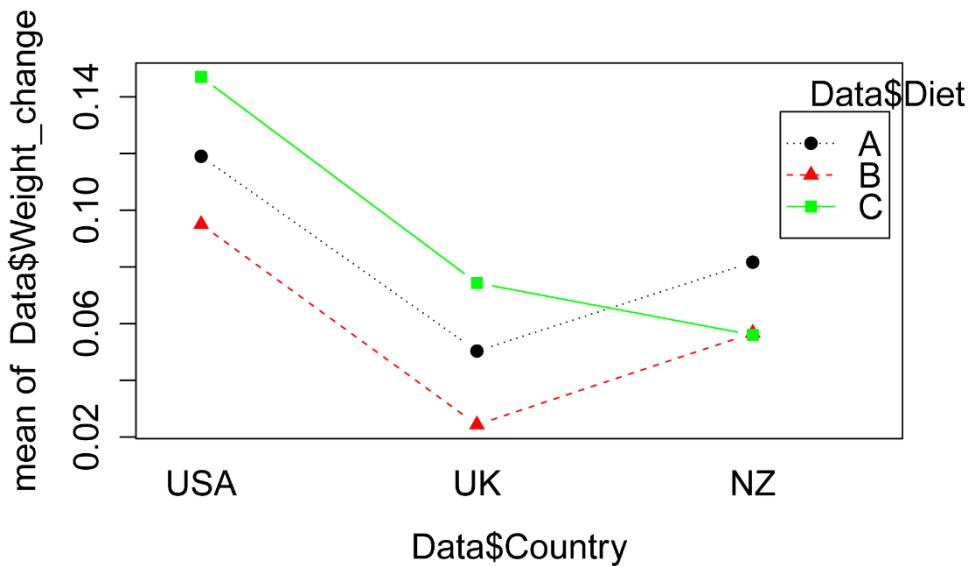
### Remove unnecessary objects

rm(Input)
```

Simple interaction plot

The plot suggests that the effect of diet is not consistent across all three countries. While Diet C showed the greatest mean weight gain for USA and UK, for NZ it has a lower mean than Diet A. This suggests there may be a meaningful or significant interaction effect, but we will need to do a statistical test to confirm this hypothesis.

```
interaction.plot(x.factor      = Data$Country,
                  trace.factor = Data$Diet,
                  response     = Data$Weight_change,
                  fun = mean,
                  type="b",
                  col=c("black","red","green"),   ### Colors for levels of trace var.
                  pch=c(19, 17, 15),             ### Symbols for levels of trace var.
                  fixed=TRUE,                   ### Order by factor order in data
                  leg.bty = "o")
```



Specify the linear model and conduct an analysis of variance

The ANOVA table indicates that the interaction effect is significant, as are both main effects.

```
model = lm(weight_change ~ Country + Diet + Country:Diet,
            data = Data)
```

```
library(car)
```

```
Anova(model,
      type = "II")
```

Anova Table (Type II tests)

	Sum Sq	Df	F value	Pr(>F)
Country	0.0256761	2	318.715	2.426e-15 ***
Diet	0.0051534	2	63.969	3.634e-09 ***
Country:Diet	0.0040162	4	24.926	2.477e-07 ***
Residuals	0.0007653	19		

Post-hoc testing with emmeans

Because the interaction effect was significant, we would like to compare all group means from the interaction. Even though the main effects were significant, the typical advice is to not conduct pairwise comparisons for main effects when their interaction is significant. This is because focusing on the groups in the interaction better describe the results of the analysis.

We will use the *emmeans* package, and ask for a compact letter display with the *cld* function. First we create an object, named *marginal*, with the results of the call to *emmeans*. Notice here that the formula indicates that comparisons should be conducted for the interaction of *Country* and *Diet*, indicated with *Country:Diet*.

Be sure to read the *Estimated Marginal Means for Multiple Comparisons* chapter for correct interpretation of estimated marginal means. For lm model objects, the values for *emmean*, *SE*, *LCL*, and *UCL* values are meaningful.

```
library(emmeans)

marginal = emmeans(model,
                     ~ Country:Diet)

pairs(marginal,
      adjust="tukey")

cld(marginal,
     alpha=0.05,
     Letters=letters,      ### Use lower-case letters for .group
     adjust="tukey")       ### Tukey-adjusted comparisons



| Country | Diet | emmean     | SE          | df | lower.CL   | upper.CL   | .group |
|---------|------|------------|-------------|----|------------|------------|--------|
| UK      | B    | 0.02433333 | 0.003664274 | 19 | 0.01291388 | 0.03575278 | a      |
| UK      | A    | 0.05033333 | 0.003664274 | 19 | 0.03891388 | 0.06175278 | b      |
| NZ      | C    | 0.05600000 | 0.003173354 | 19 | 0.04611047 | 0.06588953 | b      |
| NZ      | B    | 0.05666667 | 0.003664274 | 19 | 0.04524722 | 0.06808612 | bc     |
| UK      | C    | 0.07433333 | 0.003664274 | 19 | 0.06291388 | 0.08575278 | cd     |
| NZ      | A    | 0.08166667 | 0.003664274 | 19 | 0.07024722 | 0.09308612 | de     |
| USA     | B    | 0.09500000 | 0.003664274 | 19 | 0.08358055 | 0.10641945 | e      |
| USA     | A    | 0.11900000 | 0.003664274 | 19 | 0.10758055 | 0.13041945 | f      |
| USA     | C    | 0.14700000 | 0.003664274 | 19 | 0.13558055 | 0.15841945 | g      |



Confidence level used: 0.95  

  Conf-level adjustment: sidak method for 9 estimates  

  P value adjustment: tukey method for comparing a family of 9 estimates  

  significance level used: alpha = 0.05



### Groups sharing a letter are not significantly different  

  ### at the alpha = 0.05 level.


```

Interaction plot with error bars using ggplot2

The interaction plots created with the *interaction.plot* function above are handy to investigate trends in the data, but have the disadvantage of not showing the variability in data within each group.

The package *ggplot2* can be used to create attractive interaction plots with error bars. Here, we will use standard error of each mean for the error bars.

```
### Create a data frame called Sum with means and standard deviations

library(FSA)

Sum = Summarize(weight_change ~ Country + Diet,
                 data=Data,
                 digits=3)
```

```
### Add standard error of the mean to the Sum data frame
```

```
Sum$se = Sum$sd / sqrt(Sum$n)
```

```
Sum$se = signif(Sum$se, digits=3)
```

```
Sum
```

	Country	Diet	n	nvalid	mean	sd	min	Q1	median	Q3	max	percZero	se
1	USA	A	3	3	0.119	0.007	0.112	0.116	0.120	0.122	0.125	0	0.00404
2	UK	A	3	3	0.050	0.006	0.044	0.048	0.052	0.054	0.055	0	0.00346
3	NZ	A	3	3	0.082	0.008	0.075	0.078	0.080	0.085	0.090	0	0.00462
4	USA	B	3	3	0.095	0.006	0.089	0.092	0.096	0.098	0.100	0	0.00346
5	UK	B	3	3	0.024	0.005	0.019	0.022	0.025	0.027	0.029	0	0.00289
6	NZ	B	3	3	0.057	0.008	0.050	0.052	0.055	0.060	0.065	0	0.00462
7	USA	C	3	3	0.147	0.004	0.142	0.146	0.149	0.150	0.150	0	0.00231
8	UK	C	3	3	0.074	0.007	0.066	0.072	0.077	0.078	0.080	0	0.00404
9	NZ	C	4	4	0.056	0.006	0.050	0.053	0.054	0.058	0.065	0	0.00300

```
### Order levels of the factor; otherwise R will alphabetize them
```

```
Sum$Country = factor(Sum$Country,  
                      levels=unique(Sum$Country))
```

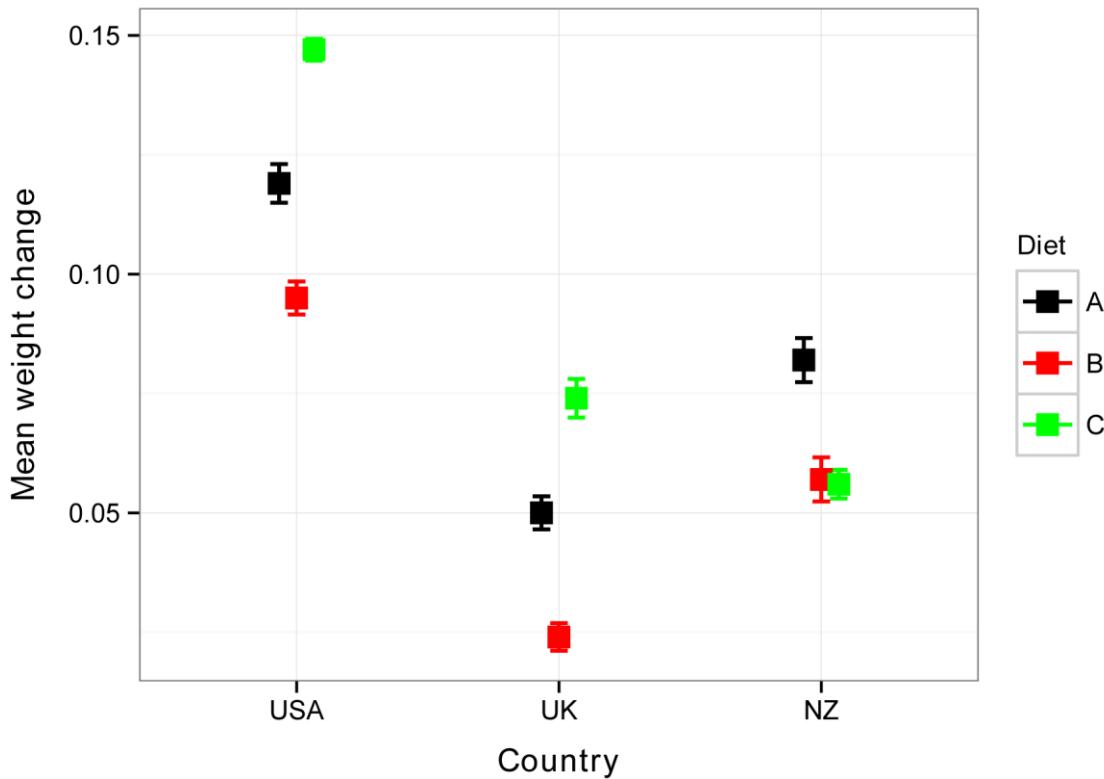
```
### Produce interaction plot
```

```
library(ggplot2)
```

```
pd = position_dodge(.2)  
  
ggplot(Sum, aes(x = Country,  
                 y = mean,  
                 color = Diet)) +  
  geom_errorbar(aes(ymin = mean - se,  
                    ymax = mean + se),  
                width=.2, size=0.7, position=pd) +  
  geom_point(shape=15, size=4, position=pd) +  
  theme_bw() +  
  theme(axis.title = element_text(face = "bold")) +  
  scale_colour_manual(values= c("black","red","green")) +  
  ylab("Mean weight change")
```

```
## You may see an error, "ymax not defined"
```

```
## In this case, it does not appear to affect anything
```

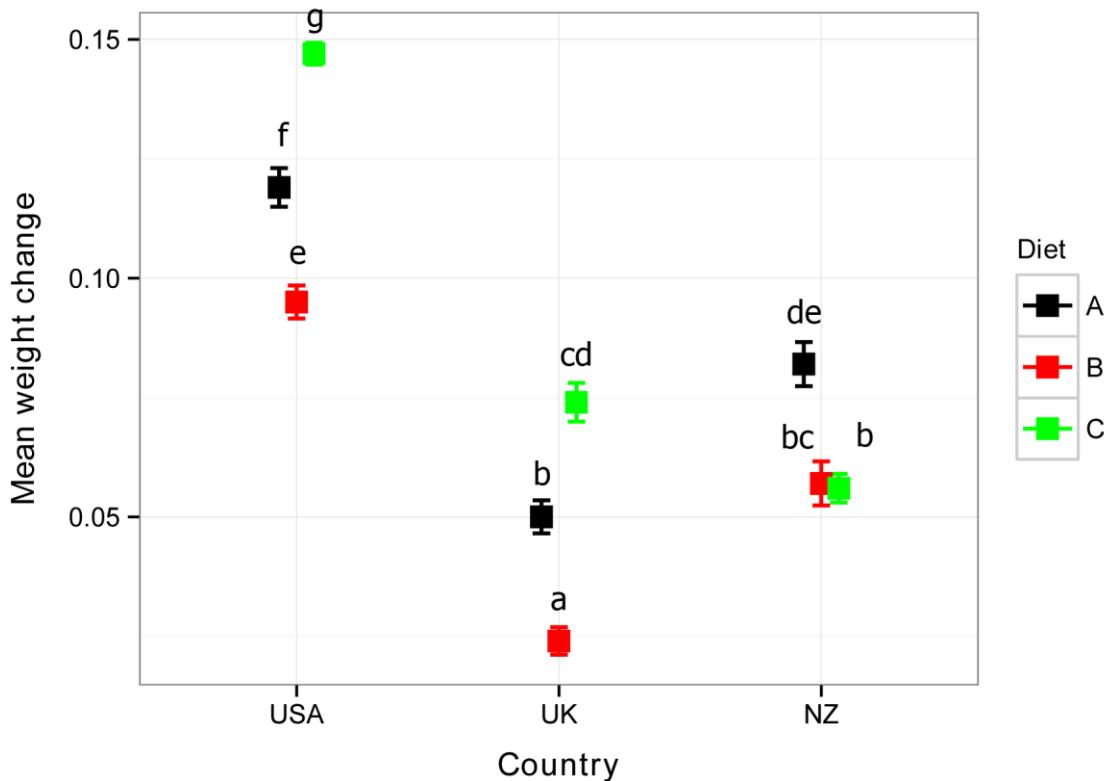


Interaction plot with mean separation letters manually added

It is common to add mean separation letters from post-hoc analyses to interaction plots. One option is to add letters manually in either image manipulation software like Photoshop or GIMP, or in a word processor or other software that can handle graphic manipulation.

Letters can also be added to the plot by *ggplot2* with the *annotate* or *geom_text* options. See “Optional: Interaction plot of estimated marginal means with mean separation letters” in the *Estimated Marginal Means for Multiple Comparisons* chapter for examples.

It is probably more common for means to be lettered so that the *greatest* mean is indicated with *a*. However, *emmeans* by default labels the *least* mean with *a*. The order of letters can be reversed manually.



Plot of mean weight change for three diets in three countries. Means sharing a letter are not significantly different according to pairwise comparisons of estimated marginal means with Tukey adjustment for multiple comparisons.

Interaction plot with the phia package

The *phia* package can be used to create interaction plots quickly. By default the error bars indicate standard error of the means. For additional options, see *?interactionMeans*.

Note that *model* is the linear model specified above.

```
library(phia)

IM = interactionMeans(model)

IM
```

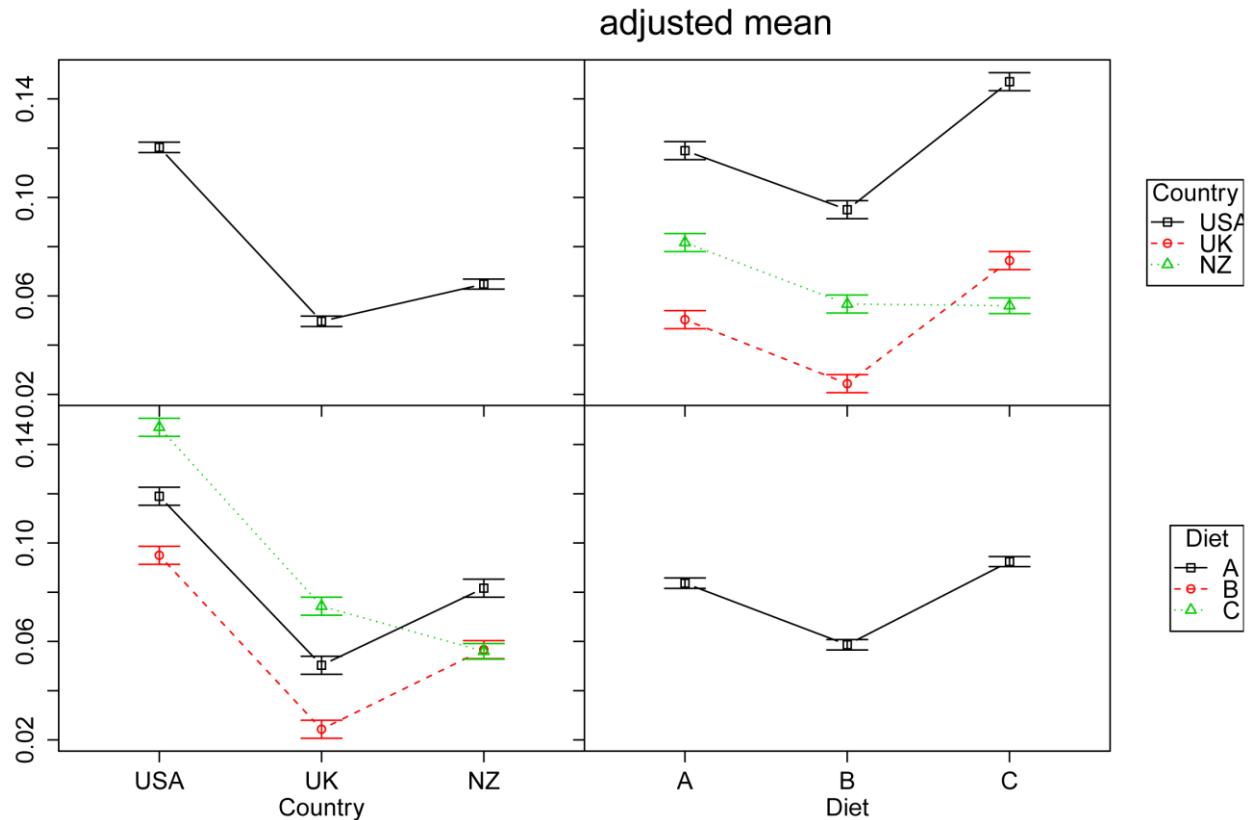
	Country	Diet	adjusted mean	std. error
1	USA	A	0.11900000	0.003664274
2	UK	A	0.05033333	0.003664274
3	NZ	A	0.08166667	0.003664274
4	USA	B	0.09500000	0.003664274
5	UK	B	0.02433333	0.003664274
6	NZ	B	0.05666667	0.003664274
7	USA	C	0.14700000	0.003664274
8	UK	C	0.07433333	0.003664274
9	NZ	C	0.05600000	0.003173354

```
### Produce interaction plot
```

```
plot(IM)
```

```
### Return the graphics device to its default 1-plot-per-window state
```

```
par(mfrow=c(1,1))
```



p-values and R-square Values for Models

When developing more complex models it is often desirable to report a *p*-value for the model as a whole as well as an *R-square* for the model.

p-values for models

The *p*-value for a model determines the significance of the model compared with a null model. For a linear model, the null model is defined as the dependent variable being equal to its mean. So, the *p*-value for the model is answering the question, *Does this model explain the data significantly better than would just looking at the average value of the dependent variable?*

R-squared and pseudo R-squared

The *R-squared* value is a measure of how well the model explains the data. It is an example of a *goodness-of-fit* statistic.

R-squared for linear (ordinary least squares) models

In R, models fit with the *lm* function are linear models fit with *ordinary least squares* (OLS). For these models, *R-squared* indicates the proportion of the variability in the dependent variable that is explained by model. That is, an *R-squared* of 0.60 indicates that 60% of the variability in the dependent variable is explained by the model.

Pseudo R-squared

For many types of models, *R-squared* is not defined. These include relatively common models like *logistic regression* and the *cumulative link models* used in this book. For these models, pseudo *R-squared* measures can be calculated. A pseudo *R-squared* is not directly comparable to the *R-squared* for OLS models. Nor can it be interpreted as the proportion of the variability in the dependent variable that is explained by model. Instead, pseudo *R-squared* measures are relative measures among similar models indicating how well the model explains the data.

This book uses three pseudo *R-squared* measures: McFadden, Cox and Snell (also referred to as ML), Nagelkerke (also referred to as Cragg and Uhler).

In general I favor the Nagelkerke pseudo *R-squared*, but there is no agreement as to which pseudo *R-squared* measurement should be used.

p-values and R-squared values.

p-values and *R-squared* values measure different things. The *p*-value indicates if there is a significant relationship described by the model, and the *R-squared* measures the degree to which the data is explained by the model. It is therefore possible to get a significant *p*-value with a low *R-squared* value. This often happens when there is a lot of variability in the dependent variable, but there are enough data points for a significant relationship to be indicated.

Packages used in this chapter

The packages used in this chapter include:

- psych
- lmtest
- boot
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(lmtest)){install.packages("lmtest")}
if(!require(boot)){install.packages("boot")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Example of model *p-value*, *R-squared*, and pseudo *R-squared*

The following example uses some hypothetical data of a sample of people for which typing speed (*Words.per.minute*) and age were measured. After plotting the data, we decide to construct a polynomial model with *Words.per.minute* as the dependent variable and *Age* and *Age²* as the independent variables. Notice in this example that all variables are treated as interval/ratio variables, and that the independent variables are also continuous variables.

The data will first be fit with a linear model with the *lm* function. Passing this model to the *summary* function will display the *p-value* for the model and the *R-squared* value for the model.

The same data will then be fit with a *generalized linear model* with the *glm* function. This type of model allows more latitude in the types of data that can be fit, but in this example, we'll use the *family=gaussian* option, which will mimic the model fit with the *lm* function, though the underlying math is different.

Importantly, the *summary* of the *glm* function does not produce a *p-value* for the model nor an *R-squared* for the model.

For the model fit with *glm*, the *p-value* can be determined with the *anova* function comparing the fitted model to a null model. The null model is fit with only an intercept term on the right side of the model. As an alternative, the *nagelkerke* function described below also reports a *p-value* for the model, using the likelihood ratio test.

There is no *R-squared* defined for a *glm* model. Instead a pseudo *R-squared* can be calculated. The function *nagelkerke* produces pseudo *R-squared* values for a variety of models. It reports three types: McFadden, Cox and Snell, and Nagelkerke. In general I recommend using the Nagelkerke measure, though there is no agreement on which pseudo *R-squared* measurement to use, if any at all.

The Nagelkerke is the same as the Cox and Snell, except that the value is adjusted upward so that the Nagelkerke has a maximum value of 1.

It has been suggested that a McFadden value of 0.2–0.4 indicates a good fit.

Note that these models makes certain assumptions about the distribution of the data, but for simplicity, this example will ignore the need to determine if the data met these assumptions.

```
Input ="
Age  Words.per.minute
12   23
12   32
12   25
13   27
13   30
15   29
15   33
16   37
18   29
```

```

22    33
23    37
24    33
25    43
27    35
33    30
42    25
53    22
")

```

```

Data = read.table(textConnection(Input), header=TRUE)

### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

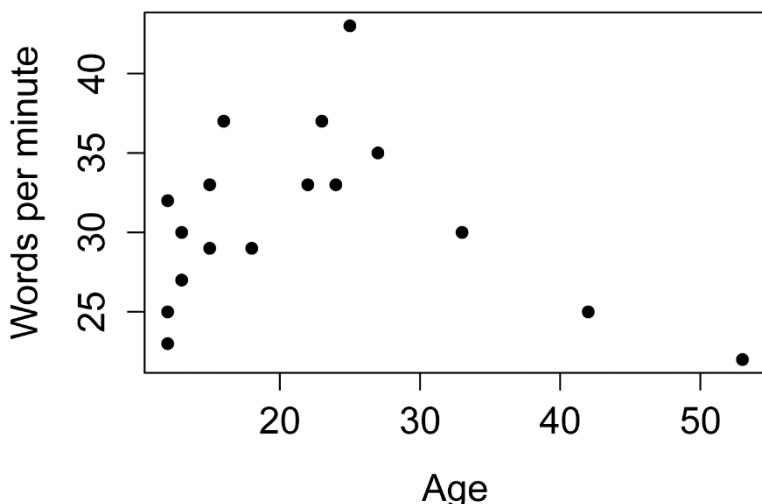
```

Plot the data

```

plot(Words.per.minute ~ Age,
      data = Data,
      pch=16,
      xlab = "Age",
      ylab = "Words per minute")

```



Prepare data

```
### Create new variable for the square of Age
Data$Age2 = Data$Age ^ 2

### Double check data frame
library(psych)
headTail(Data)

  Age Words.per.minute Age2
1   12             23  144
2   12             32  144
3   12             25  144
4   13             27  169
...
14  27             35  729
15  33             30 1089
16  42             25 1764
17  53             22 2809
```

Linear model

```
model = lm (Words.per.minute ~ Age + Age2,
            data=Data)

summary(model)      ### Shows coefficients,
                    ### p-value for model, and R-squared

Multiple R-squared:  0.5009, Adjusted R-squared:  0.4296
F-statistic: 7.026 on 2 and 14 DF,  p-value: 0.00771

### p-value and (multiple) R-squared value
```

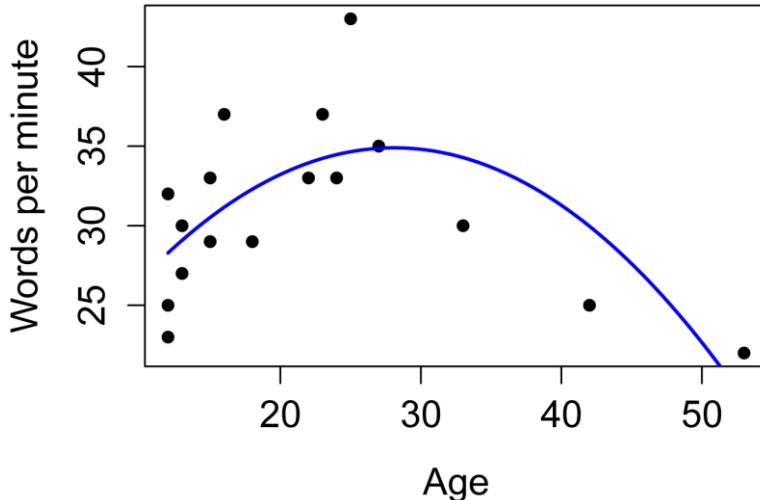
Simple plot of data and model

For bivariate data, the function *plotPredy* will plot the data and the predicted line for the model. It also works for polynomial functions, if the *order* option is changed.

```
library(rcompanion)

plotPredy(data  = Data,
          y     = Words.per.minute,
          x     = Age,
          x2    = Age2,
          model = model,
          order = 2,
```

```
xlab = "words per minute",
ylab = "Age")
```



Generalized linear model

```
model = glm (words.per.minute ~ Age + Age2,
             data = Data,
             family = gaussian)

summary(model)      ### Shows coefficients
```

Calculate p-value for model

In R, the most common way to calculate the *p*-value for a fitted model is to compare the fitted model to a null model with the *anova* function. The null model is usually formulated with just a constant on the right side.

```
null = glm (words.per.minute ~ 1,      ### Create null model
             data = Data,          ### with only a constant on the right side
             family = gaussian)

anova (model,
       null,
       test="Chisq")           ### Tests options "Rao", "LRT",
                                ### "Chisq", "F", "Cp"
                                ### But some work with only some model types

Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        14    243.07
2        16    487.06 -2   -243.99 0.0008882 ***
```

Calculate pseudo *R-squared* and *p*-value for model

An alternative test for the *p*-value for a fitted model, the *nagelkerke* function will report the *p*-value for a model using the likelihood ratio test.

The *nagelkerke* function also reports the McFadden, Cox and Snell, and Nagelkerke pseudo *R-squared* values for the model.

```
library(rcompanion)

nagelkerke(model)

$Pseudo.R.squared.for.model.vs.null
                                         Pseudo.R.squared
McFadden                               0.112227
Cox and Snell (ML)                     0.500939
Nagelkerke (Cragg and Uhler)           0.501964

$Likelihood.ratio.test
Df.diff LogLik.diff   Chisq   p.value
-2      -5.9077  11.815  0.0027184
```

Likelihood ratio test for *p*-value for model

The *p*-value for a model by the likelihood ratio test can also be determined with the *lrtest* function in the *lmtest* package.

```
library(lmtest)

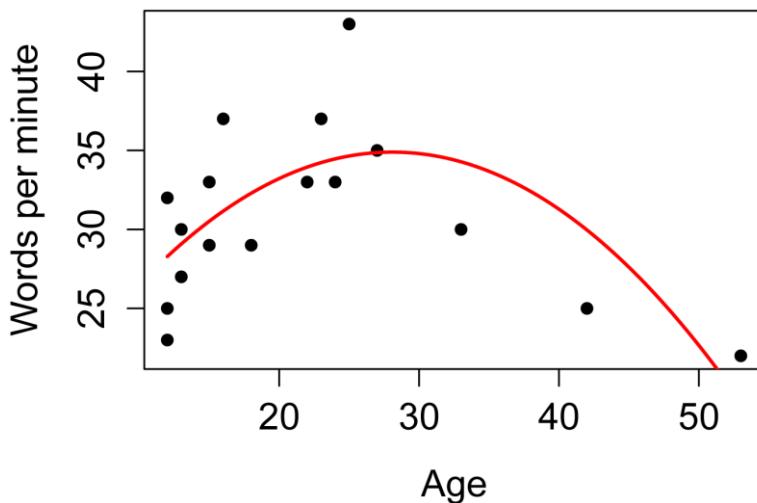
lrtest(model)

#DF  LogLik DF  Chisq Pr(>Chisq)
1    4 -46.733
2    2 -52.641 -2 11.815  0.002718 **
```

Simple plot of data and model

```
library(rcompanion)

plotPredy(data  = Data,
          y      = Words.per.minute,
          x      = Age,
          x2     = Age2,
          model  = model,
          order  = 2,
          xlab   = "Words per minute",
          ylab   = "Age",
          col    = "red")                 ### line color
```



Optional analyses: Confidence intervals for R-squared values

It is relatively easy to produce confidence intervals for *R-squared* values or other parameters from model fitting, such as coefficients for regression terms. This can be accomplished with bootstrapping. Here the *boot.ci* function from the *boot* package is used.

The code below is a little complicated, but relatively flexible.

Function can contain any function of interest, as long as it includes an input vector or data frame (*input* in this case) and an indexing variable (*index* in this case). *Stat* is set to produce the actual statistic of interest on which to perform the bootstrap (*r.squared* from the *summary* of the *lm* in this case).

The code *Function(Data, 1:n)* is there simply to test *Function* on the data frame *Data*. In this case, it will produce the output of *Function* for the first *n* rows of *Data*. Since *n* is defined as the length of the first column in *Data*, this should return the value for *Stat* for the whole data frame, if *Function* is set up correctly.

```
Input ="
Age  Words.per.minute
12   23
12   32
12   25
13   27
13   30
15   29
15   33
16   37
18   29
22   33
23   37
24   33
25   43
27   35
```

```

33    30
42    25
53    22
")

Data = read.table(textConnection(Input), header=TRUE)
Data$Age2 = Data$Age ^ 2

### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

```

Linear model**R-squared value**

```

library(boot)

Function = function(input, index){
  Input = input[index,]
  Result = lm (Words.per.minute ~ Age + Age2,
               data = Input)
  Stat = summary(Result)$r.squared
  return(Stat)}

### Test Function
n = length(Data[,1])
Function(Data, 1:n)
[1] 0.5009385

### Produce Stat estimate by bootstrap
Boot = boot(Data,
            Function,
            R=5000)

```

```

mean(Boot$t[,1])
[1] 0.5754582

### Produce confidence interval by bootstrap

boot.ci(Boot,
        conf = 0.95,
        type = "perc")

      ### Options: "norm", "basic", "stud", "perc", "bca", "all"

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

Intervals :
Level      Percentile
95%    ( 0.3796,  0.7802 )
Calculations and Intervals on Original Scale

### Other information

Boot

hist(Boot$t[,1],
     col = "darkgray")

```

Generalized linear model

Pseudo R-squared value

The *nagelkerke* function produces a list. The second item in the list is a matrix named

```
Pseudo.R.squared.for.model1.vs.null.
```

The third element of this matrix is the value for the Nagelkerke pseudo *R-squared*. So,

```
nagelkerke(Result, Null)[[2]][3]
```

yields the value of the Nagelkerke pseudo *R-squared*.

```

library(boot)

library(rcompanion)

Function = function(input, index){
  Input = input[index,]
  Result = glm (Words.per.minute ~ Age + Age2,
                data = Input,
                family="gaussian")

```

```

Null = glm (words.per.minute ~ 1,
            data = Input,
            family="gaussian")
Stat = nagelkerke(Result, Null)[[2]][3]
return(Stat)}

### Test Function

n = length(Data[,1])

Function(Data, 1:n)

[1] 0.501964

### Produce Stat estimate by bootstrap

Boot = boot(Data,
             Function,
             R=1000)

      ### In this case, even 1000 iterations can take a while

mean(Boot$t[,1])

[1] 0.5803598

### Produce confidence interval by bootstrap

boot.ci(Boot,
        conf = 0.95,
        type = "perc")

      ### Options: "norm", "basic", "stud", "perc", "bca", "all"

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

Intervals :
Level    Percentile
95%    ( 0.3836,  0.7860 )
Calculations and Intervals on Original Scale

### Other information

Boot

hist(Boot$t[,1],
     col = "darkgray")

```

References

Kabacoff, R.I. Bootstrapping. Quick-R. www.statmethods.net/advstats/bootstrapping.html.

Accuracy and Errors for Models

Measures of accuracy and error can be used to determine how well a given model fits the data. They are distinct from the *R-squared* and pseudo *R-squared* measures discussed in the last chapter.

These statistics are useful to compare a wide variety of models where the dependent variable is continuous. In general, one would want to compare only among models with the same data for the dependent variable.

Measure	Units	Interpretation
Min.max.accuracy	Unitless	1 is perfect fit
Mean absolute error (MAE)	Same as variable	0 is perfect fit
Mean absolute percent error (MAPE)	Unitless	0 is perfect fit
Mean square error (MSE)	Square of variable units	0 is perfect fit
Root mean square error (RMSE)	Same as variable	0 is perfect fit
Normalized root mean square error (NRMSE)	Unitless	0 is perfect fit
NRMSE Accuracy	Unitless	1 is perfect fit
Efron's R-squared	Unitless	1 is perfect fit

Packages used in this chapter

The packages used in this chapter include:

- rcompanion
- betareg
- nlme
- lme4
- quantreg
- mgcv
- MASS
- robust

The following commands will install these packages if they are not already installed:

```
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(betareg)){install.packages("betareg")}
if(!require(nlme)){install.packages("nlme")}
if(!require(lme4)){install.packages("lme4")}
if(!require(quantreg)){install.packages("quantreg")}
if(!require(mgcv)){install.packages("mgcv")}
if(!require(MASS)){install.packages("MASS")}
if(!require(robust)){install.packages("robust")}
```

Examples of accuracy and error

The following example fits various models to the *BrendonSmall* data set in the *rcompanion* package, and then uses the *accuracy* function to produce accuracy and error statistics for these models.

Note that *model.5* has a different dependent variable than the others.

```
### Prepare data set

library(rcompanion)

data(BrendonSmall)

BrendonSmall$Calories = as.numeric(BrendonSmall$Calories)
BrendonSmall$Sodium = as.numeric(BrendonSmall$Sodium)
BrendonSmall$Calories2 = BrendonSmall$Calories ^ 2
BrendonSmall$Sodium.ratio = BrendonSmall$Sodium / 1500
BrendonSmall$Grade = factor(BrendonSmall$Grade)

head(BrendonSmall)

  Instructor Grade Weight Calories Sodium Score Calories2 Sodium.ratio
1 Brendon Small     6      43    2069    1287     77   4280761     0.8580000
2 Brendon Small     6      41    1990    1164     76   3960100     0.7760000
3 Brendon Small     6      40    1975    1177     76   3900625     0.7846667
4 Brendon Small     6      44    2116    1262     84   4477456     0.8413333
5 Brendon Small     6      45    2161    1271     86   4669921     0.8473333
6 Brendon Small     6      44    2091    1222     87   4372281     0.8146667
```

```
### Fit various models

model.1 = lm(Sodium ~ Calories, data = BrendonSmall)

model.2 = lm(Sodium ~ Calories + Calories2, data = BrendonSmall)

model.3 = glm(Sodium ~ Calories, data = BrendonSmall, family="Gamma")

quadplat = function(x, a, b, c1x) {
  ifelse(x < c1x, a + b * x + (-0.5*b/c1x) * x * x,
         a + b * c1x + (-0.5*b/c1x) * c1x * c1x)}

model.4 = nls(Sodium ~ quadplat(Calories, a, b, c1x),
              data = BrendonSmall,
```

```

start = list(a    = 519,
            b    = 0.359,
            c1x = 2300)

library(betareg)
model.5 = betareg(Sodium.ratio ~ Calories, data = BrendonSmall)

library(nlme)
model.6 = gls(Sodium ~ Calories, data = BrendonSmall, method="REML")

model.7 = lme(Sodium ~ Calories,
              random = ~1|Instructor,
              data = BrendonSmall,
              method="REML")

library(lme4)
model.8 = lmer(Sodium ~ Calories + (1|Instructor),
               data=BrendonSmall,
               REML=TRUE)

library(lme4)
model.9 = suppressWarnings(glmer(Sodium ~ Calories + (1|Instructor),
                                 data=BrendonSmall,
                                 family = gaussian))

library(quantreg)
model.10 = rq(Sodium ~ Calories,
               data = BrendonSmall,
               tau = 0.5)

model.11 = loess(Sodium ~ Calories,
                  data = BrendonSmall,
                  span = 0.75,
                  degree=2,
                  family="gaussian")

library(mgcv)
model.12 = gam(Sodium ~ s(Calories),
                data = BrendonSmall,
                family=gaussian())

library(MASS)
model.13 = glm.nb(Sodium ~ Grade,
                  data = BrendonSmall)

library(robust)
model.14 = glmRob(Sodium ~ Grade,
                  data = BrendonSmall,
                  family = "poisson")

### Produce accuracy and error statisitcs

library(rcompanion)

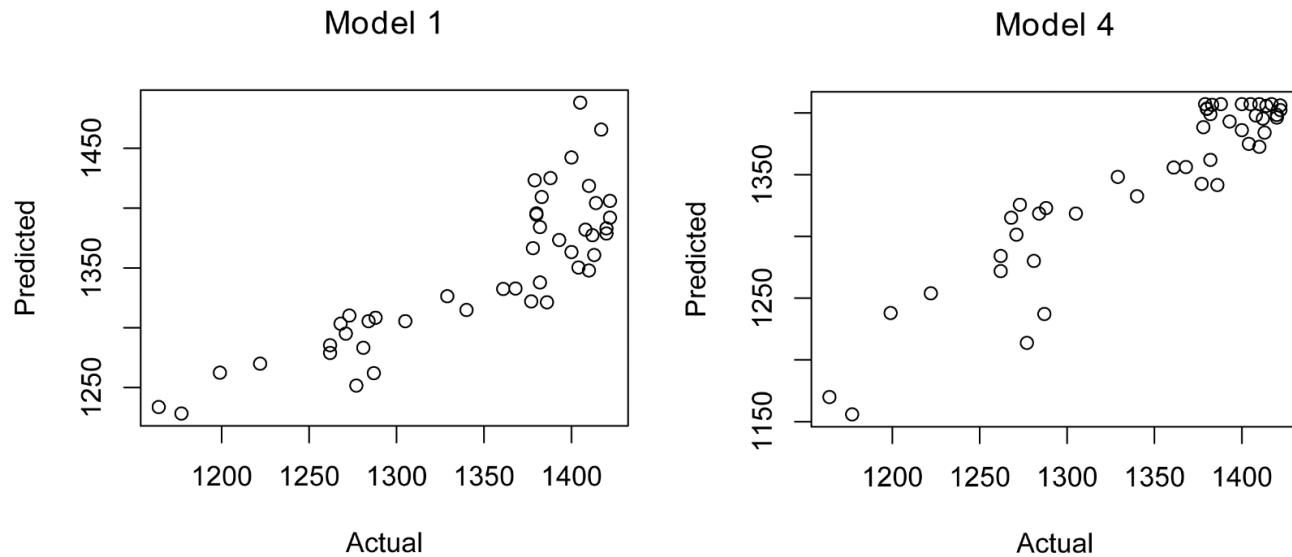
```

```
accuracy(list(model.1, model.2, model.3, model.4, model.5, model.6,
model.7, model.8, model.9, model.10, model.11, model.12,
model.13, model.14),
plotit=TRUE, digits=3)
```

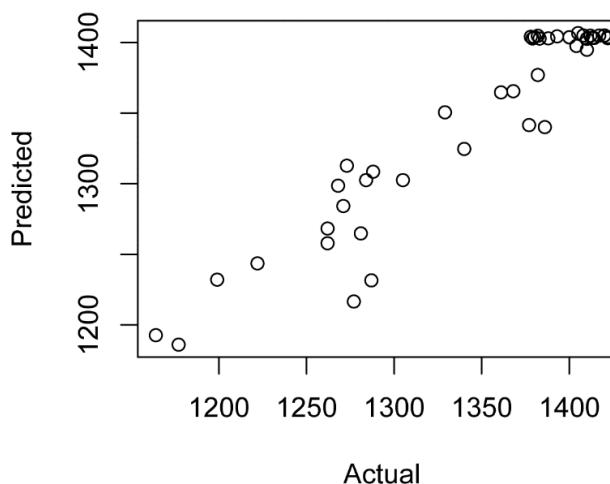
	\$Fit.criteria											
	Min.max.accuracy	MAE	MAPE	MSE	RMSE	NRMSE.mean	NRMSE.median	NRMSE.mean.accuracy	NRMSE.median.accuracy	Efron.r.squared		
1	0.976	32.7000	0.0244	1.44e+03	38.0000	0.0282	0.0275	0.972	0.972	0.721		
2	0.983	22.6000	0.0170	7.13e+02	26.7000	0.0198	0.0194	0.980	0.981	0.862		
3	0.975	34.4000	0.0257	1.63e+03	40.4000	0.0300	0.0293	0.970	0.971	0.684		
4	0.984	22.0000	0.0166	7.00e+02	26.5000	0.0196	0.0192	0.980	0.981	0.865		
5	0.980	0.0179	0.0201	4.09e-04	0.0202	0.0225	0.0220	0.977	0.978	0.822		
6	0.976	32.7000	0.0244	1.44e+03	38.0000	0.0282	0.0275	0.972	0.972	0.721		
7	0.979	28.5000	0.0213	1.15e+03	33.9000	0.0252	0.0246	0.975	0.975	0.778		
8	0.979	28.5000	0.0213	1.15e+03	33.9000	0.0252	0.0246	0.975	0.975	0.778		
9	0.979	28.5000	0.0213	1.15e+03	33.9000	0.0252	0.0246	0.975	0.975	0.778		
10	0.976	32.6000	0.0243	1.50e+03	38.7000	0.0287	0.0280	0.971	0.972	0.710		
11	0.987	17.9000	0.0135	5.03e+02	22.4000	0.0167	0.0163	0.983	0.984	0.903		
12	0.987	17.1000	0.0129	4.53e+02	21.3000	0.0158	0.0154	0.984	0.985	0.913		
13	0.979	27.7000	0.0213	1.61e+03	40.1000	0.0298	0.0290	0.970	0.971	0.690		
14	0.743	343.0000	0.2570	1.87e+05	433.0000	0.3210	0.3140	0.679	0.686	-35.200		

Note in the previous results that *model.14*, the *glmRob* model, did not fit the data well. In this case, fitting a Poisson regression model is probably not appropriate for the data here, but is included since this type of model is accepted by the *accuracy* function.

It is useful to examine plots of the predicted values vs. the actual values to see how well the model reflects the actual values, and to see if patterns in the plots suggest another model may be better.



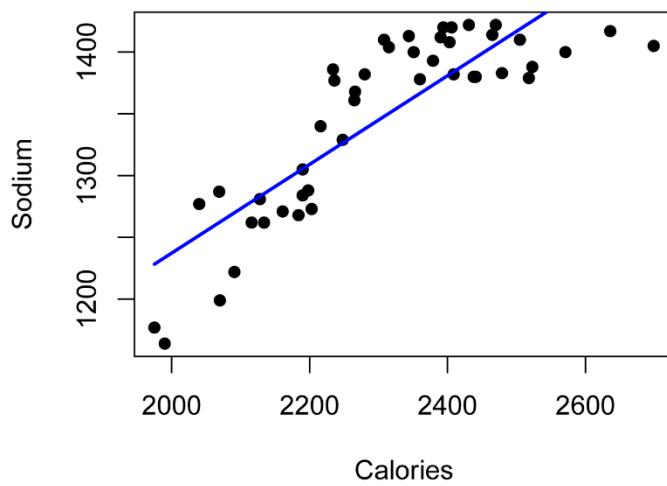
Model 11



The accuracy measures produced here are different in magnitude than their corresponding *R-squared* or pseudo *R-squared* measures. The following plots and captions illustrate this point.

```
plotPredy(data = BrendonSmall,
          x      = Calories,
          y      = Sodium,
          model = model.1,
          xlab  = "Calories",
          ylab  = "Sodium")
```

```
summary(model.1)
```



Plot of *model.1*, a linear OLS model. *R-squared* = 0.721. Accuracy with *RMSE / median* = 0.972.

```
plotPredy(data = BrendonSmall,
          x      = Calories,
          y      = Sodium,
          model = model.4,
```

```

xlab  = "Calories",
ylab  = "Sodium")

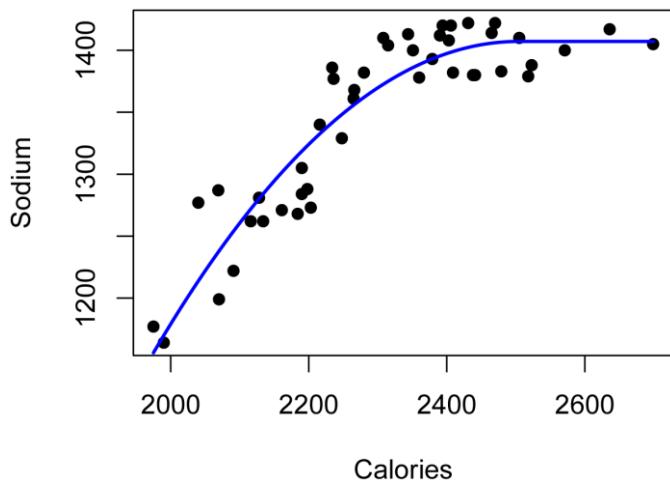
nullfunct = function(x, m){m}

m.ini     = 1300

null = nls(Sodium ~ nullfunct(Calories, m),
           data = BrendonSmall,
           start = list(m = m.ini),
           trace = FALSE,
           nls.control(maxiter = 1000))

nagelkerke(model.4, null)

```

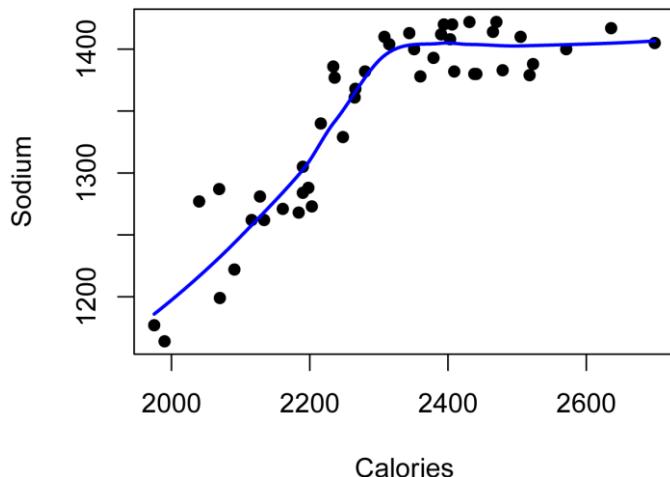


Plot of *model.4*, a nonlinear regression model. Nagelkerke pseudo *R-squared* = 0.865. Accuracy with *RMSE / median* = 0.978.

```

plotPredy(data  = BrendonSmall,
          x      = Calories,
          y      = Sodium,
          model = model.11,
          xlab  = "Calories",
          ylab  = "Sodium")

```



Plot of *model.11*, a loess model. Accuracy with $RMSE / median = 0.984$.

Ordinal Tests with Cumulative Link Models

Introduction to Cumulative Link Models (CLM) for Ordinal Data

In the section on nonparametric tests in this book, each test is used for data from a specific situation or design, such as comparing groups from two-sample unpaired data, or two-sample paired data, or with an unreplicated complete block design.

Cumulative link models are a different approach to analyzing ordinal data. Models can be chosen to handle simple or more complex designs.

This approach is very powerful and flexible, and might be considered the best approach for data with ordinal dependent variables in many cases.

However, a few disadvantages to using these models are that 1) your audience may not familiar with them, 2) their results can be somewhat tricky to interpret or explain, and 3) some models won't converge or model assumptions won't be met for some data sets.

These models are also called *ordinal regression models*, or *proportional odds models*.

The ordinal package

These models and tests will use the *ordinal* package, and either of two functions, *clm* and *clmm*.

A few notes on using cumulative link models:

- The dependent variable must be an ordered factor variable. It does not need to have numerals for levels. For example it could have levels *doctorate > masters > bachelors > associates > high.school*. But also it could have the levels *5 > 4 > 3 > 2 > 1*.

- Independent variables can be factors, ordered factors, or interval/ratio variables.
- The general interpretation for significant results of these models is that there is a significant effect of the independent variable on the dependent variable, or that there is a significant difference among groups.
- Post-hoc tests for factors or groups can be conducted with the *emmeans* package. An optional approach for post-hoc tests is to use pairwise ordinal tests of groups. These can be conducted with the functions *pairwiseOrdinalTest* and *pairwiseOrdinalPairedTest* from the *rcompanion* package.
- The *threshold = "equidistant"* and *threshold = "symmetric"* options can be used to indicate to the software that levels of the response variable are equally spaced or symmetrically spaced, respectively. This is useful to indicate when these conditions are assumed to be true, but are also useful to try if the model procedure produces errors. Likert items using symmetrical language in the range of responses could be considered symmetric. Likert items with several numbered options with anchor terms only at the ends of scale might be considered equidistant.

Analysis of deviance

The significance of the effects of independent variables will be tested with an *analysis of deviance* (ANODe) approach. This is analogous to the *analysis of variance* (ANOVA) used in linear models.

Model assumptions for CLM

Most statistical models have some assumptions about the underlying data. In order for the model to be valid, these assumptions have to be met.

For CLM, the assumption of concern is called the *proportional odds assumption*. An explanation of this assumption can be found in the Wikipedia or IDRE articles cited below.

The ordinal package can test for the proportional odds assumption with the *nominal_test* and *scale_test* functions (Christensen 2015b). If any independent variable fails these tests (that is, a significant *p*-value is returned), that variable can be handled differently in the model using the *nominal* and *scale* options in the *clm* function.

References

For more information on these models and the *ordinal* package, see:

- Christensen, H.R.B. 2015a. *Analysis of ordinal data with cumulative link models—estimation with the R-package ordinal*. cran.r-project.org/web/packages/ordinal/vignettes/clm_intro.pdf.
- `library(ordinal); help(package="ordinal")`

- Wikipedia. 2015. "The model and the proportional odds assumption" in *Ordered logit*. en.wikipedia.org/wiki/Ordered_logit#The_model_and_the_proportional_odds_assumption.
- IDRE . 2015. *R Data Analysis Examples: Ordinal Logistic Regression*. UCLA. www.ats.ucla.edu/stat/r/dae/ologit.htm.
- Christensen, R.H.B. 2015b. *Package 'ordinal'*. cran.r-project.org/web/packages/ordinal/ordinal.pdf.
- Hervé, M. 2014. "72. Analyser des notes" in *Aide-mémoire de statistique appliquée à la biologie*. cran.r-project.org/doc/contrib/Herve-Aide-memoire-statistique.pdf.

Two-sample Ordinal Test with CLM

Appropriate data

- Two-sample data. That is, one-way data with two groups only
- Dependent variable is ordered factor
- Independent variable is a factor with two levels. That is, two groups
- Observations between groups are independent. That is, not paired or repeated measures data

Interpretation

A significant result can be interpreted as, "There was a significant difference between groups." Or, "There was a significant effect of Independent Variable."

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- ordinal
- car
- RVAideMemoire

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(ordinal)){install.packages("ordinal")}
if(!require(car)){install.packages("car")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
```

Two-sample ordinal model example

```

Input ="
  Speaker Likert
  Pooh    3
  Pooh    5
  Pooh    4
  Pooh    4
  Pooh    4
  Pooh    4
  Pooh    4
  Pooh    4
  Pooh    5
  Pooh    5
  Piglet  2
  Piglet  4
  Piglet  2
  Piglet  2
  Piglet  1
  Piglet  2
  Piglet  3
  Piglet  2
  Piglet  2
  Piglet  3
")
Data = read.table(textConnection(Input),header=TRUE)

### Create a new variable which is the Likert scores as an ordered factor
Data$Likert.f = factor(Data$Likert,
                       ordered = TRUE)

### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

```

Summarize data treating Likert scores as factors

```

xtabs( ~ Speaker + Likert.f,
      data = Data)

  Likert.f
Speaker 1 2 3 4 5
Piglet 1 6 2 1 0
Pooh   0 0 1 6 3

XT = xtabs( ~ Speaker + Likert.f,
            data = Data)

prop.table(XT,
           margin = 1)

  Likert.f
Speaker 1 2 3 4 5
Piglet 0.1 0.6 0.2 0.1 0.0
Pooh   0.0 0.0 0.1 0.6 0.3

```

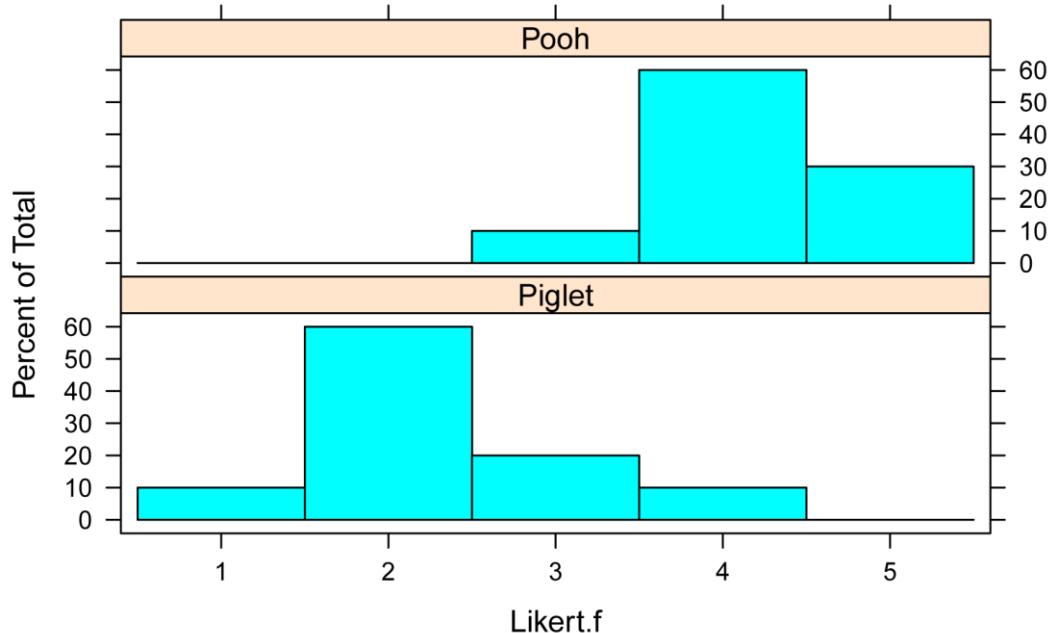
Bar plots of data by group

```

library(lattice)

histogram(~ Likert.f | Speaker,
          data=Data,
          layout=c(1,2)      # columns and rows of individual plots
)

```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Speaker,
          data=Data,
          digits=3)
```

Speaker	n	mean	sd	min	Q1	median	Q3	max	percZero
Piglet	10	2.3	0.823	1	2	2	2.75	4	0
Pooh	10	4.2	0.632	3	4	4	4.75	5	0

Two-sample ordinal model example

The model is specified using formula notation. Here, *Likert.f* is the dependent variable and *Speaker* is the independent variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?clm*.

The *p*-value for the effect of *Speaker* is determined using the *Anova* function in the packages *RVAideMemoire* and *car*.

Define model

```
library(ordinal)

model = clm(Likert.f ~ Speaker,
            data = Data)
```

Analysis of deviance

```
library(car)
library(RVAideMemoire)

Anova(model,
      type = "II")

  Analysis of Deviance Table (Type II tests)

    LR Chisq Df Pr(>Chisq)
Speaker 17.505  1 2.866e-05 ***

```

Check model assumptions

```
nominal_test(model)

  Tests of nominal effects

  formula: Likert.f ~ Speaker
            Df logLik     AIC LRT Pr(>Chi)
<none>      -20.199 50.397
Speaker
  #### No p-value produced for this example.

scale_test(model)

  Tests of scale effects

  formula: Likert.f ~ Speaker
            Df logLik     AIC     LRT Pr(>Chi)
<none>      -20.199 50.397
Speaker   1 -20.148 52.295 0.1019  0.7496
  #### No violation of model assumptions
```

Two-sample Paired Ordinal Test with CLMM

Note that the order of the data doesn't matter, as it did in the paired signed-rank test example, because here the blocking variable, *Student*, is entered explicitly in the model.

Note that the *clmm* function is used here instead of the *clm* function. The *clmm* function specifies a *mixed effects model*. This type of model is appropriate for paired and repeated measures analyses.

Appropriate data

- Two-sample data. That is, one-way data with two groups only
- Dependent variable is ordered factor

- Independent variable is a factor with two levels. That is, two groups
- Observations between groups are paired

Interpretation

A significant result can be interpreted as, "There was a significant difference between groups."
Or, "There was a significant effect of Independent Variable."

Packages used in this chapter

The packages used in this chapter include:

- psych
- ordinal
- car
- RVAideMemoire

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(ordinal)){install.packages("ordinal")}
if(!require(car)){install.packages("car")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
```

Two-sample paired ordinal model example

```
Input ="
Speaker Time Student Likert
Pooh    1      a      1
Pooh    1      b      4
Pooh    1      c      3
Pooh    1      d      3
Pooh    1      e      3
Pooh    1      f      3
Pooh    1      g      4
Pooh    1      h      3
Pooh    1      i      3
Pooh    1      j      3
Pooh    2      a      4
Pooh    2      b      5
Pooh    2      c      4
Pooh    2      d      5
Pooh    2      e      4
Pooh    2      f      5
Pooh    2      g      3
Pooh    2      h      4
Pooh    2      i      3
Pooh    2      j      4
")
Data = read.table(textConnection(Input),header=TRUE)
```

```

### Create a new variable which is the Likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                      ordered = TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Plot the paired data

Scatter plot with one-to-one line

Paired data can be visualized with a scatter plot of the paired cases. In the plot below, points that fall above and to the left of the blue line indicate cases for which the value for *Time 2* was greater than for *Time 1*.

Note that the points in the plot are jittered slightly so that points that would fall directly on top of one another can be seen.

First, two new variables, *Time.1* and *Time.2*, are created by extracting the values of *Likert* for observations with the *Time* variable equal to 1 or 2, respectively.

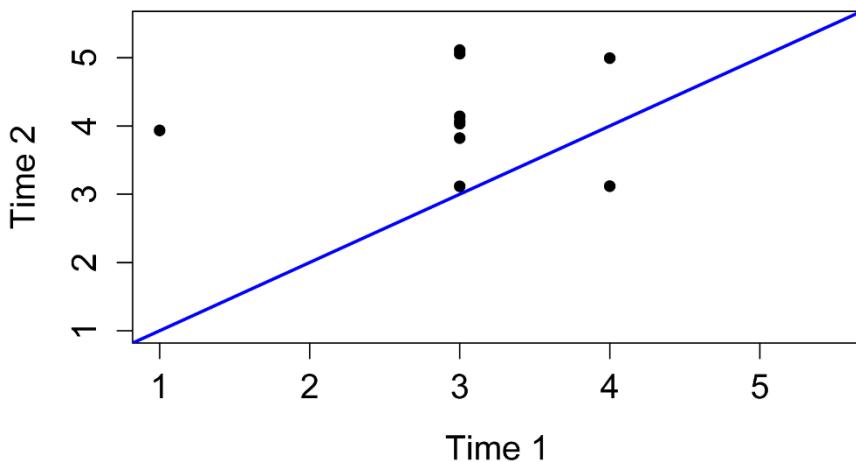
Note that for this plot, the data must be ordered so that the first observation where *Time* = 1 is paired to the first observation where *Time* = 2, and so on.

```

Time.1 = Data$Likert[Data$Time==1]
Time.2 = Data$Likert[Data$Time==2]

plot(Time.1, jitter(Time.2),   # jitter offsets points so you can see them all
      pch = 16,                # shape of points
      cex = 1.0,                # size of points
      xlim=c(1, 5.5),          # limits of x axis
      ylim=c(1, 5.5),          # limits of y axis
      xlab="Time 1",
      ylab="Time 2"
    )
abline(0,1, col="blue", lwd=2) # line with intercept of 0 and slope of 1

```

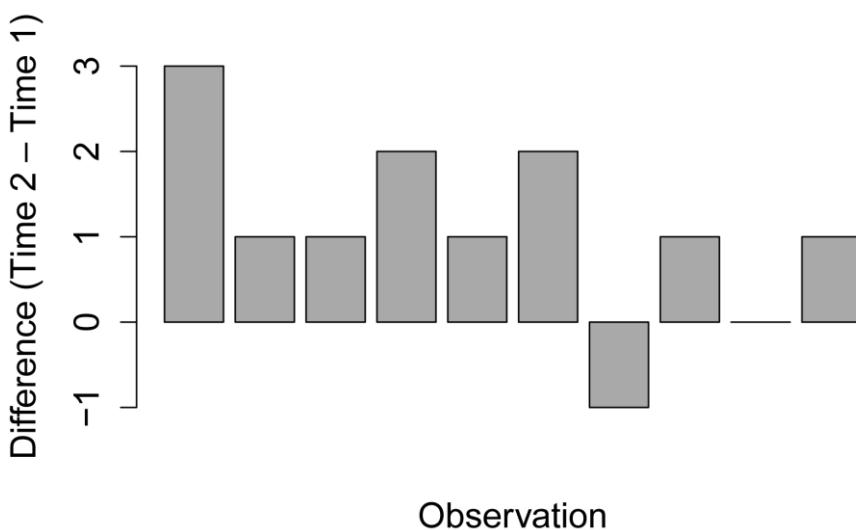
Bar plot of differences

Note that for this plot, the data must be ordered so that the first observation where $Time = 1$ is paired to the first observation where $Time = 2$, and so on.

```
Time.1 = Data$Likert[Data$Time==1]
Time.2 = Data$Likert[Data$Time==2]

Difference = Time.2 - Time.1

barplot(Difference,
        col="dark gray",
        xlab="Observation",
        ylab="Difference (Time 2 - Time 1)")
```

**Two-sample paired ordinal model**

The model is specified using formula notation with the *clmm* function (not *clm*). Here, *Likert.f* is the dependent variable and *Time* is the independent variable. *Student* is specified as the blocking variable.

It is identified as a random variable in the mixed effects model. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?clmm*.

The *p*-value for the effect of *Time* is determined using the *Anova* function in the packages *RVAideMemoire* and *car*.

Define model

```
library(ordinal)

model = clmm(Likert.f ~ Time + (1|Student),
             data = Data)
```

Analysis of deviance

```
library(car)

library(RVAideMemoire)

Anova(model,
      type = "II")

Analysis of Deviance Table (Type II tests)

  LR Chisq Df Pr(>Chisq)
Time  9.2033  1   0.002416 **
```

Determine the effect of the random variable

The effect of the random variable (*Student*, in this case) can be determined by comparing the full model to a model with only the fixed effects (*Time*, in this case).

First, *model.fixed* is defined, and then the two models are passed to the *anova* function.

```
model.fixed = clm(Likert.f ~ Time,
                  data = Data)

anova(model,
      null = model.fixed)

  no.par    AIC  logLik LR.stat df Pr(>Chisq)
model.null     4 45.262 -18.631
model          5 47.242 -18.621    0.02  1    0.8875
```

Check model assumptions

At the time of writing, the *nominal_test* and *scale_test* functions don't work with *clmm* model objects, so we will define a similar model with *clm* and no random effects, and test the proportional odds assumption on this model. In this example, neither the *nominal_test* nor *scale_test* functions produced a *p*-value, so the results of these tests are not conclusive.

```
model.clm = clm(Likert.f ~ Time + Student,
```

```

data = Data)

nominal_test(model.clm)

Tests of nominal effects

formula: Likert.f ~ Time + Student
      Df  logLik   AIC LRT Pr(>Chi)
<none>    -11.821 49.642
Time
Student

### No p-value produced for this example.

scale_test(model.clm)

Tests of scale effects

formula: Likert.f ~ Time + Student
      Df  logLik   AIC LRT Pr(>Chi)
<none>    -11.821 49.642
Time
Student

Warning messages:
1: (-2) Model failed to converge: degenerate Hessian with 1 negative eigenvalues
In addition: maximum number of consecutive Newton modifications reached
2: (-1) Model failed to converge with max|grad| = 6.85558e-06 (tol = 1e-06)
In addition: maximum number of consecutive Newton modifications reached

### No p-value produced for this example.
### Errors produced for this example.

```

One-way Ordinal Regression with CLM

Appropriate data

- One-way data with two or more groups
- Dependent variable is ordered factor
- Independent variable is a factor with at least two levels or groups
- Observations between groups are not paired or repeated measures

Interpretation

A significant result can be interpreted as, "There was a significant difference among groups." Or, "There was a significant effect of Independent Variable."

A significant post-hoc analysis indicates, "There was a significant difference between Group A and Group B", and so on.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- ordinal
- car
- RVAideMemoire
- multcompView
- emmeans
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(ordinal)){install.packages("ordinal")}
if(!require(car)){install.packages("car")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

One-way ordinal regression example

```
Input =""
Speaker Likert
Pooh      3
Pooh      5
Pooh      4
Pooh      5
Pooh      5
Piglet    2
Piglet    4
Piglet    2
Piglet    2
Piglet    1
Piglet    2
Piglet    3
Piglet    2
Piglet    2
Piglet    3
Tigger    4
```

```

Tigger    4
Tigger    4
Tigger    4
Tigger    5
Tigger    3
Tigger    5
Tigger    4
Tigger    4
Tigger    3
")

Data = read.table(textConnection(Input),header=TRUE)

### Order levels of the factor; otherwise R will alphabetize them

Data$Speaker = factor(Data$Speaker,
                      levels=unique(Data$Speaker))

### Create a new variable which is the likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                       ordered = TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Summarize data treating Likert scores as factors

```

xtabs(~ Speaker + Likert.f,
      data = Data)

      Likert.f
Speaker 1 2 3 4 5
Pooh    0 0 1 6 3
Piglet  1 6 2 1 0
Tigger  0 0 2 6 2

```

```
XT = xtabs(~ Speaker + Likert.f,
           data = Data)
```

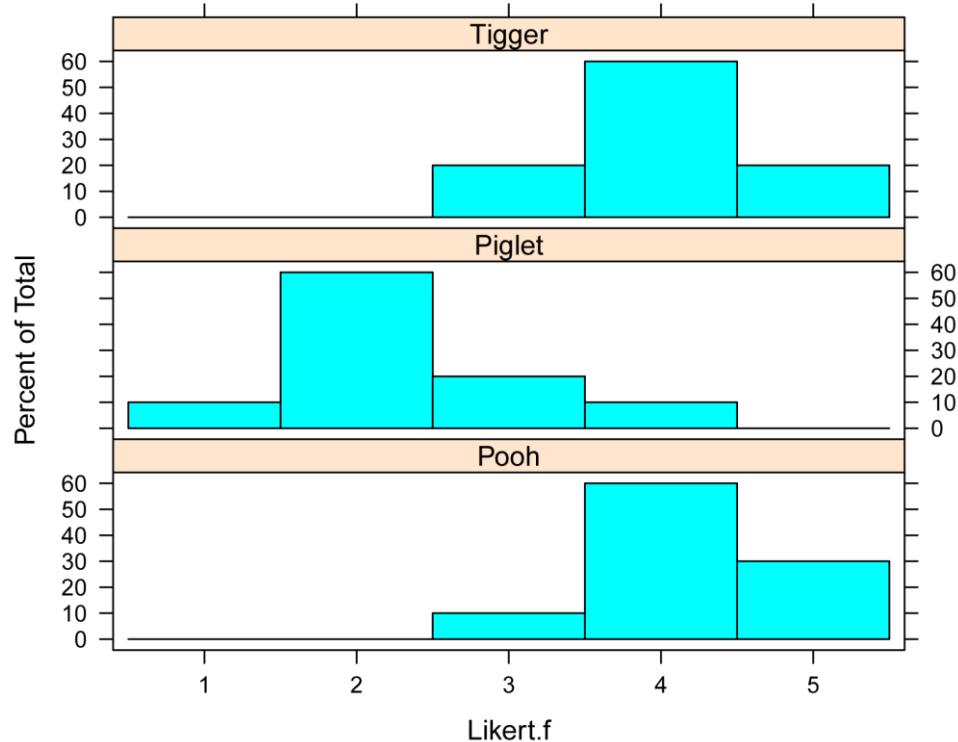
```
prop.table(XT,
           margin = 1)
```

	Likert.f				
Speaker	1	2	3	4	5
Pooh	0.0	0.0	0.1	0.6	0.3
Piglet	0.1	0.6	0.2	0.1	0.0
Tigger	0.0	0.0	0.2	0.6	0.2

Bar plots by group

```
library(lattice)

histogram(~ Likert.f | speaker,
          data=Data,
          layout=c(1,3)      # columns and rows of individual plots
         )
```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Speaker,
          data=Data,
          digits=3)
```

	Speaker	n	mean	sd	min	Q1	median	Q3	max	percZero
1	Pooh	10	4.2	0.632	3	4	4	4.75	5	0
2	Piglet	10	2.3	0.823	1	2	2	2.75	4	0
3	Tigger	10	4.0	0.667	3	4	4	4.00	5	0

One-way ordinal regression

The model is specified using formula notation. Here, *Likert.f* is the dependent variable and *Speaker* is the independent variable. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?clm*.

The *p*-value for the effect of *Speaker* is determined using the *Anova* function in the packages *RVAideMemoire* and *car*.

Define model

```
library(ordinal)

model = clm(Likert.f ~ Speaker,
            data = Data)
```

Analysis of deviance

```
library(car)

library(RVAideMemoire)

Anova(model,
      type = "II")

Analysis of Deviance Table (Type II tests)

LR Chisq Df Pr(>Chisq)
Speaker 23.395 2 8.315e-06 ***
```

Post-hoc test with emmeans

In the *emmeans* function, *model* specifies the model object that was previously fitted. Note the specialized formula where *pairwise* indicates that all pairwise comparisons should be conducted, and *Instructor* indicates the variable whose levels will be compared.

Here we'll create an object of the *emmeans* output called *marginal*. Then we'll pass this object to the *cld* function to create a compact letter display.

Be sure to read the *Estimated Marginal Means for Multiple Comparisons* chapter for correct interpretation of estimated marginal means. For *clm* model objects, the *emmean*, *SE*, *LCL*, and *UCL* values should be ignored, unless specific options in *emmeans* are selected.

```

library(multcompView)
library(emmeans)

marginal = emmeans(model,
~ Speaker)

pairs(marginal,
adjust="tukey")

cld(marginal,
alpha=0.05,
Letters=letters,      ### Use lower-case letters for .group
adjust="tukey")       ### Tukey-adjusted comparisons

Speaker    emmean      SE df  asymp.LCL  asymp.UCL .group
Piglet    -1.783599 0.7755958 NA -3.6355182 0.06832068   a
Tigger     2.526492 0.8413200 NA  0.5176407 4.53534421   b
Pooh      3.160223 0.8912968 NA  1.0320404 5.28840661   b

Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05

### Groups sharing a letter in .group are not significantly different
### Remember to ignore "emmean", "SE", "LCL", and "UCL" for CLM

```

Check model assumptions

```

nominal_test(model)

Tests of nominal effects

formula: Likert.f ~ speaker
          Df  logLik      AIC LRT Pr(>Chi)
<none>      -30.149 72.298
Speaker

### No p-value

scale_test(model)

Tests of scale effects

formula: Likert.f ~ speaker
          Df  logLik      AIC      LRT Pr(>Chi)
<none>      -30.149 72.298
Speaker    2 -29.839 75.677 0.62096  0.7331

### No violation in assumptions

```

One-way Repeated Ordinal Regression with CLMM

Appropriate data

- Two-way data with one treatment variable and one blocking variable
- Dependent variable is ordered factor
- Treatment variable is a factor with at least two levels or groups
- Blocking variable is a factor with at least two levels
- Observations can be paired or repeated measures within blocks

Interpretation

A significant result can be interpreted as, "There was a significant difference among groups." Or, "There was a significant effect of Independent Variable."

A significant post-hoc analysis indicates e.g. "There was a significant difference between Group A and Group B", and so on.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- ordinal
- car
- RVAideMemoire
- multcompView
- emmeans
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(ordinal)){install.packages("ordinal")}
if(!require(car)){install.packages("car")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(multcompView)){install.packages("multcompView")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

One-way repeated ordinal regression example

In this example, we want to determine if there is a difference in scores among five instructors from the Belcher family. Because we know which rater gave each score, we will use the *Rater* as a blocking variable, with the suspicion that one rater might rate consistently higher than another one. If we hadn't recorded this information about raters, we could use a one-way ordinal regression.

```
Input =("
Instructor      Rater Likert
'Bob Belcher'   a     4
'Bob Belcher'   b     5
'Bob Belcher'   c     4
'Bob Belcher'   d     6
'Bob Belcher'   e     6
'Bob Belcher'   f     6
'Bob Belcher'   g     10
'Bob Belcher'   h     6
'Linda Belcher' a     8
'Linda Belcher' b     6
'Linda Belcher' c     8
'Linda Belcher' d     8
'Linda Belcher' e     8
'Linda Belcher' f     7
'Linda Belcher' g     10
'Linda Belcher' h     9
'Tina Belcher'  a     7
'Tina Belcher'  b     5
'Tina Belcher'  c     7
'Tina Belcher'  d     8
'Tina Belcher'  e     8
'Tina Belcher'  f     9
'Tina Belcher'  g     10
'Tina Belcher'  h     9
'Gene Belcher'  a     6
'Gene Belcher'  b     4
'Gene Belcher'  c     5
'Gene Belcher'  d     5
'Gene Belcher'  e     6
'Gene Belcher'  f     6
'Gene Belcher'  g     5
'Gene Belcher'  h     5
'Louise Belcher' a     8
'Louise Belcher' b     7
'Louise Belcher' c     8
'Louise Belcher' d     8
'Louise Belcher' e     9
'Louise Belcher' f     9
'Louise Belcher' g     8
'Louise Belcher' h     10
")
```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```

### Order levels of the factor; otherwise R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                          levels=unique(Data$Instructor))

### Create a new variable which is the likert scores as an ordered factor

Data$Likert.f = factor(Data$Likert,
                       ordered=TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Summarize data treating Likert scores as factors

```

XT = xtabs(~ Instructor + Likert.f,
           data = Data)

XT

      Likert.f
Instructor   4 5 6 7 8 9 10
  Bob Belcher 2 1 4 0 0 0 1
  Linda Belcher 0 0 1 1 4 1 1
  Tina Belcher 0 1 0 2 2 2 1
  Gene Belcher 1 4 3 0 0 0 0
  Louise Belcher 0 0 0 1 4 2 1

prop.table(XT,
           margin = 1)

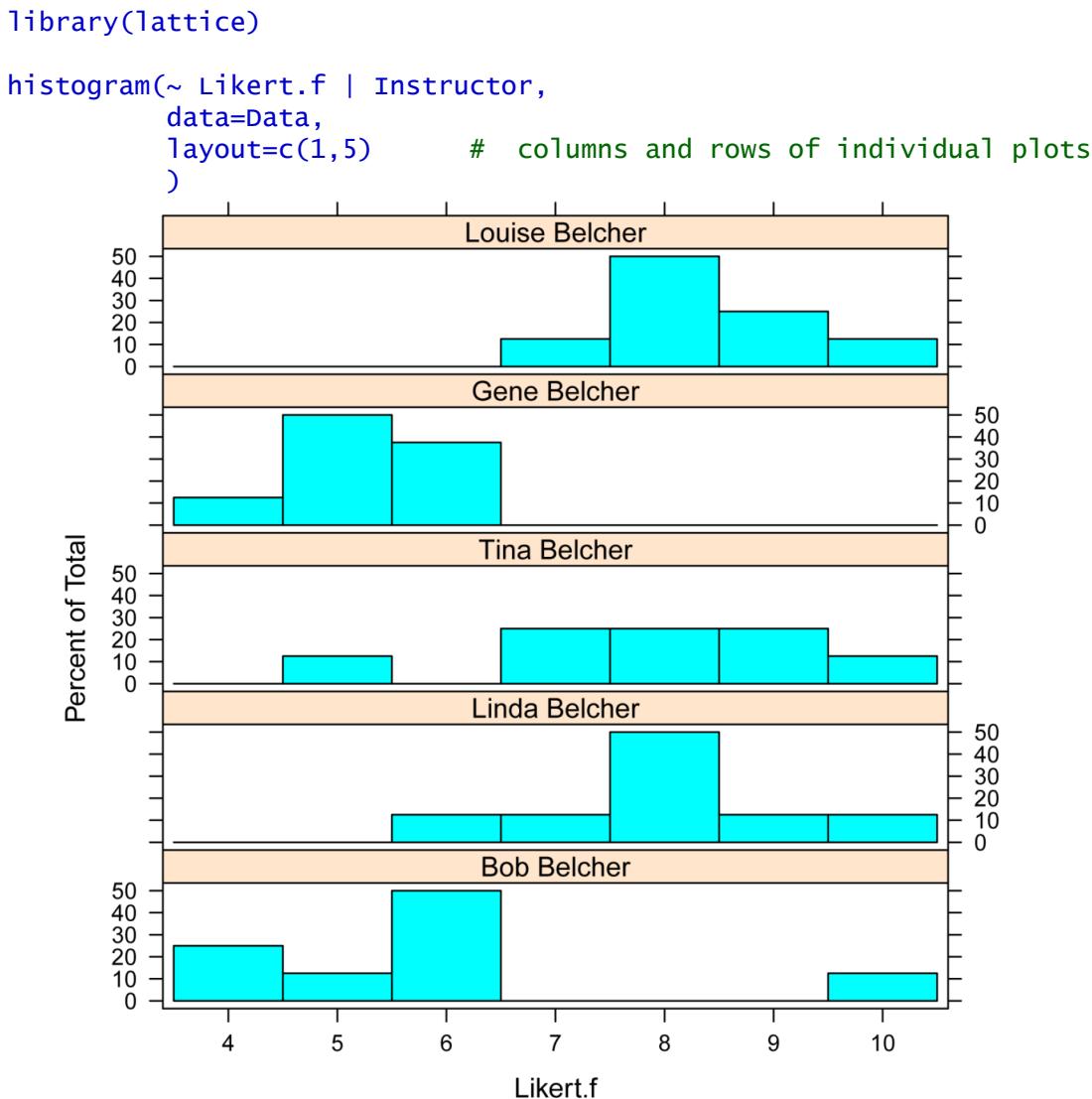
      Likert.f
Instructor   4      5      6      7      8      9      10
  Bob Belcher 0.250 0.125 0.500 0.000 0.000 0.000 0.125
  Linda Belcher 0.000 0.000 0.125 0.125 0.500 0.125 0.125
  Tina Belcher 0.000 0.125 0.000 0.250 0.250 0.250 0.125
  Gene Belcher 0.125 0.500 0.375 0.000 0.000 0.000 0.000
  Louise Belcher 0.000 0.000 0.000 0.125 0.500 0.250 0.125

```

```
### Proportion with each row
```

Bar plots by group

Note that the bar plots don't show the effect of the blocking variable.



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Instructor,
           data=Data,
           digits=3)
```

Instructor	n	mean	sd	min	Q1	median	Q3	max	percZero
Bob Belcher	8	5.875	1.885	4	4.75	6	6.00	10	0
Linda Belcher	8	8.000	1.195	6	7.75	8	8.25	10	0
Tina Belcher	8	7.875	1.553	5	7.00	8	9.00	10	0

4 Gene Belcher	8 5.250	0.707	4 5.00	5 6.00	6	0
5 Louise Belcher	8 8.375	0.916	7 8.00	8 9.00	10	0

One-way repeated ordinal regression

The model is specified using formula notation with the *clmm* function (not *clm*). Here, *Likert.f* is the dependent variable, and an ordered factor in R, and *Instructor* is the independent variable. *Rater* is specified as the blocking variable. Technically, it is identified as the *random* variable in the mixed model. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?clmm*.

The *p*-value for the effect of *Instructor* is determined using the *Anova* function in the packages *RVAideMemoire* and *car*.

Define model and conduct analysis of deviance

Here the *threshold = "equidistant"* option is used in order to avoid errors in the optional post-hoc tests. This option does not need to be used routinely.

```
library(ordinal)

model = clmm(Likert.f ~ Instructor + (1|Rater),
             data = Data,
             threshold = "equidistant")

library(car)

library(RVAideMemoire)

Anova(model,
      type = "II")

Analysis of Deviance Table (Type II tests)

          LR Chisq Df Pr(>Chisq)
Instructor 38.51  4  8.794e-08 ***
```

Determine the effect of the random variable

The effect of the random variable (*Rater*, in this case) can be determined by comparing the full model to a model with only the fixed effects (*Instructor*, in this case).

First, *model.fixed* is defined, and then the two models are passed to the *anova* function.

Be sure that the *threshold* option in *model.fixed* is the same as in the original model.

```
model.fixed = clm(Likert.f ~ Instructor,
                  data = Data,
                  threshold = "equidistant")
```

```
anova(model,
      null = model.fixed)
```

Likelihood ratio tests of cumulative link models:

	no.par	AIC	logLik	LR.stat	df	Pr(>Chisq)
model.fixed	6	135.19	-61.596			
model	7	129.46	-57.732	7.7275	1	0.005439 **

Check model assumptions

At the time of writing, the *nominal_test* and *scale_test* functions don't work with clmm model objects, so we will define a similar model with *clm* and no random effects, and test the proportional odds assumption on this model.

```
model.clm = clm(Likert.f ~ Instructor + Rater,
                  data = Data,
                  threshold = "equidistant")
```

```
nominal_test(model.clm)
```

Tests of nominal effects

	Df	logLik	AIC	LRT	Pr(>Chi)
<none>		-47.032	120.06		
Instructor	4	-42.680	119.36	8.7053	0.0689 .
Rater	7	-39.544	119.09	14.9766	0.0363 *

No violation in assumptions for *Instructor*.

We can ignore *Rater* since it was a random effect in the original model.

```
scale_test(model.clm)
```

Warning messages:

1: (-2) Model failed to converge: degenerate Hessian with 1 negative eigenvalues
In addition: maximum number of consecutive Newton modifications reached

This test failed. It would be possible to adjust the fitting parameters,
and try to get the model to converge.

Post-hoc comparisons with emmeans

Because the main effect of *Instructor* in the model was significant, we want to know which instructor has statistically different scores than which other instructors. Comparing EM means for each instructor will allow us to do this.

Be sure to read the *Estimated Marginal Means for Multiple Comparisons* chapter for correct interpretation of estimated marginal means. For clmm model objects, the *emmean*, *SE*, *LCL*, and *UCL* values should be ignored, unless specific options in *emmeans* are selected.

Here we'll create an object of the *emmeans* output called *marginal*. Then we'll pass this object to the

cld function to create a compact letter display.

The ordinal model under consideration is called *model*, created with the *clmm* function above. The formula in the *emmeans* function indicates that *pairwise* comparisons should be conducted for the variable *Instructor*.

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
~ Instructor)

pairs(marginal,
adjust="tukey")

cld(marginal,
alpha=0.05,
Letters=letters,      ### Use lower-case letters for .group
adjust="tukey")       ### Tukey-adjusted comparisons

Instructor      emmean        SE df  asymp.LCL  asymp.UCL .group
Gene Belcher   -3.957035 0.9610526 NA -6.4257521 -1.4883188 a
Bob Belcher    -2.893186 0.8879872 NA -5.1742153 -0.6121576 a
Tina Belcher   2.018826 0.7027604 NA  0.2136007  3.8240505 b
Linda Belcher  2.378266 0.7417485 NA  0.4728901  4.2836425 b
Louise Belcher 3.329647 0.8470188 NA  1.1538560  5.5054375 b

Results are averaged over the levels of: Rater
Confidence level used: 0.95
Conf-level adjustment: sidak method for 5 estimates
P value adjustment: tukey method for comparing a family of 5 estimates
significance level used: alpha = 0.05

### Groups sharing a letter in .group are not significantly different
### Remember to ignore "emmean", "SE", "LCL", and "UCL" for CLMM
```

Optional analyses

Post-hoc test: pairwise paired ordinal tests for multiple comparisons

A post-hoc analysis can be accomplished with the *pairwiseOrdinalPairedTest* or *pairwiseOrdinalPairedMatrix* functions. These functions extract the *p*-values from paired cumulative link models for each pair of treatments.

Usually you would want to first order your treatment variable (*Speaker*) by median or other location statistic for the letters in the compact letter display to be in their proper order.

Note that for this function, the data must be ordered by the blocking variable so that the first observation where *Instructor* = *Bob* is paired to the first observation where *Instructor* = *Linda*, and so

on.

Here the *p*-values for the pairwise comparisons are adjusted with the *fdr* method. See *?p.adjust* for options and details on *p*-value adjustment methods.

Here the *threshold = "equidistant"* option is used in order to avoid errors. This option does not need to be used routinely.

```
### Order groups by median

Data$Instructor = factor(Data$Instructor,
                         levels = c("Linda Belcher", "Louise Belcher",
                                   "Tina Belcher", "Bob Belcher",
                                   "Gene Belcher"))

### Pairwise ordinal regression tests

library(rcompanion)

PT = pairwiseOrdinalPairedTest(Likert.f ~ Instructor | Rater,
                               data      = Data,
                               threshold = "equidistant",
                               method    = "fdr")
                               # Adjusts p-values for multiple comparisons;
                               # See ?p.adjust for options

PT

      Comparison   p.value   p.adjust
1 Linda Belcher - Louise Belcher = 0 0.3565 4.456e-01
2 Linda Belcher - Tina Belcher = 0 0.5593 5.593e-01
3 Linda Belcher - Bob Belcher = 0 0.00231 3.850e-03
4 Linda Belcher - Gene Belcher = 0 1.485e-05 7.425e-05
5 Louise Belcher - Tina Belcher = 0 0.2347 3.353e-01
6 Louise Belcher - Bob Belcher = 0 0.001071 2.142e-03
7 Louise Belcher - Gene Belcher = 0 1.247e-07 1.247e-06
8 Tina Belcher - Bob Belcher = 0 0.001013 2.142e-03
9 Tina Belcher - Gene Belcher = 0 0.0001283 4.277e-04
10 Bob Belcher - Gene Belcher = 0 0.5152 5.593e-01

### Compact letter display

library(rcompanion)

cldList(p.adjust ~ Comparison,
        data = PT,
        threshold = 0.05)

      Group Letter MonoLetter
1 LindaBelcher     a       a
2 LouiseBelcher    a       a
3 TinaBelcher      a       a
4 BobBelcher       b       b
```

5	Gene Belcher	b	b	
---	--------------	---	---	--

Groups sharing a letter are not significantly different (alpha = 0.05).

Matrix output and compact letter display

```
### Order groups by median

Data$Instructor = factor(Data$Instructor,
                          levels = c("Linda Belcher", "Louise Belcher",
                                    "Tina Belcher", "Bob Belcher",
                                    "Gene Belcher"))

### Pairwise ordinal regression tests

library(rcompanion)

PM = pairwiseordinalPairedMatrix(Likert.f ~ Instructor | Rater,
                                   data      = Data,
                                   threshold = "equidistant",
                                   method    = "fdr")
# Adjusts p-values for multiple comparisons;
# See ?p.adjust for options

PM

$Adjusted
      Linda Belcher Louise Belcher Tina Belcher Bob Belcher Gene Belcher
Linda Belcher 1.000e+00 4.456e-01 0.5593000 0.003850 7.425e-05
Louise Belcher 4.456e-01 1.000e+00 0.3353000 0.002142 1.247e-06
Tina Belcher 5.593e-01 3.353e-01 1.0000000 0.002142 4.277e-04
Bob Belcher 3.850e-03 2.142e-03 0.0021420 1.000000 5.593e-01
Gene Belcher 7.425e-05 1.247e-06 0.0004277 0.559300 1.000e+00

library(multcompView)

multcompLetters(PM$Adjusted,
                compare = "<",
                threshold = 0.05, # p-value to use as significance threshold
                Letters = letters,
                reversed = FALSE)

Linda Belcher Louise Belcher Tina Belcher Bob Belcher Gene Belcher
      "a"          "a"          "a"          "b"          "b"

### Values sharing a letter are not significantly different
```

Two-way Ordinal Regression with CLM

A two-way ordinal analysis of variance can address an experimental design with two independent variables, each of which is a factor variable. The main effect of each independent variable can be tested, as well as the effect of the interaction of the two factors.

The example here looks at ratings for three instructors across four different questions. The analysis will attempt to answer the questions: a) Is there a significant difference in scores for different instructors? b) Is there a significant difference in scores for different questions? c) Is there a significant interaction effect of *Instructor* and *Question*?

Main effects and interaction effects are explained further in the *Factorial ANOVA: Main Effects, Interaction Effects, and Interaction Plots* chapter.

The *clm* function can specify more complex models with multiple independent variables of different types, but this book will not explore more complex examples.

The *p*-values for the main and interaction effects can be determined with the *Anova* function from *RVAideMemoire*, which produces an analysis of deviance table for these effects. In addition, a *p*-value for the model as a whole will be determined, along with a pseudo *R-squared* for the model as a whole.

Post-hoc analysis to determine which groups are different can be conducted on each significant main effect and on the interaction effect if it is significant.

Appropriate data

- Two-way data
- Dependent variable is ordered factor
- Independent variables are factors with at least two levels or groups each
- Observations between groups are not paired or repeated measures

Interpretation

A significant main effect can be interpreted as, “There was a significant difference among groups.” Or, “There was a significant effect of Independent Variable.”

A significant interaction effect can be interpreted as, “There was a significant interaction effect between Independent Variable A and Independent Variable B.”

A significant post-hoc analysis indicates, “There was a significant difference between Group A and Group B”, and so on.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA

- lattice
- ggplot2
- ordinal
- car
- RVAideMemoire
- multcompView
- emmeans
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(ordinal)){install.packages("ordinal")}
if(!require(car)){install.packages("car")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(multcompView)){install.packages("multcompView")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Two-way ordinal regression example

```
Input =""
Instructor Question Likert
Fuu      Informative 8
Fuu      Informative 9
Fuu      Informative 9
Fuu      Informative 8
Fuu      Delivery    10
Fuu      Delivery    9
Fuu      Delivery    8
Fuu      Delivery    8
Fuu      VisualAides 7
Fuu      VisualAides 7
Fuu      VisualAides 6
Fuu      VisualAides 7
Fuu      AnswerQuest 8
Fuu      AnswerQuest 9
Fuu      AnswerQuest 9
Fuu      AnswerQuest 8
Jin      Informative 7
Jin      Informative 8
Jin      Informative 6
Jin      Informative 5
Jin      Delivery    8
Jin      Delivery    8
Jin      Delivery    9
Jin      Delivery    6
Jin      VisualAides 5
Jin      VisualAides 6
```

```

Jin      VisualAides    7
Jin      VisualAides    7
Jin      AnswerQuest    8
Jin      AnswerQuest    7
Jin      AnswerQuest    6
Jin      AnswerQuest    6
Mugen   Informative     5
Mugen   Informative     4
Mugen   Informative     3
Mugen   Informative     4
Mugen   Delivery        8
Mugen   Delivery        9
Mugen   Delivery        8
Mugen   Delivery        7
Mugen   VisualAides    5
Mugen   VisualAides    4
Mugen   VisualAides    4
Mugen   VisualAides    5
Mugen   AnswerQuest    6
Mugen   AnswerQuest    7
Mugen   AnswerQuest    6
Mugen   AnswerQuest    7
"))

```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
### Order levels of the factor; otherwise R will alphabetize them
```

```
Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))
```

```
### Create a new variable which is the likert scores as an ordered factor
```

```
Data$Likert.f = factor(Data$Likert,
                      ordered = TRUE)
```

```
### Check the data frame
```

```
library(psych)
```

```
headTail(Data)
```

```
str(Data)
```

```
summary(Data)
```

```
### Remove unnecessary objects
```

```
rm(Input)
```

Summarize data treating Likert scores as factors

```
xtabs( ~ Instructor + Likert.f,
      data = Data)
```

	Likert.f									
Instructor	3	4	5	6	7	8	9	10		
Fuu	0	0	0	1	3	6	5	1		
Jin	0	0	2	5	4	4	1	0		
Mugen	1	4	3	2	3	2	1	0		

```
xtabs( ~ Question + Likert.f,
      data = Data)
```

	Likert.f									
Question	3	4	5	6	7	8	9	10		
AnswerQuest	0	0	0	4	3	3	2	0		
Delivery	0	0	0	1	1	6	3	1		
Informative	1	2	2	1	1	3	2	0		
visualAides	0	2	3	2	5	0	0	0		

```
xtabs( ~ Instructor + Likert.f + Question,
      data = Data)
```

```
, , Question = AnswerQuest
```

	Likert.f									
Instructor	3	4	5	6	7	8	9	10		
Fuu	0	0	0	0	0	2	2	0		
Jin	0	0	0	2	1	1	0	0		
Mugen	0	0	0	2	2	0	0	0		

```
, , Question = Delivery
```

	Likert.f									
Instructor	3	4	5	6	7	8	9	10		
Fuu	0	0	0	0	0	2	1	1		
Jin	0	0	0	1	0	2	1	0		
Mugen	0	0	0	0	1	2	1	0		

```
, , Question = Informative
```

	Likert.f									
Instructor	3	4	5	6	7	8	9	10		
Fuu	0	0	0	0	0	2	2	0		
Jin	0	0	1	1	1	1	0	0		
Mugen	1	2	1	0	0	0	0	0		

```
, , Question = visualAides
```

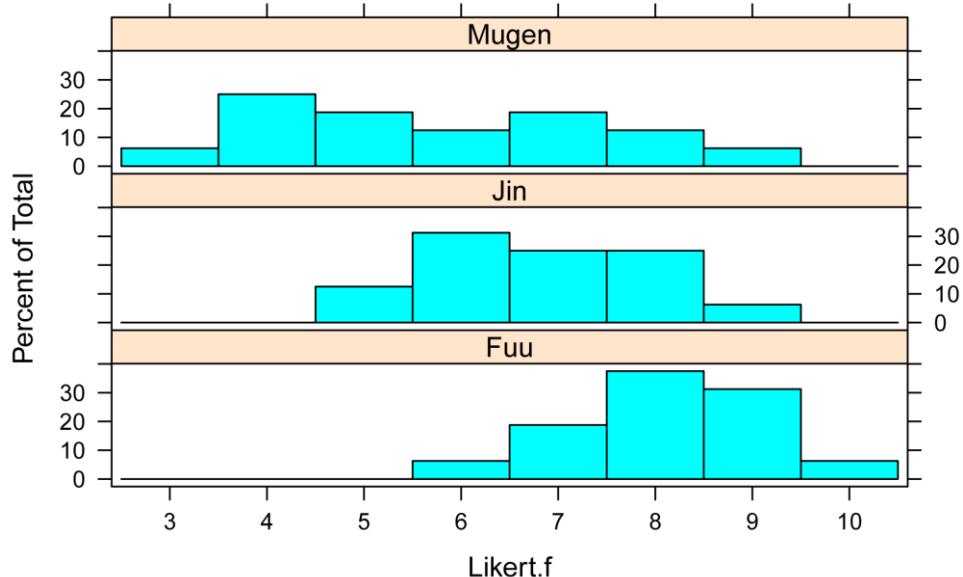
	Likert.f									
--	----------	--	--	--	--	--	--	--	--	--

Instructor	3	4	5	6	7	8	9	10
Fuu	0	0	0	1	3	0	0	0
Jin	0	0	1	1	2	0	0	0
Mugen	0	2	2	0	0	0	0	0

Bar plots by group

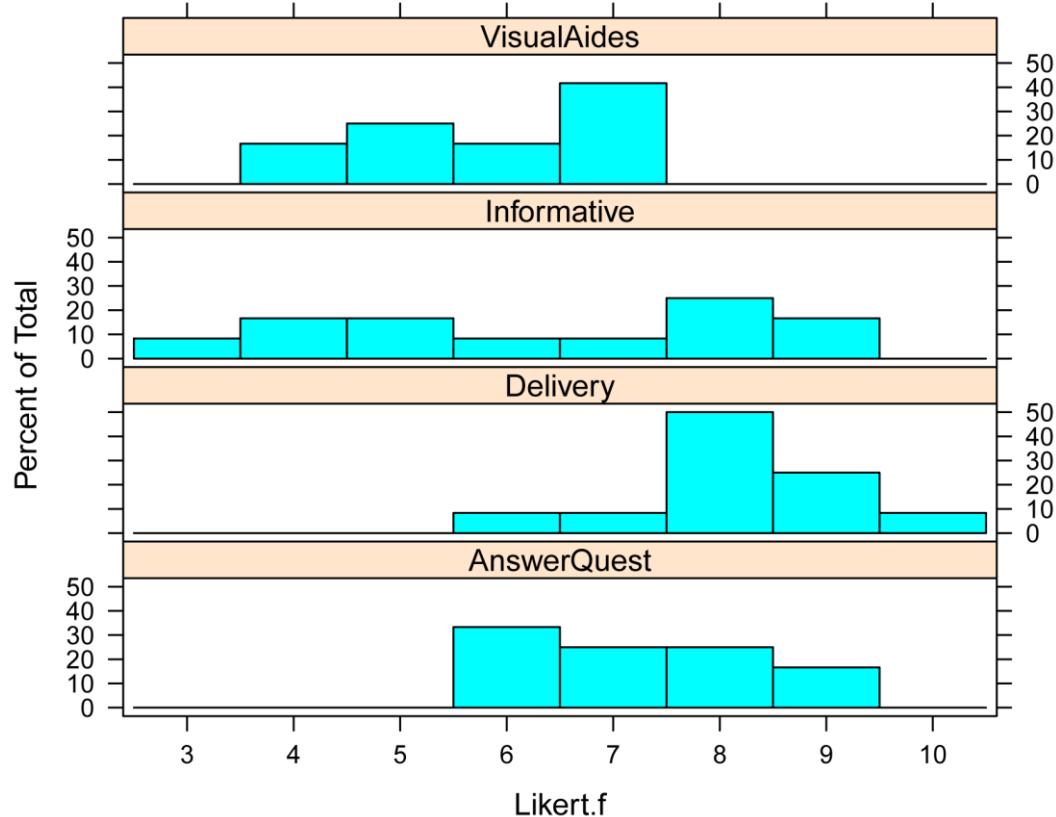
```
library(lattice)

histogram(~ Likert.f | Instructor,
          data=Data,
          layout=c(1,3)      # columns and rows of individual plots
)
```



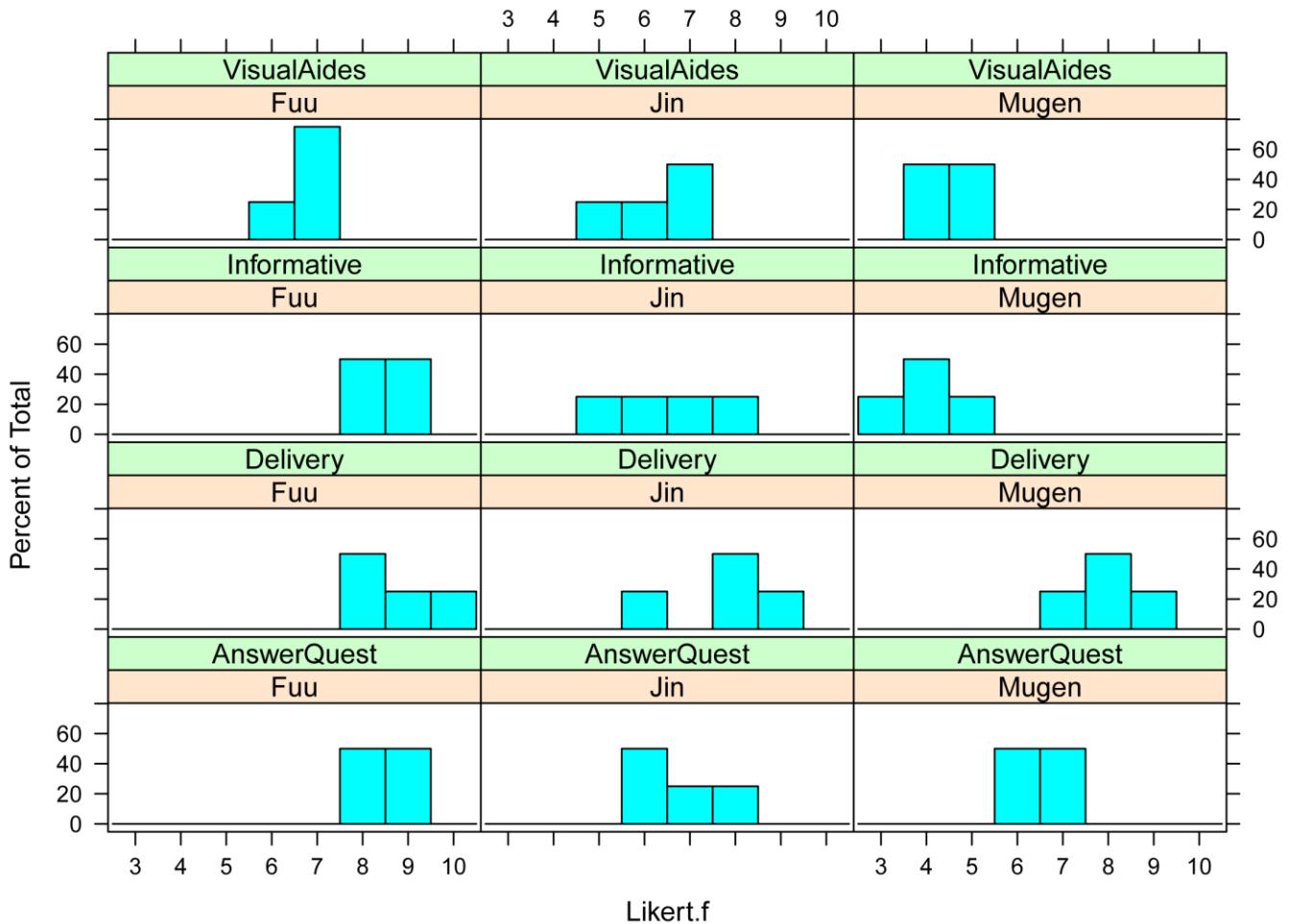
```
library(lattice)

histogram(~ Likert.f | Question,
          data=Data,
          layout=c(1,4)      # columns and rows of individual plots
)
```



```
library(lattice)

histogram(~ Likert.f | Instructor + Question,
          data=Data,
          layout=c(3,4)      # columns and rows of individual plots
)
```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Instructor,
          data=Data,
          digits=3)

  Instructor n  mean      sd min   Q1 median Q3 max percZero
1       Fuu 16 8.125 1.025   6 7.75    8.0  9  10      0
2       Jin 16 6.812 1.167   5 6.00    7.0  8  9      0
3     Mugen 16 5.750 1.770   3 4.00    5.5  7  9      0
```

```
library(FSA)
```

```
Summarize(Likert ~ Question,
          data=Data,
          digits=3)

  Question n  mean      sd min   Q1 median Q3 max percZero
1 AnswerQuest 12 7.250 1.138   6 6.00    7.0  8  9      0
```

2	Delivery	12	8.167	1.030	6	8.00	8.0	9	10	0
3	Informative	12	6.333	2.103	3	4.75	6.5	8	9	0
4	VisualAides	12	5.833	1.193	4	5.00	6.0	7	7	0

```
library(FSA)
```

```
Summarize(Likert ~ Instructor + Question,
          data=Data,
          digits=3)
```

	Instructor	Question	n	mean	sd	min	Q1	median	Q3	max	percZero
1	Fuu	AnswerQuest	4	8.50	0.577	8	8.00	8.5	9.00	9	0
2	Jin	AnswerQuest	4	6.75	0.957	6	6.00	6.5	7.25	8	0
3	Mugen	AnswerQuest	4	6.50	0.577	6	6.00	6.5	7.00	7	0
4	Fuu	Delivery	4	8.75	0.957	8	8.00	8.5	9.25	10	0
5	Jin	Delivery	4	7.75	1.258	6	7.50	8.0	8.25	9	0
6	Mugen	Delivery	4	8.00	0.816	7	7.75	8.0	8.25	9	0
7	Fuu	Informative	4	8.50	0.577	8	8.00	8.5	9.00	9	0
8	Jin	Informative	4	6.50	1.291	5	5.75	6.5	7.25	8	0
9	Mugen	Informative	4	4.00	0.816	3	3.75	4.0	4.25	5	0
10	Fuu	VisualAides	4	6.75	0.500	6	6.75	7.0	7.00	7	0
11	Jin	VisualAides	4	6.25	0.957	5	5.75	6.5	7.00	7	0
12	Mugen	VisualAides	4	4.50	0.577	4	4.00	4.5	5.00	5	0

Produce interaction plot with medians and quantiles

```
library(FSA)
```

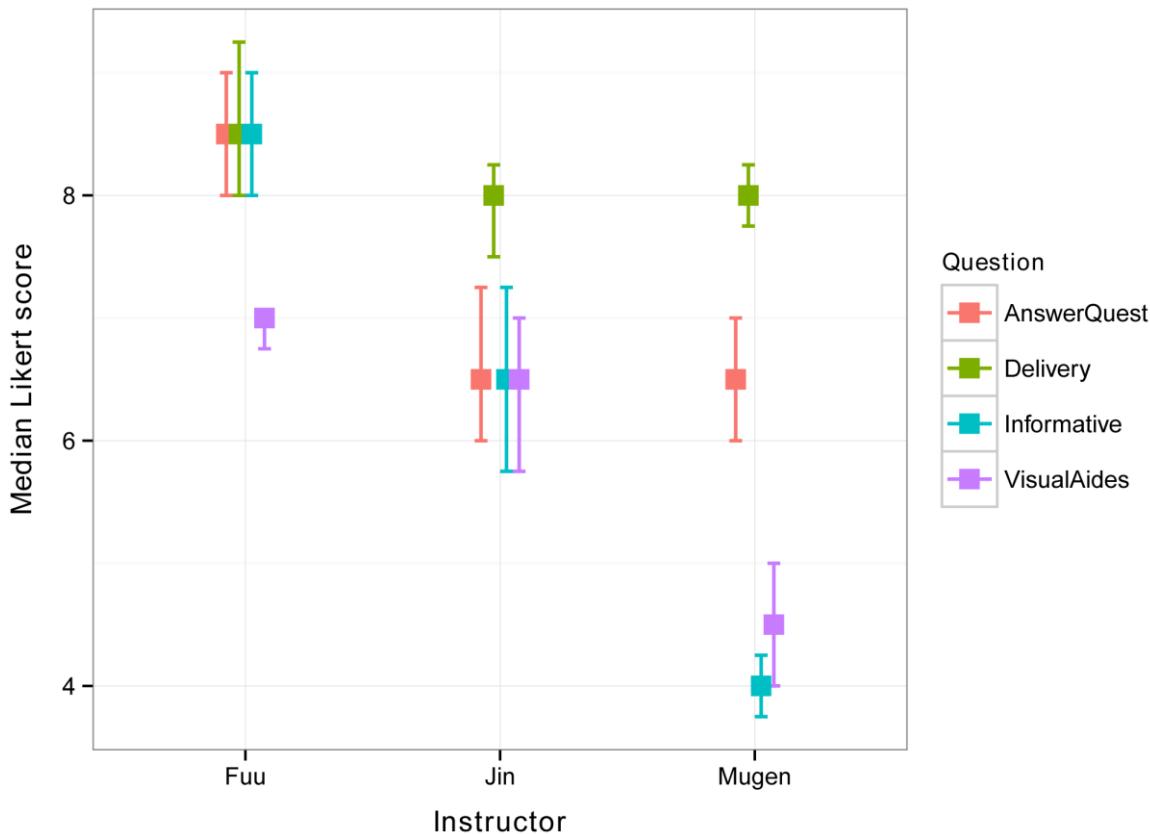
```
Sum = Summarize(Likert ~ Instructor + Question,
                 data=Data,
                 digits=3)
```

```
Sum
```

```
library(ggplot2)
```

```
pd = position_dodge(.2)

ggplot(Sum, aes(x=Instructor,
                 y=median,
                 color=Question)) +
  geom_errorbar(aes(ymin=Q1,
                     ymax=Q3),
                width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold"))+
  ylab("Median Likert score")
```



Two-way ordinal regression

In the model notation in the *clm* function, here, *Likert.f* is the dependent variable and *Instructor* and *Question* are the independent variables. The term *Instructor:Question* adds the interaction effect of these two independent variables to the model. The *data=* option indicates the data frame that contains the variables. For the meaning of other options, see *?clm*.

The *p*-values for the two main effects and the interaction effect is determined using the *Anova* function in the packages *RVAideMemoire* and *car*.

Here the *threshold = "symmetric"* option is used in order to avoid errors. This option does not need to be used routinely.

Define model and analyses of deviance

```
library(ordinal)

model = clm(Likert.f ~ Instructor + Question + Instructor:Question,
            data = Data,
            threshold="symmetric")
```

```
library(car)
```

```
library(RVAideMemoire)
```

```
Anova(model,
      type = "II")

Analysis of Deviance Table (Type II tests)

          LR Chisq Df Pr(>Chisq)
Instructor      32.157  2  1.040e-07 ***
Question        28.248  3  3.221e-06 ***
Instructor:Question 24.326  6  0.0004548 ***
```

p-value for model and pseudo R-squared

The *p*-value for the model and a pseudo *R-squared* value can be determined with the *nagelkerke* function.

```
library(rcompanion)

nagelkerke(model)

$Pseudo.R.squared.for.model.vs.null

          Pseudo.R.squared
McFadden           0.400602
Cox and Snell (ML) 0.775956
Nagelkerke (Cragg and Uhler) 0.794950

$Likelihood.ratio.test

Df.diff LogLik.diff  chisq   p.value
-11      -35.902 71.804 5.5398e-11
```

Post-hoc tests with emmeans for multiple comparisons of groups

Be sure to read the *Estimated Marginal Means for Multiple Comparisons* chapter for correct interpretation of estimated marginal means. For clm model objects, the *emmean*, *SE*, *LCL*, and *UCL* values should be ignored, unless specific options in *emmeans* are selected.

Because the interaction term in the model was significant, the group separation for the interaction effect is explored.

```
library(emmeans)

marginal = emmeans(model,
                    ~ Instructor + Question)

pairs(marginal,
      adjust="tukey")

cld(marginal,
     alpha    = 0.05,
```

```

Letters = letters,      ### Use lower-case letters for .group
adjust  = "tukey")     ### Tukey-adjusted comparisons

Instructor Question      emmean      SE df  asymp.LCL asymp.UCL .group
Mugen   Informative -6.663413e+00 1.4186237 NA -10.7176160 -2.609209 a
Mugen   VisualAides -5.262834e+00 1.2789949 NA -8.9180007 -1.607668 ab
Jin    VisualAides -4.347138e-01 0.9435048 NA -3.1311021 2.261675 bc
Jin    Informative -8.326673e-17 1.0546366 NA -3.0139854 3.013985 bcd
Mugen   AnswerQuest 4.718448e-16 0.8484277 NA -2.4246729 2.424673 c
Jin    AnswerQuest 4.347138e-01 0.9435048 NA -2.2616745 3.131102 cde
Fuu    VisualAides 5.525651e-01 0.8464847 NA -1.8665549 2.971685 cd
Jin    Delivery 3.490051e+00 1.3194708 NA -0.2807891 7.260890 cde
Mugen   Delivery 3.713121e+00 1.2254534 NA 0.2109685 7.215274 cde
Fuu    AnswerQuest 5.262834e+00 1.2789949 NA 1.6076682 8.918001 de
Fuu    Informative 5.262834e+00 1.2789949 NA 1.6076682 8.918001 de
Fuu    Delivery 5.782817e+00 1.3782347 NA 1.8440397 9.721595 e

```

Confidence level used: 0.95

Conf-level adjustment: sidak method for 12 estimates

P value adjustment: tukey method for comparing a family of 12 estimates

significance level used: alpha = 0.05

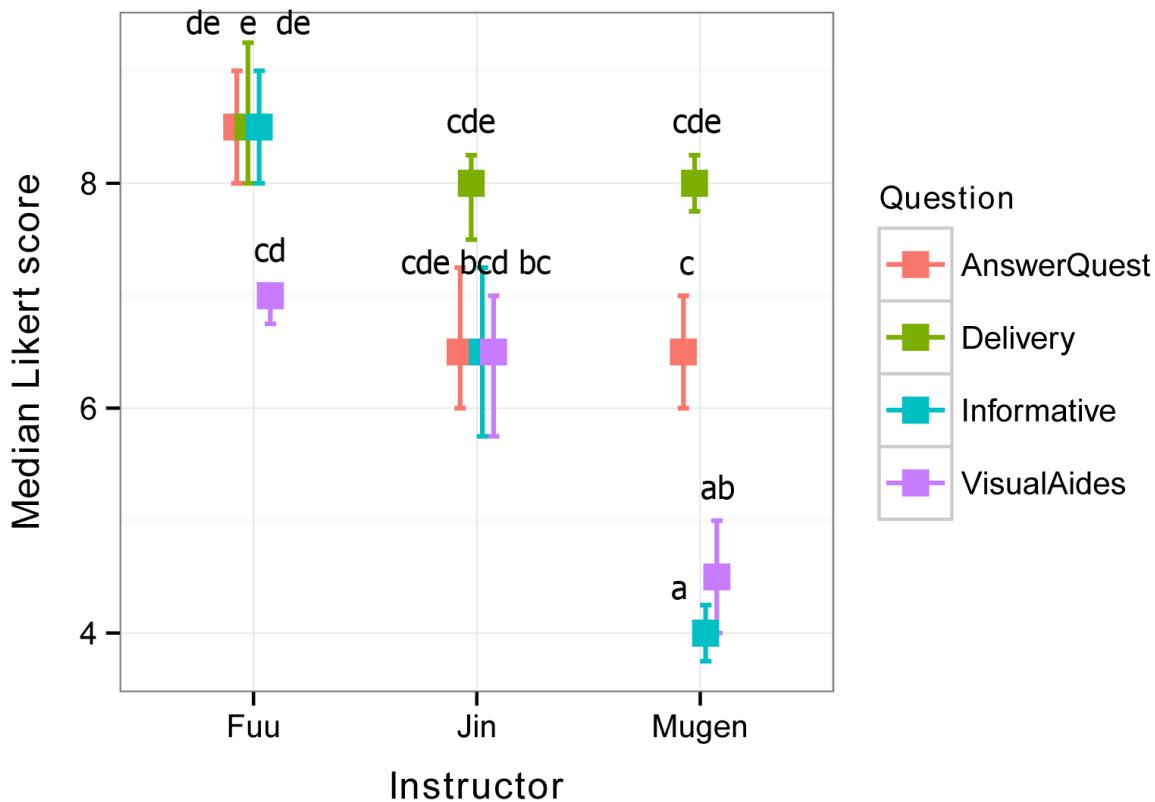
Groups sharing a letter in .group are not significantly different

Remember to ignore “emmean”, “SE”, “LCL”, and “UCL” with CLM

Interaction plot with group separation letters

Group separation letters can be added manually to an interaction plot.

Groups sharing a letter are not significantly different. In this case, because so many groups share a letter, it is difficult to interpret this plot. One approach would be to look at differences among instructors for each question. Looking at *AnswerQuest*, Fuu's scores are not statistically different than Jin's (because they share the letters *d* and *e*), but Fuu's scores are different than Mugen's (because they share no letters). So, we can conclude for this question, that Fuu's scores are significantly greater than Mugen's. And so on.



Check model assumptions

```
nominal_test(model)

Tests of nominal effects

formula: Likert.f ~ Instructor + Question + Instructor:Question
          Df  logLik     AIC      LRT Pr(>Chi)
<none>           -53.718 137.44
Instructor        6  -49.812 141.62  7.8121   0.2522
Question          9  -50.182 148.37  7.0711   0.6297
Instructor:Question
```

No violation in assumptions.

```
scale_test(model)

Tests of scale effects

formula: Likert.f ~ Instructor + Question + Instructor:Question
          Df  logLik     AIC      LRT Pr(>Chi)
<none>           -53.718 137.44
Instructor        2  -51.669 137.34  4.0985   0.12883
Question          3  -50.534 137.07  6.3669   0.09506 .
Instructor:Question
```

Warning message:

(-1) Model failed to converge with max|grad| = 1.70325e-06 (tol = 1e-06)
In addition: maximum number of consecutive Newton modifications reached

This test failed, but the results suggest no violation of assumptions.

Two-way Repeated Ordinal Regression with CLMM

A two-way repeated ordinal analysis of variance can address an experimental design with two independent variables, each of which is a factor variable, plus a blocking variable. The main effect of each independent variable can be tested, as well as the effect of the interaction of the two factors.

The example here looks at students' knowledge scores for three speakers across two different times. Because we know which student has each score at each time, we will use the *Student* as a blocking variable, with the suspicion that one student might have consistently lower knowledge than another student. If we hadn't recorded this information about students, we could use a two-way ordinal regression.

As a matter of practical interpretation, we may not care about the absolute scores for each of the three speakers, only if the knowledge scores for students increased from *Time 1* to *Time 2*. At *Time 1*, students' knowledge scores are lower for Piglet than for the other two speakers. This isn't the fault of Piglet; he is just speaking about a topic that the students have little knowledge of.

The *clmm* function can specify more complex models with multiple independent variables of different types, but this book will not explore more complex examples.

The *p*-values for the main and interaction effects can be determined with the *Anova* function from *RVAideMemoire*, which produces an analysis of deviance table for these effects. In addition, a *p*-value for the model as a whole will be determined, along with a pseudo *R-squared* for the model as a whole.

Post-hoc analysis to determine which groups are different can be conducted on each significant main effect and on the interaction effect if it is significant.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- ggplot2
- ordinal
- car

- RVAideMemoire
- emmeans
- multcompView
- plyr
- boot
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(ordinal)){install.packages("ordinal")}
if(!require(car)){install.packages("car")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(multcompview)){install.packages("multcompview")}
if(!require(plyr)){install.packages("plyr")}
if(!require(boot)){install.packages("boot")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Two-way repeated ordinal regression example

```
Input ="
Speaker Student Time Likert
Pooh      a     1     3
Pooh      b     1     5
Pooh      c     1     4
Pooh      d     1     4
Pooh      e     1     4
Pooh      f     1     3
Pooh      g     1     4
Pooh      h     1     4

Pooh      a     2     4
Pooh      b     2     5
Pooh      c     2     5
Pooh      d     2     5
Pooh      e     2     5
Pooh      f     2     4
Pooh      g     2     5
Pooh      h     2     5

Piglet    i     1     1
Piglet    j     1     2
Piglet    k     1     1
Piglet    l     1     1
Piglet    m     1     1
Piglet    n     1     2
Piglet    o     1     3
Piglet    p     1     1"
```

```

Piglet    i      2      2
Piglet    j      2      4
Piglet    k      2      2
Piglet    l      2      1
Piglet    m      2      2
Piglet    n      2      2
Piglet    o      2      4
Piglet    p      2      2

Eeyore   q      1      4
Eeyore   r      1      5
Eeyore   s      1      4
Eeyore   t      1      4
Eeyore   u      1      4
Eeyore   v      1      4
Eeyore   w      1      4
Eeyore   x      1      4

Eeyore   q      2      5
Eeyore   r      2      4
Eeyore   s      2      4
Eeyore   t      2      4
Eeyore   u      2      3
Eeyore   v      2      4
Eeyore   w      2      4
Eeyore   x      2      4
")

```

```

Data = read.table(textConnection(Input),header=TRUE)

### Change Time into a factor variable
Data$Time = factor(Data$Time)

### Order levels of the factor; otherwise R will alphabetize them
Data$Speaker = factor(Data$Speaker,
                      levels=unique(Data$Speaker))

### Create a new variable which is the Likert scores as an ordered factor
Data$Likert.f = factor(Data$Likert,
                       ordered = TRUE)

### Check the data frame
library(psych)
headTail(Data)

```

```
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)
```

Summarize data treating Likert scores as factors

```
xtabs( ~ Time + Likert.f + Speaker,
      data = Data)

  Likert.f
Time 1 2 3 4 5
  1 0 0 2 5 1
  2 0 0 0 2 6

, , Speaker = Piglet

  Likert.f
Time 1 2 3 4 5
  1 5 2 1 0 0
  2 1 5 0 2 0

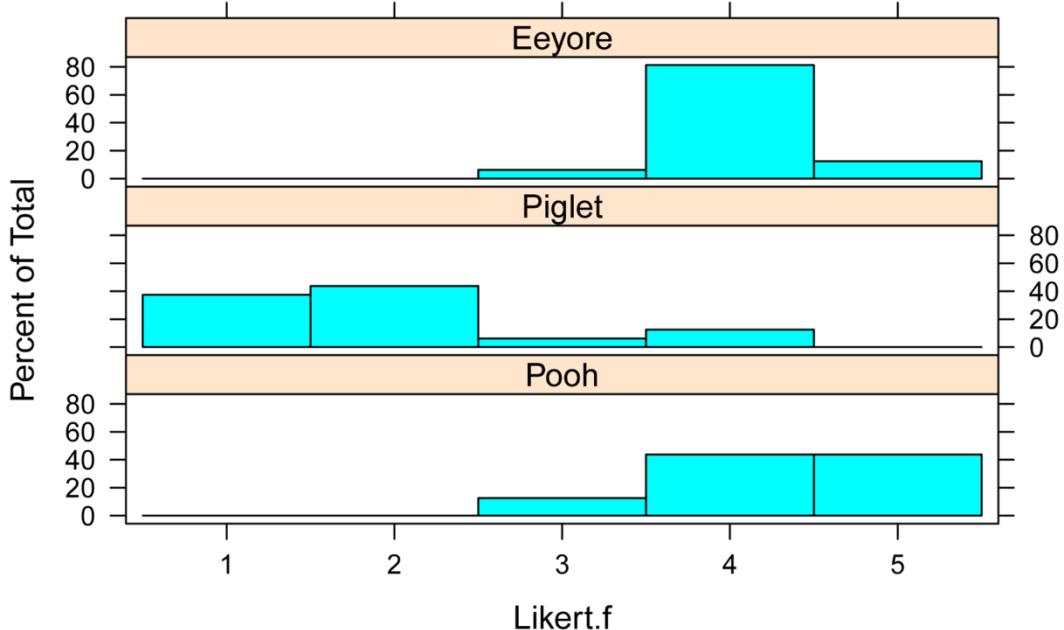
, , Speaker = Eeyore

  Likert.f
Time 1 2 3 4 5
  1 0 0 0 7 1
  2 0 0 1 6 1
```

Bar plots by group

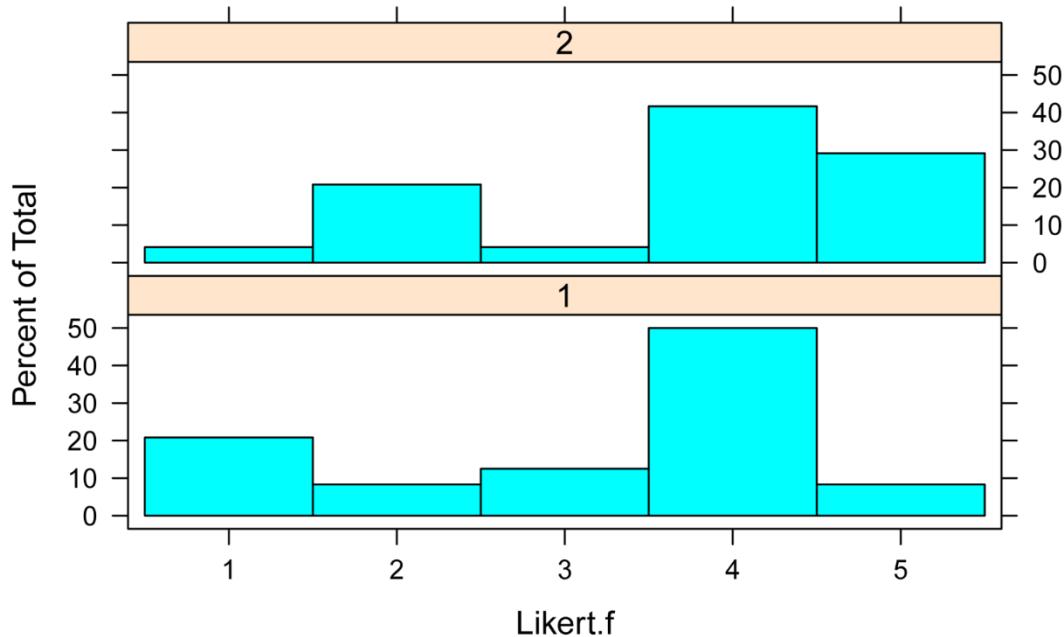
```
library(lattice)

histogram(~ Likert.f | speaker,
          data=Data,
          layout=c(1,3)      # columns and rows of individual plots
        )
```



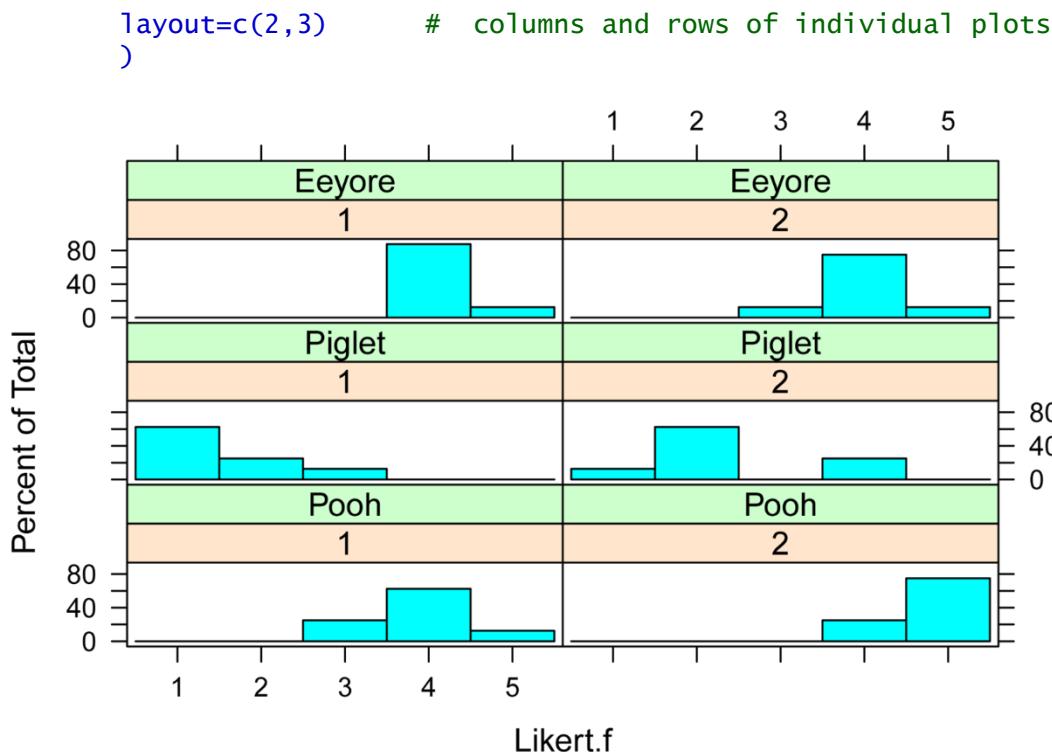
```
library(lattice)

histogram(~ Likert.f | Time,
  data=Data,
  layout=c(1,2)      # columns and rows of individual plots
)
```



```
library(lattice)

histogram(~ Likert.f | Time + Speaker,
  data=Data,
```



Summarize data treating Likert scores as numeric

```
library(FSA)

Summarize(Likert ~ Speaker + Time,
          data=Data,
          digits=3)
```

	Speaker	Time	n	nvalid	mean	sd	min	Q1	median	Q3	max	percZero
1	Pooh	1	8	8	3.875	0.641	3	3.75	4	4.0	5	0
2	Piglet	1	8	8	1.500	0.756	1	1.00	1	2.0	3	0
3	Eeyore	1	8	8	4.125	0.354	4	4.00	4	4.0	5	0
4	Pooh	2	8	8	4.750	0.463	4	4.75	5	5.0	5	0
5	Piglet	2	8	8	2.375	1.061	1	2.00	2	2.5	4	0
6	Eeyore	2	8	8	4.000	0.535	3	4.00	4	4.0	5	0

Plot Likert scores before and after

Since we are most interested in pairing scores from *Time 1* to *Time 2*, we can plot the data in a way that pairs points.

Points above and to left of the blue line indicate cases in which the score for *Time 2* is greater than that for *Time 1*.

Note that the data must be ordered by *Student* and *Speaker* for this code to plot the data correctly. That is, the first observation for *Time==1* must be the same student and speaker as the first observation for *Time==2*.

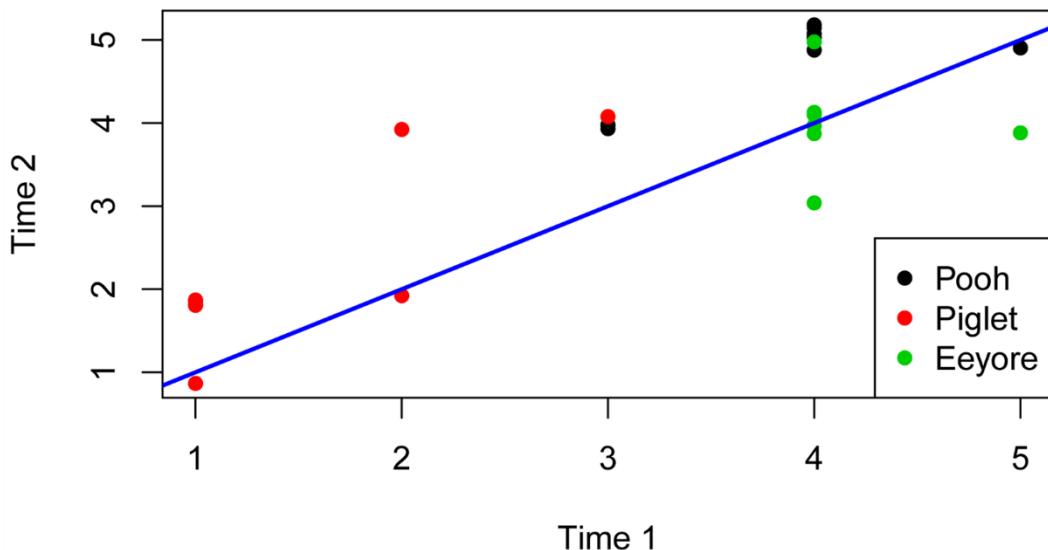
Also note that the points on the plot are jittered so that multiple points with the same values will be visible.

```
Time.1 = Data$Likert[Data$Time==1]
Time.2 = Data$Likert[Data$Time==2]
Speaker = Data$Speaker[Data$Time==2]

plot(Time.1, jitter(Time.2),
      pch = 16,
      cex = 1,
      col = Speaker,
      xlab="Time 1",
      ylab="Time 2")

abline(0,1, col="blue", lwd=2)

legend('bottomright',
       legend = unique(Speaker),
       col = 1:3,
       cex = 1,
       pch = 16)
```



Produce interaction plot with medians and confidence intervals

The following plot will show the interaction between *Speaker* and *Time*. The median for each *Speaker* \times *Time* combination is plotted, and the confidence intervals for each are shown with error bars. In this case some of the medians and confidence limits are the same values, so some error bars are not visible.

The plot makes it easy to see the change in median score for each speaker from *Time 1* to *Time 2*.

```
library(rcompanion)

Sum = groupwiseMedian(Likert ~ Speaker + Time,
                      data      = Data,
```

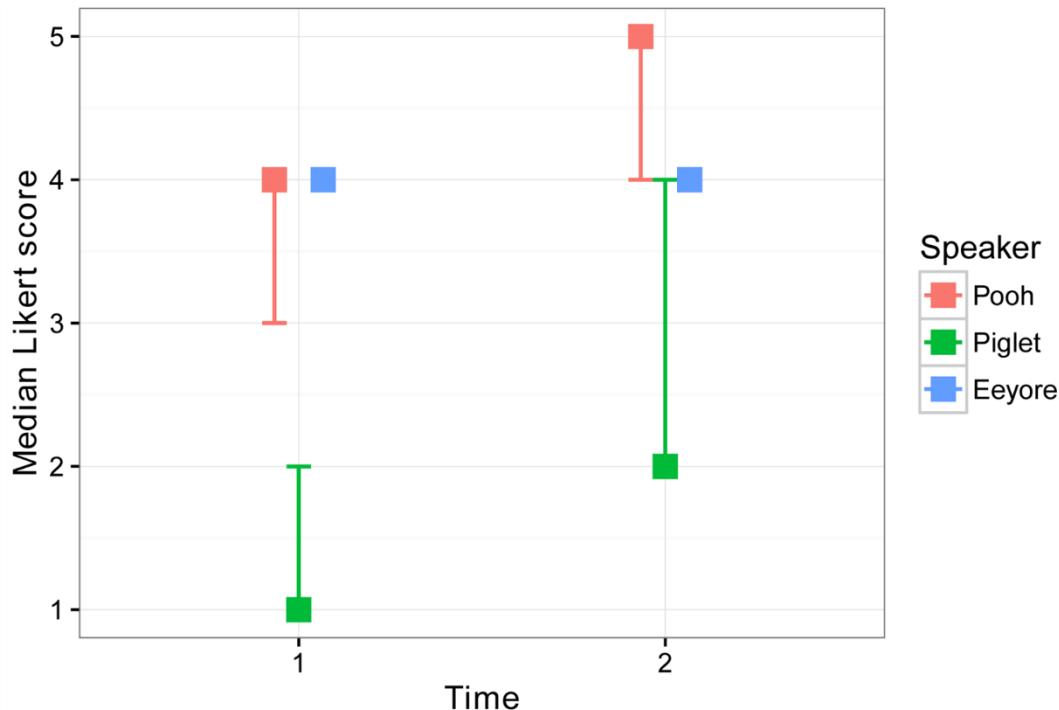
```
conf      = 0.95,
R        = 5000,
percentile = TRUE,
bca      = FALSE,
digits    = 3)
```

Sum

```
library(ggplot2)

pd = position_dodge(.2)

ggplot(Sum, aes(x=Time,
                 y=Median,
                 color=Speaker)) +
  geom_errorbar(aes(ymin=Percentile.lower,
                     ymax=Percentile.upper),
                width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Median Likert score")
```



Two-way repeated ordinal regression

In the model notation in the *clmm* function, here, *Likert.f* is the dependent variable and *Speaker* and *Time* are the independent variables. The term *Time:Speaker* adds the interaction effect of these two independent variables to the model. *Student* is used as a blocking variable, and is entered as a random variable. The *data=* option indicates the data frame that contains the variables. For the meaning of

other options, see `?clmm`.

The *p*-values for the two main effects and the interaction effect is determined using the *Anova* function in the packages *RVAideMemoire* and *car*.

Here the *threshold = "equidistant"* option is used in order to avoid errors for this data set. This option does not need to be used routinely.

Optional technical note

In the model below *Student* is treated as a simple random blocking variable, specified by *(1|Student)*. Note that in the data, students were lettered *a–x* straight through the *Speakers*. If the students had been lettered *a–h* for each *Speaker*, but there were actually 24 students, we would need to tell the model that *Student* was nested within *Speaker*, by specifying that the random effect was *(1|Speaker/Student)*, or that the random effect of interest was the interaction of *Speaker* and *Student*, by specifying *(1|Speaker:Student)*. This second specification is equivalent to that in the code below, with the original data, and will produce the same results.

I consider it a good practice to assign unique individuals with unique labels. This practice is not always followed, but following the practice will avoid mis-specifying models. In this case with 24 students, if they had been lettered *a–h* for each *Speaker*, and the original model with the *(1|Student)* effect had been used, the results would be wrong because the model would treat Pooh's student *a* as the same individual as Piglet's student *a*.

Define model and conduct analysis of deviance

```
library(ordinal)

model = clmm(Likert.f ~ Time + Speaker + Time:Speaker + (1|Student),
             data = Data,
             threshold = "equidistant")

library(car)

library(RVAideMemoire)

Anova(model,
      type = "II")

Analysis of Deviance Table (Type II tests)

          LR Chisq Df Pr(>Chisq)
Time        9.774  1   0.00177 ***
Speaker     34.485  2   3.249e-08 ***
Time:Speaker 28.202  2   7.517e-07 ***
```

p-value for model and pseudo R-squared

In order to use the *nagelkerke* function for a *clmm* model object, a null model has to be specified explicitly. (This is not the case with a *clm* model object).

We want the null model to have no fixed effects except for an intercept, indicated with a 1 on the right side of the ~. But we also may want to include the random effects, in this case (1 / Student).

We can compare our full model against a model with an overall intercept and an intercept for each student.

Be sure that the *threshold* option in *model.null* is the same as in the original model.

```
model.null = clmm(Likert.f ~ 1 + (1 | student),
                   data = Data,
                   threshold = "equidistant")

library(rcompanion)

nagelkerke(fit = model,
           null = model.null)

$Pseudo.R.squared.for.model.vs.null
                                         Pseudo.R.squared
McFadden                               0.544198
Cox and Snell (ML)                     0.787504
Nagelkerke (Cragg and Uhler)          0.836055

$Likelihood.ratio.test
Df.diff LogLik.diff Chisq   p.value
-5      -37.172 74.344 1.275e-14
```

Another approach to determining the *p*-value and pseudo *R-squared* for a clmm model is comparing the model to a null model with only an intercept and neither the fixed nor the random effects.

```
model.null.2 = clm(Likert.f ~ 1,
                    data = Data,
                    threshold = "equidistant")

library(rcompanion)

nagelkerke(fit = model,
           null = model.null.2)

$Pseudo.R.squared.for.model.vs.null
                                         Pseudo.R.squared
McFadden                               0.587734
Cox and Snell (ML)                     0.842667
Nagelkerke (Cragg and Uhler)          0.880526

$Likelihood.ratio.test
Df.diff LogLik.diff Chisq   p.value
-6      -44.385 88.771 5.4539e-17
```

Determine the effect of the random variable

The effect of the random variable (*Student*, in this case) can be determined by comparing the full model to a model with only the fixed effects (*Speaker*, *Time*, and *Speaker:Time* in this case).

First, *model.fixed* is defined, and then the two models are passed to the *anova* function.

Be sure that the *threshold* option in *model.fixed* is the same as in the original model.

```
model.fixed = clm(Likert.f ~ Time + Speaker + Time:Speaker,
                   data = Data,
                   threshold = "equidistant")
```

```
anova(model,
      model.fixed)
```

Likelihood ratio tests of cumulative link models:

	no.par	AIC	logLik	LR.stat	df	Pr(>chisq)
model.fixed	7	101.197	-43.598			
model	8	78.268	-31.134	24.928	1	5.95e-07 ***

Post-hoc tests with emmeans for multiple comparisons of groups

Be sure to read the *Estimated Marginal Means for Multiple Comparisons* chapter for correct interpretation of estimated marginal means. For clmm model objects, the *emmean*, *SE*, *LCL*, and *UCL* values should be ignored, unless specific options in *emmeans* are selected. See *?emmeans::models* for more information.

Because the interaction term in the model was significant, the group separations for the interaction is explored.

```
library(emmeans)

marginal = emmeans(model,
                     ~ Speaker + Time)
pairs(marginal,
      adjust="tukey")

cld(marginal,
     alpha = 0.05,
     Letters = letters,      ### Use lower-case letters for .group
     adjust = "tukey")       ### Tukey-adjusted comparisons

  Speaker  Time    emmean        SE df asympt.LCL asympt.UCL .group
  Piglet   1    -36.08909 0.002735251 NA -36.09629 -36.08190  a
  Piglet   2    -17.88982 0.003252687 NA -17.89838 -17.88127  b
  Eeyore   2     17.46501 2.047095879 NA  12.07902  22.85100  c
  Pooh     1     18.19627 0.002535027 NA  18.18960  18.20294  c
  Eeyore   1     19.30751 2.030625019 NA  13.96486  24.65016  c
  Pooh     2     36.72566 0.002910484 NA  36.71800  36.73332  d
```

Confidence level used: 0.95

Conf-level adjustment: sidak method for 6 estimates
 P value adjustment: tukey method for comparing a family of 6 estimates
 significance level used: alpha = 0.05

Groups sharing a letter in .group are not significantly different

Remember to ignore "emmean", "LCL", "LCL", and "UCL" with CLMM

Focusing on meaningful comparisons

Because we wanted to focus on the comparing each speaker at *Time1* and *Time 2*, let's rearrange the *emmeans cld* table to focus on these comparisons.

With the results in this format, it is easy to see that the scores for both Pooh and Piglet improved from Time 1 to Time 2, but that the scores for poor Eeyore did not.

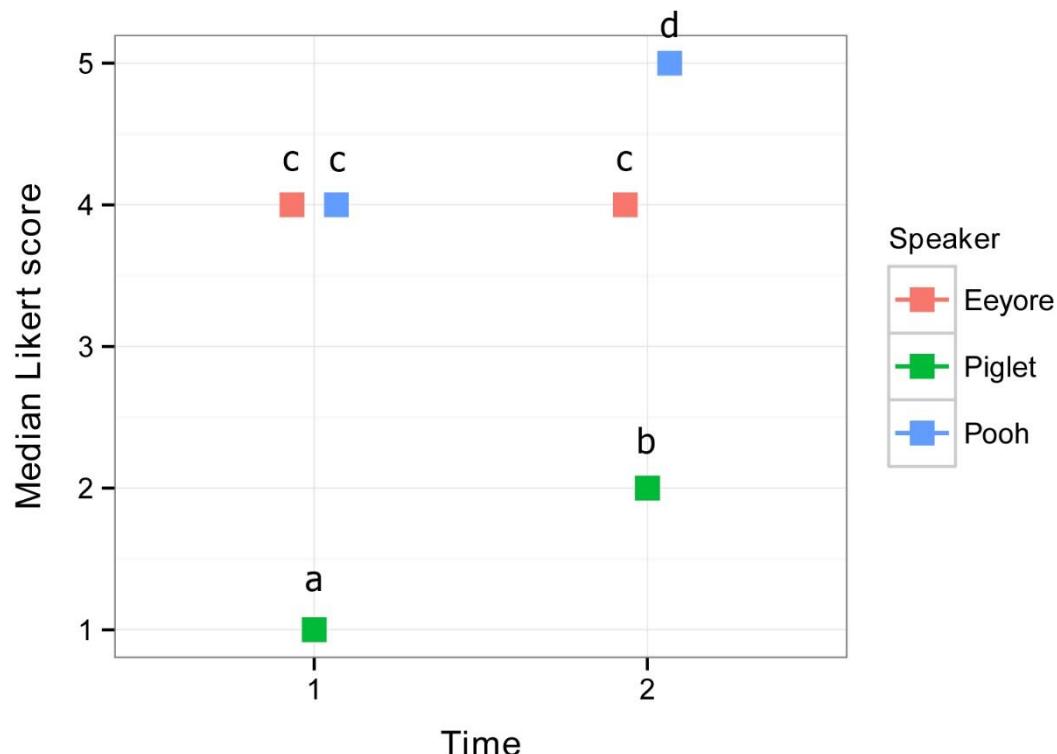
Speaker	Time	emmean	SE	df	asymp.LCL	asymp.UCL	.group
Pooh	1	18.19627	0.002535027	NA	18.18960	18.20294	c
Pooh	2	36.72566	0.002910484	NA	36.71800	36.73332	d

Speaker	Time	emmean	SE	df	asymp.LCL	asymp.UCL	.group
Piglet	1	-36.08909	0.002735251	NA	-36.09629	-36.08190	a
Piglet	2	-17.88982	0.003252687	NA	-17.89838	-17.88127	b

Speaker	Time	emmean	SE	df	asymp.LCL	asymp.UCL	.group
Eeyore	1	19.30751	2.030625019	NA	13.96486	24.65016	c
Eeyore	2	17.46501	2.047095879	NA	12.07902	22.85100	c

Interaction plot with group separation letters

Group separation letters can be added manually to the plot of medians.

**Check model assumptions**

At the time of writing, the *nominal_test* and *scale_test* functions don't work with clmm model objects, so we will define a similar model with *clm* and no random effects, and test the proportional odds assumption on this model.

```
model.clm = clm(Likert.f ~ Time + Speaker + Time:Speaker + Student,
                  data = Data,
                  threshold = "equidistant")
```

Warning message:

(1) Hessian is numerically singular: parameters are not uniquely determined
In addition: Absolute convergence criterion was met, but relative criterion was not met.

```
### The fitting of this model failed. We'll try a model without student.
```

```
model.clm = clm(Likert.f ~ Time + Speaker + Time:Speaker,
                  data = Data,
                  threshold = "equidistant")
```

```
nominal_test(model.clm)
```

Tests of nominal effects

```
formula: Likert.f ~ Time + Speaker + Time:Speaker
      Df logLik   AIC    LRT Pr(>Chi)
<none>     -43.598 101.197
Time        1 -43.149 102.298 0.8983  0.34325
Speaker      2 -40.053  98.106 7.0903  0.02886 *
Time:Speaker 5 -39.393 102.786 8.4110  0.13499
```

There is a violation of proportional odds assumption for this model.
 ### Results from this model fitting may not be reliable.

```
scale_test(model.clm)
```

Warning messages:

1: (-1) Model failed to converge with max|grad| = 2.26258e-05 (tol = 1e-06)
 In addition: iteration limit reached

This test failed. It would be possible to adjust the fitting parameters,
 ### and try to get the model to converge.

Tests for Nominal Data

Introduction to Tests for Nominal Variables

The tests for nominal variables presented in this book are commonly used. They might be used to determine if there is an association between two nominal variables (“association tests”), or if counts of observations for a nominal variable match a theoretical set of proportions for that variable (“goodness-of-fit tests”).

Tests of symmetry, or marginal homogeneity, can determine if frequencies for one nominal variable are greater than that for another, or if there was a change in frequencies from sampling at one time to another. These are described here as “tests for paired nominal data.”

For tests of association, a measure of association, or effect size, should be reported.

When contingency tables include one or more ordinal variables, different tests of association are called for. (See *Association Tests for Ordinal Tables*). Effect sizes are specific for these situations (See *Measures of Association for Ordinal Tables*).

As a more advanced approach, models can be specified with nominal dependent variables. A common type of model with a nominal dependent variable is logistic regression.

Descriptive statistics and plots for nominal data

Descriptive statistics for nominal data are discussed in the “Descriptive statistics for nominal data” section in the *Descriptive Statistics* chapter.

Descriptive plots for nominal data are discussed in the “Examples of basic plots for nominal data” section in the *Basic Plots* chapter.

Contingency tables and matrices

Nominal data are often arranged in a contingency table of counts of observations for each cell of the table. For example, if there were 6 males and 4 females reading Sappho, 3 males and 4 females reading Stephen Crane, and 2 males and 5 females reading Judith Viorst, the data could be arranged as:

	Sex	
	Male	Female
Poet		
Sappho	6	4
Crane	3	4
Viorst	2	5

This data can be read into R in the following manner as a matrix.

```
Input ="
Poet    Male    Female
Sappho  6      4
```

```
Crane      3      4
Viorst     2      5
")
```

```
Matrix = as.matrix(read.table(textConnection(Input),
                               header=TRUE,
                               row.names=1))
```

```
Matrix
```

	Male	Female
Sappho	6	4
Crane	3	4
Viorst	2	5

It is helpful to look at totals for columns and rows.

```
colSums(Matrix)
```

	Male	Female
	11	13

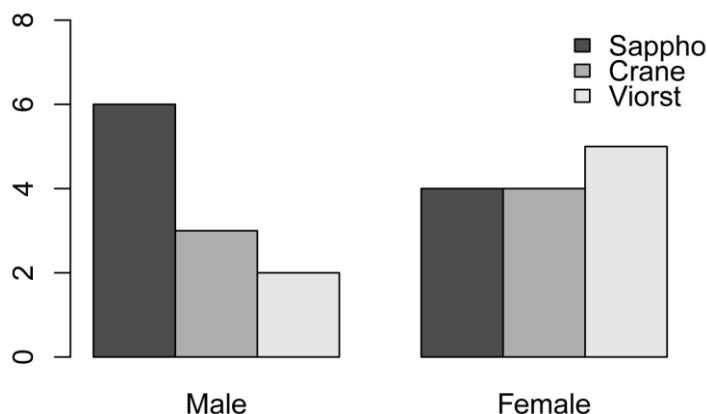
```
rowSums(Matrix)
```

	Sappho	Crane	Viorst
	10	7	7

Bar plots

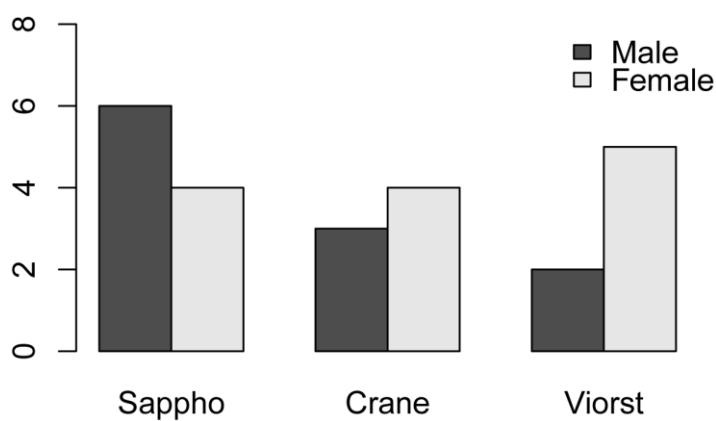
Simple bar charts and mosaic plots are also helpful.

```
barplot(Matrix,
        beside = TRUE,
        legend = TRUE,
        ylim = c(0, 8),    ### y-axis: used to prevent legend overlapping bars
        cex.names = 0.8,   ### Text size for bars
        cex.axis = 0.8,    ### Text size for axis
        args.legend = list(x = "topright",    ### Legend location
                           cex = 0.8,          ### Legend text size
                           bty = "n"))         ### Remove legend box
```



```
Matrix.t = t(Matrix)      ### Transpose Matrix for the next plot

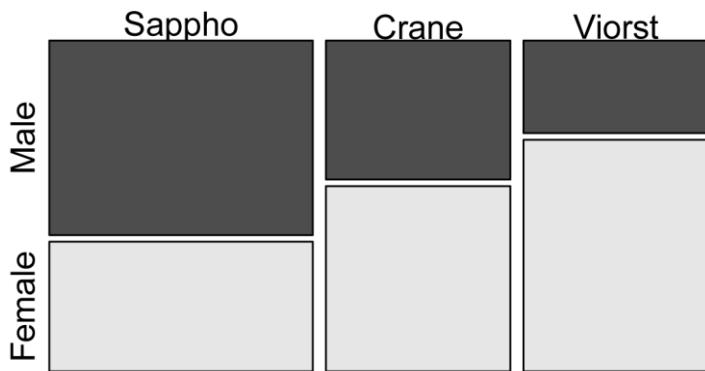
barplot(Matrix.t,
        beside = TRUE,
        legend = TRUE,
        ylim = c(0, 8),    ### y-axis: used to prevent legend overlapping bars
        cex.names = 0.8,   ### Text size for bars
        cex.axis = 0.8,    ### Text size for axis
        args.legend = list(x = "topright",   ### Legend location
                           cex = 0.8,          ### Legend text size
                           bty = "n"))         ### Remove legend box
```



Mosaic plots

Mosaic plots are very useful for visualizing the association between two nominal variables, but can be somewhat tricky to interpret for those unfamiliar with them. Note that the column width is determined by the number of observation in that category. In this case, the Sappho column is wider because more students are reading Sappho than the other two poets. Note, too, that the number of observations in each cell is determined by the area of the cell, not its height. In this case, the Sappho-Female cell and the Crane-Female cell have the same count (4), and so the same area. The Crane-Female cell is taller than the Sappho-Female because it is a higher proportion of observations for that author (4 out of 7 Crane readers compared with 4 out of 10 Sappho readers).

```
mosaicplot(Matrix,
            color=TRUE,
            cex.axis=0.8)
```



Working with proportions

It is often useful to look at proportions of counts within nominal tables.

In this example we may want to look at the proportion of each *Sex* within each *Poet*. That is, the proportions in each row of the first table below sum to 1.

```
Props = prop.table(Matrix, margin = 1)
```

```
Props
```

	Male	Female
Sappho	0.6000000	0.4000000
Crane	0.4285714	0.5714286
Viorst	0.2857143	0.7142857

To plot these proportions, we will first transpose the table.

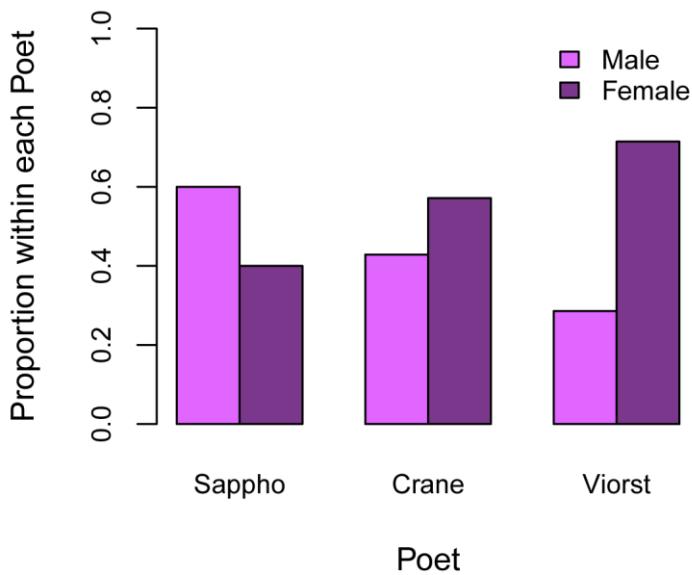
```
Props.t = t(Props)
```

```
Props.t
```

	Sappho	Crane	Viorst
Male	0.6	0.4285714	0.2857143
Female	0.4	0.5714286	0.7142857

```
barplot(Props.t,
        beside = TRUE,
        legend = TRUE,
        ylim = c(0, 1),    ### y-axis: used to prevent legend overlapping bars
        cex.names = 0.8,      ### Text size for bars
```

```
cex.axis = 0.8,      ### Text size for axis
col     = c("mediumorchid1","mediumorchid4"),
ylab     = "Proportion within each Poet",
xlab     = "Poet",
args.legend = list(x = "topright",    ### Legend location
                   cex = 0.8,          ### Legend text size
                   bty = "n"))         ### Remove box
```



Optional analyses: converting among matrices, tables, counts, and cases

In R, most simple analyses for nominal data expect the data to be in a matrix format. However, data may be in a long format, either with each row representing a single observation (*cases*), or with each row containing a count of observations (*counts*).

It is relatively easy to convert among these different forms of data.

Long-format with each row as an observation (cases)

```
Input ="
Poet   Sex
Sappho Male
Sappho Female
Sappho Female
Sappho Female
Sappho Female
Crane  Male
Crane  Male
Crane  Male
```

```

Crane   Female
Crane   Female
Crane   Female
Crane   Female
Viorst  Male
Viorst  Male
Viorst  Female
Viorst  Female
Viorst  Female
Viorst  Female
Viorst  Female
Viorst  Female
")  
  

Data = read.table(textConnection(Input),header=TRUE)  
  

### Order factors by the order in data frame
### Otherwise, xtabs will alphabetize them  
  

Data$Poet = factor(Data$Poet,
                     levels=unique(Data$Poet))  
  

Data$Sex = factor(Data$Sex,
                   levels=unique(Data$Sex))

```

Cases to table

```
Table = xtabs(~ Poet + Sex,
              data=Data)
```

Table

Poet	Sex	
	Male	Female
Sappho	6	4
Crane	3	4
Viorst	2	5

Cases to Counts

```
Table = xtabs(~ Poet + Sex,
              data=Data)
```

```
Counts = as.data.frame(Table)
```

Counts

Poet	Sex	Freq
1 Sappho	Male	6
2 Crane	Male	3
3 Viorst	Male	2
4 Sappho	Female	4
5 Crane	Female	4

```
6 Viorst Female      5
```

Long-format with counts of observations (counts)

```
Input ="
Poet      Sex      Freq
Sappho    Male     6
Sappho    Female   4
Crane     Male     3
Crane     Female   4
Viorst    Male     2
Viorst    Female   5
")

Counts = read.table(textConnection(Input), header=TRUE)

### Order factors by the order in data frame
### Otherwise, xtabs will alphabetize them

Counts$Poet = factor(Counts$Poet,
                      levels=unique(Counts$Poet))

Counts$Sex = factor(Counts$Sex,
                     levels=unique(Counts$Sex))
```

Counts to Table

```
Table = xtabs(Freq ~ Poet + Sex,
               data=Counts)
```

Table

Sex		
Poet	Male	Female
Sappho	6	4
Crane	3	4
Viorst	2	5

Counts to Cases

```
Long = Counts[rep(row.names(Counts), Counts$Freq), c("Poet", "Sex")]
```

```
rownames(Long) = seq(1:nrow(Long))
```

Long

	Poet	Sex
1	Sappho	Male
2	Sappho	Male
3	Sappho	Male
4	Sappho	Male

```

5 Sappho Male
6 Sappho Male
7 Sappho Female
8 Sappho Female
9 Sappho Female
10 Sappho Female
11 Crane Male
12 Crane Male
13 Crane Male
14 Crane Female
15 Crane Female
16 Crane Female
17 Crane Female
18 Viorst Male
19 Viorst Male
20 Viorst Female
21 Viorst Female
22 Viorst Female
23 Viorst Female
24 Viorst Female

```

Matrix form

```

Input ="
Poet      Male      Female
Sappho    6          4
Crane     3          4
Viorst    2          5
")

Matrix = as.matrix(read.table(textConnection(Input),
                               header=TRUE,
                               row.names=1))

Matrix

      Male Female
Sappho   6     4
Crane    3     4
Viorst   2     5

```

Matrix to table

```

Table = as.table(Matrix)

Table

      Male Female
Sappho   6     4
Crane    3     4
Viorst   2     5

```

Matrix to counts

```
Table = as.table(Matrix)
Counts = as.data.frame(Table)
colnames(Counts) = c("Poet", "Sex", "Freq")
Counts
```

	Poet	Sex	Freq
1	Sappho	Male	6
2	Crane	Male	3
3	Viorst	Male	2
4	Sappho	Female	4
5	Crane	Female	4
6	Viorst	Female	5

Matrix to Cases

```
Table = as.table(Matrix)
Counts = as.data.frame(Table)
colnames(Counts) = c("Poet", "Sex", "Freq")
Long = Counts[rep(row.names(Counts), Counts$Freq), c("Poet", "Sex")]
rownames(Long) = seq(1:nrow(Long))
Long
```

	Poet	Sex
1	Sappho	Male
2	Sappho	Male
3	Sappho	Male
4	Sappho	Male
5	Sappho	Male
6	Sappho	Male
7	Crane	Male
8	Crane	Male
9	Crane	Male
10	Viorst	Male
11	Viorst	Male
12	Sappho	Female
13	Sappho	Female
14	Sappho	Female
15	Sappho	Female
16	Crane	Female
17	Crane	Female
18	Crane	Female

```
19 Crane Female
20 Viorst Female
21 Viorst Female
22 Viorst Female
23 Viorst Female
24 Viorst Female
```

Table to matrix

```
Matrix = as.matrix(Table)
```

```
Matrix
```

	Male	Female
Sappho	6	4
Crane	3	4
Viorst	2	5

Optional analyses: obtaining information about a matrix object

```
Matrix
```

	Male	Female
Sappho	6	4
Crane	3	4
Viorst	2	5

```
class(Matrix)
```

```
[1] "matrix"
```

```
typeof(Matrix)
```

```
[1] "integer"
```

```
attributes(Matrix)
```

```
$dim
[1] 3 2
```

```
$dimnames
```

```
$dimnames[[1]]
[1] "Sappho" "Crane"  "Viorst"
```

```
$dimnames[[2]]
[1] "Male"   "Female"
```

```

str(Matrix)

int [1:3, 1:2] 6 3 2 4 4 5
- attr(*, "dimnames")=List of 2
..$ : chr [1:3] "Sappho" "Crane" "Viorst"
..$ : chr [1:2] "Male" "Female"

colnames(Matrix)

[1] "Male"    "Female"

rownames(Matrix)

[1] "Sappho"  "Crane"   "Viorst"

```

References

Replicate each row of data.frame and specify the number of replications for each row. Stack Overflow. 2011. stackoverflow.com/questions/2894775/replicate-each-row-of-data-frame-and-specify-the-number-of-replications-for-each.

Confidence Intervals for Proportions

A binomial proportion has counts for two levels of a nominal variable. An example would be counts of students of only two sexes, male and female. If there are 20 students in a class, and 12 are female, then the proportion of females are 12/20, or 0.6, and the proportion of males are 8/20 or 0.4. This is a binomial proportion.

Sex	Count	Proportion
Female	12	0.60
Male	8	0.40
<hr/>		
Total	20	1.00

A multinomial proportion has counts for more than two levels of a nominal variable. For example, we might have the following levels and counts for sex for students in a class:

Sex	Count	Proportion
Female	12	0.60
Male	6	0.30
Other	1	0.05
Prefer not to answer	1	0.05
<hr/>		
Total	20	1.00

Confidence intervals can be produced for either binomial or multinomial proportions.

The *binom.test* function in the native *stats* package will provide the Clopper-Pearson confidence interval for a binomial proportion. Other methods for a binomial proportion are provided by the *BinomCI* function in the *DescTools* package, as well as by various functions in the *PropCIs* package.

Confidence intervals for multinomial proportions can be produced with the *MultinomCI* function in the *DescTools* package.

Packages used in this chapter

The packages used in this chapter include:

- DescTools
- PropCIs

The following commands will install these packages if they are not already installed:

```
if(!require(DescTools)){install.packages("DescTools")}
if(!require(PropCIs)){install.packages("PropCIs")}
```

Example of confidence intervals for a binomial proportion

As part of a demographic survey of her scrapbooking 4-H course, Seras Victoria asks students if they have ever done scrapbooking before. The following are the data from her course:

Experience	Count
Yes	7
No	14
-----	--
Total	21

Note that when calculating confidence intervals for a binomial variable, one level of the nominal variable is chosen to be the “success” level. This is an arbitrary decision, but you should be cautious to remember that the confidence interval is reported for the proportion of “success” responses. The *BinomCI* function in the *DescTools* package can produce the confidence interval for both “success” and “failure” in one step.

The *binom.test* function output includes a confidence interval for the proportion, and the proportion of “success” as a decimal number. The *binom.test* function uses the Clopper-Pearson method for confidence intervals.

```
binom.test(7, 21,
          0.5,
          alternative="two.sided",
          conf.level=0.95)
```

95 percent confidence interval:

```
0.1458769 0.5696755
```

```
sample estimates:
probability of success
0.3333333
```

The *BinomCI* function in the *DescTools* package has several methods for calculating confidence intervals for a binomial proportion.

```
library(DescTools)

BinomCI(7, 21,
        conf.level = 0.95,
        method = "clopper-peerson")

#### Methods: "wilson", "wald", "agresti-coull", "jeffreys",
#### "modified wilson", "modified jeffreys",
#### "clopper-peerson", "arcsine", "logit", "witting", "pratt"

      est    lwr.ci    upr.ci
[1,] 0.3333333 0.1458769 0.5696755
```

The *BinomCI* function in the *DescTools* package can also produce the confidence intervals for “success” and “failure” in one step.

```
library(DescTools)

observed = c(7, 14)

total = sum(observed)

BinomCI(observed, total,
        conf.level = 0.95,
        method = "clopper-peerson")

#### Methods: "wilson", "wald", "agresti-coull", "jeffreys",
#### "modified wilson", "modified jeffreys",
#### "clopper-peerson", "arcsine", "logit", "witting", "pratt"

      est    lwr.ci    upr.ci
[1,] 0.3333333 0.1458769 0.5696755
[2,] 0.6666667 0.4303245 0.8541231
```

The *PropCIs* package has functions for calculating confidence intervals for a binomial proportion.

The *exactci* function uses the Clopper–Pearson exact method.

7/21

```
[1] 0.3333333
```

```
library(PropCIs)

exactci(7, 21,
        conf.level=0.95)

95 percent confidence interval:
0.1458769 0.5696755
```

The *blakerci* function uses the Blaker exact method.

```
7/21

[1] 0.3333333
```

```
library(PropCIs)

blakerci(7, 21,
          conf.level=0.95)

95 percent confidence interval:
0.1523669 0.5510455
```

Example of confidence intervals for a multinomial proportion

As part of a demographic survey of her scrapbooking 4-H course, Seras Victoria asks students to report their sex. The following are the data from her course:

Sex	Count
Female	10
Male	9
Other	1
No answer	1
-----	-----
Total	21

```
library(DescTools)

observed = c(10, 9, 1, 1)

MultinomCI(observed,
            conf.level=0.95,
            method="sisonglaz")

### Methods: "sisonglaz", "cplus1", "goodman"

           est      lwr.ci      upr.ci
[1,] 0.47619048 0.2857143 0.7009460
[2,] 0.42857143 0.2380952 0.6533270
```

```
[3,] 0.04761905 0.0000000 0.2723746
[4,] 0.04761905 0.0000000 0.2723746
```

Optional analysis: confidence intervals for a difference in proportions

As part of a demographic survey of their scrapbooking 4-H courses, Seras Victoria and Integra Hellsing ask students if they have experience in scrapbooking. They want to determine the difference of proportions of students having experience in each class, and calculate a confidence interval for that difference. The following are the data:

Seras Victoria		Integra Hellsing	
Experience	Count	Experience	Count
Yes	7	Yes	13
No	14	No	4
-----	-----	-----	-----
Total	21	Total	17

Two functions in the *PropCIs* package can determine a confidence interval for a difference for independent proportions.

7/21

```
[1] 0.3333333
```

13/17

```
[1] 0.7647059
```

(7/21) - (13/17)

```
[1] -0.4313725
```

```
library(PropCIs)
```

```
diffscoreci(7, 21, 13, 17,
conf.level=0.95)
```

```
95 percent confidence interval:
-0.6685985 -0.1103804
```

```
wald2ci(7, 21, 13, 17,
conf.level=0.95,
adjust = "Wald")
```

```
### adjust = "AC", "Wald"
```

95 percent confidence interval:
-0.7165199 -0.1462252

sample estimates:
[1] -0.4313725

References

"Confidence Limits" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/c_04.html.

Goodness-of-Fit Tests for Nominal Variables

Goodness-of-fit tests are used to compare proportions of levels of a nominal variable to theoretical proportions. Common goodness-of-fit tests are G-test, chi-square, and binomial or multinomial exact tests.

In general, there are no assumptions about the distribution of data for these tests. However, the results of chi-square tests and G-tests can be inaccurate if statistically expected cell counts are low. A rule of thumb is that all statistically expected cell counts should be 5 or greater for chi-square- and G-tests. For a more complete discussion, see McDonald in the "Optional Readings" section for details on what constitutes low cell counts.

One approach is to use exact tests, which are not bothered by low cell counts. However, if there are not low cell counts, using G-test or chi-square test is fine. The advantage of chi-square tests is that your audience may be more familiar with them.

G-tests are also called likelihood ratio tests, and "Likelihood Ratio Chi-Square" by SAS.

Appropriate data

- A nominal variable with two or more levels
- Theoretical, typical, or neutral values for the proportions for this variable are needed for comparison
- G-test and chi-square tests may not be appropriate if there are cells with low expected counts

Hypotheses

- Null hypothesis: The proportions for the levels for the nominal variable are not different from the theoretical proportions.
- Alternative hypothesis (two-sided): The proportions for the levels for the nominal variable are different from the theoretical proportions.

Interpretation

Significant results can be reported as “The proportions for the levels for the nominal variable were statistically different from the theoretical proportions.”

Packages used in this chapter

The packages used in this chapter include:

- EMT
- DescTools
- ggplot2
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(EMT)){install.packages("EMT")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Goodness-of-fit tests for nominal variables example

As part of a demographic survey of students in his environmental issues webinar series, Alucard recorded the race and ethnicity of his students. He wants to compare the data for his class to the demographic data for Cumberland County, New Jersey as a whole

Race	Alucard's_class	County_proportion
White	20	0.775
Black	9	0.132
American Indian	9	0.012
Asian	1	0.054
Pacific Islander	1	0.002
Two or more races	1	0.025
<hr/>		<hr/>
Total	41	1.000

Ethnicity	Alucard's_class	County_proportion
Hispanic	7	0.174
Not Hispanic	34	0.826
<hr/>		<hr/>
Total	41	1.000

Exact tests for goodness-of-fit

Race data

```
observed = c(20, 9, 9, 1, 1, 1)
theoretical = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

library(EMT)
```

```

multinomial.test(observed, theoretical)

### This can take a long time!

Exact Multinomial Test, distance measure: p

Events      pobs      p.value
1370754        0          0

### A faster, approximate test by Monte Carlo simulation

observed = c(20, 9, 9, 1, 1, 1)
theoretical = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

library(EMT)

multinomial.test(observed, theoretical,
MonteCarlo = TRUE)

Exact Multinomial Test, distance measure: p

Events      pobs      p.value
1370754        0          0

```

Ethnicity data

```

x = 7
n = 41
theoretical = 0.174

binom.test(x, n, theoretical)

Exact binomial test

number of successes = 7, number of trials = 41, p-value = 1

```

G-test for goodness-of-fit

Race data

```

observed = c(20, 9, 9, 1, 1, 1)
theoretical = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

library(DescTools)

GTest(x = observed,
p = theoretical,
correct="none")

```

```
### Correct: "none" "williams" "yates"

Log likelihood ratio (G-test) goodness of fit test

data: observed
G = 46.317, X-squared df = 5, p-value = 7.827e-09
```

Check expected counts

```
Test = GTest(x = observed,
             p = theoretical,
             correct="none")

Test$expected
[1] 31.775 5.412 0.492 2.214 0.082 1.025

### Note the low expected counts: 0.492, 0.082, and 1.025,
### suggesting the test may not be valid.
```

Ethnicity data

```
observed      = c(7, 34)
theoretical   = c(0.174, 0.826)

library(DescTools)

GTest(x = observed,
      p = theoretical,
      correct = "none")

### Correct: "none" "williams" "yates"

Log likelihood ratio (G-test) goodness of fit test

G = 0.0030624, X-squared df = 1, p-value = 0.9559
```

Check expected counts

```
Test = GTest(x = observed,
             p = theoretical,
             correct = "none")

Test$expected
[1] 7.134 33.866

### There are no low expected counts.
```

Chi-square test for goodness-of-fitRace data

```
observed      = c(20, 9, 9, 1, 1, 1)
theoretical = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

chisq.test(x = observed,
            p = theoretical)

Chi-squared test for given probabilities

X-squared = 164.81, df = 5, p-value < 2.2e-16
```

Check expected counts

```
Test = chisq.test(x = observed,
                    p = theoretical)

Test$expected

[1] 31.775 5.412 0.492 2.214 0.082 1.025

### Note the low expected counts: 0.492, 0.082, and 1.025,
### suggesting the test may not be valid.
```

Ethnicity data

```
observed      = c(7, 34)
theoretical = c(0.174, 0.826)

chisq.test(x = observed,
            p = theoretical)

Chi-squared test for given probabilities

X-squared = 0.0030472, df = 1, p-value = 0.956
```

Check expected counts

```
Test = chisq.test(x = observed,
                    p = theoretical)

Test$expected

[1] 7.134 33.866

### There are no low expected counts.
```

Effect size for goodness-of-fit tests

Cramér's V , a measure of association used for 2-dimensional contingency tables, can be modified for use in goodness-of-fit tests for nominal variables. In this context, a value of Cramér's V of 0 indicates that observed values match theoretical values perfectly. If theoretical proportions are equally distributed, the maximum for value for Cramér's V is 1. However, if theoretical proportions vary among categories, Cramér's V can exceed 1.

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment, but for behavioral studies, the guidelines proposed by Cohen (1988) are sometimes followed. The following guidelines are based on the suggestions of Cohen ($2 \leq k \leq 6$) and an extension of these values ($7 \leq k \leq 10$). They should not be considered universal.

The following table is appropriate for *chi-square* goodness-of-fit tests, and similar tests, ***where the theoretical proportions are equally distributed among categories.*** k is the number of categories.

	<u>Small</u>	<u>Medium</u>	<u>Large</u>
$k = 2$	0.100 – < 0.300	0.300 – < 0.500	≥ 0.500
$k = 3$	0.071 – < 0.212	0.212 – < 0.354	≥ 0.354
$k = 4$	0.058 – < 0.173	0.173 – < 0.289	≥ 0.289
$k = 5$	0.050 – < 0.150	0.150 – < 0.250	≥ 0.250
$k = 6$	0.045 – < 0.134	0.134 – < 0.224	≥ 0.224
$k = 7$	0.043 – < 0.130	0.130 – < 0.217	≥ 0.217
$k = 8$	0.042 – < 0.127	0.127 – < 0.212	≥ 0.212
$k = 9$	0.042 – < 0.125	0.125 – < 0.209	≥ 0.209
$k = 10$	0.041 – < 0.124	0.124 – < 0.207	≥ 0.207

Examples of effect size for goodness-of-fit tests

Since the theoretical proportions in these example aren't equally distributed among categories, these values of Cramér's V may be difficult to interpret.

Race data

```
observed    = c(20, 9, 9, 1, 1, 1)
theoretical = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

library(rcompanion)

cramerVFit(x = observed,
            p = theoretical)

Cramer V
0.8966
```

Ethnicity data

```

observed      = c(7, 34)
theoretical   = c(0.174, 0.826)

cramerVFit (x = observed,
             p = theoretical)

Cramer V
0.008621

```

Post-hoc analysis

If the goodness of fit test is significant, a post-hoc analysis can be conducted to determine which counts differ from their theoretical proportions.

Standardized residuals

One approach is to look at the standardized residuals from the chi-square analysis. Cells with a standardized residual whose absolute value is greater than 1.96 indicate a cell differing from theoretical proportions. (The 1.96 cutoff is analogous to *alpha* = 0.05 for a hypothesis test, or 2.58 for *alpha* = 0.01.)

Race data

Here, cells with standardized residuals whose absolute value is greater than 1.96 are White, American Indian, and Pacific Islander, suggesting the counts in these cells differ from theoretical proportions.

```

observed      = c(20, 9, 9, 1, 1, 1)
theoretical   = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

chisq.test(x = observed, p = theoretical)$stdres

[1] -4.40379280  1.65544089 12.20299568 -0.83885000  3.20900567 -0.02500782

###  white        Black       AI        Asian       PI        Two+

```

Ethnicity data

Here, no cells have standardized residuals with absolute value is greater than 1.96.

```

observed      = c(7, 34)
theoretical   = c(0.174, 0.826)

chisq.test(x = observed, p = theoretical)$stdres

[1] -0.05520116  0.05520116

###  Hispanic    Not Hispanic

```

Multinomial confidence intervals

Race data

It may be a reasonable approach to examine the confidence intervals for multinomial proportions. Categories whose confidence intervals do not contain the theoretical proportion are identified as different from the theoretical. This approach may make more sense when theoretical proportions are equally distributed across categories.

Here, the confidence interval for [1,] (White) doesn't contain 0.775 and [3,] (American Indian) doesn't contain 0.012. So these categories are likely different than the theoretical proportions.

```
observed      = c(20, 9, 9, 1, 1, 1)
theoretical   = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

library(DescTools)

MultinomCI(observed,
            conf.level=0.95,
            method="sisonglaz")

      est      lwr.ci      upr.ci
[1,] 0.48780488 0.34146341 0.6438374
[2,] 0.21951220 0.07317073 0.3755448
[3,] 0.21951220 0.07317073 0.3755448
[4,] 0.02439024 0.00000000 0.1804228
[5,] 0.02439024 0.00000000 0.1804228
[6,] 0.02439024 0.00000000 0.1804228
```

Ethnicity data

Here, the confidence interval for [1,] (Hispanic) contains 0.174 and [2,] (Not Hispanic) contains 0.826. So no categories are likely different than the theoretical proportions.

```
observed      = c(7, 34)
theoretical   = c(0.174, 0.826)

library(DescTools)

MultinomCI(observed,
            conf.level=0.95,
            method="sisonglaz")

      est      lwr.ci      upr.ci
[1,] 0.1707317 0.07317073 0.2802072
[2,] 0.8292683 0.73170732 0.9387438
```

Simple bar plots

Race example

In order to facilitate plotting with R base graphics, the vector of counts is converted to proportions, and the theoretical and observed proportions are combined into a table, named *XT*.

See the *Introduction to Tests for Nominal Variables* chapter for a few additional options for how bar plots are plotted.

```

observed      = c(20, 9, 9, 1, 1, 1)
theoretical   = c(0.775, 0.132, 0.012, 0.054, 0.002, 0.025)

Observed.prop = observed / sum(observed)
Theoretical.prop = theoretical

Observed.prop = round(Observed.prop, 3)
Theoretical.prop = round(Theoretical.prop, 3)

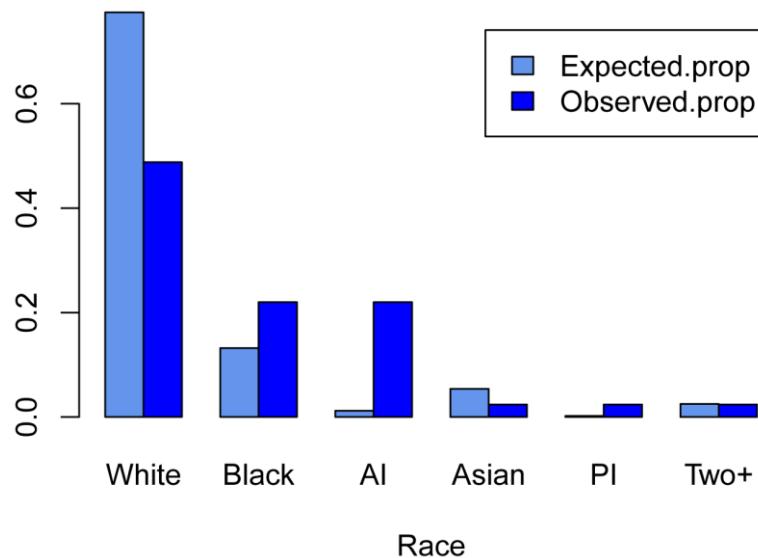
XT = rbind(Theoretical.prop, Observed.prop)

colnames(XT) = c("White", "Black", "AI", "Asian",
                 "PI", "Two+")

XT
      White Black     AI Asian      PI Two+
Theoretical.prop 0.775 0.132 0.012 0.054 0.002 0.025
Observed.prop    0.488 0.220 0.220 0.024 0.024 0.024
  
```

```

barplot(XT,
        beside = T,
        xlab   = "Race",
        col    = c("cornflowerblue", "blue"),
        legend = rownames(XT))
  
```



In order to plot with *ggplot2*, the table *XT* will be converted into a data frame in long format.

```
Race = c(colnames(XT), colnames(XT))

Race = factor(Race,
              levels=unique(colnames(XT)))

Proportion = c(rep("Theoretical", 6),
               rep("Observed", 6))

value = c(XT[1,], XT[2,])

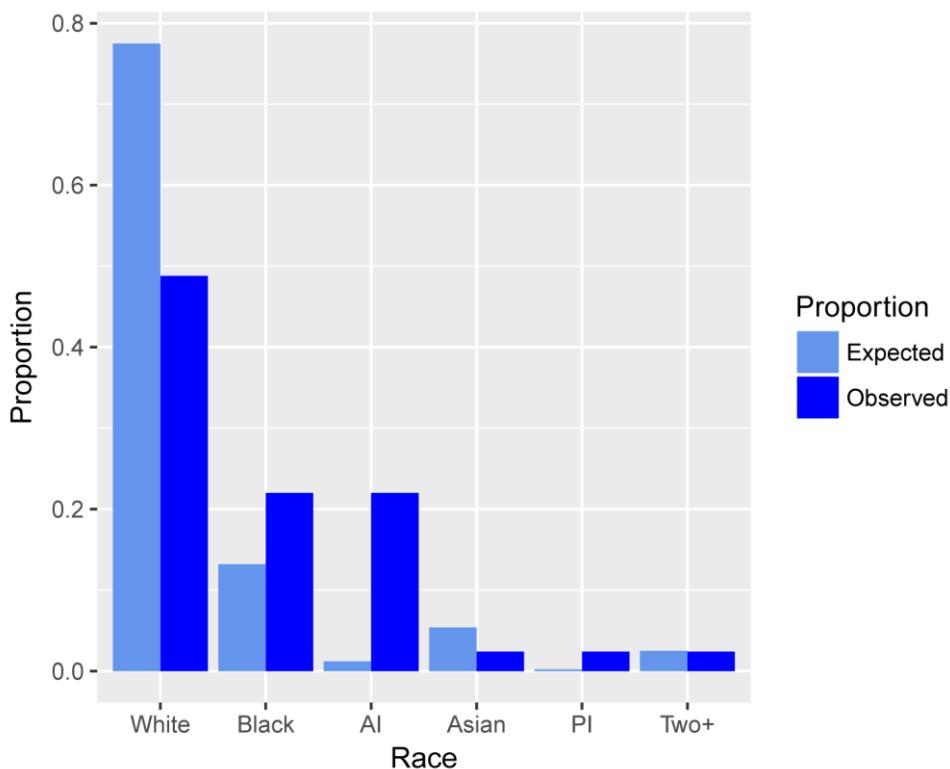
DF = data.frame(Race, Proportion, value)

DF

      Race   Proportion value
1  White Theoretical 0.775
2  Black Theoretical 0.132
3     AI Theoretical 0.012
4 Asian Theoretical 0.054
5    PI Theoretical 0.002
6 Two+ Theoretical 0.025
7  White Observed 0.488
8  Black Observed 0.220
9     AI Observed 0.220
10 Asian Observed 0.024
11    PI Observed 0.024
12 Two+ Observed 0.024

library(ggplot2)

ggplot(DF,
       aes(x = Race, y = value, fill = Proportion)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  scale_fill_manual(values = c("cornflowerblue", "blue")) +
  ylab("Proportion")
```



Ethnicity example

```

observed      = c(7, 34)
theoretical   = c(0.174, 0.826)

Observed.prop    = observed / sum(observed)
Theoretical.prop = theoretical

Observed.prop    = round(Observed.prop, 3)
Theoretical.prop = round(Theoretical.prop, 3)

XT = rbind(Theoretical.prop, Observed.prop)

colnames(XT) = c("Hispanic", "Not Hispanic")

XT

```

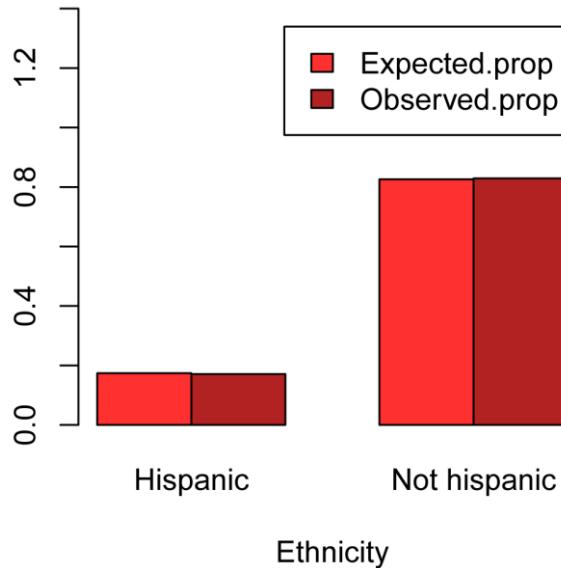
	Hispanic	Not Hispanic
Theoretical.prop	0.174	0.826
Observed.prop	0.171	0.829

```

barplot(XT,
        beside = T,
        xlab   = "Ethnicity",
        col    = c("firebrick1", "firebrick"),

```

```
legend = rownames(XT),
ylim   = c(0, 1.4))
```



```
Ethnicity = c(colnames(XT), colnames(XT))
```

```
Ethnicity = factor(Ethnicity,
levels=unique(colnames(XT)))
```

```
Proportion = c(rep("Theoretical", 2), rep("Observed", 2))
```

```
value = c(XT[1,], XT[2,])
```

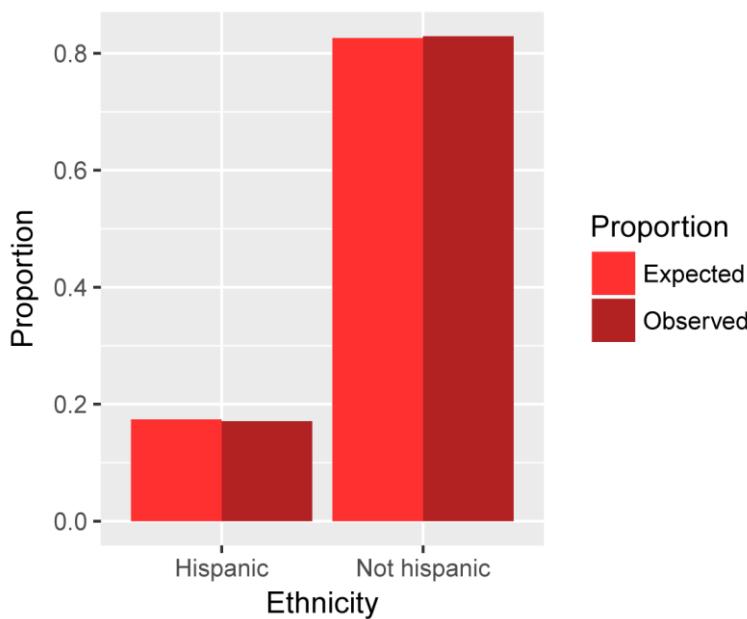
```
DF = data.frame(Ethnicity, Proportion, value)
```

```
DF
```

	Ethnicity	Proportion	value
1	Hispanic	Theoretical	0.174
2	Not Hispanic	Theoretical	0.826
3	Hispanic	Observed	0.171
4	Not Hispanic	Observed	0.829

```
library(ggplot2)
```

```
ggplot(DF,
aes(x = Ethnicity, y = Value, fill = Proportion)) +
geom_bar(stat = 'identity', position = 'dodge') +
scale_fill_manual(values = c("firebrick1","firebrick")) +
ylab("Proportion")
```



Optional: Multinomial test example with plot and confidence intervals

This is an example of a multinomial test that includes a bar plot showing confidence intervals. The data is a simple vector of counts.

For a similar example using two-way count data which is organized into a data frame, see the "Examples of basic plots for nominal data" section in the *Basic Plots* chapter.

Walking to the store, Jerry Coyne observed the colors of nail polish on women's toes (Coyne, 2016). Presumably because that's the kind of thing retired professors are apt to do. He concluded that red was a more popular color but didn't do any statistical analysis to support his conclusion.

<u>color of polish</u>	<u>Count</u>
Red	19
None or clear	3
White	1
Green	1
Purple	2
Blue	2

We will use a multinomial goodness of fit test to determine if there is an overall difference in the proportion of colors (*multinom.test* function in the *EMT* package). The confidence intervals for each proportion can be found with the *MultinomCI* function in the *DescTools* package. The data then needs to be manipulated some so that we can plot the data as counts and not proportions.

Note here that the theoretical counts are simply 1 divided by the number of treatments. In this case the null hypothesis is that the observed proportions are all the same. In the examples above, the null

hypothesis was that the observed proportions were not different than theoretical proportions. It's two ways to think of the same null hypothesis.

The confidence intervals for each proportion can be used as a post-hoc test, to determine which proportions differ from each other, or to determine which proportions differ from θ .

```
nail.color = c("Red", "None", "white", "Green", "Purple", "Blue")
observed   = c( 19,    3,    1,    1,    2,    2 )
theoretical = c( 1/6,   1/6,   1/6,   1/6,   1/6,   1/6 )
```

```
library(EMT)

multinomial.test(observed,
                  theoretical)

### This may take a while.  Use Monte Carlo for large numbers.

Exact Multinomial Test, distance measure: p

Events      pobs      p.value
237336          0            0
```

```
library(rcompanion)

cramerVFit(observed)

### Assumes equal proportions for theoretical values.

Cramer V
0.6178
```

```
library(DescTools)

MCI = MultinomCI(observed,
                  conf.level=0.95,
                  method="sisonglaz")

MCI

      est      lwr.ci      upr.ci
[1,] 0.67857143 0.5357143 0.8423162
[2,] 0.10714286 0.0000000 0.2708876
[3,] 0.03571429 0.0000000 0.1994590
[4,] 0.03571429 0.0000000 0.1994590
[5,] 0.07142857 0.0000000 0.2351733
[6,] 0.07142857 0.0000000 0.2351733
```

```
### Order the levels, otherwise R will alphabetize them

Nail.color = factor(nail.color,
```

```

  levels=unique(nail.color))

### For plot, Create variables of counts, and then wrap them into a data frame

Total = sum(observed)
Count = observed
Lower = MCI[, 'lwr.ci'] * Total
Upper = MCI[, 'upr.ci'] * Total

Data = data.frame(Count, Lower, Upper)

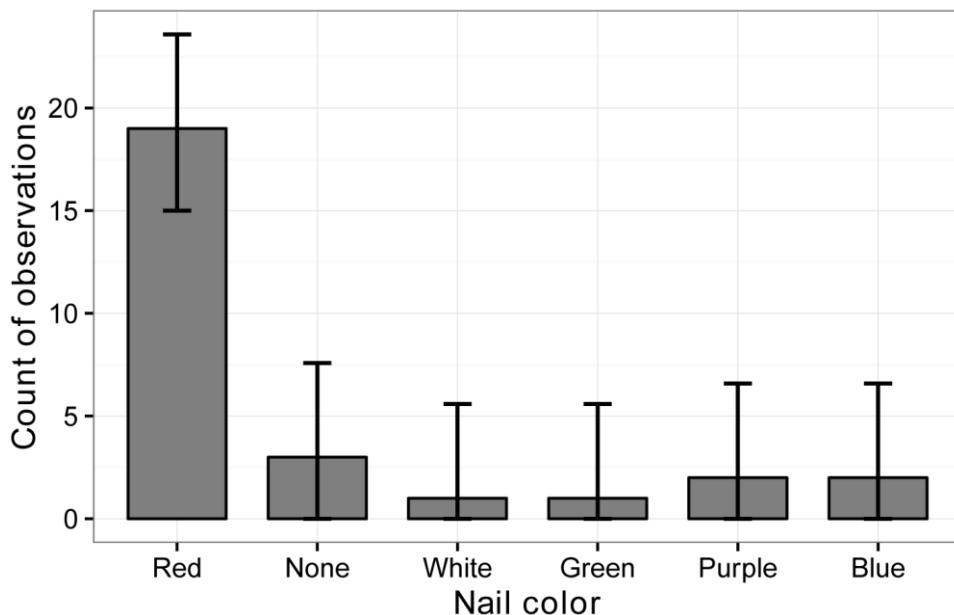
Data

  Count Lower     Upper
1    19    15 23.584853
2     3     0  7.584853
3     1     0  5.584853
4     1     0  5.584853
5     2     0  6.584853
6     2     0  6.584853

library(ggplot2)

ggplot(Data,           ### The data frame to use.
       aes(x      = Nail.color,
            y      = Count)) +
  geom_bar(stat = "identity",
            color = "black",
            fill  = "gray50",
            width = 0.7) +
  geom_errorbar(aes(ymin = Lower,
                    ymax  = Upper),
                width = 0.2,
                size  = 0.7,
                position = pd,
                color = "black"
                ) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Count of observations") +
  xlab("Nail color")

```



Bar plot of the count of the color of women's toenail polish observed by Jerry Coyne while walking to the store. Error bars indicate 95% confidence intervals (Sison and Glaz method).

Optional readings

"Small numbers in chi-square and G-tests" in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/small.html.

References

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.

Coyne, J. A. 2016. Why is red nail polish so popular? *Why Evolution is True*. whyevolutionisttrue.wordpress.com/2016/07/02/why-is-red-nail-polish-so-popular/.

"Chi-square Test of Goodness-of-Fit" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_03.html.

"Exact Test of Goodness-of-Fit" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_01.html.

"G-test of Goodness-of-Fit" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_04.html.

"Repeated G-tests of Goodness-of-Fit" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_09.html.

Exercises Beta

1. What do you conclude about whether *the race of* Alucard's students match the proportions in the county? Include, as appropriate:

- The statistical test you are using, *p*-value from that test, conclusion of the test, and any other comments on the test.
- The effect size and interpretation, or perhaps why you don't want to rely on that information.
- Description of the observed counts or proportions relative to the theoretical proportions. Is there anything notable?
- Any other practical conclusions you wish to note.

2. What do you conclude about whether *the ethnicity of* Alucard's students match the proportions in the county? Include, as appropriate:

- The statistical test you are using, *p*-value from that test, conclusion of the test, and any other comments on the test.
- The effect size and interpretation, or perhaps why you don't want to rely on that information.
- Description of the observed counts or proportions relative to the theoretical proportions. Is there anything notable?
- Any other practical conclusions you wish to note.

Association Tests for Nominal Variables

These tests for nominal variables are used to determine if two nominal variables are associated. Sometimes the term “independent” is used to mean that there is no association.

In general, there are no assumptions about the distribution of data for these tests.

Note that for these tests of association there shouldn't be paired values. For example, if experimental units—the things you are counting—are “students before” and “students after”, or “left hands” and “right hands”, the tests in the chapter *Tests for Paired Nominal Data* may be more appropriate.

Also note that these tests will not be accurate if there are “structural zeros” in the contingency table. If you were counting pregnant and non-pregnant individuals across categories like male and female, the male–pregnant cell may contain a structural zero if you assume your population cannot have pregnant males.

Low cell counts

The results of chi-square tests and G-tests can be inaccurate if expected cell counts are low. A rule of thumb is that all expected counts should be 5 or greater for chi-square- and G-tests. For a more complete discussion, see McDonald in the “Optional Readings” section on what constitutes low cell counts.

For tables larger than 2×2 , a rule of thumb is that all expected counts are 1 or greater and that no more than 20% of expected counts are less than 5.

One approach when there are low expected counts is to use exact tests, such as Fisher’s exact test, which are not bothered by low cell counts.

Another approach is to apply a continuity correction to the test. The *chisq.test* function automatically applies the Yates’s continuity correction for 2×2 tables. The *GTest* function has options for Yates or Williams corrections.

Fisher’s exact test

Technically, Fisher’s exact test assumes that the table has fixed margins, in at least in one dimension. As one example, if the table has counts of males and females, the test could assume that the number of males tested and the number of females tested was determined ahead of time.

It appears to me that this assumption is commonly ignored, but I don’t know if this is because of ignorance of this assumption, or just that some people assume that advantages of the exact test outweigh any violation of the assumption.

Choosing among tests

If there are not low expected counts, using G-test or chi-square test is fine. The advantage of chi-square tests is that your audience may be more familiar with them. That being said, some authors recommend the using Fisher’s exact test routinely unless the number of observations is so great that the analysis takes too long.

Appropriate data

- Two nominal variables with two or more levels each.
- Experimental units aren’t paired.
- There are no structural zeros in the contingency table.
- G-test and chi-square test may not be appropriate if there are cells with low expected counts.

Hypotheses

- Null hypothesis: There is no association between the two variables.
- Alternative hypothesis (two-sided): There is an association between the two variables.

Interpretation

Significant results can be reported as “There was a significant association between variable A and variable B.”

Post-hoc analysis

Post hoc analysis for tests on a contingency table larger than 2 x 2 can be conducted by conducting tests for the component 2 x n tables. A correction for multiple tests should be applied.

Other notes and alternative tests

- For paired data, see the tests in the chapter *Tests for Paired Nominal Data*.
- For tables with 3 dimensions, the Cochran–Mantel–Haenszel test can be used.

Packages used in this chapter

The packages used in this chapter include:

- DescTools
- multcompView
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(DescTools)){install.packages("DescTools")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Association tests for nominal variables example

Alexander Anderson runs the pesticide safety training course in four counties. Students must pass in order to obtain their pesticide applicator's license. He wishes to see if there is an association between the county in which the course was held and the rate of passing the test. The following are his data.

County	Pass	Fail
Bloom County	21	5
Cobblestone County	6	11
Dougal County	7	8
Heimlich County	27	5

Reading the data as a matrix

```
Input ="
County      Pass   Fail
Bloom        21     5
Cobblestone  6      11
Dougal       7      8
Heimlich     27     5
")

Matrix = as.matrix(read.table(textConnection(Input),
                               header=TRUE,
                               row.names=1))

Matrix
```

Expected cell counts

The *chisq.test* function can be used to identify the expected counts for a contingency table.

Note in the results here that one cell has an expected count below 5, but that all expected counts are at least 1, and that cells with expected counts below 5 are less than 20% of cells. (1 / 8 cells = 13%).

```
Test = chisq.test(Matrix)

Test$expected

      Pass     Fail
Bloom    17.62222 8.377778
Cobblestone 11.52222 5.477778
Dougal   10.16667 4.833333
Heimlich 21.68889 10.311111
```

Effect size

See the chapter *Measures of Association for Nominal Variables* for a discussion of effect size statistics and their interpretation for contingency tables of nominal variables.

```
library(rcompanion)

cramerV(Matrix,
         digits=3)

Cramer V
0.439

### Note: k = 2 for this table,
### as the minimum categories in one dimension is 2.
```

Fisher Exact test of association

```
fisher.test(Matrix)

Fisher's Exact Test for Count Data

p-value = 0.000668
alternative hypothesis: two.sided
```

Post-hoc analysis

Post-hoc analysis can be conducted with pairwise Fisher's exact tests. The function *pairwiseNominalIndependence* in the *rcompanion* package can be used to conduct this analysis.

```
### Order matrix

Matrix = Matrix[(c("Heimlich", "Bloom", "Dougal", "Cobblestone")), ]
```

Matrix

```
### Pairwise tests of association

library(rcompanion)

PT = pairwiseNominalIndependence(Matrix,
                                   compare = "row",
                                   fisher = TRUE,
                                   gtest = FALSE,
                                   chisq = FALSE,
                                   method = "fdr", # see ?p.adjust for options
                                   digits = 3)
```

PT

	Comparison	p.Fisher	p.adj.Fisher
1	Heimlich : Bloom	0.740000	0.74000
2	Heimlich : Dougal	0.013100	0.02620
3	Heimlich : Cobblestone	0.000994	0.00596
4	Bloom : Dougal	0.037600	0.05640
5	Bloom : Cobblestone	0.003960	0.01190
6	Dougal : Cobblestone	0.720000	0.74000

Compact letter display

```
library(rcompanion)

cldList(p.adj.Fisher ~ Comparison,
        data      = PT,
        threshold = 0.05)

Group Letter MonoLetter
1 Heimlich    a      a
2 Bloom       ab     ab
3 Dougal      bc     bc
4 Cobblestone c      c
```

The table of adjusted *p*-values can be summarized to a table of letters indicating which treatments are not significantly different.

County	Percent passing	Letter
Heimlich County	84%	a
Bloom County	81	ab
Dougal County	47	bc
Cobblestone County	35	c

Counties sharing a letter are not significantly different by Fisher exact test, with *p*-values adjusted by FDR method for multiple comparisons (Benjamini-Hochberg false discovery rate).

This table of letters can also be found using the *pairwiseNominalMatrix* function along with the *multcompLetters* function in the *multcompView* package.

```
### Order matrix

Matrix = Matrix[(c("Heimlich", "Bloom", "Dougal", "Cobblestone")),]

Matrix

### Pairwise tests of association

library(rcompanion)

PM = pairwiseNominalMatrix(Matrix,
                           compare = "row",
                           fisher = TRUE,
                           gtest = FALSE,
                           chisq = FALSE,
                           method = "fdr", # see ?p.adjust for options
                           digits = 3)

PM

$Test
[1] "Fisher exact test"

$Method
[1] "fdr"

$Adjusted
      Heimlich Bloom Dougal Cobblestone
Heimlich 1.00000 0.7400 0.0262 0.00596
Bloom    0.74000 1.0000 0.0564 0.01190
Dougal   0.02620 0.0564 1.0000 0.74000
Cobblestone 0.00596 0.0119 0.7400 1.00000

library(multcompView)

multcompLetters(PM$Adjusted,
                compare = "<",
                threshold=0.05, ### p-value to use as significance threshold
                Letters=letters,
                reversed = FALSE)

Heimlich      Bloom      Dougal Cobblestone
      "a"        "ab"       "bc"      "c"
```

G-test of association

```
library(DescTools)
```

```
GTest(Matrix)
```

Log likelihood ratio (G-test) test of independence without correction
 $G = 17.14$, X^2 -squared $df = 3$, $p\text{-value} = 0.0006615$

Post-hoc analysis

```
### Order matrix

Matrix = Matrix[(c("Heimlich", "Bloom", "Dougal", "Cobblestone")),]

Matrix

### Pairwise tests of association

pairwiseNominalIndependence(Matrix,
                           compare = "row",
                           fisher = FALSE,
                           gtest = TRUE,
                           chisq = FALSE,
                           method = "fdr", # see ?p.adjust for options
                           digits = 3)

      Comparison p.Gtest p.adj.Gtest
1 Heimlich : Bloom 0.718000 0.71800
2 Heimlich : Dougal 0.008300 0.01660
3 Heimlich : Cobblestone 0.000506 0.00304
4 Bloom : Dougal 0.024800 0.03720
5 Bloom : Cobblestone 0.002380 0.00714
6 Dougal : Cobblestone 0.513000 0.61600
```

The table of adjusted p -values can be summarized to a table of letters indicating which treatments are not significantly different.

County	Percent passing	Letter
Heimlich County	84%	a
Bloom County	81	a
Dougal County	47	b
Cobblestone County	35	b

Counties sharing a letter are not significantly different by G-test for association, with p -values adjusted by FDR method for multiple comparisons (Benjamini-Hochberg false discovery rate).

Chi-square test of association

```
chisq.test(Matrix)

Pearson's Chi-squared test
```

X-squared = 17.32, df = 3, p-value = 0.0006072

Post-hoc analysis

```
### Order matrix

Matrix = Matrix[(c("Heimlich", "Bloom", "Dougal", "Cobblestone")),]

Matrix

### Pairwise tests of association

library(rcompanion)

pairwiseNominalIndependence(Matrix,
                           compare = "row",
                           fisher = FALSE,
                           gtest = FALSE,
                           chisq = TRUE,
                           method = "fdr", # see ?p.adjust for options
                           digits = 3)

      Comparison p.Chisq p.adj.Chisq
1 Heimlich : Bloom 0.99000 0.99000
2 Heimlich : Dougal 0.01910 0.03820
3 Heimlich : Cobblestone 0.00154 0.00924
4 Bloom : Dougal 0.05590 0.08380
5 Bloom : Cobblestone 0.00707 0.02120
6 Dougal : Cobblestone 0.77000 0.92400
```

The table of adjusted *p*-values can be summarized to a table of letters indicating which treatments are not significantly different.

County	Percent passing	Letter
Heimlich County	84%	a
Bloom County	81	ab
Dougal County	47	bc
Cobblestone County	35	c

Counties sharing a letter are not significantly different by chi-square test of association, with *p*-values adjusted by FDR method for multiple comparisons (Benjamini-Hochberg false discovery rate).

Optional readings

"Small numbers in chi-square and G-tests" in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/small.html.

References

"Fisher's Exact Test of Independence" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_07.html.

"G-test of Independence" in Mangiafico, S.S. 2015b. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_06.html.

"Chi-square Test of Independence" in Mangiafico, S.S. 2015c. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_05.html.

"Small Numbers in Chi-square and G-tests" in Mangiafico, S.S. 2015d. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/b_08.html.

Exercises M

1. Considering Alexander Anderson's data,

- a. Numerically, which county had the greatest percentage of passing students?
- b. Numerically, which county had the greatest number of total students?
- c. Was there an association between county and student success? Report the test used, why you chose this test, the p -value, and the conclusion.
- d. What was the effect size? What statistic was used? What is the interpretation of this value?
- e. Statistically, which counties performed the best? Be sure to indicate which post-hoc test you are using.
- f. Statistically, which performed the worst?
- g. Plot the data in an appropriate way, and submit the plot.
- h. Practically speaking, what are your conclusions? Consider all the information above that is relevant, and include descriptive statistics and observations about your plot that support your conclusions.

2. Ryuk and Rem held a workshop on planting habitat for pollinators like bees and butterflies. They wish to know if there is an association between the profession of the attendees and their willingness to undertake a conservation planting. The following are the data.

	will plant?	
Profession	Yes	No
Homeowner	13	14
Landscape	27	6

Farmer	7	19
NGO	6	6

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- a. Numerically, which profession had the greatest percentage of answering yes?
- b. Numerically, which profession had the greatest number of total attendees?
- c. Was there an association between profession and willingness to undertake a conservation planting? Report the test used, why you chose this test, the *p*-value, and the conclusion.
- d. What was the effect size? What statistic was used? What is the interpretation of this value?
- e. Statistically, which professions were most willing? Be sure to indicate which post-hoc test you are using.
- f. Statistically, which were least willing?
- g. Plot the data in an appropriate way, and submit the plot.
- h. Practically speaking, what are your conclusions? Consider all the information above that is relevant, and include descriptive statistics and observations about your plot that support your conclusions.

Measures of Association for Nominal Variables

There are several statistics that can be used to gauge the strength of the association between two nominal variables. They are used as measures of effect size for tests of association for nominal variables.

The statistics *phi* and Cramér's *V* are commonly used. Cramér's *V* varies from 0 to 1, with a 1 indicating a perfect association. *phi* varies from -1 to 1, with -1 and 1 indicating perfect associations. *phi* is available only for 2 x 2 tables.

Cohen's *w* is similar to Cramér's *V* in use, but its upper value is not limited to 1.

The odds ratio is appropriate for some contingency tables. It is useful when the table describes a bivariate response among groups. For example, disease or no disease among males and females. Or, pass or fail among locations.

Cohen's *h* is used to compare the difference in two proportions. It can be used in 2 x 2 contingency tables in cases where it makes sense to compare the proportions in rows or columns. It can also be used in cases where proportions are known but the actual counts are not. A value of 0 indicates no

difference in proportions, and the difference between proportions of 0.00 and 1.00 results in a value of $\pm \pi$ (c. 3.14).

Goodman and Kruskal's *lambda* statistic is also used to gauge the strength of the association between two nominal variables. It is formulated so that one dimension on the table is considered the independent variable, and one is considered the dependent variable, so that the independent variable is used to predict the dependent variable. It varies from 0 to 1.

Another measure of association is Tschuprow's *T*. It is similar to Cramér's *V*, and they are equivalent for square tables (one with an equal number of rows and columns).

Appropriate data

- Two nominal variables with two or more levels each. Usually expressed as a contingency table.
- Experimental units aren't paired.
- For *phi*, the table is 2 x 2 only.
- For odds ratio, one variable is bivariate. That is, it has two levels.

Hypotheses

- There are no hypotheses tested directly with these statistics.

Other notes and alternative tests

- Freeman's *theta* and *epsilon* squared are used for tables with one ordinal variable and one nominal variable.
- For tables with two ordinal variables, Kendall's *Tau-b*, Goodman and Kruskal's *gamma*, and Somers' *D* are used.

Interpretation of statistics

The interpretation of measures of association is always relative to the discipline, the specific data, and the aims of the analyst. Sometimes guidelines are given for "small", "medium", and "large" effects—for example from Cohen (1988) for behavioral sciences—but it is important to remember that these are still relative to the discipline and type of data. A smaller effect size may be considered "large" in psychology or behavioral science, but may be considered quite small in a physical science such as chemistry. The specific conditions of the study are important as well. For example, one would expect the difference in knowledge between a group completely ignorant of a subject and one educated in the subject to be large, but the difference between two groups educated in the same subject with different methods might be small.

The interpretation for odds ratio presented here was derived from that for Cohen's *h*. Note that the cutoffs for interpretation for odds ratio shown here are different from those given in the *Tests for Paired Nominal Data* chapter.

Small

Medium

Large

Cohen's w	0.10 – < 0.30	0.30 – < 0.50	≥ 0.50
phi	0.10 – < 0.30	0.30 – < 0.50	≥ 0.50
Cohen's h	0.20 – < 0.50	0.50 – < 0.80	≥ 0.80
Cramér's V, k = 2*	0.10 – < 0.30	0.30 – < 0.50	≥ 0.50
Cramér's V, k = 3*	0.07 – < 0.20	0.20 – < 0.35	≥ 0.35
Cramér's V, k = 4*	0.06 – < 0.17	0.17 – < 0.29	≥ 0.29
Odds ratio	1.55 – < 2.8	2.8 – < 5	≥ 5

Adapted from Cohen (1988).

* k is the minimum number of categories in either rows or columns.

Packages used in this chapter

The packages used in this chapter include:

- rcompanion
- vcd
- psych
- DescTools
- epitools

The following commands will install these packages if they are not already installed:

```
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(vcd)){install.packages("vcd")}
if(!require(psych)){install.packages("psych")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(epitools)){install.packages("epitools")}
```

Examples for measures of association for nominal variables

Cramér's V

```
Input =""
County      Pass   Fail
Bloom       21     5
Cobblestone 6      11
Dougal      7      8
Heimlich    27     5
")

Matrix = as.matrix(read.table(textConnection(Input),
                               header=TRUE,
                               row.names=1))

Matrix
```

```
library(rcompanion)

cramerV(Matrix,
         digits=4)

Cramer V
0.4387
```

```
library(vcd)

assocstats(Matrix)

Phi-Coefficient : NA
Cramer's V       : 0.439
```

```
library(DescTools)

CramerV(Matrix,
         conf.level=0.95)

Cramer V    lwr.ci    upr.ci
0.4386881  0.1944364  0.6239856
```

phi

```
Input ="
Sex      Pass  Fail
  Male     49   64
 Female    44   24
")

Matrix.2 = as.matrix(read.table(textConnection(Input),
                                 header=TRUE,
                                 row.names=1))
```

```
Matrix.2
```

```
library(psych)

phi(Matrix.2,
     digits = 4)

[1] -0.2068
```

```
library(DescTools)

Phi(Matrix.2)

[1] 0.206808
```

```
### It appears that DescTools always produces a positive value.
```

```
library(rcompanion)
```

```
cramerV(Matrix.2)
```

Cramer V
0.2068

```
### Note that Cramer's V is the same as the absolute value  
### of phi for 2 x 2 tables.
```

Odds ratio

The *oddsratio* function in the *epitools* package calculates the odds ratio for a contingency table of two responses, given in the columns, and treatments or groups given in rows. By default, the function assumes that the top row is the control group. The function can calculate the odds ratio a few different ways. If the option *method*=“wald” is used, the result for the first example would be the same as the odds of Female failing (24/44) divided by the odds of Male failing (64/49), or 0.418. The calculation by the default median unbiased estimation is slightly different. For tables larger than 2 x 2, the top row is used as the control for each other row.

```
Input ="  
Sex      Pass Fail  
  Male     49   64  
Female    44   24  
")  
  
Matrix.2 = as.matrix(read.table(textConnection(Input),  
                                header=TRUE,  
                                row.names=1))  
  
Matrix.2
```

```
library(epitools)  
oddsratio(Matrix.2)$measure
```

	odds ratio with 95% C.I.	estimate	lower	upper
Male	1.000000	NA	NA	
Female	0.420623	0.2229345	0.7791587	

```
Input ="  
County      Pass Fail  
Bloom       21    5  
Cobblestone 6    11  
Dougal      7    8  
Heimlich    27    5  
")
```

```

Matrix = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

Matrix

library(epitools)

oddsratio(Matrix)$measure

odds ratio with 95% C.I. estimate      lower      upper
Bloom          1.0000000    NA        NA
Cobblestone   7.1542726 1.846641 32.389543
Dougal         4.5405648 1.123581 20.551625
Heimlich       0.7813714 0.186609  3.265344

```

Cohen's w

```

Input =("
County          Pass  Fail
Bloom           21    5
Cobblestone     6     11
Dougal          7     8
Heimlich        27    5
")

Matrix = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

```

Matrix

library(rcompanion)

cohenW(Matrix)

Cohen w
0.4387

Note that because the smallest dimension in the table is 2,
the value of Cohen's w is the same as that for
Cramer's V.

Cohen's h

```

Input =("
Sex      Pass  Fail
Male     49    64
Female   44    24
")

```

```

")
```

```

Matrix.2 = as.matrix(read.table(textConnection(Input),
                                header=TRUE,
                                row.names=1))
```

```

Matrix.2
```

```

library(rcompanion)
```

```

cohenH(Matrix.2,
       observation = "row",
       digits = 3)
```

	Group	Proportion
1	Male	0.434
2	Female	0.647

```

Cohen's h
```

```

0.432
```

Goodman Kruskal lambda

```

Input =""
County          Pass   Fail
Bloom           21     5
Cobblestone     6      11
Dougal          7      8
Heimlich        27     5
")
```

```

Matrix = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))
```

```

Matrix
```

```

library(DescTools)
```

```

Lambda(Matrix,
       direction="column")
```

```

[1] 0.2068966
```

```

### County predicts Pass/Fail
```

Tschuprow's T

```

Input =""
County          Pass   Fail
```

```
Bloom           21      5
Cobblestone     6      11
Dougal          7      8
Heimlich        27      5
")
```

```
Matrix = as.matrix(read.table(textConnection(Input),
                               header=TRUE,
                               row.names=1))
```

```
Matrix
```

```
library(DescTools)
TschuprowT(Matrix)
```

```
[1] 0.3333309
```

Optional analysis: using effect size statistics when counts are not known

One property of effect size statistics is that they are not affected by sample size. This allows some to be used when proportions are known for a contingency table, but actual counts are not known.

As an example, we could analyze the proportions of votes for the Democratic candidate in 2016 and 2018 in Pennsylvania's 18th Congressional District. The 2016 data are for the presidential race where Hilary Clinton was the Democratic candidate. The 2018 data are for the House of Representatives election where Conor Lamb was the Democratic candidate.

Note that the effect sizes here are considered “small” by Cohen’s guidelines, but notable for American politics at the national level.

```
Input =""
  Democrat Not.Democrat
2016    0.380      0.620
2018    0.498      0.502
")
```

```
Penn18 = as.matrix(read.table(textConnection(Input),
                               header=TRUE,
                               row.names=1))
```

```
Penn18
```

	Democrat	Not.Democrat
2016	0.380	0.620
2018	0.498	0.502

```
library(rcompanion)
```

```
cohenH(Penn18,
```

```
observation = "row",
digits = 3)
```

Group	Proportion
1 2016	0.380
2 2018	0.498

Cohen's h
0.238

```
library(psych)
```

```
phi(Penn18,
digits = 3)
```

[1] -0.119

```
library(rcompanion)
```

```
cramerV(Penn18,
digits=3)
```

Cramer V
0.119

Optional analysis: Cohen's *h* and *prop.test* for paired data when subject matching isn't known

In the chapter on *Tests for Paired Nominal Data*, there is an example measuring rain barrel adoption before and after a class.

In this example, the identity of subjects was matched before and after, and the table could be analyzed with McNemar's test or a similar test, and the effect size could be determined with Cohen's *g*.

Before	After.yes	After.no
Before.yes	9	5
Before.no	17	15

However, if the subjects identities were not recorded in a manner that allowed for this matching, we could still measure the difference in proportions with Cohen's *h* and with *prop.test*.

```
Input =""
Time      Yes   No
Before    14   32
After     26   20
")
```

```
Unmatched = as.matrix(read.table(textConnection(Input),
header=TRUE,
```

```
row.names=1))
```

Unmatched

```
library(rcompanion)
```

```
cohenH(Unmatched,
  observation = "row",
  digits = 3)
```

	Group Proportion
1 Before	0.304
2 After	0.565

Cohen's h
0.533

```
Yes = c(14, 26)
Trials = c(14+32, 26+20)
prop.test(Yes, Trials)
```

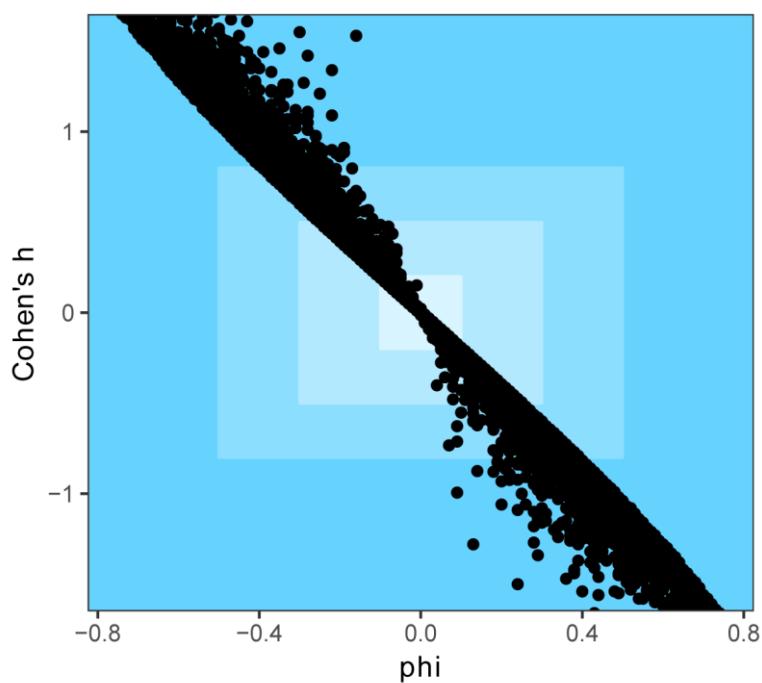
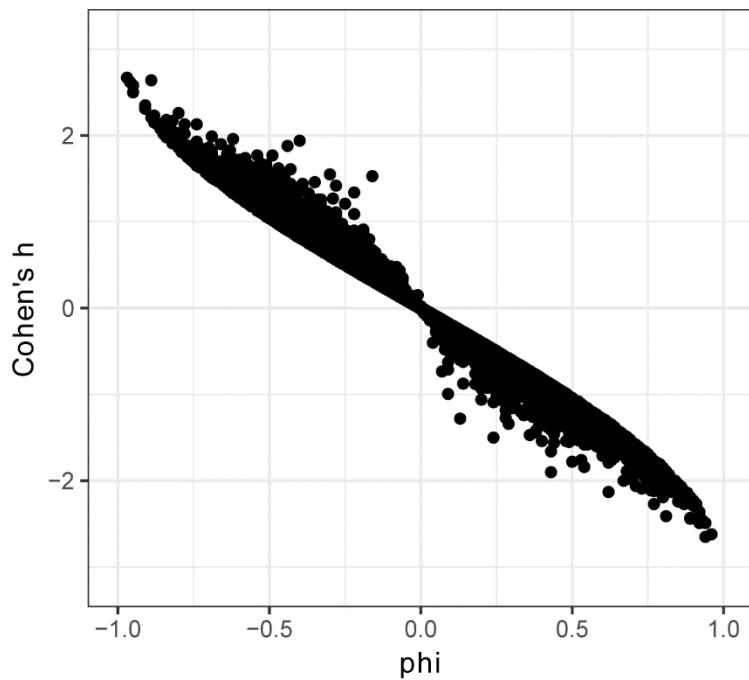
2-sample test for equality of proportions with continuity correction

```
data: Yes out of Trials
X-squared = 5.3519, df = 1, p-value = 0.0207
alternative hypothesis: two.sided
95 percent confidence interval:
-0.47806470 -0.04367443
sample estimates:
prop 1   prop 2
0.3043478 0.5652174
```

Optional: Comparing *phi* and Cohen's *h* for a 2 x 2 table

For a 2 x 2 table, Cohen's *h* and *phi* do not follow a strictly monotonic relationship. Both statistics equal zero when there is no relationship, and Cohen's *h* approaches *pi* as *phi* approaches -1. (As calculated here, the statistics have a negative relationship.) But there is variability in their values in between these endpoints for some simulated data.

In the second figure below, the colors indicate Cohen's interpretation of less-than-small, small, medium, and large as the blue becomes darker. Note that the interpretations across the two statistics may vary for the same data, as is evidenced by the fact that the plotted values do not go through the corner of the rectangle delineating the medium and large interpretations. For example, for a *phi* of approximately -0.40 (medium), Cohen's *h* are likely to range from 0.82 to 1.17, which would be interpreted as a large effect.



Optional analysis: changing the order of the table

Note that if we change the order of the rows in the table, the results for Cramér's V , Cohen's w , and λ (with $direction=column$) do not change. This is because these statistics treat the variables as nominal and not ordinal.

```

Input ="
County          Pass   Fail
Heimlich        27     5
Bloom           21     5
Dougal          7      8
Cobblestone     6      11
")

Matrix.3 = as.matrix(read.table(textConnection(Input),
                                header=TRUE,
                                row.names=1))

Matrix.3

### Cramer's V
library(rcompanion)
cramerV(Matrix.3,
         digits=4)

Cramer V
0.4387

### Cohen's w
cohenW(Matrix.3)

Cohen w
0.4387

### Goodman Kruskal Lambda
library(DescTools)
Lambda(Matrix.3,
       direction="column")

### Treat County as independent variable
[1] 0.2068966

```

Optional analysis: comparing statistics

The following examples may give some sense of the difference between Cramér's V , and λ .

```

Input =""
x   Y1   Y2
x1  10   0
x2  0    10
")

Matrix.x = as.matrix(read.table(textConnection(Input),
                                 header=TRUE,
                                 row.names=1))

library(rcompanion)
cramerV(Matrix.x)

Cramer V
1

library(rcompanion)
cohenW(Matrix.x)

Cohen W
1

library(DescTools)
Lambda(Matrix.x,
       direction="column")

[1] 1

Input =""
x   Y1   Y2
x1  10   0
x2  10   10
")

Matrix.y = as.matrix(read.table(textConnection(Input),
                                 header=TRUE,
                                 row.names=1))

library(rcompanion)
cramerV(Matrix.y)

Cramer V
0.5

library(rcompanion)
cohenW(Matrix.y)

```

```

Cohen w
0.5

library(DescTools)

Lambda(Matrix.y,
       direction="column")

[1] 0

### X predicts Y.

Input =""
X   Y1   Y2
x1  10   0
x2  10   10
x3  10   20
x4  10   30
")

Matrix.z = as.matrix(read.table(textConnection(Input),
                                 header=TRUE,
                                 row.names=1))

library(rcompanion)

cramerV(Matrix.z)

Cramer V
0.4488

library(rcompanion)

cohenW(Matrix.z)

Cohen w
0.4488

library(DescTools)

Lambda(Matrix.z,
       direction="column")

[1] 0.25

### X predicts Y.

```

References

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.

Tests for Paired Nominal Data

Tests of symmetry—or marginal homogeneity—for nominal data are used when the counts on a contingency table represent values that are paired or repeated in time.

As an example, consider a question repeated on a pre-test and a post-test. We may want to know if the number of correct responses changed from the pre-test to the post-test.

Did students have the correct answer to the question?

	After	
Before	Correct	Incorrect
Correct	2	0
Incorrect	21	7

Note that the row names and column names have the same levels, and that counts represent paired responses. That is, for each observation you must know the individual's response before and after.

Also note that the number of students included in the table are 30, or the sum of the cell counts.

In essence, those students with the same response before and after don't affect the assessment of the change in responses. We would focus on the "discordant" counts. That is, How many students had incorrect answers before and correct answers after, in contrast to those who had the reverse trend. Here, because 21 changed from incorrect to correct, and 0 changed from correct to incorrect, we might suspect that there was a significant change in responses from incorrect to correct.

To grasp the difference between nominal tests of association and nominal tests of symmetry, be sure to visit the coffee and tea example below in the section "An example without repeated measures, comparing test of symmetry with test of association".

Appropriate data

- Two nominal variables with two or more levels each, and each with the same levels.
- Observations are paired or matched between the two variables.
- McNemar and McNemar–Bowker tests may not be appropriate if discordant cells have low counts.

Hypotheses

- Null hypothesis: The contingency table is symmetric. That is, the probability of cell $[i, j]$ is equal to the probability of cell $[j, i]$.
- Alternative hypothesis (two-sided): The contingency table is not symmetric.

Interpretation

Depending on the context, significant results can be reported as e.g. "There was a significant change from answer A to answer B." Or, "X was more popular than Y."

Post-hoc analysis

Post hoc analysis for tests on a contingency table larger than 2×2 can be conducted by conducting tests for the component 2×2 tables. A correction for multiple tests should be applied.

Other notes and alternative tests

- For unpaired data, see the tests in the chapter *Association Tests for Nominal Data*.
- For multiple times or groups, Cochran's Q test can be used.

Packages used in this chapter

The packages used in this chapter include:

- EMT
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(EMT)){install.packages("EMT")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

McNemar and McNemar-Bowker tests

For a 2×2 table, the most common test for symmetry is McNemar's test. For larger tables, McNemar's test is generalized as the McNemar–Bowker symmetry test. One drawback to the latter test is that it may fail if there are 0's in certain locations in the matrix.

McNemar's test may not be reliable if there are low counts in the "discordant" cells. Authors recommend that these cells to sum to at least 5 or 10 or 25.

Exact tests

Exact tests of symmetry reduce to exact tests for goodness-of-fit. A 2×2 table is analyzed with a binomial exact test, and a larger table is analyzed with a multinomial exact test. Examples of these are shown in this chapter in the "Optional analyses: conducting exact tests for symmetry" section.

The *nominalSymmetryTest* function can conduct these exact tests easily.

Example of tests for paired data nominal data

Alucard teaches a Master Gardener training on rain gardens for stormwater management and one on rain barrels. He wishes to assess if people are more willing to install these green infrastructure practices after attending the training. His data follow. Note that there are 46 attendees answering each question.

Are you planning to install a rain barrel?

	After	
Before	Yes	No
Yes	9	5
No	17	15

Are you planning to install a rain garden?

	After		
Before	Yes	No	Maybe
Yes	6	0	1
No	5	3	7
Maybe	11	1	12

Rain barrel

```
Input ="
Before      After.yes   After.no
Before.yes    9          5
Before.no     17         15
")

Matrix.1 = as.matrix(read.table(textConnection(Input),
                                 header=TRUE,
                                 row.names=1))

Matrix.1

sum(Matrix.1)

[1] 46
```

Rain garden

```
Input ="
Before      Yes.after   No.after   Maybe.after
Yes.before   6           0           1
No.before    5           3           7
Maybe.before 11          1           12
")

Matrix.2 = as.matrix(read.table(textConnection(Input),
                                 header=TRUE,
                                 row.names=1))

Matrix.2

sum(Matrix.2)

[1] 46
```

Exact tests for symmetryRain barrel

```
library(rcompanion)

nominalSymmetryTest(Matrix.1,
                      digits = 3)

$Global.test.for.symmetry

  Dimensions p.value
1      2 x 2  0.0169
```

Rain garden

```
library(rcompanion)

nominalSymmetryTest (Matrix.2,
                      method="fdr",
                      digits = 3)

### Note: This may take a long time
###       Use MonteCarlo option for large matrices or counts

$Global.test.for.symmetry

  Dimensions p.value
1      3 x 3  2e-04

$Pairwise.symmetry.tests
                                         Comparison p.value p.adjust
1 Yes.before/Yes.after : No.before/No.after  0.0625  0.0703
2 Yes.before/Yes.after : Maybe.before/Maybe.after 0.00635  0.0190
3 No.before/No.after : Maybe.before/Maybe.after  0.0703  0.0703
```

\$p.adjustment

Method
1 fdr

Maybe to Yes, p.value = 0.0190

Before	Yes.after	No.after	Maybe.after
Yes.before	6	0	1
No.before	5	3	7
Maybe.before	11	1	12

Effect size

Appropriate effect sizes for data subjected to McNemar, McNemar–Bowker, or equivalent exact tests include Cohen's g and odds ratio.

Considering a 2×2 table, with a and d being the concordant cells and b and c being the discordant cells, the odds ratio is simply the greater of (b/c) or (c/b) , and P is the greater of $(b/(b+c))$ or $(c/(b+c))$. Cohen's g is $P - 0.5$. These statistics can be extended to larger tables.

Cohen (1988) lists interpretations for Cohen's g . These can easily translated into interpretations for the odds ratio. For example, the cutoff for the “small” interpretation is $g \geq 0.05$, which would be 0.55 : 0.45 odds, or 0.55/0.45, or 1.22.

	<u>Small</u>	<u>Medium</u>	<u>Large</u>
Cohen's g	$0.05 - < 0.15$	$0.15 - < 0.25$	≥ 0.25
Odds ratio (OR)	$1.22 - < 1.86$	$1.86 - < 3.00$	≥ 3.00

Rain barrel

In the output, OR is the odds ratio, and g is Cohen's g .

```
library(rcompanion)

cohenG(Matrix.1)

$Global.statistics
  Dimensions   OR      P      g
  1           2 x 2  3.4  0.773  0.273
```

Rain garden

```
library(rcompanion)

cohenG(Matrix.2)

$Global.statistics
  Dimensions   OR      P      g
  1           3 x 3 11.5  0.92  0.42

$Pairwise.statistics
                                         Comparison   OR      P      g
  1     Yes.before/Yes.after : No.before/No.after Inf      1  0.5
  2     Yes.before/Yes.after : Maybe.before/Maybe.after 11  0.917  0.417
  3     No.before/No.after : Maybe.before/Maybe.after    7  0.875  0.375
```

McNemar and McNemar–Bowker chi-square tests for symmetryRain barrel

```
mcnemar.test(Matrix.1)
```

McNemar's Chi-squared test with continuity correction

McNemar's chi-squared = 5.5, df = 1, p-value = 0.01902

Rain garden

```
mcnemar.test(Matrix.2)
```

McNemar's Chi-squared test

McNemar's chi-squared = 17.833, df = 3, p-value = 0.0004761

An example without repeated measures, comparing test of symmetry with test of association

As another example, consider a survey of tea and coffee drinking, in which each respondent is asked both if they drink coffee, and if they drink tea.

	Tea	
Coffee	Yes	No
Yes	37	17
No	9	25

We would use a test of symmetry in this case if the question we wanted to answer was, Is coffee more popular than tea? That is, is it more common for someone to drink coffee and not tea than to drink tea and not coffee? (Those who drink both or drink neither are not relevant to this question.)

This is an example of using a test of symmetry to test the relative frequency of two dichotomous variables when the same subjects are surveyed.

```
Input ="
Coffee  Yes  No
Yes     37   17
No      9    25
")

Matrix.3 = as.matrix(read.table(textConnection(Input),
                                header=TRUE,
                                row.names=1))
```

```
mcnemar.test(Matrix.3)
```

McNemar's Chi-squared test with continuity correction

McNemar's chi-squared = 1.8846, df = 1, p-value = 0.1698

Neither coffee nor tea is more popular, specifically because

```
### neither the 9 nor the 17 in the table are large relative to
### the other.
```

A test of association answers a very different question. Namely, Is coffee drinking associated with tea drinking? That is, is someone more likely to drink tea if they drink coffee?

```
chisq.test(Matrix.3)
```

```
Pearson's Chi-squared test with Yates' continuity correction
```

```
X-squared = 13.148, df = 1, p-value = 0.0002878
```

```
### Coffee drinking and tea drinking are associated, in this case people who
### drink coffee are likely to drink tea. This is a positive association.
### A negative association could also be significant.
```

Optional analysis: a 4 x 4 example with several 0's

As an additional example, imagine a religious caucusing event in which advocates try to sway attendees to switch their religions.

Matrix row names are the attendees' original religions, and the column names, with a "2" added, are attendees' new religions after the caucus. Note there are several 0 counts in the matrix.

Note first that the *mcnemar.test* function fails, namely because of the position of some 0 counts in the matrix.

Second, note that the *multinomial.test* function used by the *nominalSymmetryTest* function would take a long time to calculate an exact *p*-value for this matrix, so the *MonteCarlo=TRUE* option is used. The number of samples used in the Monte Carlo approach can be adjusted with the *ntrial* option. Some of the *p*-values in the post-hoc analysis cannot be produced because of the placement of 0 counts, but these should also be considered non-significant results.

```
Input ="
Before      Pastafarian2   Discordant2   Dudist2    Jedi2
Pastafarian  7             0              23          0
Discordant   0             7              0           33
Dudist       3             0              7           1
Jedi         0             1              0           7
")

Matrix.4 = as.matrix(read.table(textConnection(Input),
                                header=TRUE,
                                row.names=1))

Matrix.4
```

McNemar -Bowker test

```
mcnemar.test(Matrix.4)

McNemar's Chi-squared test

McNemar's chi-squared = NaN, df = 6, p-value = NA
```

Exact test

```
library(rcompanion)

nominalSymmetryTest(Matrix.4,
                      method="fdr",
                      digits = 3,
                      MonteCarlo = TRUE,
                      ntrial = 100000)

$Global.test.for.symmetry

  Dimensions p.value
1        4 x 4      0

$Pairwise.symmetry.tests
                                         Comparison p.value p.adjust
1 Pastafarian/Pastafarian2 : Discordant/Discordant2      <NA>     NA
2             Pastafarian/Pastafarian2 : Dudist/Dudist2  8.8e-05 1.32e-04
3             Pastafarian/Pastafarian2 : Jedi/Jedi2      <NA>     NA
4       Discordant/Discordant2 : Dudist/Dudist2      <NA>     NA
5       Discordant/Discordant2 : Jedi/Jedi2  4.07e-09 1.22e-08
6             Dudist/Dudist2 : Jedi/Jedi2            1 1.00e+00

$p.adjustment

  Method
1   fdr
```

A look at the significant results

Pastafarian to Dudist, p-value = 0.000132

Before	Pastafarian2	Discordiant2	Dudist2	Jedi2
Pastafarian	7	0	23	0
Discordiant	0	7	0	33
Dudist	3	0	7	1
Jedi	0	1	0	7

Discordant to Jedi, p-value < 0.0001

Before	Pastafarian2	Discordiant2	Dudist2	Jedi2
Pastafarian	7	0	23	0
Discordiant	0	7	0	33

Dudist	3	0	7	1
Jedi	0	1	0	7

Optional analyses: conducting exact tests for symmetry

The exact symmetry tests can be conducted directly with the *binom.test* and *multinomial.test* functions. These results match the results of the *nominalSymmetryTest* function.

Rain barrel

		After	
Before	Yes	No	
Yes	9	5	
No	17	15	

Rain garden

		After		
Before	Yes	No	Maybe	
Yes	6	0	1	
No	5	3	7	
Maybe	11	1	12	

Rain barrel

For a 2×2 matrix, x = the count in one of the discordant cells, and n = the sum of the counts in discordant cells. The *expected* proportion is 0.50.

You could also follow the method for an $n \times n$ matrix below.

```
x = 5
n = 5 + 17
expected = 0.50
```

```
binom.test(x, n, expected)
```

Exact binomial test

number of successes = 5, number of trials = 22, p-value = 0.0169

Rain garden

For an $n \times n$ matrix, *observed* = a vector of the counts of discordant cells. *Expected* is a vector of length $(n * n - n)$, with each value equal to $(1 / (n * n - n))$.

```
observed = c(0, 1, 5, 7, 11, 1)
expected = c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
```

```
library(EMT)
```

```
multinomial.test(observed, expected)
```

```
Exact Multinomial Test, distance measure: p
```

Events	pobs	p.value
142506	0	2e-04

Yes-No

For post-hoc testing, reduce the matrix to a 2 x 2 matrix.

```
x = 0
n = 0 + 5
expected = 0.50

binom.test(x, n, expected)

Exact binomial test

number of successes = 0, number of trials = 5, p-value = 0.0625
```

Yes-Maybe

```
x = 1
n = 1 + 11
expected = 0.50

binom.test(x, n, expected)

Exact binomial test

number of successes = 1, number of trials = 12, p-value = 0.006348
```

No-Maybe

```
x = 7
n = 7 + 1
expected = 0.50

binom.test(x, n, expected)

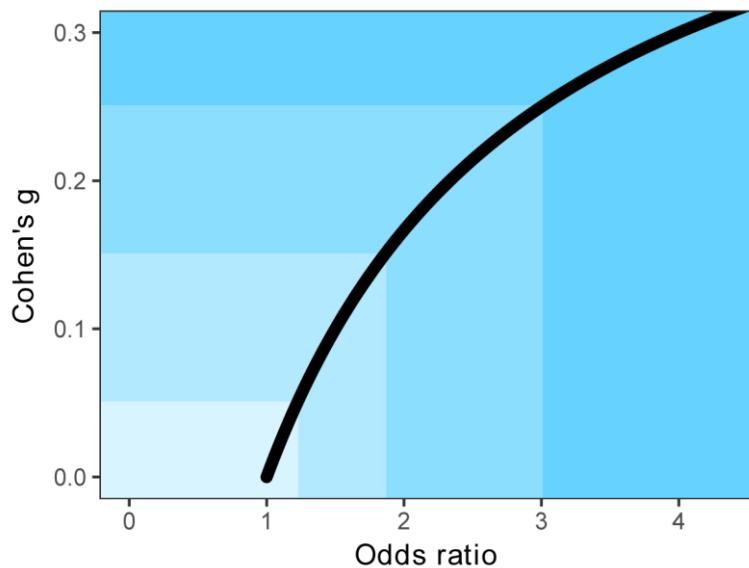
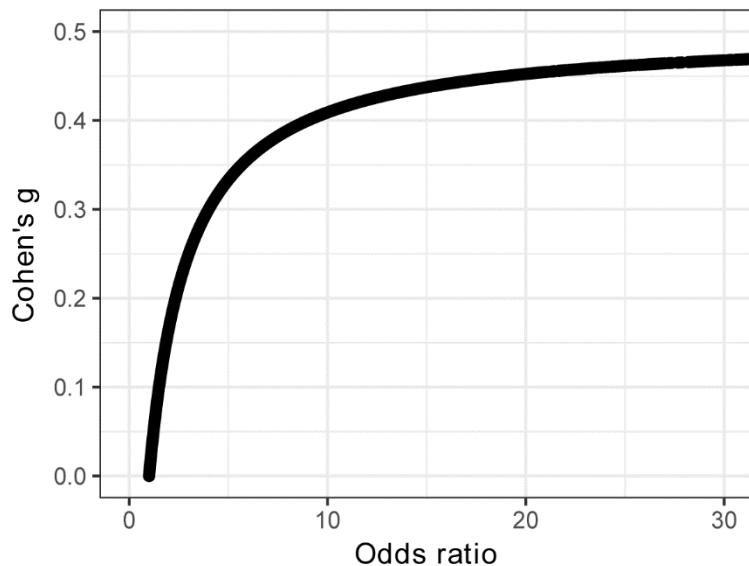
Exact binomial test

number of successes = 7, number of trials = 8, p-value = 0.07031
```

Optional: Comparing odds ratio and Cohen's *g*

For a 2 x 2 table of matched counts, odds ratio and Cohen's *g* are precisely related, and their interpretations according to Cohen will always coincide. An odds ratio value of 1 corresponds to a Cohen's *g* of 0. Cohen's *g* approaches 0.5 as odds ratio approaches infinity. In the second figure below,

the colors indicate Cohen's interpretation of less-than-small, small, medium, and large as the blue becomes darker.



References

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.

Exercises N

1. Considering Alucard's data,

- a. How many students responded to the ***rain barrel*** question?
- b. How many students changed their answer on the ***rain barrel*** question from no to yes?
- c. What do you conclude about the results on the ***rain barrel*** question? Include:
 - The statistical test you are using, *p*-value from that test, conclusion of the test, and any other comments on the test.
 - The effect size and interpretation.
 - Description of the observed counts or proportions. Is there anything notable?
 - Any other practical conclusions you wish to note.

- d. What do you conclude about the results of the ***global test*** of the ***rain garden*** question?
- e. What do you conclude about the results of the ***post-hoc*** analysis of the ***rain garden*** question?

2. Considering the coffee and tea example, Do you understand the difference between the hypotheses for tests of association and tests of symmetry? Would you be comfortable choosing the correct approach?

3. Ryuk and Rem held a workshop on planting habitat for pollinators like bees and butterflies. They wish to know if attendees were more likely to do a planting after the workshop than before.

will plant?

	After		
Before	Yes.after	No.after	Maybe.after
Yes	17	0	0
No	5	9	13
Maybe	15	0	7

For each of the following, answer the question, and ***show the output from the analyses you used to answer the question.***

- a. How many students responded to this question?
- b. How many students changed their answer from no to yes?
- c. What do you conclude about the results of the ***global test***?

d. What do you conclude about the results of the *post-hoc* analysis?

Cochran–Mantel–Haenszel Test for 3-Dimensional Tables

The Cochran–Mantel–Haenszel test is an extension of the chi-square test of association. It is used for multiple chi-square tests across multiple groups or times. The data are stratified so that each chi-square table is within one group or time.

The following data investigate whether there is a link between listening to podcasts and using public transportation to get to work, collected across three cities. This can be thought of as a 2×2 contingency table in each of the three cities.

Data can be arranged in a table of counts or can be arranged in long-format with or without counts. If a table is used for input, it should follow R's *ftable* format, as shown.

Table format

		City	Bikini.Bottom	Frostbite.Falls	New.New.York
Listen	Transport				
Podcast	Drive		13	17	5
	Public		27	25	27
No.podcast	Drive		23	22	17
	Public		44	31	22

Long-format with counts

City	Listen	Transport	Count
Bikini.Bottom	Podcast	Drive	13
Bikini.Bottom	Podcast	Public	27
Bikini.Bottom	No.podcast	Drive	23
Bikini.Bottom	No.podcast	Public	44
Frostbite.Falls	Podcast	Drive	17
Frostbite.Falls	Podcast	Public	25
Frostbite.Falls	No.podcast	Drive	22
Frostbite.Falls	No.podcast	Public	31
New.New.York	Podcast	Drive	5
New.New.York	Podcast	Public	27
New.New.York	No.podcast	Drive	17
New.New.York	No.podcast	Public	22

The test can be conducted with the *mantelhaen.test* function in the native *stats* package.

One assumption of the test is that there are no three-way interactions in the data. This is confirmed with a non-significant result from a test such as the Woolf test or Breslow-Day test.

Post-hoc analysis can include looking at the individual chi-square, Fisher exact, or G-test for association for each time or group.

The component $n \times n$ tables can be 2 x 2 or larger.

Appropriate data

- Three nominal variables with two or more levels each.
- Data can be stratified as $n \times n$ tables with the third time or grouping variable

Hypotheses

- Null hypothesis: There is no association between the two inner variables.
- Alternative hypothesis (two-sided): There is an association between the two inner variables.

Interpretation

Significant results can be reported as “There was a significant association between variable A and variable B [across groups].”

Post-hoc analysis

Post-hoc analysis can include looking at the individual chi-square, Fisher exact, or g-test for association for each time or group.

Packages used in this chapter

The packages used in this chapter include:

- psych
- vcd
- DescTools
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(vcd)){install.packages("vcd")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

C-M-H test example: long-format with counts

Alexander Anderson is concerned that there is a bias in his teaching methods for his pesticide applicator's course. He wants to know if there is an association between students' sex and passing the course across the four counties in which he teaches. The following are his data.

```
Input = "
County      Sex    Result  Count
Bloom       Female Pass     9
```

```
Bloom      Female  Fail    5
Bloom      Male    Pass   7
Bloom      Male    Fail  17
Cobblestone Female Pass  11
Cobblestone Female Fail   4
Cobblestone Male   Pass  9
Cobblestone Male   Fail  21
Dougal     Female Pass  9
Dougal     Female Fail  7
Dougal     Male   Pass  19
Dougal     Male   Fail  9
Heimlich   Female Pass 15
Heimlich   Female Fail  8
Heimlich   Male   Pass 14
Heimlich   Male   Fail 17
")
```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
### Order factors otherwise R will alphabetize them
```

```
Data$County = factor(Data$County,
                      levels=unique(Data$County))
```

```
Data$Sex     = factor(Data$Sex,
                      levels=unique(Data$Sex))
```

```
Data$Result = factor(Data$Result,
                      levels=unique(Data$Result))
```

```
### Check the data frame
```

```
library(psych)
```

```
headTail(Data)
```

```
str(Data)
```

```
summary(Data)
```

```
### Remove unnecessary objects
```

```
rm(Input)
```

Convert data to a table

```
Table = xtabs(Count ~ Sex + Result + County,
              data=Data)
```

```
### Note that the grouping variable is last in the xtabs function
```

```
ftable(Table) # Display a flattened table

      County Bloom Cobblestone Dougal Heimlich
Sex   Result
Female Pass      9       11      9      15
        Fail      5       4       7      8
Male   Pass      7       9      19      14
        Fail     17      21      9      17
```

Cochran-Mantel-Haenszel test

```
mantelhaen.test(Table)

Mantel-Haenszel chi-squared test with continuity correction
Mantel-Haenszel X-squared = 6.7314, df = 1, p-value = 0.009473
alternative hypothesis: true common odds ratio is not equal to 1
```

Woolf test

```
library(vcd)
woolf_test(Table)

### Woolf test for homogeneity of odds ratios across strata.
### If significant, C-M-H test is not appropriate

Woolf-test on Homogeneity of Odds Ratios (no 3-way assoc.)

X-squared = 7.1376, df = 3, p-value = 0.06764
```

Post-hoc analysis

The *groupwiseCMH* function will conduct analysis of the component $n \times n$ tables with Fisher exact, g-test, or chi-square tests of association. It accepts only a 3-dimensional table. The *group* option indicates which dimension should be considered the grouping variable (1, 2, or 3). It will conduct only one type of test at a time. That is, if multiple of the options *fisher*, *gtest*, or *chisq* are set to *TRUE*, it will conduct only one of them. As usual, *method* is the *p*-value adjustment method (see *?p.adjust* for options), and *digits* indicates the number of digits in the output. The *correct* option is used by the chi-square test function.

```
library(rcompanion)

groupwiseCMH(Table,
  group = 3,
  fisher = TRUE,
  gtest = FALSE,
  chisq = FALSE,
  method = "fdr",
```

```

correct = "none",
digits = 3)

      Group   Test p.value adj.p
1    Bloom Fisher  0.0468 0.0936
2 Cobblestone Fisher  0.0102 0.0408
3     Dougal Fisher  0.5230 0.5230
4    Heimlich Fisher  0.1750 0.2330

```

C-M-H test example: table format

The *read.ftable* function can be very fussy about the formatting of the input. 1) It seems to not like a blank first line, so the double quote symbol in the input should be on the same line as the column names. 2) It doesn't like leading spaces on the input lines. These may appear when you paste the code in to the RStudio Console or R Script area. One solution is to manually delete these spaces. R Script files that are saved without these leading spaces should be able to be opened and run without further modification.

The Cochran–Mantel–Haenszel test, Woolf test, and post-hoc analysis would be the same as those conducted on *Table* above.

```

Input =(
"          County Bloom Cobblestone Dougal Heimlich
Sex   Result
Female Pass        9       11       9      15
         Fail        5        4       7       8
Male   Pass        7       9      19      14
         Fail       17       21       9      17
")
Table = as.table(read.ftable(textConnection(Input)))
ftable(Table)
### Display a flattened table

```

Cochran's Q Test for Paired Nominal Data

Cochran's Q test is an extension of the McNemar test, when the response variable is dichotomous and there are either multiple times for a repeated measure or multiple categories with paired responses. A dichotomous variable is a nominal variable with only two levels.

One case would be extending the rain barrel question from the previous chapter to multiple times. A more common case would be extending the coffee and tea example to multiple beverages. In this case, we would test among several beverages which is more popular than the others.

Data are not typically arranged in a contingency table, but are either organized in long-format, with each row representing a single observation, or organized in a short-format matrix with each row representing an experimental unit with the repeated measures across columns.

Long-format

Student	Beverage	Response
a	Coffee	Yes
a	Tea	Yes
a	Cola	No
b	Coffee	No
b	Tea	No
b	Cola	Yes
c	Coffee	Yes
c	Tea	No
c	Cola	No

Short-format

Student	Coffee	Tea	Cola
a	Yes	Yes	No
b	No	No	Yes
c	Yes	No	No

Data need to be arranged in an unreplicated complete block design. That is, for each experimental unit (*Student* in this case) there needs to be one and only one response per question or time.

The package *RVAideMemoire* has a function *cochran.qtest* which will conduct Cochran's Q test on long-format data. The function *symmetry_test* in the *coin* package will also conduct the test using permutation, which a type of resampling technique.

A word of caution about the *symmetry_test* function: be sure not to use it in cases where the response variable is multinomial; that is, where there are more than two levels in the response. The function will return a result, but it is not the same kind of symmetry result as the McNemar–Bowker test. I suspect that for a multinomial response variable, *coin* treats it like an ordered factor, not like a nominal variable.

When there are two questions or times, Cochran's Q test is equivalent to the McNemar test.

Post-hoc analysis can be conducted with pairwise McNemar or exact tests on 2 x 2 tables. The function *pairwiseMcnemar* in the *rcompanion* package will conduct pairwise McNemar, exact, or permutation tests.

Appropriate data

- The response variable is a dichotomous nominal variable.
- The responses are paired by experimental unit.
- The responses are measured across two or more times or factors.

- The data follow an unreplicated complete block design, with each experimental unit treated as the block.

Hypotheses

- Null hypothesis: The marginal probability of a [positive] response is unchanged across the times or factors. That is, a positive response is equally likely across times or factors.
- Alternative hypothesis: Positive responses are not equally likely across times or factors.

Interpretation

Significant results can be reported as "There was a significant difference in the proportion of positive responses across times or factors." Or, "A positive response for X was more likely than a positive response for Y."

Post-hoc analysis

Post hoc analysis can be conducted by conducting tests for the component 2 x 2 tables. A correction for multiple tests should be applied.

Packages used in this chapter

The packages used in this chapter include:

- psych
- RVAideMemoire
- coin
- reshape2
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(RVAideMemoire)){install.packages("RVAideMemoire")}
if(!require(coin)){install.packages("coin")}
if(!require(reshape2)){install.packages("reshape2")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Cochran's Q test example: long-format

Hayley Smith teaches an extension course on environmentally-friendly lawn care. After the course, she surveys her students about their willingness to adopt good practices: Soil testing, measuring the amount of irrigation, raising the mowing height, and returning the grass clippings to the lawn. She wants to know which practice students were more willing to adopt.

```
Input = "
Practice      Student    Response
SoilTest      a          Yes
SoilTest      b          No
SoilTest      c          Yes
SoilTest      d          Yes
SoilTest      e          No
```

```

SoilTest    f      Yes
SoilTest    g      Yes
SoilTest    h      Yes
SoilTest    i      No
SoilTest    j      Yes
SoilTest    k      Yes
SoilTest    l      Yes
SoilTest    m      Yes
SoilTest    n      Yes
Irrigation  a      Yes
Irrigation  b      No
Irrigation  c      No
Irrigation  d      No
Irrigation  e      No
Irrigation  f      No
Irrigation  g      No
Irrigation  h      No
Irrigation  i      Yes
Irrigation  j      No
Irrigation  k      No
Irrigation  l      No
Irrigation  m      No
Irrigation  n      No
MowHeight   a      Yes
MowHeight   b      No
MowHeight   c      Yes
MowHeight   d      Yes
MowHeight   e      Yes
MowHeight   f      Yes
MowHeight   g      Yes
MowHeight   h      Yes
MowHeight   i      No
MowHeight   j      Yes
MowHeight   k      Yes
MowHeight   l      Yes
MowHeight   m      Yes
MowHeight   n      Yes
Clippings   a      Yes
Clippings   b      No
Clippings   c      No
Clippings   d      Yes
Clippings   e      Yes
Clippings   f      No
Clippings   g      No
Clippings   h      Yes
Clippings   i      Yes
Clippings   j      Yes
Clippings   k      Yes
Clippings   l      No
Clippings   m      Yes
Clippings   n      No
")

```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```

### Order factors otherwise R will alphabetize them

Data$Practice = factor(Data$Practice,
                       levels=unique(Data$Practice))

Data$Response = factor(Data$Response,
                       levels=c("Yes", "No"))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

### View data as a table

Data$Response.n = as.numeric(Data$Response) - 1    ### Creates a new numeric variable
                                                       ### that is Response as a 0 or 1

Table = xtabs(Response.n ~ Student + Practice,
               data=Data)

Table

      Practice
Student SoilTest Irrigation MowHeight Clippings
      a        0        0        0        0
      b        1        1        1        1
      c        0        1        0        1
      d        0        1        0        0
      e        1        1        0        0
      f        0        1        0        1
      g        0        1        0        1
      h        0        1        0        0
      i        1        0        1        0
      j        0        1        0        0
      k        0        1        0        0
      l        0        1        0        1
      m        0        1        0        0
      n        0        1        0        1

```

```
### View counts of responses
```

```
xtabs( ~ Practice + Response,
      data=Data)
```

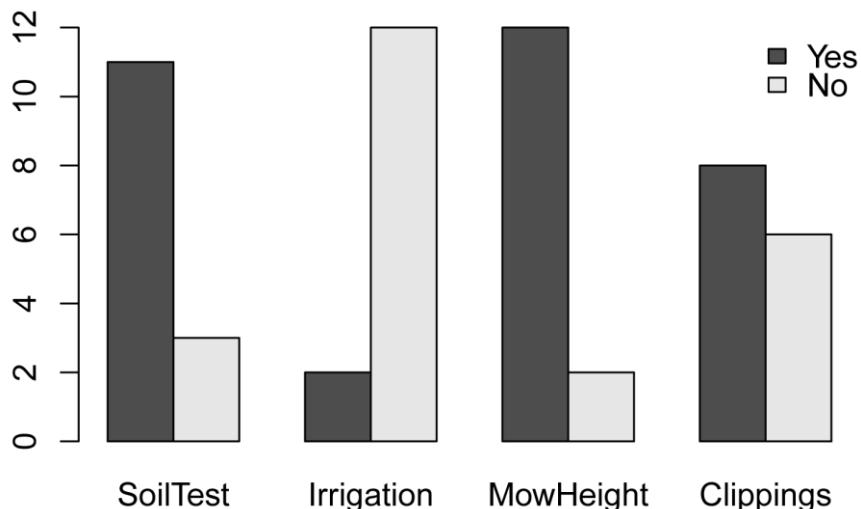
Practice	Response	
	Yes	No
SoilTest	11	3
Irrigation	2	12
MowHeight	12	2
Clippings	8	6

```
### Create bar plot
```

```
Table = xtabs( ~ Response + Practice,
               data=Data)
```

```
Table
```

```
barplot(Table,
        beside = TRUE,
        legend = TRUE,
        ylim = c(0, 12),    ### y-axis: used to prevent legend overlapping bars
        cex.names = 0.8,    ### Text size for bars
        cex.axis = 0.8,     ### Text size for axis
        args.legend = list(x = "topright",   ### Legend location
                           cex = 0.8,          ### Legend text size
                           bty = "n"))         ### Remove legend box
```



Cochran's Q test

```
library(RVAideMemoire)
```

```
cochran.qtest(Response ~ Practice | Student,
               data = Data)
```

Cochran's Q test

Q = 16.9535, df = 3, p-value = 0.0007225

```
### Note that the function also gives you proportion of responses for the
### positive response, which is always the second of the nominal response,
### which in this case is "no", or the response coded as "1".
```

```
### I wouldn't use the post-hoc analysis included with the output of the
### function in RVAideMemoire
```

```
library(coin)
```

```
symmetry_test(Response ~ Practice | Student,
               data = Data,
               teststat = "quad")
```

Asymptotic General Symmetry Test

chi-squared = 16.953, df = 3, p-value = 0.0007225

Post-hoc analysis for Cochran's Q test

The *pairwiseMcNemar* function will conduct pairwise McNemar, binomial exact, or permutation tests analogous to uncorrected McNemar tests. The permutation tests require the *coin* package. As usual, *method* is the *p*-value adjustment method (see *?p.adjust* for options), and *digits* indicates the number of digits in the output. The *correct* option is used by the chi-square test function.

```
### Order groups
```

```
Data$Practice = factor(Data$Practice,
                      levels = c("MowHeight", "SoilTest",
                                "Clippings", "Irrigation"))
```

```
### Pairwise McNemar tests
```

```
library(rcompanion)
```

```
PT = pairwiseMcNemar(Response ~ Practice | Student,
                      data = Data,
                      test = "permutation",
                      method = "fdr",
                      digits = 3)
```

```
PT
```

	\$Pairwise	Comparison	p.value	p.adjust
1	MowHeight - SoilTest	= 0	0.317	0.3170

```

2 MowHeight - Clippings = 0  0.102  0.1530
3 MowHeight - Irrigation = 0 0.00389 0.0200
4 SoilTest - Clippings = 0  0.257  0.3080
5 SoilTest - Irrigation = 0 0.00666 0.0200
6 Clippings - Irrigation = 0  0.0143 0.0286

```

```
### Compact letter display
```

```
PT = PT$Pairwise
```

```
library(rcompanion)
```

```
cldList(p.adjust ~ Comparison,
        data      = PT,
        threshold = 0.05)
```

	Group	Letter	MonoLetter
1 MowHeight	a	a	
2 SoilTest	a	a	
3 Clippings	a	a	
4 Irrigation	b	b	

Table of results

	Proportion "Yes"	Grouping letter
MowHeight	0.86	a
SoilTest	0.79	a
Clippings	0.57	a
Irrigation	0.14	b

Optional analysis: Converting a matrix of p-values to a compact letter display

This compact letter display could also be generated with the *multcompView* package. To perform this well, it is helpful to first order the factors by the numeric order of their results, in this case by proportion of "yes" responses. Then the data need to be arranged into a matrix of pairwise p-values.

```

Input =""
Column    MowHeight  SoilTest  Clippings  Irrigation
MowHeight NA        0.317    0.153     0.0200
SoilTest   NA        NA       0.308     0.0200
Clippings NA        NA       NA        0.0286
Irrigation NA        NA       NA        NA
")

```

```
PT = as.matrix(read.table(textConnection(Input),
                           header=TRUE,
                           row.names=1))
```

```
PT
```

```
### Transform the upper matrix to a full matrix
```

```

diag(PT) = 1

PT1 = t(PT)

PT[lower.tri(PT)] = PT1[lower.tri(PT1)]

PT

### Produce compact letter display

library(multcompView)

multcompLetters(PT,
                compare="<",
                threshold=0.05, ### p-value to use as significance threshold
                Letters=letters,
                reversed = FALSE)

MowHeight  SoilTest  Clippings Irrigation
      "a"        "a"       "a"       "b"

```

Cochran's Q test example: short-format in matrix

```

Input =""
Student  SoilTest  Irrigation  MowHeight  Clippings
a        0         0          0          0
b        1         1          1          1
c        0         1          0          1
d        0         1          0          0
e        1         1          0          0
f        0         1          0          1
g        0         1          0          1
h        0         1          0          0
i        1         0          1          0
j        0         1          0          0
k        0         1          0          0
l        0         1          0          1
m        0         1          0          0
n        0         1          0          1
")
Matrix = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

### Add names to matrix dimensions that will be names of variables in long-format

names(dimnames(Matrix))[1] = "Student"
names(dimnames(Matrix))[2] = "Practice"

Matrix

```

```

### Convert matrix to long-format

library(reshape2)

Data = melt(Matrix)

### View data frame

library(psych)

headTail(Data)

  Student Practice value
1         a SoilTest   0
2         b SoilTest   1
3         c SoilTest   0
4         d SoilTest   0
...     <NA>    <NA> ...
53        k Clippings  0
54        l Clippings  1
55        m Clippings  0
56        n Clippings  1

### Note response variable is called "value"

```

Cochran's Q test

```

library(RVAideMemoire)

cochran.qtest(value ~ Practice | Student,
              data = Data)

Cochran's Q test

Q = 16.9535, df = 3, p-value = 0.0007225

library(coin)

symmetry_test(value ~ Practice | Student,
               data = Data,
               teststat = "quad")

Asymptotic General Symmetry Test

chi-squared = 16.953, df = 3, p-value = 0.0007225

```

Post-hoc analysis for Cochran's Q test

```

### Order groups

Data$Practice = factor(Data$Practice,
                       levels = c("MowHeight", "SoilTest",
                                  "Clippings", "Irrigation"))

### Pairwise McNemar tests

library(rcompanion)

pairwiseMcnemar(Response ~ Practice | Student,
                 data   = Data,
                 test   = "permutation",
                 method = "fdr",
                 digits = 3)

### test = "exact", "mcnemar", "permutation"
### method: p-value adjustment, see ?p.adjust
### correct: TRUE, to apply continuity correction for McNemar test

$Pairwise
      Comparison p.value p.adjust
1 MowHeight - SoilTest = 0  0.317  0.3170
2 MowHeight - Clippings = 0  0.102  0.1530
3 MowHeight - Irrigation = 0 0.00389  0.0200
4 SoilTest - Clippings = 0  0.257  0.3080
5 SoilTest - Irrigation = 0 0.00666  0.0200
6 Clippings - Irrigation = 0  0.0143  0.0286

```

Models for Nominal Data

Packages used in this chapter

The packages used in this chapter include:

- car
- multcompView
- emmeans
- VGAM
- lmtest
- psych
- lme4
- MASS
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(VGAM)){install.packages("VGAM")}
if(!require(lmtest)){install.packages("lmtest")}
if(!require(psych)){install.packages("psych")}
if(!require(lme4)){install.packages("lme4")}
if(!require(MASS)){install.packages("MASS")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Log-linear models for tests of association

Analysis of contingency tables with two or more dimensions can be accomplished with log-linear models. These will test for association. When there are multiple dimensions, this approach is sometimes called *multiway frequency analysis*. It is analogous to *chi-square* tests for two-way tables or Cochran–Mantel–Haenszel test for three way tables.

One advantage of log-linear models is that they handle tables with a higher number of dimensions.

Another advantage is that the models can be specified to test for more complicated patterns of association. When each term in the model is included, and there are no interactions, the model tests for *mutual independence* of the terms. Models can also test for *conditional independence* and *partial independence*.

For more information on specifying models for conditional independence or partial independence, see:

Quick-R. 2014. Frequencies and Crosstabs. www.statmethods.net/stats/frequencies.html.

Agresti, A. 2007. An Introduction to Categorical Data Analysis 2nd Edition. Wiley-Interscience.

For a more complete discussion of log-linear models for multiway frequency analysis see:

Tabachnick, B.G. and S.F. Fidell. 2001. Using multivariate statistics. 4th Edition. Allyn and Bacon, Boston, MA.

Zero frequencies in cells may cause the maximum likelihood estimation to fail, or may cause bias in the results.

Structural zeros can be handled with the *start* argument in the *loglm* function. See *library(MASS)*; *?loglm*, *library(MASS)*; *?loglm1*, and *?loglin*.

Log-linear model example

The following example revisits Alexander Anderson's data of passing grades by sex within counties, for which we had used the Cochran–Mantel–Haenszel test. Here we use a log-linear model with the *loglm* function in the *MASS* package.

```
Input = ("
```

```

County      Sex    Result  Count
Bloom       Female  Pass    9
Bloom       Female  Fail    5
Bloom       Male    Pass    7
Bloom       Male    Fail    17
Cobblestone Female  Pass    11
Cobblestone Female  Fail    4
Cobblestone Male   Pass    9
Cobblestone Male   Fail    21
Dougal      Female  Pass    9
Dougal      Female  Fail    7
Dougal      Male   Pass    19
Dougal      Male   Fail    9
Heimlich    Female  Pass    15
Heimlich    Female  Fail    8
Heimlich    Male   Pass    14
Heimlich    Male   Fail    17
")

```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
### Order factors otherwise R will alphabetize them
```

```
Data$County = factor(Data$County,
                      levels=unique(Data$County))
```

```
Data$Sex     = factor(Data$Sex,
                      levels=unique(Data$Sex))
```

```
Data$Result = factor(Data$Result,
                      levels=unique(Data$Result))
```

```
### Check the data frame
```

```
library(psych)
```

```
headTail(Data)
```

```
str(Data)
```

```
summary(Data)
```

```
### Remove unnecessary objects
```

```
rm(Input)
```

Create table

```
Table = xtabs(Count ~ Sex + Result + County,
              data=Data)
```

```
ftable(Table) # Display a flattened table
```

		County	Bloom	Cobblestone	Dougal	Heimlich
Sex	Result					
Female	Pass		9	11	9	15
	Fail		5	4	7	8
Male	Pass		7	9	19	14
	Fail		17	21	9	17

Log-linear model

The model here tests for mutual independence of the three variables.

```
library(MASS)

loglm(~ Sex + Result + County,
      Table)

Statistics:
          X^2  df  P(> X^2)
Likelihood Ratio 20.96662 10 0.0213275
Pearson          20.79050 10 0.0226027
```

Post-hoc analysis

For a post-hoc analysis, you could slice up a multi-dimensional table in any way that makes sense for the hypotheses you want to test. Here, we'll look for an association of *Sex* and *Result* within each county.

Third dimension in our table is *County*, so *Table[,1]* yields the piece of the table where the third dimension is equal to 1, that is, where *County* = *Bloom*. And so on.

```
Bloom = Table[, , 1]

loglm(~ Sex + Result,
      Bloom)

          X^2  df  P(> X^2)
Likelihood Ratio 4.504069 1 0.03381429
Pearson          4.473688 1 0.03442062
```

```
Cobblestone = Table[, , 2]

loglm(~ Sex + Result,
      Cobblestone)

          X^2  df  P(> X^2)
Likelihood Ratio 7.777229 1 0.005290890
Pearson          7.605000 1 0.005820666
```

```
Dougal = Table[, , 3]

loglm(~ Sex + Result,
      Dougal)

          X^2 df  P(> X^2)
Likelihood Ratio 0.5876125 1 0.4433438
Pearson         0.5927934 1 0.4413410

Heimlich = Table[, , 4]

loglm(~ Sex + Result,
      Heimlich)

          X^2 df  P(> X^2)
Likelihood Ratio 2.158817 1 0.1417538
Pearson         2.136182 1 0.1438595
```

Summary of analysis from Cochran–Mantel–Haenszel tests and log-linear models

County	C-H-M p-value	log-linear p-value (lr)
Bloom	0.0468	0.03381
Cobblestone	0.0102	0.00529
Dougal	0.5230	0.4433
Heimlich	0.1750	0.14175

Logistic regression

Logistic regression is a common option for building models with a nominal dependent variable. For standard logistic regression, the dependent variable must have only two levels. That is, it must be dichotomous.

The following references should be useful for conducting logistic regression.

“Simple Logistic Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_06.html.

“Multiple Logistic Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_07.html.

“Simple logistic regression” in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/simplelogistic.html.

“Multiple logistic regression” in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/multiplelogistic.html.

Logistic regression example

Note that model assumptions and pitfalls of logistic regression are not discussed here. The reader is urged to read the preceding references or other appropriate materials before proceeding.

The following example revisits Alexander Anderson pesticide safety training course data across sex and four counties, for which we had used the Cochran–Mantel–Haenszel Test.

```

Input = "
County      Sex    Result  Count
Bloom       Female  Pass    9
Bloom       Female  Fail    5
Bloom       Male    Pass    7
Bloom       Male    Fail    17
Cobblestone Female  Pass    11
Cobblestone Female  Fail    4
Cobblestone Male   Pass    9
Cobblestone Male   Fail    21
Dougal      Female  Pass    9
Dougal      Female  Fail    7
Dougal      Male   Pass    19
Dougal      Male   Fail    9
Heimlich    Female  Pass    15
Heimlich    Female  Fail    8
Heimlich    Male   Pass    14
Heimlich    Male   Fail    17
")
Data = read.table(textConnection(Input),header=TRUE)

### Order factors otherwise R will alphabetize them

Data$County = factor(Data$County,
                      levels=unique(Data$County))

Data$Sex     = factor(Data$Sex,
                      levels=unique(Data$Sex))

Data$Result = factor(Data$Result,
                      levels=unique(Data$Result))

### Check the data frame

library(psych)

headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects

```

```
rm(Input)
```

Logistic regression

Logistic regression can be conducted with the *glm* function in the native *stats* package. The *summary* function will return coefficients for the model. A *p*-value and a pseudo *R-squared* value for the model can be produced with the *nagelkerke* function. The *Anova* function in the *car* package will produce an analysis of deviance table with either likelihood ratio, Wald, or F tests.

```
model = glm(Result ~ County + Sex + County:Sex,
            weight = Count,
            data = Data,
            family = binomial(link="logit")
            )
```

```
summary(model)
```

```
library(rcompanion)
```

```
nagekerke(model)
```

```
$Pseudo.R.squared.for.model.vs.null
                           Pseudo.R.squared
McFadden                  0.0797857
Cox and Snell (ML)        0.7136520
Nagelkerke (Cragg and Uhler) 0.7136520

$Likelihood.ratio.test
 Df.diff LogLik.diff Chisq p.value
 -7      -10.004 20.009 0.0055508
```

```
library(car)
```

```
Anova(model,
      type="II",
      test="LR")
```

```
Analysis of Deviance Table (Type II tests)
```

	LR	Chisq	Df	Pr(>Chisq)
County	4.9627	3	0.174548	
Sex	7.8108	1	0.005193 **	
County:Sex	7.2169	3	0.065296 .	

Post-hoc analysis

Post-hoc analysis can be conducted with the *emmeans* package. Note that estimates for emmeans, differences, or standard errors are on a log or logit scale. For notes on using *emmeans* with different model types, see *?emmeans::models*.

For this example, no adjustment in *p*-values was made for multiple comparisons. Typically, an adjustment such as Tukey would be applied. See `?emmeans::summary` for *p*-value adjustment options in `emmeans`. Note that the adjustment should be applied for each output; that is, for each of `pairs` and `cld`.

Also note that normally, because the *County* \times *Sex* interaction was not significant, we wouldn't want to explore the mean separations of the *County* \times *Sex* interaction.

The results are truncated to emphasize the differences between sexes within counties.

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
~ County:Sex)

pairs(marginal,
adjust="none")      ### No adjustment for multiple comparisons

contrast           estimate       SE df z.ratio p.value
Bloom,Female - Bloom,Male      -1.47508986 0.7160935 NA -2.060  0.0394
Cobblestone,Female - Cobblestone,Male -1.85889877 0.7068488 NA -2.630  0.0085
Dougal,Female - Dougal,Male       0.49589997 0.6463052 NA  0.767  0.4429
Heimlich,Female - Heimlich,Male   -0.82276467 0.5673787 NA -1.450  0.1470

Results are given on the log (not the response) scale.
Tests are performed on the log scale
```

```
cld(marginal,
alpha=0.05,
Letters=letters, ### Use lower-case letters for .group
adjust="none")    ### No adjustment for multiple comparisons

County     Sex      emmean       SE df  asymp.LCL  asymp.UCL .group
Cobblestone Female -1.0116009 0.5838720 NA -2.155968949 0.13276713 a
Dougal      Male   -0.7472144 0.4046506 NA -1.540315071 0.04588627 a
Heimlich    Female -0.6286087 0.4377974 NA -1.486675732 0.22945841 a
Bloom       Female -0.5877867 0.5577733 NA -1.681002268 0.50542894 a
Dougal      Female -0.2513144 0.5039526 NA -1.239043434 0.73641458 ab
Heimlich    Male   0.1941560 0.3609046 NA -0.513203923 0.90151595 ab
Cobblestone Male   0.8472979 0.3984077 NA  0.066433148 1.62816257 b
Bloom       Male   0.8873032 0.4490867 NA  0.007109497 1.76749689 b

Results are given on the logit (not the response) scale.
Confidence level used: 0.95
Results are given on the log (not the response) scale.
Tests are performed on the log scale
```

significance level used: alpha = 0.05

Summary table of results

Comparison		p-value
Bloom-Female	- Bloom-Male	0.039
Cobblestone-Female	- Cobblestone-Male	0.0085
Dougal-Female	- Dougal-Male	0.44
Heimlich-Female	- Heimlich-Male	0.15

```
p.value = c(0.039, 0.0085, 0.44, 0.15)
```

```
p.adj = p.adjust(p.value,
method = "fdr")
```

```
p.adj = signif(p.adj,
2)
```

```
p.adj
```

```
[1] 0.078 0.034 0.440 0.200
```

Comparison		p-value	p.adj
Bloom-Female	- Bloom-Male	0.039	0.078
Cobblestone-Female	- Cobblestone-Male	0.0085	0.034
Dougal-Female	- Dougal-Male	0.44	0.44
Heimlich-Female	- Heimlich-Male	0.15	0.20

Multinomial logistic regression

If the nominal dependent variable has more than two levels, multinomial logistic regression can be used.

VGAM package

The *VGAM* package provides a flexible framework for building models with categorical data. The vignette for the package provides an overview of the package, as well as brief overview of categorical analysis in R.

Yee, T.W. The VGAM package for Categorical Data Analysis. The Comprehensive R Archive Network. cran.r-project.org/web/packages/VGAM/vignettes/categoricalVGAM.pdf.

The following article may also be helpful for those interested in using the VGAM package.

Yee, T.W. 2008. The VGAM Package. R News, 8(2), 28–39. URL. cran.r-project.org/doc/Rnews/Rnews_2008-2.pdf.

The *VGAM* package can be explored through the help files for the package.

```
if(!require(VGAM)){install.packages("VGAM")}

help(package="VGAM")
```

And multinomial regression is discussed specifically in the entry for *multinomial*.

```
library(VGAM)

?multinomial
```

mlogit and nnet packages

Two other packages that can perform multinomial regression are *mlogit* and *nnet*. The packages can be explored with the following vignette, article, and the package help files.

Croissant, Y. Estimation of multinomial logit models in R: The mlogit Packages. cran.r-project.org/web/packages/mlogit/vignettes/mlogit.pdf.

```
if(!require(mlogit)){install.packages("mlogit")}

help(package="mlogit")
```

[IDRE] Institute for Digital Research and Education. 2015. "R Data Analysis Examples: Multinomial Logistic Regression". UCLA. www.ats.ucla.edu/stat/r/dae/mlogit.htm.

```
if(!require(nnet)){install.packages("nnet")}

help(package="nnet")
```

Multinomial regression example

The following example revisits Alexander Anderson's data of passing grades by sex within counties, for which we had used the Cochran–Mantel–Haenszel Test.

Because the dependent variable, *Result*, has only two levels, it could be modeled with standard binomial regression. However, a multinomial approach with VGAM will be used here as an example. Note that in multinomial regression, a reference level for the dependent variable must be specified. Coefficients of the model will change if a different reference level is specified.

Note that model assumptions and pitfalls of this approach are not discussed here. The reader is urged to understand the assumptions of this kind of modeling before proceeding.

The *summary* function will provide estimates of coefficients and standard errors for the coefficients. A *p*-value for the overall model as well as pseudo *R-squared* value is provided by the *nagelkerke* function.

The *lrtest* function in *VGAM* can be used to compare two nested models with likelihood ratio test. If only one model is given, it provides a *p*-value for the overall model, compared with a null model. The *Anova* function in the *car* package will provide chi-square tests for individual factors in the model. The user might prefer to use likelihood ratio tests with nested models instead of chi-square tests. This

could be accomplished with the *lrtest* function in the *lmtest* package, or by specifying the second model with the *null=* option in the *nagelkerke* function.

```

Input = "
County      Sex    Result  Count
Bloom       Female  Pass    9
Bloom       Female  Fail    5
Bloom       Male   Pass    7
Bloom       Male   Fail    17
Cobblestone Female  Pass   11
Cobblestone Female  Fail   4
Cobblestone Male   Pass   9
Cobblestone Male   Fail   21
Dougal      Female  Pass   9
Dougal      Female  Fail   7
Dougal      Male   Pass   19
Dougal      Male   Fail   9
Heimlich    Female  Pass   15
Heimlich    Female  Fail   8
Heimlich    Male   Pass   14
Heimlich    Male   Fail   17
")

Data = read.table(textConnection(Input),header=TRUE)

### Order factors otherwise R will alphabetize them

Data$County = factor(Data$County,
                      levels=unique(Data$County))

Data$Sex     = factor(Data$Sex,
                      levels=unique(Data$Sex))

Data$Result = factor(Data$Result,
                      levels=unique(Data$Result))

### Check the data frame

library(psych)

headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects

rm(Input)

```

Multinomial regression

```
library(VGAM)

model = vglm(Result ~ Sex + County + Sex:County,
             family=multinomial(refLevel=1),
             weights = Count,
             data = Data)
```

```
summary(model)
```

```
library(car)
```

```
Anova(model,
      type="II",
      test="Chisq")
```

Analysis of Deviance Table (Type II tests)

	Response: Result	df	Chisq	Pr(>Chisq)
Sex	1	6.7132	0.00957	**
County	3	4.1947	0.24120	
Sex:County	3	7.1376	0.06764	.

```
library(rcompanion)
```

```
nagelkerke(model)
```

	\$Pseudo.R.squared.for.model1.vs.null	Pseudo.R.squared
McFadden		0.0797857
Cox and Snell (ML)		0.7136520
Nagelkerke (Cragg and Uhler)		0.7136520

	\$Likelihood.ratio.test		
Df.diff	LogLik.diff	Chisq	p.value
7	-10.004	20.009	0.0055508

```
library(lmtest)
```

```
lrtest(model)
```

Likelihood ratio test

Model 1: Result ~ Sex + County + Sex:County
Model 2: Result ~ 1

#Df	LogLik	Df	Chisq	Pr(>Chisq)
1	8	-115.39		
2	15	-125.39	7	20.009 0.005551 **

Post-hoc analysis

At the time of writing, the *emmeans* package cannot be used with *vglm* models.

One option for post-hoc analysis would be to conduct analyses on reduced models, including only two levels of a factor. For example, if the variable *County* x *Sex* term had been significant, the following code could be used to create a reduced dataset with only Bloom–Female and Bloom–Male, and analyze this data with *vglm*.

```
Data.b      = Data[Data$County=="Bloom" &
                  (Data$Sex=="Female" | Data$Sex=="Male") , ]

Data.b$County = factor(Data.b$County)
Data.b$Sex    = factor(Data.b$Sex)

summary(Data.b)

  County     Sex     Result     Count
Bloom:4 Female:2   Pass:2   Min.   : 5.0
           Male  :2    Fail:2   1st Qu.: 6.5
                           Median : 8.0
                           Mean   : 9.5
                           3rd Qu.:11.0
                           Max.   :17.0

library(VGAM)

model.b = vglm(Result ~ Sex,
               family=multinomial(refLevel=1),
               weights = Count,
               data = Data.b)

lrtest(model.b)

Likelihood ratio test

#DF  LogLik DF  Chisq Pr(>Chisq)
1    2 -23.612
2    3 -25.864  1 4.5041   0.03381 *
```

Summary table of results

Comparison		p-value
Bloom–Female	- Bloom–Male	0.034
Cobblestone–Female	- Cobblestone–Male	0.0052
Dougal–Female	- Dougal–Male	0.44
Heimlich–Female	- Heimlich–Male	0.14

```

p.value = c(0.034, 0.0052, 0.44, 0.14)

p.adj = p.adjust(p.value,
                  method = "fdr")

p.adj = signif(p.adj,
               2)

p.adj

[1] 0.068 0.021 0.440 0.190

Comparison          p-value  p.adj
Bloom-Female - Bloom-Male 0.034    0.068
Cobblestone-Female - Cobblestone-Male 0.0052   0.021
Dougal-Female - Dougal-Male 0.44     0.44
Heimlich-Female - Heimlich-Male 0.14     0.19

```

Mixed-effects logistic regression example

A mixed-effects generalized linear model, as in the case of logistic regression with random effects, can be specified. This example revisits Hayley Smith's friendly lawn care course, for which we had used Cochran's Q test.

This example uses the *glmer* function in the package *mle4*, which can fit binomial dependent variables, with the *binomial* family of models, or other families of models. As far as I know, it will not fit multinomial regression. For more information on families of models, see *?family* and *?glm*. For more information on *glmer*, see *?glmer*. For a list of methods used to extract information from the model object, see *methods(class="merMod")*.

The *Anova* function in the *car* package will provide chi-square tests for individual factors in the model. The user might prefer to use likelihood ratio tests with nested models instead of chi-square tests. This could be accomplished with the *anova* function, or by specifying the second model with the *null=* option in the *nagelkerke* function.

```

Input = "
Practice   Student  Response
SoilTest   a        Yes
SoilTest   b        No
SoilTest   c        Yes
SoilTest   d        Yes
SoilTest   e        No
SoilTest   f        Yes
SoilTest   g        Yes
SoilTest   h        Yes
SoilTest   i        No
SoilTest   j        Yes
SoilTest   k        Yes
SoilTest   l        Yes
SoilTest   m        Yes"

```

```

SoilTest    n      Yes
Irrigation  a      Yes
Irrigation  b      No
Irrigation  c      No
Irrigation  d      No
Irrigation  e      No
Irrigation  f      No
Irrigation  g      No
Irrigation  h      No
Irrigation  i      Yes
Irrigation  j      No
Irrigation  k      No
Irrigation  l      No
Irrigation  m      No
Irrigation  n      No
MowHeight   a      Yes
MowHeight   b      No
MowHeight   c      Yes
MowHeight   d      Yes
MowHeight   e      Yes
MowHeight   f      Yes
MowHeight   g      Yes
MowHeight   h      Yes
MowHeight   i      No
MowHeight   j      Yes
MowHeight   k      Yes
MowHeight   l      Yes
MowHeight   m      Yes
MowHeight   n      Yes
Clippings   a      Yes
Clippings   b      No
Clippings   c      No
Clippings   d      Yes
Clippings   e      Yes
Clippings   f      No
Clippings   g      No
Clippings   h      Yes
Clippings   i      Yes
Clippings   j      Yes
Clippings   k      Yes
Clippings   l      No
Clippings   m      Yes
Clippings   n      No
")

```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```
### Order factors otherwise R will alphabetize them
```

```
Data$Practice = factor(Data$Practice,
                      levels=unique(Data$Practice))
```

```
Data$Response = factor(Data$Response,
                      levels=c("Yes", "No"))
```

```
### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)
```

Mixed-effects logistic regression

```
library(lme4)

model = glmer(Response ~ Practice + (1 | Student),
               data = Data,
               family = binomial,
               nAGQ = 1)      ### Must be 1 to compare with glm model objects
```

```
summary(model)
```

```
library(car)
```

```
Anova(model,
      type="II",
      test="Chisq")
```

Analysis of Deviance Table (Type II Wald chisquare tests)

	Chisq	Df	Pr(>Chisq)
Practice	10.713	3	0.01338 *

```
### p-value for model and pseudo R-squared
```

```
model.null = glm(Response ~ 1,
                  data = Data,
                  family = binomial)
```

```
library(rcompanion)
```

```
nagelkerke(model,
            model.null)
```

\$Pseudo.R.squared.for.model.vs.null	Pseudo.R.squared
--------------------------------------	------------------

```

McFadden          0.258315
Cox and Snell (ML) 0.295184
Nagelkerke (Cragg and Uhler) 0.397900

$Likelihood.ratio.test
Df.diff LogLik.diff Chisq   p.value
-4      -9.7949 19.59 0.00060164

### Test the random effects

model.fixed = glm(Response ~ Practice,
                   data = Data,
                   family = binomial)

anova(model, model.fixed)

model.fixed: Response ~ Practice
model: Response ~ Practice + (1 | Student)

      Df     AIC     BIC logLik deviance chisq Chi Df Pr(>chisq)
model.fixed 4 64.636 72.738 -28.318    56.636
model       5 66.247 76.374 -28.124    56.247 0.3889      1      0.5329

```

Post-hoc analysis

Post-hoc analysis can be conducted with the *emmeans* package. Note that estimates for emmeans, differences, or standard errors are on a log or logit scale. For notes on using emmeans with different model types, see *?emmeans::models*.

```

library(multcompView)
library(emmeans)

marginal = emmeans(model,
                    ~ Practice)

pairs(marginal,
      adjust="tukey")

contrast           estimate      SE df z.ratio p.value
SoilTest - Irrigation -3.4004979 1.1888138 NA -2.860  0.0220
SoilTest - MowHeight  0.5256967 1.0384143 NA  0.506  0.9576
SoilTest - Clippings -1.1165227 0.9087443 NA -1.229  0.6086
Irrigation - MowHeight 3.9261946 1.2853129 NA  3.055  0.0121
Irrigation - Clippings 2.2839752 1.0431115 NA  2.190  0.1261
MowHeight - Clippings -1.6422194 1.0071891 NA -1.630  0.3614

```

Results are given on the log (not the response) scale.

P value adjustment: tukey method for comparing a family of 4 estimates
Tests are performed on the log scale

```
cld(marginal,
alpha=0.05,
Letters=letters, ### use lower-case letters for .group
adjust="tukey") ### Tukey adjustment for multiple comparisons
```

MowHeight	-1.9640077	0.8712261	NA	-4.1341580	0.2061426	a
SoilTest	-1.4383110	0.7498776	NA	-3.3061924	0.4295705	a
Clippings	-0.3217883	0.6055520	NA	-1.8301670	1.1865905	ab
Irrigation	1.9621870	0.8672121	NA	-0.1979648	4.1223388	b

Results are given on the logit (not the response) scale.
Confidence level used: 0.95

P value adjustment: tukey method for comparing a family of 4 estimates
Tests are performed on the log scale
significance level used: alpha = 0.05

Other tools for categorical analysis

The *vcd* package has several function that are useful when working with categorical data. The package can be explored with the following vignette and the package help files.

Friendly, M. Working with categorical data with R and the vcd and vcdExtra packages. The Comprehensive R Archive Network. cran.r-project.org/web/packages/vcdExtra/vignettes/vcd-tutorial.pdf.

```
if(!require(vcd)){install.packages("vcd")}
help(package="vcd")
```

Introduction to Parametric Tests

Packages used in this chapter

The packages used in this chapter include:

- psych
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}  
if(!require(rcompanion)){install.packages("rcompanion")}
```

When to use parametric tests

Parametric statistical tests are among the most common you'll encounter. They include *t*-test, analysis of variance, and linear regression.

They are used when the dependent variable is an interval/ratio data variable. This might include variables measured in science such as fish length, child height, crop yield weight, or pollutant concentration in water.

One advantage of using parametric statistical tests is that your audience will likely be familiar with the techniques and interpretation of the results. These tests are also often more flexible and more powerful than their nonparametric analogues.

Their major drawback is that all parametric tests assume something about the distribution of the underlying data. If these assumptions are violated, the resultant test statistics will not be valid, and the tests will not be as powerful as for cases when assumptions are met.

Count data may not be appropriate for common parametric tests

A frequent error is to use common parametric models and tests with count data for the dependent variable. Instead, count data could be analyzed either by using tests for nominal data or by using regression methods appropriate for count data. These include Poisson regression, negative binomial regression, and zero-inflated Poisson regression. See the *Regression for Count Data* chapter.

When to use parametric tests for count data

It is sometimes permissible to use common parametric tests for count data or other discrete data. They can be used in cases where counts are used as a type of measurement of some property of subjects, provided that 1) the distribution of data or residuals from the analysis approximately meet test assumptions; and 2) there are few or no counts at or close to zero, or close to a maximum, if one exists. Permissible examples might include test scores, age, or number of steps taken during the day. Technically, each of these measurements is bound by zero, and are discrete rather than continuous measurements. However, if other conditions are met, it is reasonable to handle them as if they were continuous measurement variables.

This kind of count data will sometimes need to be transformed to meet the assumptions of parametric analysis. Square root and logarithmic transformations are common. However, if there are many counts at or near zero, transformation is unlikely to help. It is usually not worth the effort to attempt to force count data to meet the assumptions of parametric analysis with transformations, since there are more appropriate methods available.

Percentage and proportion data

Percentage and proportion data are often inappropriate for parametric statistics for the same reasons given for count data. They will often not meet the assumptions of the tests, and are particularly problematic if there are some or many observations close to 0 or 1. The arcsine transformation was used traditionally, but logistic regression or beta regression may be more appropriate. However, if assumptions are approximately met, parametric analyses could be used.

Assumptions in parametric statistics

All parametric analyses have assumptions about the underlying data, and these assumptions should be confirmed or assumed with good reason when using these tests. If these assumptions are violated, the resulting statistics and conclusions will not be valid, and the tests may lack power relative to alternative tests.

The assumptions vary among tests, but a general discussion follows.

For examples in this section, we'll revisit the Catbus data.

```
Input = "
Student Sex Teacher Steps Rating
a female Catbus 8000 7
b female Catbus 9000 10
c female Catbus 10000 9
d female Catbus 7000 5
e female Catbus 6000 4
f female Catbus 8000 8
g male Catbus 7000 6
h male Catbus 5000 5
i male Catbus 9000 10
j male Catbus 7000 8
k female Satsuki 8000 7
l female Satsuki 9000 8
m female Satsuki 9000 8
n female Satsuki 8000 9
o male Satsuki 6000 5
p male Satsuki 8000 9
q male Satsuki 7000 6
r female Totoro 10000 10
s female Totoro 9000 10
t female Totoro 8000 8
u female Totoro 8000 7
v female Totoro 6000 7
w male Totoro 6000 8
x male Totoro 8000 10
```

```

y      male    Totoro    7000    7
z      male    Totoro    7000    7
")

Data = read.table(textConnection(Input), header=TRUE)

Data

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Random sampling

All statistical tests assume that the data captured in the sample are randomly chosen from the population as a whole. Selection bias will obviously affect the validity of the outcome of the analysis.

Independent observations

Tests will also assume that observations are independent of one another, except when the analysis takes non-independence into account. One common case of non-independent observations is in repeated measures experiments, in which the same subject is observed over time. If you were measuring test scores of students over time, you might expect students with a high test score on one date to have a high test score on subsequent dates. In this case the observation on one date would not be independent of observations on other dates.

The independence of observation is often assumed from good experimental design. Also, data or residuals can be plotted, for example to see if observations from one date are correlated to those for another date.

Normal distribution of data or residuals

Parametric tests assume that the data come from a population of known distribution. In particular, the tests discussed in this section assume that the distribution of the data are *conditionally* normal in distribution. That is, the data are normally distributed *once the effects of the variables in the model are taken into account*. Practically speaking, this means that the residuals from the analysis should be normally distributed. This will usually be assessed with a histogram of residuals, a density plot as shown below, or with quantile-quantile plot.

A select number of tests will require that data itself be normally distributed. This will be limited to one-sample t -test, two-sample t -test, and paired t -test. For other tests, the distribution of the residuals will be investigated.

Residuals from an analysis are also commonly called *errors*. They are the difference between the observations and the value predicted by the model. For example, if the calculated mean of a sample is 10, and one observation is 12, the residual for this observation is 2. If another observation is 7, the residual for this observation is -3.

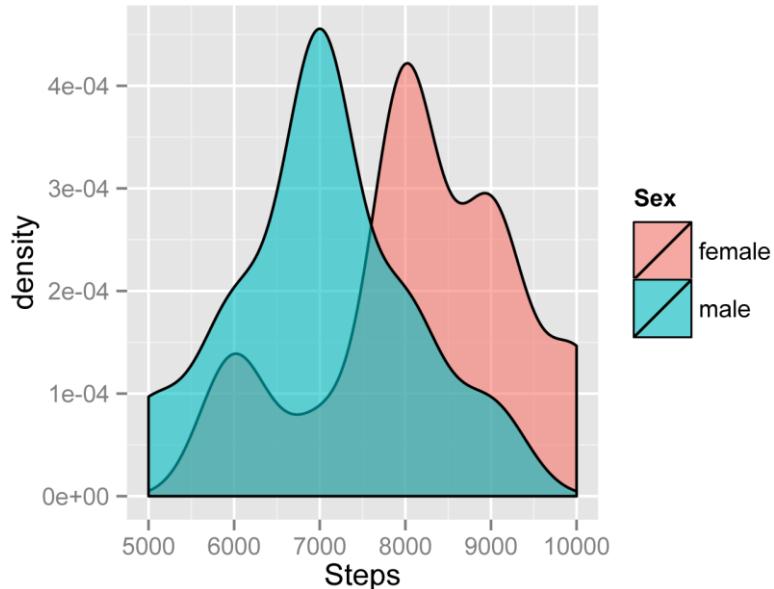
Be careful not to get confused about this assumption. You may see discussion about how “data” should be normally distributed for parametric tests. This is usually wrong-headed. The t -test assumes that the observations for *each group* are normally distributed, but if there is a difference in the groups, we might expect a bi-modal distribution, not a simple normal distribution, for the combined data. This is why in most cases we look at the distribution of the residuals, not the raw data.

Optional: Considering the distributions of data for groups and their residuals

The following code is just to illustrate this principle, and isn’t normally used in analysis. First, the distributions of *Steps* by *Sex* in the *Catbus* data set are examined, and then the distribution of residuals — group means subtracted from each observation — is examined.

```
### Density plot of observations by sex for Catbus data set
```

```
ggplot(Data,
       aes(Steps, fill = Sex)) +
  geom_density(position="dodge",
               alpha = 0.6)
```



The following code creates a variable *Mean* for *Steps* for each *Sex* in the *Catbus* data set, and then subtracts *Steps* from each mean, calling the result *Residual*.

```
M1 = mean(Data$Steps[Data$Sex=="female"])
```

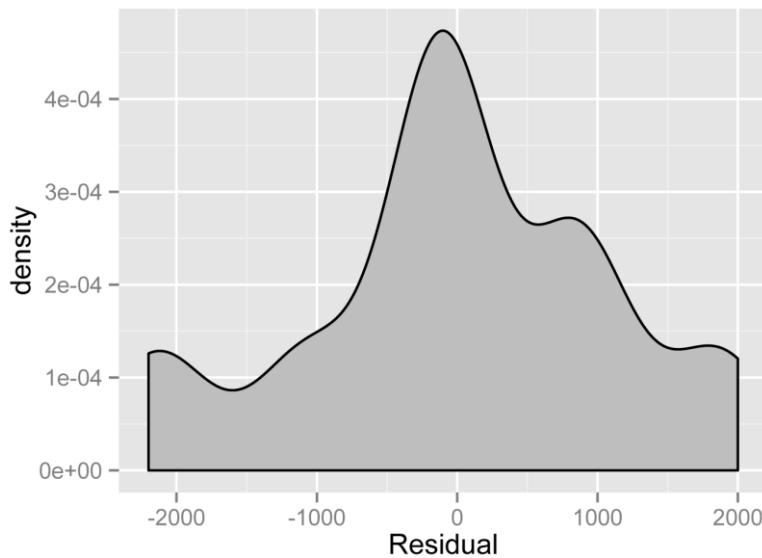
```
M2 = mean(Data$Steps[Data$Sex=="male"])

Data$Mean[Data$Sex=="female"] = M1
Data$Mean[Data$Sex=="male"] = M2

Data$Residual = Data$Steps - Data$Mean
```

Density plot of residuals for mean for each sex for the Catbus data set

```
ggplot(Data,
       aes(Residual)) +
  geom_density(fill = "gray")
```



Homogeneity of variance

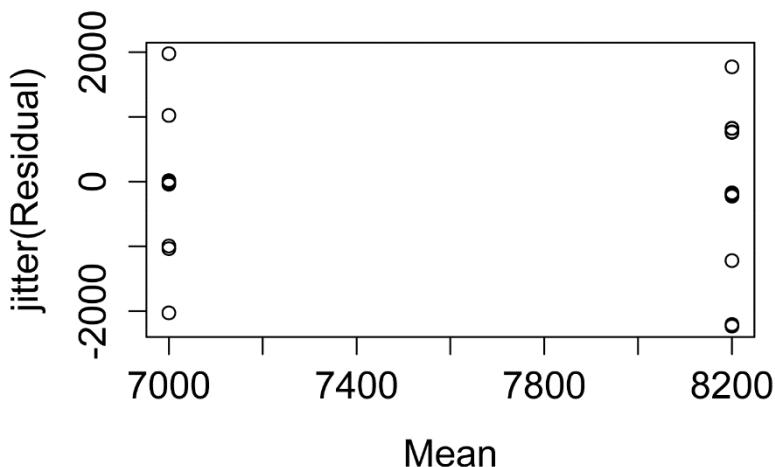
Parametric analyses will also assume a homogeneity of variance among groups. That is, for Student's *t*-test comparing two groups, each group should have the same variance.

A good approach to assess this assumption is to plot residuals vs. predicted values. The residuals should have approximately the same spread across all predicted values.

Homogeneity of variance is also called *homoscedasticity*. The opposite is *heteroscedasticity*.

```
### Residuals from mean for each sex vs. mean for Catbus data set

plot(jitter(Residual) ~ Mean,
     data = Data)
```



As a technical note, by default R conducts a variant of the t -test called Welch's t -test. This test does not assume homogeneity of variance and so can be used to compare two groups with unequal variances.

Additivity of treatment effects

Models for two-way analysis of variance and similar analyses are constructed as linear models in which the dependent variable is predicted as a linear combination of the independent variables.

A violation of this assumption is sometimes indicated when a plot of residuals versus predicted values exhibits a curved pattern.

Outliers

Outliers are observations whose value is far outside what is expected. They can play havoc with parametric analyses since they affect the distribution of the data and strongly influence the mean.

There are a variety of formal tests for detecting outliers, but they will not be discussed here. The best approach is one that looks at residuals after an analysis. Good tools are the “Residuals vs. leverage” plot and other plots in the “Other diagnostic plots” section below.

It's my opinion that outliers should not be removed from data unless there is a good reason, usually when a value is impossible or a measurement error of some kind is suspected.

Parametric tests are somewhat robust

Some parametric tests are somewhat robust to violations of certain assumptions. For example, the t -test is reasonably robust to violations of normality for symmetric distributions, but not to samples having unequal variances (unless Welch's t -test is used). A one-way analysis of variance is likewise reasonably robust to violations in normality.

The upshot is that model assumptions should always be checked, but you may be able to tolerate small violations in the distribution of residuals or homoscedasticity. Large violations will make the test invalid, though. It is important to be honest with your assessments when checking model

assumptions. It is better to transform data, change your model, use a robust method, or use a nonparametric test than to not have confidence in your analysis.

Assessing model assumptions

For this example, we'll revisit the Catbus data. We'll then define a linear model where *Steps* is the dependent variable and *Sex* and *Teacher* are the independent variables.

```
Input = "
Student  Sex    Teacher  Steps  Rating
a        female  Catbus   8000   7
b        female  Catbus   9000   10
c        female  Catbus  10000   9
d        female  Catbus   7000   5
e        female  Catbus   6000   4
f        female  Catbus   8000   8
g        male    Catbus   7000   6
h        male    Catbus   5000   5
i        male    Catbus   9000   10
j        male    Catbus   7000   8
k        female  Satsuki  8000   7
l        female  Satsuki  9000   8
m        female  Satsuki  9000   8
n        female  Satsuki  8000   9
o        male    Satsuki  6000   5
p        male    Satsuki  8000   9
q        male    Satsuki  7000   6
r        female  Totoro   10000  10
s        female  Totoro   9000   10
t        female  Totoro   8000   8
u        female  Totoro   8000   7
v        female  Totoro   6000   7
w        male    Totoro   6000   8
x        male    Totoro   8000   10
y        male    Totoro   7000   7
z        male    Totoro   7000   7
")

Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame

library(psych)

headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
```

```
rm(Input)
```

Define a linear model

```
model = lm(Steps ~ Sex + Teacher,  
          data = Data)
```

Using formal tests to assess normality of residuals

There are formal tests to assess the normality of residuals. Common tests include Shapiro-Wilk, Anderson-Darling, Kolmogorov-Smirnov, and D'Agostino-Pearson. These are presented in the "Optional analyses: formal tests for normality" section.

In general, I don't recommend using these tests because their results are dependent on sample size. When the sample size is large, the tests may indicate a statistically significant departure from normality, even if that departure is small. And when sample sizes are small, they won't detect departures from normality.

The article from the *Fells Stats* blog, in the "References" section has pretty convincing examples of these problems.

Skew and kurtosis

There are no definitive guidelines as to what range of skew or kurtosis are acceptable for considering residuals to be normally distributed.

In general, I would not recommend relying on skew and kurtosis calculations, but instead use histograms and other plots.

If I were forced to give advice for skewness calculations, I might say, be cautious if the absolute value is > 0.5 , and consider it not normally distributed if the absolute value is > 1.0 . Some authors use 2.0 as a cutoff for normality, and others use a higher limit for kurtosis.

```
x = residuals(model)  
  
library(psych)  
  
describe(x,  
         type=2)      # Type of skew and kurtosis
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis
1	1	26	0	1132.26	-39.58	11.07	1114.18	-2202.4	2123.25	4325.65	-0.13	-0.11

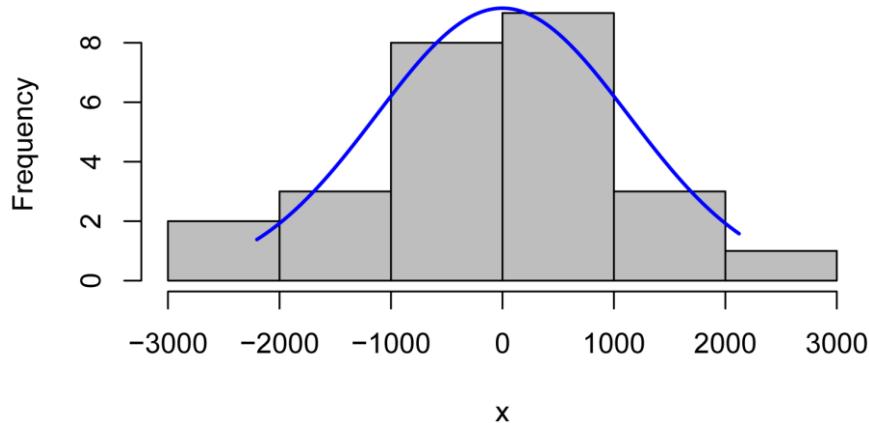
Using visual inspection to assess the normality of residuals

Usually, the best method to see if model residuals meet the assumptions of normal distribution and homoscedasticity are to plot them and inspect the plots visually.

Histogram with normal curve

A histogram of the residuals should be approximately normal, without excessive skew or kurtosis. Adding a normal curve with the same mean and standard deviation as the data helps to assess the histogram.

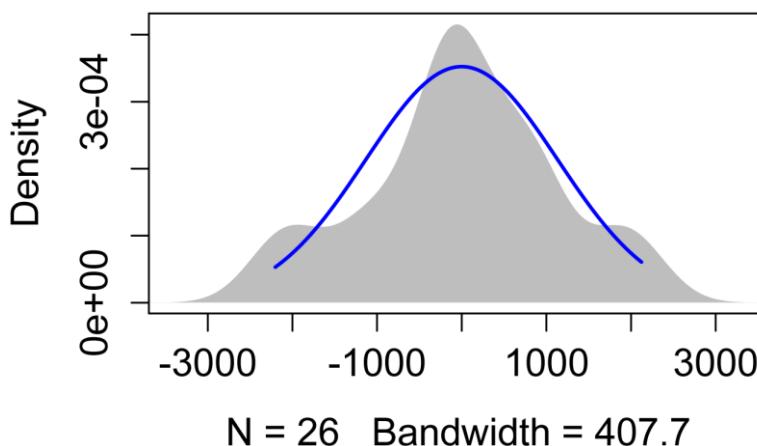
```
x = residuals(model)
library(rcompanion)
plotNormalHistogram(x)
```

Kernel density plot with normal curve

A kernel density plot is similar to a histogram, but is smoothed into a curve. Sometimes a density plot gives a better representation of the distribution of data, because the appearance of the histogram depends upon how many bins are used.

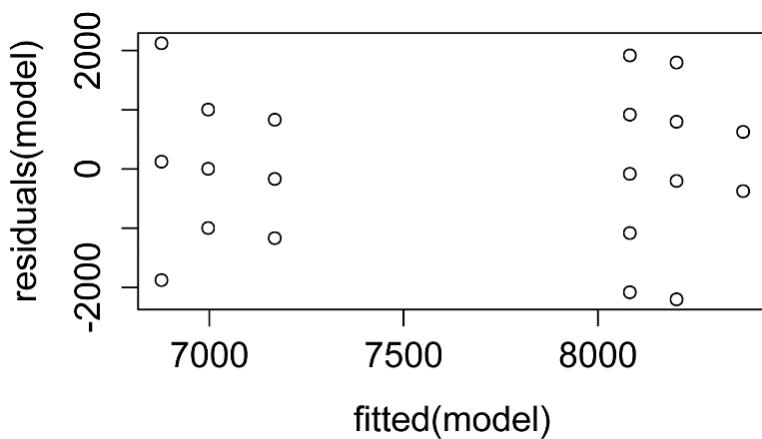
The *plotNormalDensity* function will produce this plot. Options include those for the *plot* function, as well as *adjust*, *bw*, and *kernel* which are passed to the *density* function. *col1*, *col2*, and *col3* change plot colors, and *lwd* changes line thickness.

```
x = residuals(model)
library(rcompanion)
plotNormalDensity(x,
                  adjust = 1)  ### Decrease this number
                  ### to make line less smooth
```

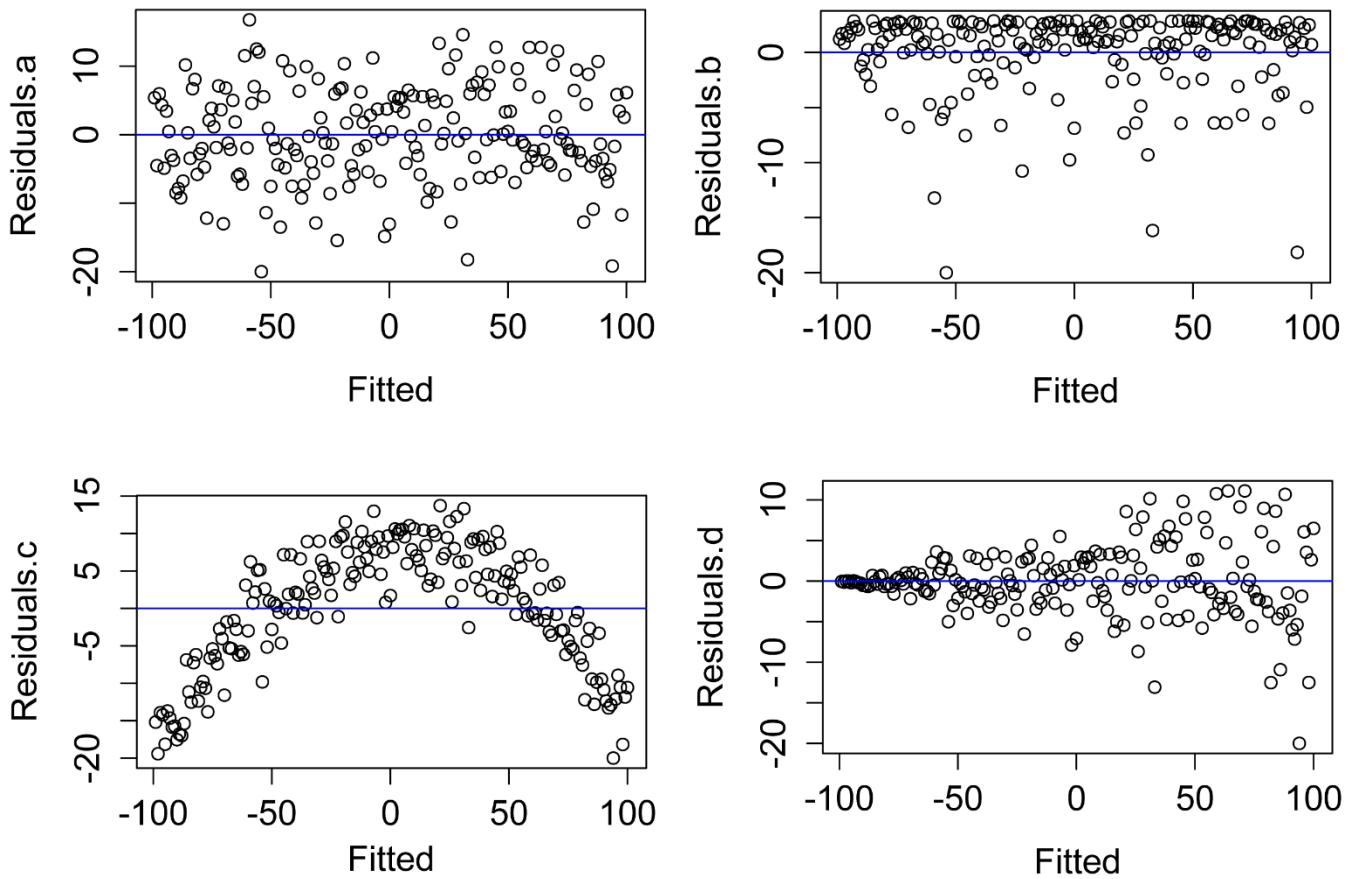
Plot of residuals vs. fitted values

Patterns in the plot of residuals versus fitted values can indicate a lack of homoscedasticity or that errors are not independent of fitted values.

```
plot(fitted(model),
      residuals(model))
```

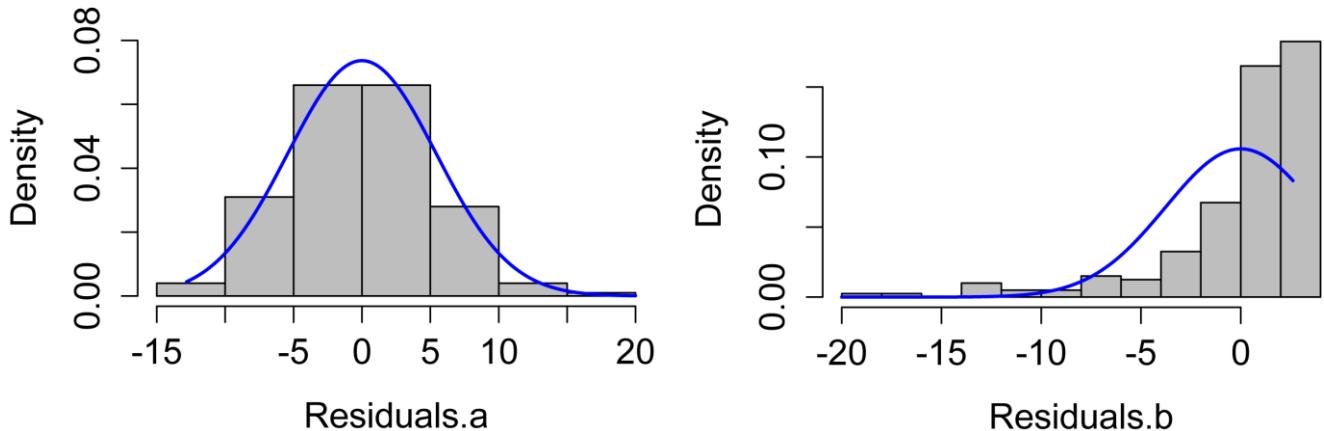
Examples of residual plots and histograms

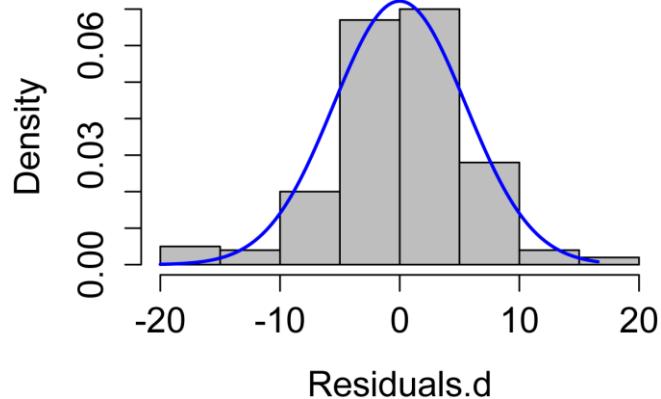
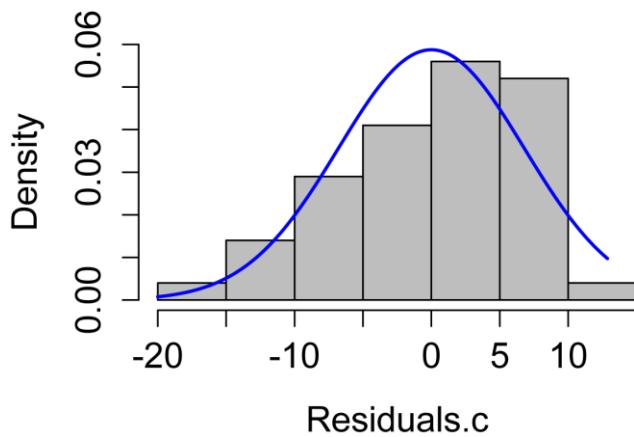
In the four plots below, A) *Residuals.a* show normally distributed and homoscedastic residuals, suggesting model assumptions were met. B) *Residuals.b* show a non-normal distribution of residuals. C) *Residuals.c* show that the residuals are not independent of the fitted values. In this case, the model needs to be modified in order to describe the data well. D) *Residuals.d* show heteroscedasticity, since variability in the residuals is greater for large fitted values than for small fitted values. (Adapted from similar plots in Tabachnick, 2001).



The following plots are histograms of the same residuals shown in the previous plots. A) *Residuals.a* are reasonably-close to normally distributed. B) *Residuals.b* are highly skewed (left, or negative). C) *Residuals.c* are moderately negatively skewed. This distribution would probably not cause too much havoc with most parametric tests, but, depending on the circumstances, I would probably try to transform a variable or find a better-fitting model. D) *Residuals.d* are symmetric, but leptokurtic. I probably wouldn't be too concerned with the distribution.

Results from the *describe* function in the *psych* package are shown below the histograms.





```
### describe(Residuals.a)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	200	0	6.82	0.63	0.07	6.91	-20	18.51	38.51	-0.14	0.03	0.48

```
### describe(Residuals.b)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	200	0	3.77	1.38	0.86	1.62	-20	2.62	22.62	-2.53	7.2	0.27

```
### describe(Residuals.c)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	200	0	6.79	1.25	0.48	7.75	-20	12.88	32.88	-0.54	-0.42	0.48

```
### describe(Residuals.d)
```

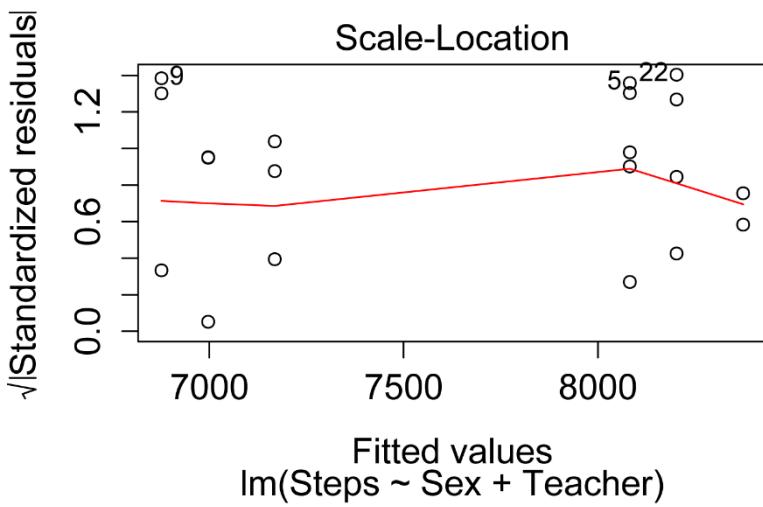
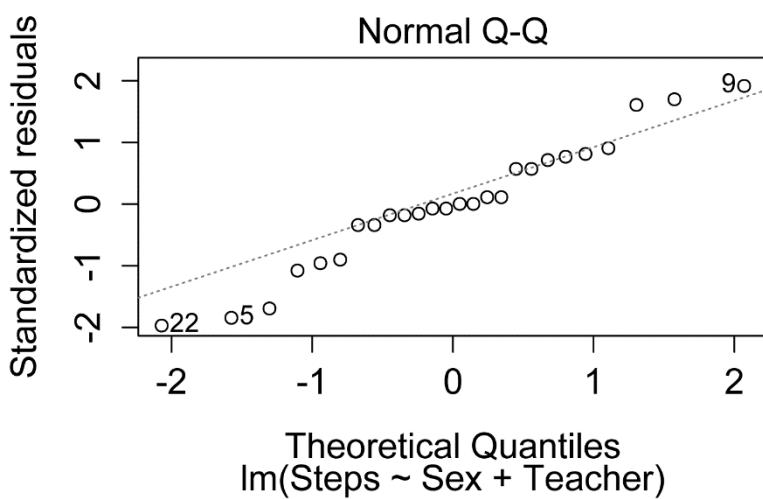
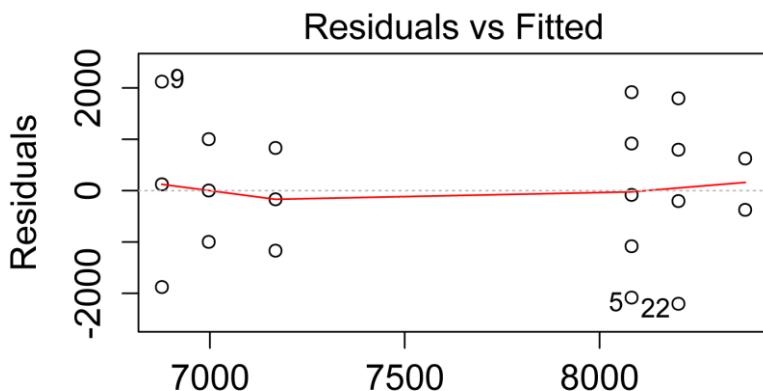
	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
1	1	200	0	5.53	0.11	0.21	3.83	-20	16.57	36.57	-0.48	1.94	0.39

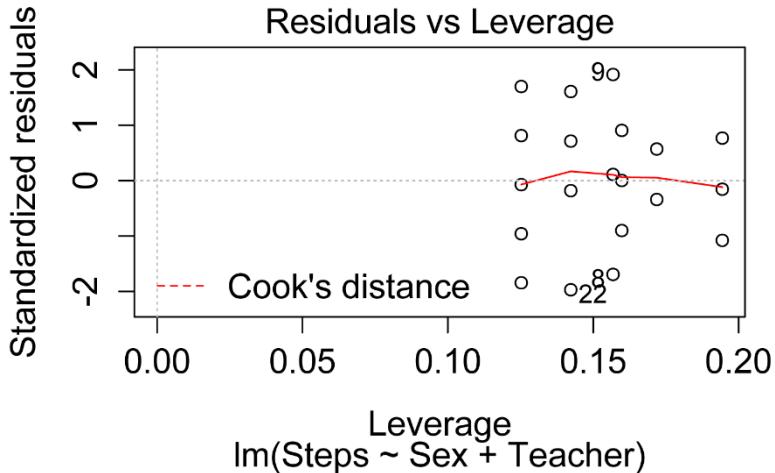
Other diagnostic plots

Using the *plot* function for most common types of models will show diagnostic plots or other useful plots for the model. For linear models, the *plot* function produces plots of 1) residuals vs. fitted values; 2) normal quantile-quantile plot; 3) square root of standardized residuals vs. fitted values; and 4) standardized residuals vs. leverage.

While these are useful diagnostic plots, their interpretation will not be discussed here.

```
plot(model)
```





Steps to handle violations of assumptions

If residuals show heteroscedasticity or a non-normal distribution, the analysis must be modified in some way to get valid results. Some options include the following.

1. Select the correct parametric test and check assumptions

2. Change your model

Problems with distribution of residuals is sometimes related to the model not describing the data well. Adding or removing terms from the model may correct this. As an example, imagine you had plotted bivariate data and they formed a shape like a parabola. If you fit a straight line to this data, you might get residuals that look like *Residuals.c* above. Adding an x^2 term, or other term to add curvature to the fitted model, might correct this. A similar situation can be encountered in analysis of variance and similar tests; it is sometimes helpful to add other terms or covariates.

3. Transform data and repeat your original analysis

The dependent variable, or other variables, can be transformed so that model assumptions are met. The tests are then performed on these transformed data. See the *Transforming Data* chapter for details.

4. Use nonparametric tests

Nonparametric tests include those discussed in the *Traditional Nonparametric Tests* section of this book and those chapters on *permutation tests*. Quantile regression and generalized additive model may be options as well.

5. Use robust methods

There are other statistical methods which are robust to violations of parametric assumptions or are nonparametric. See [Mangiafico \(2015a\)](#) and [\(2015b\)](#) in the “References” section for examples.

Descriptive Statistics for Parametric Statistics

Descriptive statistics for interval/ratio data are discussed in the “Descriptive statistics for interval/ratio data” section in the *Descriptive Statistics* chapter.

Descriptive plots for interval/ratio data are discussed in the “Examples of basic plots for interval/ratio and ordinal data” section in the *Basic Plots* chapter.

Optional readings

“**Normality**” in McDonald, J.H. 2014. *Handbook of Biological Statistics*.
www.biostathandbook.com/normality.html.

“**Independence**” in McDonald, J.H. 2014. *Handbook of Biological Statistics*.
www.biostathandbook.com/independence.html.

“**Homoscedasticity and heteroscedasticity**” in McDonald, J.H. 2014. *Handbook of Biological Statistics*.
www.biostathandbook.com/homoscedasticity.html.

References

“One-way Analysis with Permutation Test” in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_06a.html.

“Two-way Anova with Robust Estimation” in Mangiafico, S.S. 2015b. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_08a.html.

Tabachnick, B.G. and S.F. Fidell. 2001. *Using multivariate statistics*. 4th Edition. Allyn and Bacon, Boston, MA.

“Normality tests don't do what you think they do.” 2012. *Fells Stats*. blog.fellstat.com/?p=61.

Optional analyses: formal tests for normality

Code for conducting formal tests of normality are included here. I don't recommend using them, but they may be instructive in training the eye to interpret histograms and Q-Q plots, or to respond to a less-savvy reviewer.

In each case, the null hypothesis is that the data distribution is not different from normal. That is, a significant *p*-value (*p* < 0.05) suggests that data are not normally distributed.

As mentioned previously, a limitation to using these tests is that as the number of observations are increased, the ability of the test to return a significant *p*-value increases, even for small deviations from a normal distribution.

```
Input = "
Student  Sex      Teacher  Steps   Rating
a        female   Catbus    8000    7
b        female   Catbus    9000   10
c        female   Catbus   10000   9
d        female   Catbus    7000   5"
```

```

e      female  Catbus    6000    4
f      female  Catbus    8000    8
g      male   Catbus    7000    6
h      male   Catbus    5000    5
i      male   Catbus    9000   10
j      male   Catbus    7000    8
k      female Satsuki   8000    7
l      female Satsuki   9000    8
m      female Satsuki   9000    8
n      female Satsuki   8000    9
o      male   Satsuki   6000    5
p      male   Satsuki   8000    9
q      male   Satsuki   7000    6
r      female Totoro   10000   10
s      female Totoro   9000   10
t      female Totoro   8000    8
u      female Totoro   8000    7
v      female Totoro   6000    7
w      male   Totoro   6000    8
x      male   Totoro   8000   10
y      male   Totoro   7000    7
z      male   Totoro   7000    7
")

```

```
Data = read.table(textConnection(Input), header=TRUE)
```

```

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Define a linear model

```
model = lm(Steps ~ Sex + Teacher,
           data = Data)
```

Shapiro-Wilk normality test

```
x = residuals(model)

shapiro.test(x)
```

Shapiro-wilk normality test

w = 0.96256, p-value = 0.4444

Anderson-Darling normality test

```
if(!require(nortest)){install.packages("nortest")}

library(nortest)

x = residuals(model)

ad.test(x)

Anderson-Darling normality test

A = 0.39351, p-value = 0.3506
```

One-sample Kolmogorov-Smirnov test

```
x = residuals(model)

ks.test(x,
        "pnorm",
        mean = mean(x),
        sd   = sd(x))

One-sample Kolmogorov-Smirnov test

D = 0.1399, p-value = 0.6889
```

D'Agostino Normality Test

```
if(!require(fBasics)){install.packages("fBasics")}

library(fBasics)

x = residuals(model)

dagoTest(x)

Title:
D'Agostino Normality Test

Test Results:
STATISTIC:
Chi2 | Omnibus: 0.1071
Z3 | Skewness: -0.3132
Z4 | Kurtosis: 0.0948
P VALUE:
```

```
Omnibus Test: 0.9479
Skewness Test: 0.7541
Kurtosis Test: 0.9245
```

Optional analyses: formal tests for homogeneity of variance

Code for formal tests of homogeneity of variance among groups is presented here. I don't recommend using them, but instead recommend using diagnostic plots.

In each case, the null hypothesis is that the variance among groups is not different. That is, a significant p -value ($p < 0.05$) suggests that the variance among groups is different.

```
Input = "
Student Sex Teacher Steps Rating
a female Catbus 8000 7
b female Catbus 9000 10
c female Catbus 10000 9
d female Catbus 7000 5
e female Catbus 6000 4
f female Catbus 8000 8
g male Catbus 7000 6
h male Catbus 5000 5
i male Catbus 9000 10
j male Catbus 7000 8
k female Satsuki 8000 7
l female Satsuki 9000 8
m female Satsuki 9000 8
n female Satsuki 8000 9
o male Satsuki 6000 5
p male Satsuki 8000 9
q male Satsuki 7000 6
r female Totoro 10000 10
s female Totoro 9000 10
t female Totoro 8000 8
u female Totoro 8000 7
v female Totoro 6000 7
w male Totoro 6000 8
x male Totoro 8000 10
y male Totoro 7000 7
z male Totoro 7000 7
")

Data = read.table(textConnection(Input), header=TRUE)

### Check the data frame

library(psych)

headTail(Data)

str(Data)
```

```
summary(Data)

### Remove unnecessary objects

rm(Input)
```

Define a linear model

```
model1 = lm(Steps ~ Sex + Teacher,
            data = Data)
```

Bartlett's test for homogeneity of variance

Bartlett's test is known to be sensitive to non-normality in samples. That is, non-normal samples can result in a significant test due to the non-normality.

```
x = residuals(model1)

bartlett.test(x ~ interaction(Sex, Teacher),
              data=Data)

Bartlett test of homogeneity of variances

Bartlett's K-squared = 3.7499, df = 5, p-value = 0.586
```

Levene's test for homogeneity of variance

Levene's test is an alternative to Bartlett's that is supposedly less sensitive to departures from normality in the data.

```
if(!require(car)){install.packages("car")}

library(car)

x = residuals(model1)

leveneTest(x ~ Sex * Teacher
           data=Data,
           center=mean)      ### Use the original Levene's test

Levene's Test for Homogeneity of Variance (center = mean)

  Df F value Pr(>F)
group  5  0.4457 0.8113
      20
```

Brown-Forsythe or robust Levene's test

The Brown-Forsythe modification of Levene's test makes it more robust to departures in normality of the data.

```
if(!require(car)){install.packages("car")}

library(car)

x = residuals(model)

leveneTest(x ~ Sex * Teacher, data=Data)

Levene's Test for Homogeneity of Variance (center = median)

  Df F value Pr(>F)
group  5  0.4015  0.842
      20

if(!require(lawstat)){install.packages("lawstat")}

library(lawstat)

x = residuals(model)

levene.test(x, interaction(Data$Sex, Data$Teacher))

modified robust Brown-Forsythe Levene-type test based on the absolute deviations
from the median

Test Statistic = 0.4015, p-value = 0.842
```

Fligner-Killeen test

The Fligner-Killeen test is another test for homogeneity of variances that is robust to departures in normality of the data.

```
x = residuals(model)

fligner.test(x ~ interaction(Sex, Teacher), data=Data)

Fligner-Killeen test of homogeneity of variances

Fligner-Killeen:med chi-squared = 2.685, df = 5, p-value = 0.7484
```

One-sample t-test

One-sample tests are not used too often, but are useful to compare a set of values to a given default value. For example, one might ask if a set of student scores are significantly different from a “default” or “neutral” score of 75.

Appropriate data

- One-sample data
- Data are interval/ratio, and are continuous
- Data are normally distributed
- Moderate skewness is permissible if the data distribution is unimodal without outliers

Hypotheses

- Null hypothesis: The mean of the population from which the data were sampled is equal to the default value.
- Alternative hypothesis (two-sided): The mean the population from which the data were sampled is not equal to the default value.

Interpretation

Reporting significant results as, e.g., “Mean score for Variable A was significantly different from a default value of 75” is acceptable.

Other notes and alternative tests

- The nonparametric analogue for this test is the one-sample Wilcoxon signed-rank test.
- Power analysis for the one-sample *t*-test can be found at [Mangiafico \(2015\)](#) in the References section.

Packages used in this chapter

The packages used in this chapter include:

- psych
- rcompanion
- lsr
- ggplot2

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(lsr)){install.packages("lsr")}
if(!require(ggplot2)){install.packages("ggplot2")}
```

One-sample *t*-test example

In the following example, Brendon Small has his SNAP-Ed students keep diaries of what they eat for a week, and then calculate the daily sodium intake in milligrams. As a first step in the analysis, he wants to compare mean sodium intake by his students to the American Heart Association recommendation of 1500 mg.

A one-sample *t*-test can be conducted with the *t.test* function in the native *stats* package. Conveniently the output includes the mean of the sample, a confidence interval for that mean, and a *p*-value for the *t*-test.

We will use a histogram with an imposed normal curve to confirm data are approximately normal.

```
Input = "
Instructor      Student    Sodium
'Brendon Small'  a        1200
'Brendon Small'  b        1400
'Brendon Small'  c        1350
'Brendon Small'  d        950
'Brendon Small'  e        1400
'Brendon Small'  f        1150
'Brendon Small'  g        1300
'Brendon Small'  h        1325
'Brendon Small'  i        1425
'Brendon Small'  j        1500
'Brendon Small'  k        1250
'Brendon Small'  l        1150
'Brendon Small'  m        950
'Brendon Small'  n        1150
'Brendon Small'  o        1600
'Brendon Small'  p        1300
'Brendon Small'  q        1050
'Brendon Small'  r        1300
'Brendon Small'  s        1700
'Brendon Small'  t        1300
")
Data = read.table(textConnection(Input),header=TRUE)

### Check the data frame
library(psych)
headTail(Data)
str(Data)
summary(Data)

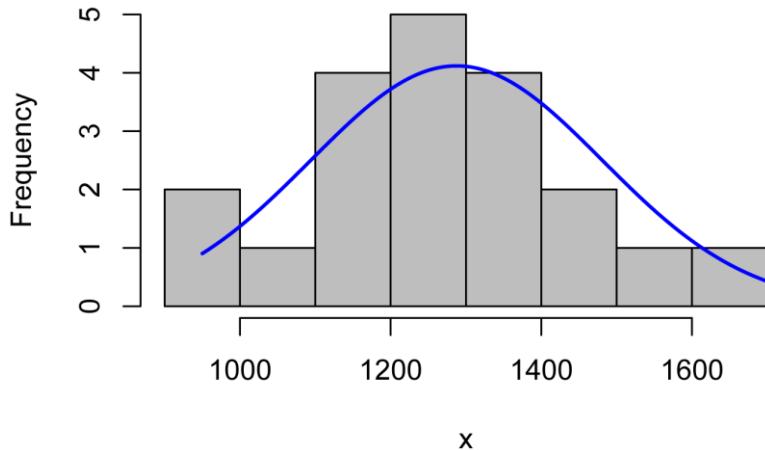
### Remove unnecessary objects
rm(Input)
```

Histogram of data

A histogram of the data can be examined to determine if the data are sufficiently normal.

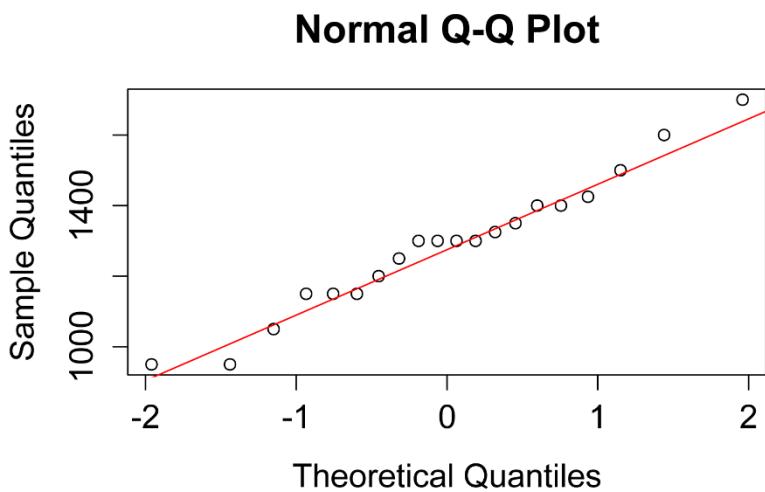
```
x = Data$Sodium
```

```
library(rcompanion)
plotNormalHistogram(x)
```



Normal quantile plot of data

```
x = Data$Sodium
qqnorm(x)
qqline(x, col="red")
```



One-sample t-test

```
t.test(Data$Sodium,
       mu = 1500)
```

One Sample t-test

```
t = -4.9053, df = 19, p-value = 9.825e-05
alternative hypothesis: true mean is not equal to 1500
95 percent confidence interval:
1196.83 1378.17
sample estimates:
mean of x
1287.5
```

Effect size

Cohen's d can be used as an effect size statistic for a one-sample t -test. It is calculated as the difference between the mean of the data and μ , the default value, all divided by the standard deviation of the data.

It ranges from 0 to infinity, with 0 indicating no effect where the mean equals μ . In some versions, Cohen's d can be positive or negative depending on whether the mean is greater than or less than μ .

A Cohen's d of 0.5 suggests that the mean and μ differ by one-half the standard deviation of the data. A Cohen's d of 1.0 suggests that the mean and μ differ by one standard deviation of the data.

Interpretation of Cohen's d

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment, but for behavioral studies, the guidelines proposed by Cohen (1988) are sometimes followed. They should not be considered universal.

	<u>Small</u>	<u>Medium</u>	<u>Large</u>
Cohen's d	0.2 – < 0.5	0.5 – < 0.8	≥ 0.8

Source: Cohen (1988).

Cohen's d for one-sample t-test

The `cohensD` function in the `lsr` package has an option for the one-sample t -test. The grammar follows that of the `t.test` function. Note that this function reports the value as a positive number.

```
library(ls)
cohensD(Data$Sodium,
        mu = 1500)

[1] 1.096864
```

It could also be calculated manually from the mean, μ , and standard deviation.

```
Mean = mean(Data$Sodium)
```

Mean

1287.5

Mu = 1500

Sd = sd(Data\$Sodium)

Diff = Mean - Mu

Diff

-212.5

CohenD = (Mean - Mu) / Sd

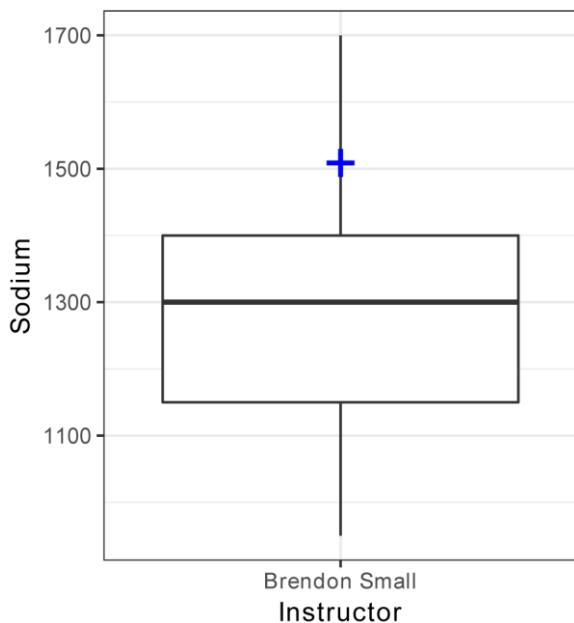
CohenD

[1] -1.096864

Box plot with default value

```
library(ggplot2)

ggplot(data=Data,
       aes(x = Instructor, y = Sodium)) +
  geom_boxplot() +
  geom_point(aes(x = 1, y = 1500),
             colour="blue",
             size = 8,
             shape = "+") +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  theme(axis.text = element_text(face = "bold"))
```



Box plot of daily sodium intake in milligrams recorded in student diaries. The blue cross indicates the American Heart Association recommendation of 1500 mg.

References

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.

"Student's t-test for One Sample" in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_01.html.

Exercises 0

1. Considering Brendon Small's data,

- What was the mean sodium intake?
- Is the data distribution reasonably normal?
- Was the mean sodium intake significantly different from the American Heart Association recommendation or 1500 mg per day?
- What do you conclude practically? Include a description of the difference between the mean of the data and the American Heart Association recommendation. If they're different, which is higher? Include effect size, any other relevant summary statistics, and your practical conclusions.

2. As part of a professional skills program, a 4-H club tests its members for typing proficiency. Dr. Katz wants to test his students' mean typing speed against a nominal speed of 40 words per minute.

Instructor	Student	words.per.minute
'Dr. Katz Professional Therapist'	a	35
'Dr. Katz Professional Therapist'	b	50
'Dr. Katz Professional Therapist'	c	55
'Dr. Katz Professional Therapist'	d	60
'Dr. Katz Professional Therapist'	e	65
'Dr. Katz Professional Therapist'	f	60
'Dr. Katz Professional Therapist'	g	70
'Dr. Katz Professional Therapist'	h	55
'Dr. Katz Professional Therapist'	i	45
'Dr. Katz Professional Therapist'	j	55
'Dr. Katz Professional Therapist'	k	60
'Dr. Katz Professional Therapist'	l	45
'Dr. Katz Professional Therapist'	m	65
'Dr. Katz Professional Therapist'	n	55
'Dr. Katz Professional Therapist'	o	50
'Dr. Katz Professional Therapist'	p	60

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- What was the mean typing speed?
- Is the data distribution reasonably normal?
- Was the mean typing speed significantly different from the nominal rate of 40 words per minute?
- What do you conclude practically? Include a description of the difference between the mean of the data and the 40 words per minute standard. If they're different, which is higher? Include effect size, any other relevant summary statistics, and your practical conclusions.

Two-sample t-test

The two-sample unpaired *t*-test is a commonly used test that compares the means of two samples.

Appropriate data

- Two-sample data. That is, one measurement variable in two groups or samples
- Dependent variable is interval/ratio, and is continuous
- Independent variable is a factor with two levels. That is, two groups
- Data for each population are normally distributed

- For Student's t -test, the two samples need to have the same variance. However, Welch's t -test, which is used by default in R, does not assume equal variances.
- Observations between groups are independent. That is, not paired or repeated measures data
- Moderate skewness is permissible if the data distribution is unimodal without outliers

Hypotheses

- Null hypothesis: The means of the populations from which the data were sampled for each group are equal.
- Alternative hypothesis (two-sided): The means of the populations from which the data were sampled for each group are not equal.

Interpretation

Reporting significant results as "Mean of variable Y for group A was different than that for group B." is acceptable.

Other notes and alternative tests

- The nonparametric analogue for this test is the two-sample Mann–Whitney U Test. Another option is to use a permutation test. See [Mangiafico \(2015b\)](#) in the "References" section.
- Power analysis for the two-sample t -test can be found at [Mangiafico \(2015a\)](#) in the "References" section.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- lattice
- lsr

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(lattice)){install.packages("lattice")}
if(!require(lsr)){install.packages("lsr")}
```

Two-sample t -test example

In the following example, Brendon Small and Coach McGuirk have their SNAP-Ed students keep diaries of what they eat for a week, and then calculate the daily sodium intake in milligrams. Since the classes have received different nutrition education programs, they want to see if the mean sodium intake is the same for both classes.

A two-sample t -test can be conducted with the `t.test` function in the native `stats` package. The default is to use Welch's t -test, which doesn't require equal variance between groups. Conveniently the output

includes the mean of each sample, a confidence interval for the difference in means, and a p -value for the t -test.

We will use histograms with imposed normal curves to confirm data are approximately normal.

```
Input = "
Instructor      Student   Sodium
'Brendon Small'  a        1200
'Brendon Small'  b        1400
'Brendon Small'  c        1350
'Brendon Small'  d        950
'Brendon Small'  e        1400
'Brendon Small'  f        1150
'Brendon Small'  g        1300
'Brendon Small'  h        1325
'Brendon Small'  i        1425
'Brendon Small'  j        1500
'Brendon Small'  k        1250
'Brendon Small'  l        1150
'Brendon Small'  m        950
'Brendon Small'  n        1150
'Brendon Small'  o        1600
'Brendon Small'  p        1300
'Brendon Small'  q        1050
'Brendon Small'  r        1300
'Brendon Small'  s        1700
'Brendon Small'  t        1300
'Coach McGuirk' u        1100
'Coach McGuirk' v        1200
'Coach McGuirk' w        1250
'Coach McGuirk' x        1050
'Coach McGuirk' y        1200
'Coach McGuirk' z        1250
'Coach McGuirk' aa       1350
'Coach McGuirk' ab       1350
'Coach McGuirk' ac       1325
'Coach McGuirk' ad       1525
'Coach McGuirk' ae       1225
'Coach McGuirk' af       1125
'Coach McGuirk' ag       1000
'Coach McGuirk' ah       1125
'Coach McGuirk' ai       1400
'Coach McGuirk' aj       1200
'Coach McGuirk' ak       1150
'Coach McGuirk' al       1400
'Coach McGuirk' am       1500
'Coach McGuirk' an       1200
")

Data = read.table(textConnection(Input), header=TRUE)

### Check the data frame
```

```
library(psych)
headTail(Data)
str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)
```

Summarize data by group

```
library(FSA)

Summarize(Sodium ~ Instructor,
          data=Data,
          digits=3)

Instructor   n  nvalid      mean       sd   min    Q1 median    Q3   max percZero
1 Brendon Small 20      20 1287.50 193.734  950 1150    1300 1400 1700      0
2 Coach McGuirk 20      20 1246.25 142.412 1000 1144    1212 1350 1525      0
```

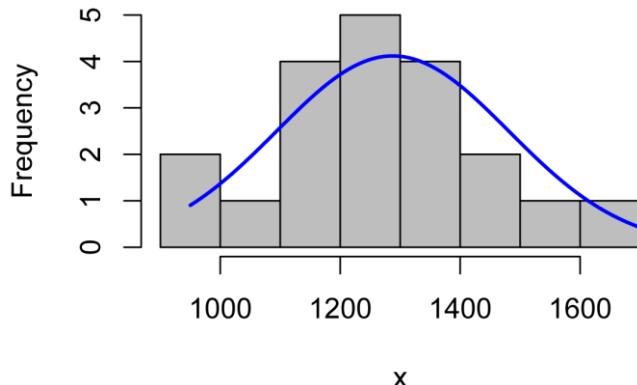
Histograms for data by group

Histograms of each group could be examined.

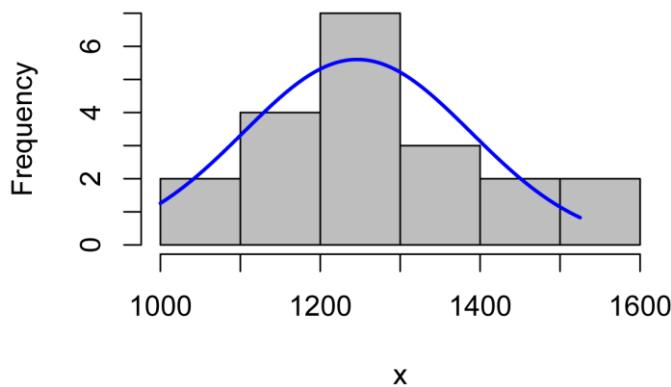
The *right = FALSE* option is used with the McGuirk data to improve the appearance of the plot in this particular case.

```
Brendon = Data$Sodium[Data$Instructor == "Brendon Small"]
McGuirk = Data$Sodium[Data$Instructor == "Coach McGuirk"]

library(rcompanion)
plotNormalHistogram(Brendon)
```



```
plotNormalHistogram(McGuirk,
                    right = FALSE)
```



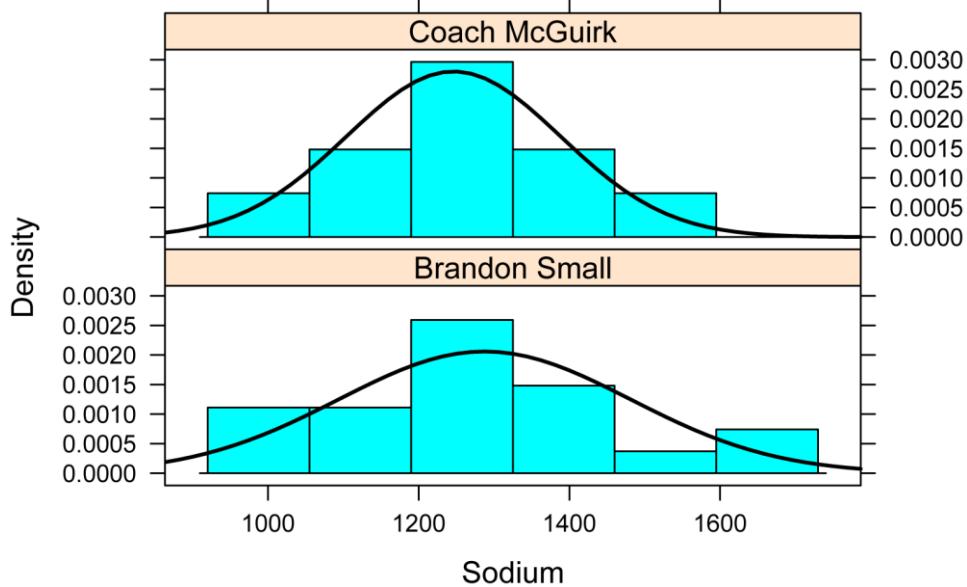
The *lattice* package could also be used, but adding normal curves to histograms by groups requires some extra code.

```
library(lattice)

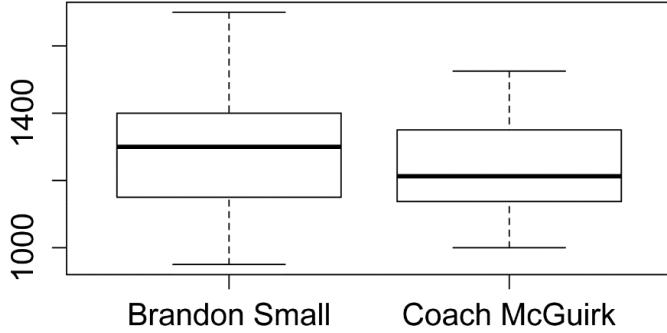
histogram(~ Sodium | Instructor,
          data    = Data,
          type    = "density",
          layout  = c(1,2),           ### columns and rows of individual plots

          panel=function(x, ...) {
            panel.histogram(x, ...)

            panel.mathdensity(dmath = dnorm,
                               col    = "blue",
                               lwd   = 2,
                               args  = list(mean=mean(x),
                                           sd=sd(x)), ...)})
```

***Box plots for data by group***

```
boxplot(Sodium ~ Instructor,
        data = Data)
```

***Two-sample unpaired t-test***

```
t.test(Sodium ~ Instructor,
       data = Data)

Welch Two Sample t-test

data: Sodium by Instructor
t = 0.76722, df = 34.893, p-value = 0.4481

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:
-67.91132 150.41132
```

```
sample estimates:
mean in group Brandon Small mean in group Coach McGuirk
1287.50                1246.25
```

Effect size

Cohen's d can be used as an effect size statistic for a two-sample t -test. It is calculated as the difference between the means of each group, all divided by the pooled standard deviation of the data.

It ranges from 0 to infinity, with 0 indicating no effect where the means are equal. In some versions, Cohen's d can be positive or negative depending on which mean is greater.

A Cohen's d of 0.5 suggests that the means differ by one-half the standard deviation of the data. A Cohen's d of 1.0 suggests that the means differ by one standard deviation of the data.

Interpretation of Cohen's d

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment, but for behavioral studies, the guidelines proposed by Cohen (1988) are sometimes followed. They should not be considered universal.

	<u>Small</u>	<u>Medium</u>	<u>Large</u>
Cohen's d	0.2 – < 0.5	0.5 – < 0.8	≥ 0.8

Source: Cohen (1988).

Cohen's d for two-sample t-test

The grammar in the `cohensD` function in the `lsr` package follows that of the `t.test` function. Note that this function reports the value as a positive number.

```
library(ls)
cohensD(Sodium ~ Instructor,
        data = Data)

[1] 0.2426174
```

Optional readings

"Student's t-test for two samples" in McDonald, J.H. 2014. *Handbook of Biological Statistics*. www.biostathandbook.com/twosampletest.html.

References

"Student's t-test for Two Samples" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_02.html.

"Mann-Whitney and Two-sample Permutation Test" in Mangiafico, S.S. 2015b. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_02a.html.

Exercises P

1. Considering Brendon and McGuirk's data,

- a. What was the mean sodium intake for each class?
- b. Are the data distributions for each sample reasonably normal?
- c. Was the mean sodium intake significantly different between classes?
- d. What do you conclude practically? Include a description of the difference between the means of the data. If they're different, which is higher? Include effect size, any other relevant summary statistics, and your practical conclusions.

2. As part of a professional skills program, a 4-H club tests its members for typing proficiency. Dr. Katz and Laura want to compare their students' mean typing speed between their classes.

Instructor	Student	words.per.minute
'Dr. Katz Professional Therapist'	a	35
'Dr. Katz Professional Therapist'	b	50
'Dr. Katz Professional Therapist'	c	55
'Dr. Katz Professional Therapist'	d	60
'Dr. Katz Professional Therapist'	e	65
'Dr. Katz Professional Therapist'	f	60
'Dr. Katz Professional Therapist'	g	70
'Dr. Katz Professional Therapist'	h	55
'Dr. Katz Professional Therapist'	i	45
'Dr. Katz Professional Therapist'	j	55
'Dr. Katz Professional Therapist'	k	60
'Dr. Katz Professional Therapist'	l	45
'Dr. Katz Professional Therapist'	m	65
'Dr. Katz Professional Therapist'	n	55
'Dr. Katz Professional Therapist'	o	50
'Dr. Katz Professional Therapist'	p	60
'Laura the Receptionist'	q	55
'Laura the Receptionist'	r	60
'Laura the Receptionist'	s	75
'Laura the Receptionist'	t	65
'Laura the Receptionist'	u	60
'Laura the Receptionist'	v	70
'Laura the Receptionist'	w	75
'Laura the Receptionist'	x	70
'Laura the Receptionist'	y	65
'Laura the Receptionist'	z	72
'Laura the Receptionist'	aa	73
'Laura the Receptionist'	ab	65

'Laura the Receptionist'	ac	80
'Laura the Receptionist'	ad	50
'Laura the Receptionist'	ae	55
'Laura the Receptionist'	af	70

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- What was the mean typing speed for each class?
- Are the data distributions for each sample reasonably normal?
- Was the mean typing speed significantly different between the classes?
- What do you conclude practically? Include a description of the difference between the means of the data. If they're different, which is higher? Include effect size, any other relevant summary statistics, and your practical conclusions.

Paired t-test

The paired *t*-test is commonly used. It compares the means of two populations of paired observations by testing if the difference between pairs is statistically different from zero.

Appropriate data

- Two-sample data. That is, one measurement variable in two groups or samples
- Dependent variable is interval/ratio, and is continuous
- Independent variable is a factor with two levels. That is, two groups
- Data are paired. That is, the measurement for each observation in one group can be paired logically or by subject to a measurement in the other group
- The distribution of the difference of paired measurements is normally distributed
- Moderate skewness is permissible if the data distribution is unimodal without outliers

Hypotheses

- Null hypothesis: The population mean of the differences between paired observations is equal to zero.
- Alternative hypothesis (two-sided): The population mean of the differences between paired observations is not equal to zero.

Interpretation

Reporting significant results as "Mean of variable Y for group A was different than that for group B." or "Variable Y increased from before to after" is acceptable.

Other notes and alternative tests

- The nonparametric analogue for this test is the two-sample paired rank-sum test.

- Power analysis for the paired *t*-test can be found at [Mangiafico \(2015\)](#) in the “References” section.

Packages used in this chapter

The packages used in this chapter include:

- psych
- rcompanion
- lsr

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(lsr)){install.packages("lsr")}
```

Paired *t*-test example

In the following example, Dumbländ Extension had adult students fill out a financial literacy knowledge questionnaire both before and after completing a home financial management workshop. Each student’s score before and after was paired by student.

Note in the following data that the students’ names are repeated, so that there is a before score for student *a* and an after score for student *a*.

Since the data is in long form, we’ll order by *Time*, then *Student* to be sure the first observation for *Before* is student *a* and the first observation for *After* is student *a*, and so on.

```
Input = "
Time   Student  Score
Before  a       65
Before  b       75
Before  c       86
Before  d       69
Before  e       60
Before  f       81
Before  g       88
Before  h       53
Before  i       75
Before  j       73
After   a       77
After   b       98
After   c       92
After   d       77
After   e       65
After   f       77
After   g       100
After   h       73
After   i       93"
```

```

After    j      75
")

Data = read.table(textConnection(Input),header=TRUE)

### Order data by Time and Student

Data = Data[order(Time, Student),]

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Check the number of paired observations

It is helpful to create a table of the counts of observations to be sure that there is one observation for each student for each time period.

```
xtabs(~ Student + Time,
      data = Data)
```

Student	Time	
	After	Before
a	1	1
b	1	1
c	1	1
d	1	1
e	1	1
f	1	1
g	1	1
h	1	1
i	1	1
j	1	1

Histogram of difference data

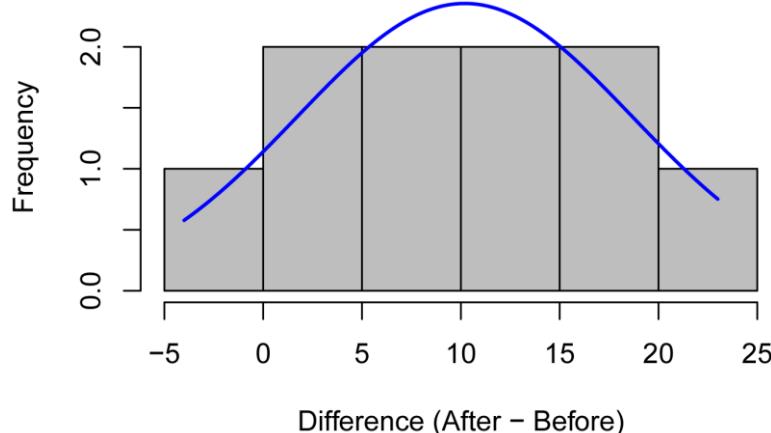
A histogram with a normal curve imposed will be used to check if the paired differences between the two populations is approximately normal in distribution.

First, two new variables, *Before* and *After*, are created by extracting the values of *Score* for observations with the *Time* variable equal to *Before* or *After*, respectively.

Note that for this code to make sense, the first observation for *Before* is student *a* and the first observation for *After* is student *a*, and so on.

```
Before = Data$Score[Data$Time=="Before"]
After = Data$Score[Data$Time=="After"]
Difference = After - Before

x = Difference
library(rcompanion)
plotNormalHistogram(x,
                    xlab="Difference (After - Before)")
```



Plot the paired data

Scatter plot with one-to-one line

Paired data can be visualized with a scatter plot of the paired cases. In the plot below, points that fall above and to the left of the blue line indicate cases for which the value for *After* was greater than for *Before*.

Note that the points in the plot are jittered slightly so that points that would fall directly on top of one another can be seen.

First, two new variables, *Before* and *After*, are created by extracting the values of *Score* for observations with the *Time* variable equal to *Before* or *After*, respectively.

Note that for this code to make sense, the first observation for *Before* is student *a* and the first observation for *After* is student *a*, and so on.

A variable *Names* is also created for point labels.

```
Before = Data$Score[Data$Time=="Before"]

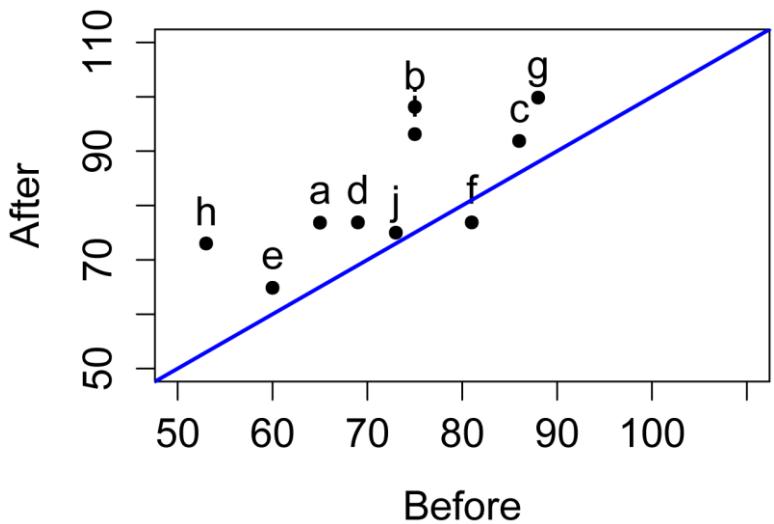
After = Data$Score[Data$Time=="After"]

Names = Data$Student[Data$Time=="Before"]

plot(Before, jitter(After),      # jitter offsets points so you can see them all
     pch = 16,                  # shape of points
     cex = 1.0,                  # size of points
     xlim=c(50, 110),            # limits of x-axis
     ylim=c(50, 110),            # limits of y-axis
     xlab="Before",              # label for x-axis
     ylab="After")               # label for y-axis

text(Before, After, labels=Names, # Label location and text
     pos=3, cex=1.0)             # Label text position and size

abline(0, 1, col="blue", lwd=2)    # line with intercept of 0 and slope of 1
```



Bar plot of differences

Paired data can also be visualized with a bar chart of differences. In the plot below, bars with a value greater than zero indicate cases for which the value for *After* was greater than for *Before*.

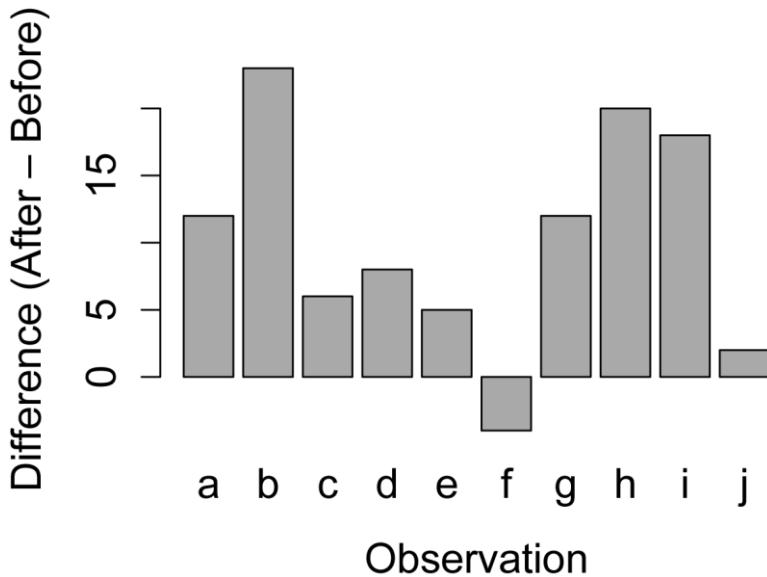
New variables are first created for *Before*, *After*, and their *Difference*.

Note that for this code to make sense, the first observation for *Before* is student *a* and the first observation for *After* is student *a*, and so on.

A variable *Names* is also created for bar labels.

```
Before = Data$Score[Data$Time=="Before"]
After = Data$Score[Data$Time=="After"]
Difference = After - Before
Names = Data$Student[Data$Time=="Before"]

barplot(Difference,
        col="dark gray",
        xlab="Observation",
        ylab="Difference (After - Before)",
        names.arg=Names) # variable to plot
# color of bars
# x-axis label
# y-axis label
# labels for bars
```



Paired t-test

Note that the output shows the *p*-value for the test, and the simple difference in the means for the two groups.

Note that for this test to be conducted correctly, the first observation for *Before* is student *a* and the first observation for *After* is student *a*, and so on.

```
t.test(Score ~ Time,
       data = Data,
       paired = TRUE)

Paired t-test

t = 3.8084, df = 9, p-value = 0.004163
alternative hypothesis: true difference in means is not equal to 0
```

95 percent confidence interval:
4.141247 16.258753

sample estimates:
mean of the differences
10.2

Effect size

Cohen's d can be used as an effect size statistic for a paired t -test. It is calculated as the difference between the means of each group, all divided by the standard deviation of the data. The standard deviation used could be calculated from the differences between observations, or, for observations across two times, the observations in the "before" group.

It ranges from 0 to infinity, with 0 indicating no effect where the means are equal. In some versions, Cohen's d can be positive or negative depending on which mean is greater.

A Cohen's d of 0.5 suggests that the means differ by one-half the standard deviation of the data. A Cohen's d of 1.0 suggests that the means differ by one standard deviation of the data.

Interpretation of Cohen's d

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment, but for behavioral studies, the guidelines proposed by Cohen (1988) are sometimes followed. They should not be considered universal.

	<u>Small</u>	<u>Medium</u>	<u>Large</u>
Cohen's d	0.2 – < 0.5	0.5 – < 0.8	≥ 0.8

Source: Cohen (1988).

Cohen's d for paired t-test

Standard deviation calculated from differences in observations

Note that for this code to make sense, the first observation for *Before* is student a and the first observation for *After* is student a , and so on.

```
library(lsrr)
cohensD(Score ~ Time,
        data = Data,
        method = "paired")
[1] 1.204314
```

Or we can calculate the value manually.

```

Before = Data$Score[Data$Time=="Before"]
After  = Data$Score[Data$Time=="After"]
Difference = After - Before

mean(Before)
[1] 72.5

mean(After)
[1] 82.7

mean(Before) - mean(After)
[1] -10.2

( mean(Before) - mean(After) ) / sd(Difference)
[1] -1.204314

```

Standard deviation calculated from the “before” observations

Note that for this code to make sense, the “before” observations must correspond to the **second** level of the *Time* variable, corresponding to *y.sd* in the function.

```

library(lsrr)
cohensD(Score ~ Time,
         data   = Data,
         method = "y.sd")

[1] 0.9174268

( mean(Before) - mean(After) ) / sd(Before)

[1] -0.9174268

```

Optional readings

“**Paired t-test**” in McDonald, J.H. 2014. *Handbook of Biological Statistics*.
www.biostathandbook.com/pairedttest.html.

References

"Paired t-test" in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_09.html.

Exercises Q

1. Consider the Dumbland Extension data. Report for each answer **How you know**, when appropriate, by reporting the values of the statistic you are using or other information you used.

- a. What was the mean of the differences in score before and after the training?
- b. Was this an increase or a decrease?
- c. Is the data distribution for the paired differences reasonably normal?
- d. Was the mean score significantly different before and after the training?
- e. What do you conclude practically? As appropriate, report the means before and after, the mean difference, the effect size and interpretation, whether the difference is large relative to the scoring system, anything notable on plots, and your practical conclusions.

2. Residential properties in Dougal County rarely need phosphorus for good turfgrass growth. As part of an extension education program, Early and Rusty Cuyler asked homeowners to report their phosphorus fertilizer use, in pounds of P₂O₅ per acre, before the program and then one year later.

Date	Homeowner	P2O5
'2014-01-01'	a	0.81
'2014-01-01'	b	0.86
'2014-01-01'	c	0.79
'2014-01-01'	d	0.59
'2014-01-01'	e	0.71
'2014-01-01'	f	0.88
'2014-01-01'	g	0.63
'2014-01-01'	h	0.72
'2014-01-01'	i	0.76
'2014-01-01'	j	0.58
'2015-01-01'	a	0.67
'2015-01-01'	b	0.83
'2015-01-01'	c	0.81
'2015-01-01'	d	0.50
'2015-01-01'	e	0.71
'2015-01-01'	f	0.72
'2015-01-01'	g	0.67
'2015-01-01'	h	0.67
'2015-01-01'	i	0.48
'2015-01-01'	j	0.68

For each of the following, answer the question, and **show the output from the analyses you used to answer the question**.

- a. What was mean of the differences in P₂O₅ before and after the training?
- b. Is this an increase or a decrease?
- c. Is the data distribution for the paired differences reasonably normal?
- d. Was the mean P₂O₅ use significantly different before and after the training?
- f. What do you conclude practically? As appropriate, report the means before and after, the mean difference, the effect size and interpretation, whether the difference is large relative to the values, anything notable on plots, and your practical conclusions.

One-way ANOVA

A one-way analysis of variance (ANOVA) is similar to an independent *t*-test, except that it is capable of comparing more than two groups.

We will conduct the ANOVA by constructing a general linear model with the *lm* function in the native *stats* package. The general linear model is the basis for more advanced parametric models that can include multiple independent variables that can be continuous or factor variables.

Appropriate data

- One-way data. That is, one measurement variable in two or more groups
- Dependent variable is interval/ratio, and is continuous
- Independent variable is a factor with two or more levels. That is, two or more groups
- In the general linear model approach, residuals are normally distributed
- In the general linear model approach, groups have the same variance. That is, homoscedasticity
- Observations among groups are independent. That is, not paired or repeated measures data
- Moderate deviation from normally-distributed residuals is permissible

Hypotheses

- Null hypothesis: The means of the measurement variable for each group are equal.
- Alternative hypothesis (two-sided): The means of the measurement variable for among groups are not equal.

Interpretation

- Reporting significant results for the omnibus test as “Significant differences were found among means for groups.” is acceptable. Alternatively, “A significant effect for Independent Variable on Dependent Variable was found.”
- Reporting significant results for mean separation post-hoc tests as “Mean of variable Y for group A was different than that for group B.” is acceptable.

Other notes and alternative tests

- The nonparametric analogue for this test is the Kruskal–Wallis test. Another nonparametric approach is to use a permutation test. See [Mangiafico \(2015b\)](#) in the “References” section.
- Power analysis for one-way anova can be found at [Mangiafico \(2015a\)](#) in the “References” section.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- Rmisc
- ggplot2
- car
- multcompView
- emmeans
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(Rmisc)){install.packages("Rmisc")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

One-way ANOVA example

In the following example, Brendon Small, Coach McGuirk, and Melissa Robbins have their SNAP-Ed students keep diaries of what they eat for a week, and then calculate the daily sodium intake in milligrams. Since the classes have received different nutrition education programs, they want to see if the mean sodium intake is the same among classes.

The analysis will be conducted by constructing a general linear model with the *lm* function in the native *stats* package, and conducting the analysis of variance with the *Anova* function in the *car* package.

Plots will be used to check model assumptions for normality of residuals and homoscedasticity.

Finally, if the effect of *Instructor* is significant, a mean comparison test will be conducted to determine which group means differ from which others.

```
Input = "
Instructor      Student  Sodium
'Brendon Small'    a      1200
```

'Brendon Small'	b	1400
'Brendon Small'	c	1350
'Brendon Small'	d	950
'Brendon Small'	e	1400
'Brendon Small'	f	1150
'Brendon Small'	g	1300
'Brendon Small'	h	1325
'Brendon Small'	i	1425
'Brendon Small'	j	1500
'Brendon Small'	k	1250
'Brendon Small'	l	1150
'Brendon Small'	m	950
'Brendon Small'	n	1150
'Brendon Small'	o	1600
'Brendon Small'	p	1300
'Brendon Small'	q	1050
'Brendon Small'	r	1300
'Brendon Small'	s	1700
'Brendon Small'	t	1300
'Coach McGuirk'	u	1100
'Coach McGuirk'	v	1200
'Coach McGuirk'	w	1250
'Coach McGuirk'	x	1050
'Coach McGuirk'	y	1200
'Coach McGuirk'	z	1250
'Coach McGuirk'	aa	1350
'Coach McGuirk'	ab	1350
'Coach McGuirk'	ac	1325
'Coach McGuirk'	ad	1525
'Coach McGuirk'	ae	1225
'Coach McGuirk'	af	1125
'Coach McGuirk'	ag	1000
'Coach McGuirk'	ah	1125
'Coach McGuirk'	ai	1400
'Coach McGuirk'	aj	1200
'Coach McGuirk'	ak	1150
'Coach McGuirk'	al	1400
'Coach McGuirk'	am	1500
'Coach McGuirk'	an	1200
'Melissa Robins'	ao	900
'Melissa Robins'	ap	1100
'Melissa Robins'	aq	1150
'Melissa Robins'	ar	950
'Melissa Robins'	as	1100
'Melissa Robins'	at	1150
'Melissa Robins'	au	1250
'Melissa Robins'	av	1250
'Melissa Robins'	aw	1225
'Melissa Robins'	ax	1325
'Melissa Robins'	ay	1125
'Melissa Robins'	az	1025
'Melissa Robins'	ba	950
'Melissa Robins'	bc	925
'Melissa Robins'	bd	1200
'Melissa Robins'	be	1100

```
'Melissa Robins'    bf   950
'Melissa Robins'    bg   1300
'Melissa Robins'    bh   1400
'Melissa Robins'    bi   1100
")

Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                           levels=unique(Data$Instructor))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)
```

Summarize data by group

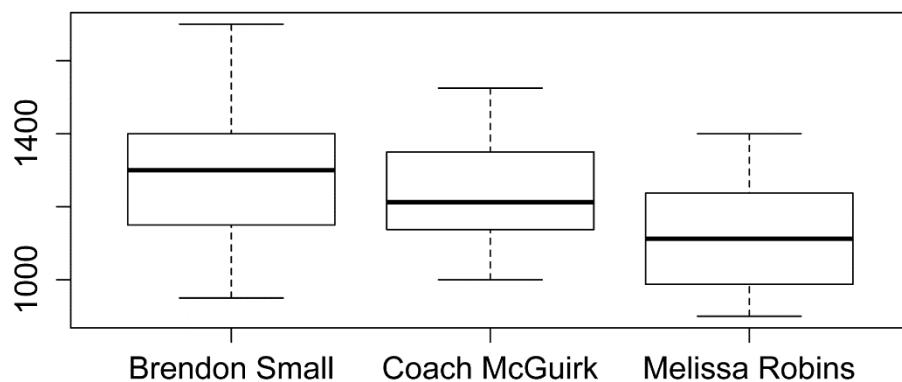
```
library(FSA)

Summarize(Sodium ~ Instructor,
          data=Data,
          digits=3)

      Instructor  n  nvalid     mean       sd   min    Q1 median    Q3 max perczero
1     Brendon Small 20      20 1287.50 193.734  950 1150    1300 1400 1700      0
2 Coach McGuirk 20      20 1246.25 142.412 1000 1144    1212 1350 1525      0
3 Melissa Robins 20      20 1123.75 143.149  900 1006    1112 1231 1400      0
```

Box plots for data by group

```
boxplot(Sodium ~ Instructor,
        data = Data)
```



Plot of means and confidence intervals

For this plot, a data frame called *Sum* will be created with the mean sodium intake and confidence interval for this mean for each instructor.

```
library(rcompanion)

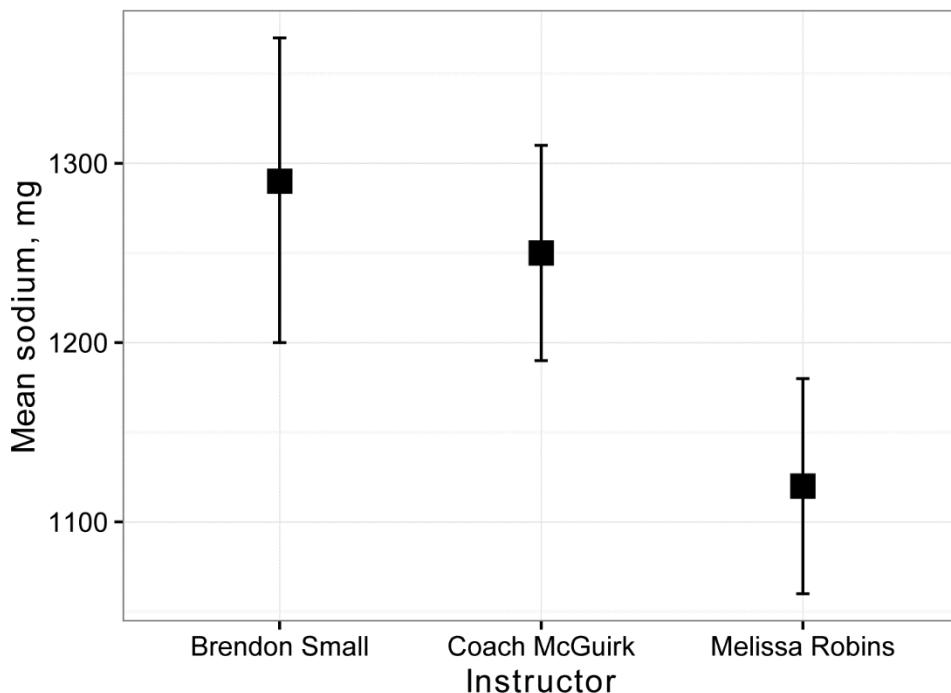
Sum = groupwiseMean(Sodium ~ Instructor,
                     data      = Data,
                     conf      = 0.95,
                     digits    = 3,
                     traditional = FALSE,
                     percentile = TRUE)

Sum

  Instructor n Mean Conf.level Percentile.lower Percentile.upper
1 Brendon Small 20 1290 0.95 1200 1370
2 Coach McGuirk 20 1250 0.95 1190 1310
3 Melissa Robins 20 1120 0.95 1060 1180

library(ggplot2)

ggplot(Sum,           ### The data frame to use.
       aes(x = Instructor,
           y = Mean)) +
  geom_errorbar(aes(ymin = Percentile.lower,
                    ymax = Percentile.upper),
                width = 0.05,
                size = 0.5) +
  geom_point(shape = 15,
             size = 4) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Mean sodium, mg")
```



Define linear model

The model for the analysis of variance is specified with the *lm* function. Typical formula notation is used, in this case with *Sodium* as the dependent variable, *Instructor* as the independent variable, and *Data* as the data frame containing these variables.

The *summary* function will report the coefficients for the model, including the intercept and other terms. Factor variables are shown with their dummy coding. For each term, an estimate, standard error of the estimate, and *t*- and *p*-values for the estimates are given. These are helpful for other linear models, but are often ignored in the case of anova.

The *summary* function also reports the *p*-value and *r-squared* value for the model as a whole. These are occasionally used in reporting results of anova.

```
model = lm(Sodium ~ Instructor,
           data = Data)

summary(model)    ### will show overall p-value and r-square
Multiple R-squared:  0.1632,  Adjusted R-squared:  0.1338
F-statistic: 5.558 on 2 and 57 DF,  p-value: 0.006235
```

Conduct analysis of variance

An analysis of variance table is reported as the result of the *Anova* function in the *car* package. By default the *Anova* function uses type-II sum of squares, which are appropriate in common anova analyses.

This table contains the *p*-value for each effect tested in the model.

```
library(car)

Anova(model,
      type = "II")  ### Type II sum of squares

  Anova Table (Type II tests)

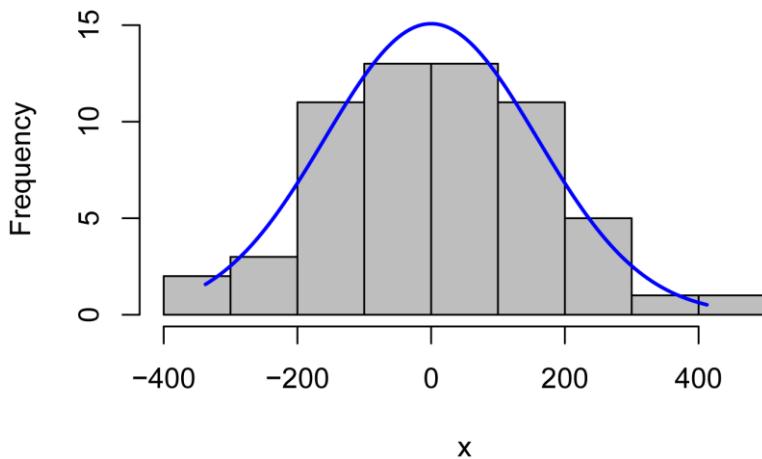
  Response: Sodium

    Sum Sq Df F value    Pr(>F)
Instructor 290146  2 5.5579 0.006235 **
Residuals 1487812 57
```

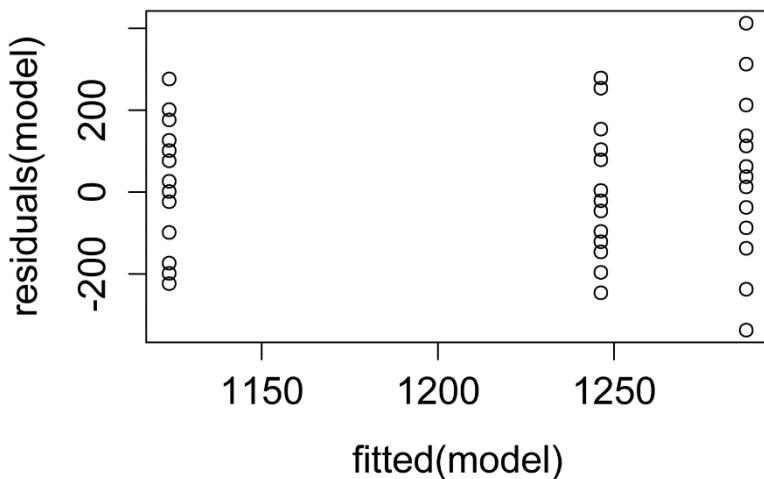
Histogram of residuals

```
x = residuals(model)
```

```
library(rcompanion)
plotNormalHistogram(x)
```



```
plot(fitted(model),
      residuals(model))
```



```
plot(model)  ### Additional model checking plots
```

Post-hoc analysis: mean separation tests

For an explanation of using estimated marginal means for multiple comparisons, review the following chapters:

- *What are Estimated Marginal Means?*
- *Estimated Marginal Means for Multiple Comparisons*

The code defines an object *marginal* which holds the output of the *emmeans* function. The *emmeans* function here calls for comparisons for the independent variable *Instructor* in the previously defined *model* object.

Using the *cld* function produces a compact letter display. EM means sharing a letter are not significantly different from one another at the *alpha* level indicated. A Tukey adjustment is made for multiple comparisons with the *adjust="tukey"* option. And *Letters* indicates the symbols to use for groups.

To adjust the confidence intervals for the EM means, use e.g. *summary(marginal, level=0.99)* or e.g. *cld(marginal, level=0.99)*.

Note that because *Instructor* is the only independent variable in the model, the EM means are equal to the arithmetic means produced by the *Summarize* function in this case.

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
                    ~ Instructor,
                    adjust = "tukey")

marginal
```

Instructor	emmean	SE	df	lower.CL	upper.CL
Brendon Small	1287.50	36.12615	57	1215.159	1359.841
Coach McGuirk	1246.25	36.12615	57	1173.909	1318.591
Melissa Robins	1123.75	36.12615	57	1051.409	1196.091

Confidence level used: 0.95

```
pairs(marginal,
      adjust="tukey")

contrast                         estimate      SE df t.ratio p.value
Brendon Small - Coach McGuirk    41.25 51.09009 57   0.807  0.7000
Brendon Small - Melissa Robins   163.75 51.09009 57   3.205  0.0062
Coach McGuirk - Melissa Robins   122.50 51.09009 57   2.398  0.0510
```

P value adjustment: tukey method for comparing a family of 3 estimates

```
CLD = cld(marginal,
            alpha = 0.05,
            Letters = letters,    ### Use lower-case letters for .group
            adjust = "tukey")    ### Tukey-adjusted comparisons
```

CLD

Instructor	emmean	SE	df	lower.CL	upper.CL	.group
Melissa Robins	1123.75	36.12615	57	1034.882	1212.618	a
Coach McGuirk	1246.25	36.12615	57	1157.382	1335.118	ab
Brendon Small	1287.50	36.12615	57	1198.632	1376.368	b

Confidence level used: 0.95

P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05

Plot of means, confidence intervals, and mean-separation letters

This plot uses the estimated marginal means, confidence intervals, and mean-separation letters from the compact letter display above.

For different data, the *nudge_y* parameter will need to be adjusted manually.

```
### Order the levels for printing

CLD$Instructor = factor(CLD$Instructor,
                        levels=c("Brendon Small",
                                "Coach McGuirk",
                                "Melissa Robins"))

### Remove spaces in .group

CLD$.group=gsub(" ", "", CLD$.group)
```

```
### Plot

library(ggplot2)

ggplot(CLD,
       aes(x      = Instructor,
            y      = emmean,
            label = .group)) +

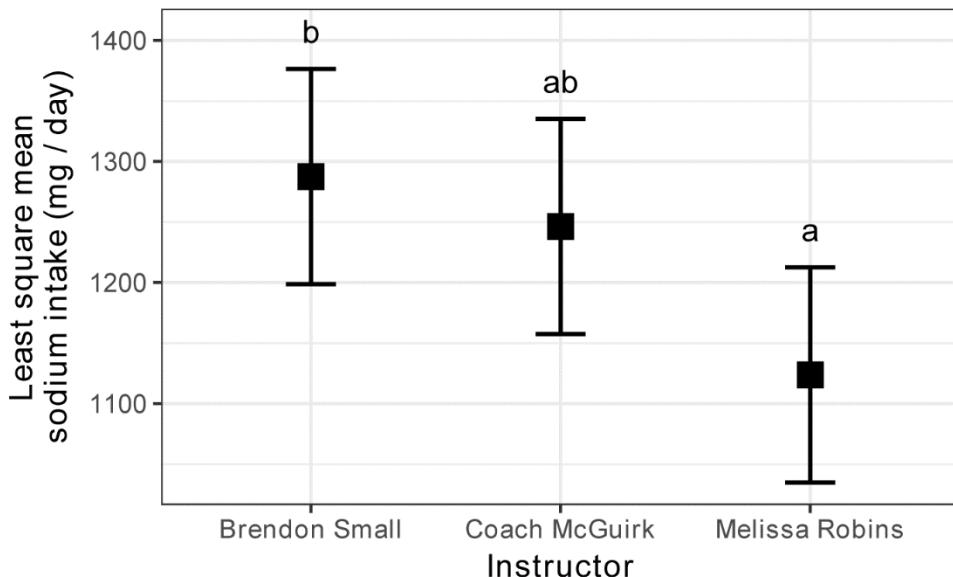
  geom_point(shape = 15,
             size = 4) +

  geom_errorbar(aes(ymin = lower.CL,
                     ymax = upper.CL,
                     width = 0.2,
                     size = 0.7) +

  theme_bw() +
  theme(axis.title = element_text(face = "bold"),
        axis.text   = element_text(face = "bold"),
        plot.caption = element_text(hjust = 0)) +

  ylab("Estimated marginal mean\nsodium intake (mg / day)") +

  geom_text(nudge_x = c(0, 0, 0),
            nudge_y = c(120, 120, 120),
            color   = "black")
```



Daily intake of sodium from diaries for students in Supplemental Nutrition Assistance Program Education (SNAP-ed) workshops. Boxes represent estimated marginal mean values for each of three classes, following one-way analysis of variance (ANOVA). Error bars indicate 95% confidence intervals of the EM means. Means sharing a letter are not significantly different ($\alpha = 0.05$, Tukey-adjusted).

References

"One-way Anova" in Mangiafico, S.S. 2015a. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_05.html.

"One-way Analysis with Permutation Test" in Mangiafico, S.S. 2015b. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_06a.html.

Exercises R

1. Considering Brendon, Melissa, and McGuirk's data,

- What was the mean sodium intake for each instructor?
- Does *Instructor* have a significant effect on sodium intake?
- Are the residuals reasonably normal and homoscedastic?
- Which instructors had classes with significantly different mean sodium intake from which others?
- Does this result correspond to what you would have thought from looking at the plot of the means and standard errors or the box plots?
- Practically speaking, what do you conclude?

2. As part of a professional skills program, a 4-H club tests its members for typing proficiency. Dr. Katz, Laura, and Ben want to compare their students' mean typing speed between their classes.

Instructor	Student	words.per.minute
'Dr. Katz Professional Therapist'	a	35
'Dr. Katz Professional Therapist'	b	50
'Dr. Katz Professional Therapist'	c	55
'Dr. Katz Professional Therapist'	d	60
'Dr. Katz Professional Therapist'	e	65
'Dr. Katz Professional Therapist'	f	60
'Dr. Katz Professional Therapist'	g	70
'Dr. Katz Professional Therapist'	h	55
'Dr. Katz Professional Therapist'	i	45
'Dr. Katz Professional Therapist'	j	55
'Dr. Katz Professional Therapist'	k	60
'Dr. Katz Professional Therapist'	l	45
'Dr. Katz Professional Therapist'	m	65
'Dr. Katz Professional Therapist'	n	55
'Dr. Katz Professional Therapist'	o	50
'Dr. Katz Professional Therapist'	p	60
'Laura the Receptionist'	q	55

'Laura the Receptionist'	r	60
'Laura the Receptionist'	s	75
'Laura the Receptionist'	t	65
'Laura the Receptionist'	u	60
'Laura the Receptionist'	v	70
'Laura the Receptionist'	w	75
'Laura the Receptionist'	x	70
'Laura the Receptionist'	y	65
'Laura the Receptionist'	z	72
'Laura the Receptionist'	aa	73
'Laura the Receptionist'	ab	65
'Laura the Receptionist'	ac	80
'Laura the Receptionist'	ad	50
'Laura the Receptionist'	ae	60
'Laura the Receptionist'	af	70
'Ben Katz'	ag	55
'Ben Katz'	ah	55
'Ben Katz'	ai	70
'Ben Katz'	aj	55
'Ben Katz'	ak	65
'Ben Katz'	al	60
'Ben Katz'	am	70
'Ben Katz'	an	60
'Ben Katz'	ao	60
'Ben Katz'	ap	62
'Ben Katz'	aq	63
'Ben Katz'	ar	65
'Ben Katz'	as	75
'Ben Katz'	at	50
'Ben Katz'	au	50
'Ben Katz'	av	65

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- What was the mean typing speed for each instructor?
- Does *Instructor* have a significant effect on typing speed?
- Are the residuals reasonably normal and homoscedastic?
- Which instructors had classes with significantly different mean typing speed from which others?
- Produce and submit a plot of means with error bars or a box plot. Include a caption.
- Does this result correspond to what you would have thought from looking at the plot of the means and standard errors or box plots?
- Practically speaking, what do you conclude?

One-way ANOVA with Blocks

Blocks are used in an analysis of variance or similar models in order to account for suspected variation from factors other than the treatments or main independent variables being investigated.

Traditionally, in agricultural experiments, plots would be arranged into blocks according to factors in the field that could not be controlled. For example, if one side of the field were close to the tree line, those plots might get less sun, or be more protected from wind. So the experimenter would consider those plots near the tree line as one block, and another set in a more open spot another block. In this way, all kinds of variation in the soils and locations could be accounted for somewhat. The experiment might be designed in a *randomized complete block design* in which each block had a plot with each treatment.

When using blocks, the experimenter isn't concerned necessarily with the effect of the blocks or even the factors behind assigning those blocks. That is, the east end of the field might be by the tree line and may have soils with slightly poorer drainage. If the experimenter were testing the effects of different fertilizers, she isn't going to try to report the effect of "next to the tree line" or "in poorly drained soil," but she does want to account for these effects statistically.

Blocks in extension education for measurements on students might include school or class or grade. There might be differences according to these factors, but we might not care to investigate if one particular school has a different result than another particular one. Still, we want to take these differences into account statistically.

Appropriate data

- One-way data, with blocks. That is, one measurement variable in two or more groups, where each group is also distributed among at least two blocks
- Dependent variable is interval/ratio, and is continuous
- Independent variable is a factor with two or more levels. That is, two or more groups
- A second independent variable is a blocking factor variable with two or more levels
- In the general linear model approach, residuals are normally distributed
- In the general linear model approach, groups have the same variance. That is, homoscedasticity
- Observations among groups are independent. That is, not paired or repeated measures data, except that observations are within blocks
- Moderate deviation from normally-distributed residuals is permissible

Hypotheses

- Null hypothesis: The means of the measurement variable for each group are equal
- Alternative hypothesis (two-sided): The means of the measurement variable for among groups are not equal

- An additional null hypothesis is tested for the effect of blocks: The means of the measurement variable for each block are equal

Interpretation

- Reporting significant results for the omnibus test as “Significant differences were found among means for groups.” is acceptable. Alternatively, “A significant effect for independent variable on dependent variable was found.”
- Reporting significant results for mean separation post-hoc tests as “Mean of variable Y for group A was different than that for group B.” is acceptable.

Other notes and alternative tests

- A permutation test is a nonparametric alternative. Only for *unreplicated complete block designs*, the traditional nonparametric analogues for this test are the Friedman and Quade tests.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- ggplot2
- car
- multcompView
- emmeans
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

One-way ANOVA with blocks example

This example will revisit the sodium intake data set with Brendon Small and the other instructors. This time, though, they have recorded the town each student is from, and they would like to use this as a blocking variable. They suspect that the town of residence may have some effect on sodium intake since each town has varying income, ethnic makeup, and other demographic factors.

The instructors are focused on the effect of their different nutrition education programs. They are not concerned about sodium intake in one specific town or another *per se*, but they want to take account this effect into account statistically.

Note that when the blocking variable is added to the model, the calculated *p*-value for the effect of *Instructor* is different than it was for the case of the one-way ANOVA without the blocking variable.

```
Input = "
Instructor      Town      Sodium
'Brendon Small' Squiggleville 1200
'Brendon Small' Squiggleville 1400
'Brendon Small' Squiggleville 1350
'Brendon Small' Metalocalypse 950
'Brendon Small' Squiggleville 1400
'Brendon Small' Squiggleville 1150
'Brendon Small' Squiggleville 1300
'Brendon Small' Metalocalypse 1325
'Brendon Small' Metalocalypse 1425
'Brendon Small' Squiggleville 1500
'Brendon Small' Squiggleville 1250
'Brendon Small' Metalocalypse 1150
'Brendon Small' Metalocalypse 950
'Brendon Small' Squiggleville 1150
'Brendon Small' Metalocalypse 1600
'Brendon Small' Metalocalypse 1300
'Brendon Small' Metalocalypse 1050
'Brendon Small' Metalocalypse 1300
'Brendon Small' Squiggleville 1700
'Brendon Small' Squiggleville 1300
'Coach McGuirk' Squiggleville 1100
'Coach McGuirk' Squiggleville 1200
'Coach McGuirk' Squiggleville 1250
'Coach McGuirk' Metalocalypse 1050
'Coach McGuirk' Metalocalypse 1200
'Coach McGuirk' Metalocalypse 1250
'Coach McGuirk' Squiggleville 1350
'Coach McGuirk' Squiggleville 1350
'Coach McGuirk' Squiggleville 1325
'Coach McGuirk' Squiggleville 1525
'Coach McGuirk' Squiggleville 1225
'Coach McGuirk' Squiggleville 1125
'Coach McGuirk' Metalocalypse 1000
'Coach McGuirk' Metalocalypse 1125
'Coach McGuirk' Squiggleville 1400
'Coach McGuirk' Metalocalypse 1200
'Coach McGuirk' Squiggleville 1150
'Coach McGuirk' Squiggleville 1400
'Coach McGuirk' Squiggleville 1500
'Coach McGuirk' Squiggleville 1200
'Melissa Robins' Metalocalypse 900
'Melissa Robins' Metalocalypse 1100
'Melissa Robins' Metalocalypse 1150
'Melissa Robins' Metalocalypse 950
'Melissa Robins' Metalocalypse 1100
'Melissa Robins' Metalocalypse 1150
'Melissa Robins' Squiggleville 1250
'Melissa Robins' Squiggleville 1250
'Melissa Robins' Squiggleville 1225
```

```
'Melissa Robins' Squiggleville    1325
'Melissa Robins' Metalocalypse   1125
'Melissa Robins' Metalocalypse   1025
'Melissa Robins' Metalocalypse    950
'Melissa Robins' Metalocalypse    925
'Melissa Robins' Squiggleville   1200
'Melissa Robins' Metalocalypse   1100
'Melissa Robins' Metalocalypse    950
'Melissa Robins' Metalocalypse   1300
'Melissa Robins' Squiggleville   1400
'Melissa Robins' Metalocalypse   1100
")
Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))

Data$Town      = factor(Data$Town,
                        levels=unique(Data$Town))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)
```

Use Summarize to check if cell sizes are balanced

We can use the *n* or *nvalid* columns in the *Summarize* output to check if cell sizes are balanced. That is, if each treatment and block combination have the same number of observations.

```
library(FSA)

Summarize(Sodium ~ Instructor + Town,
          data=Data,
          digits=3)

      Instructor        Town  n     mean      sd  min   Q1 median   Q3 max
1  Brendon Small  Squiggleville 11 1336.364 162.928 1150 1225    1300 1400 1700
2 Coach McGuirk  Squiggleville 14 1292.857 134.960 1100 1200    1288 1388 1525
```

3	Melissa Robins	Squiggleville	6	1275.000	74.162	1200	1231		1250	1306	1400		
4	Brendon Small	Metalocalypse	9	1227.778	220.597	950	1050		1300	1325	1600		
5	Coach McGuirk	Metalocalypse	6	1137.500	97.147	1000	1069		1162	1200	1250		
6	Melissa Robins	Metalocalypse	14	1058.929	112.919	900	950		1100	1119	1300		

Define linear model

```
model = lm(Sodium ~ Instructor + Town,
           data = Data)

summary(model)    ### will show overall p-value and r-squared

Multiple R-squared:  0.3485,  Adjusted R-squared:  0.3136
F-statistic: 9.987 on 3 and 56 DF,  p-value: 2.265e-05
```

Conduct analysis of variance

```
library(car)

Anova(model,
      type = "II")  ### Type II sum of squares

Anova Table (Type II tests)

Response: Sodium

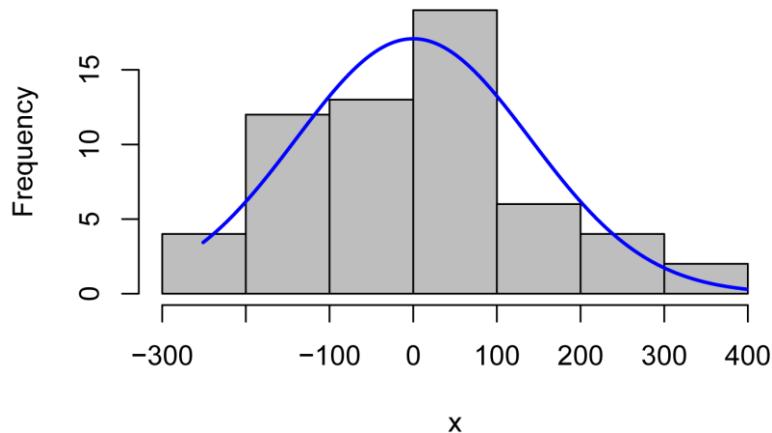
          Sum Sq Df F value    Pr(>F)
Instructor 148944  2 3.6006 0.0338033 *
Town        329551  1 15.9332 0.0001928 ***
Residuals  1158261 56
```

Histogram and plot of residuals

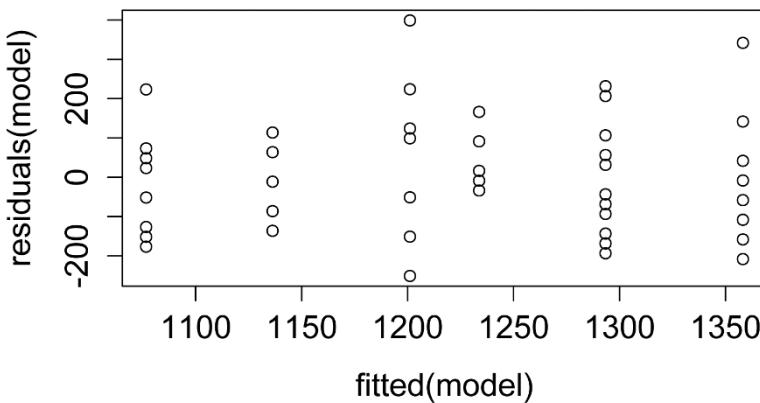
```
x = residuals(model)

library(rcompanion)

plotNormalHistogram(x)
```



```
plot(fitted(model),
      residuals(model))
```



```
plot(model) ### Additional model checking plots
```

Post-hoc analysis: mean separation tests

For an explanation of using estimated marginal means for multiple comparisons, see the section “Post-hoc analysis: mean separation tests” in the *One-way ANOVA* chapter.

In this example, you’ll note that the EM means are different from the arithmetic means calculated for *Instructor* in the last chapter. This is a result of the fact that instructors have different numbers of students from each town. That is, the design is *unbalanced*.

```
library(multcompView)
library(emmeans)
marginal = emmeans(model,
                    ~ Instructor)
```

```

pairs(marginal,
      adjust="tukey")

contrast              estimate      SE df t.ratio p.value
Brendon Small - Coach McGuirk  64.81742 45.86048 56   1.413  0.3410
Brendon Small - Melissa Robins 124.47097 46.53123 56   2.675  0.0261
Coach McGuirk - Melissa Robins 59.65356 48.12705 56   1.240  0.4352

Results are averaged over the levels of: Town
P value adjustment: tukey method for comparing a family of 3 estimates

CLD = cld(marginal,
           alpha = 0.05,
           Letters = letters, ### Use lower-case letters for .group
           adjust = "tukey") ### Tukey-adjusted p-values

CLD

Instructor      emmean      SE df lower.CL upper.CL .group
Melissa Robins 1155.173 33.10792 56 1088.850 1221.496    a
Coach McGuirk  1214.827 33.10792 56 1148.504 1281.150    ab
Brendon Small   1279.644 32.21856 56 1215.103 1344.186    b

Results are averaged over the levels of: Town
Confidence level used: 0.95
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05

```

Plot of means, confidence intervals, and mean-separation letters

This plot uses the estimated marginal means, confidence intervals, and mean-separation letters from the compact letter display above.

For different data, the *nudge_y* parameter will need to be adjusted manually.

```

### Order the levels for printing

CLD$Instructor = factor(CLD$Instructor,
                         levels=c("Brendon Small",
                                 "Coach McGuirk",
                                 "Melissa Robins"))

### Remove spaces in .group

CLD$.group=gsub(" ", "", CLD$.group)

### Plot

library(ggplot2)

ggplot(CLD,
       aes(x      = Instructor,

```

```

y      = emmean,
label = .group)) +

geom_point(shape  = 15,
           size   = 4) +

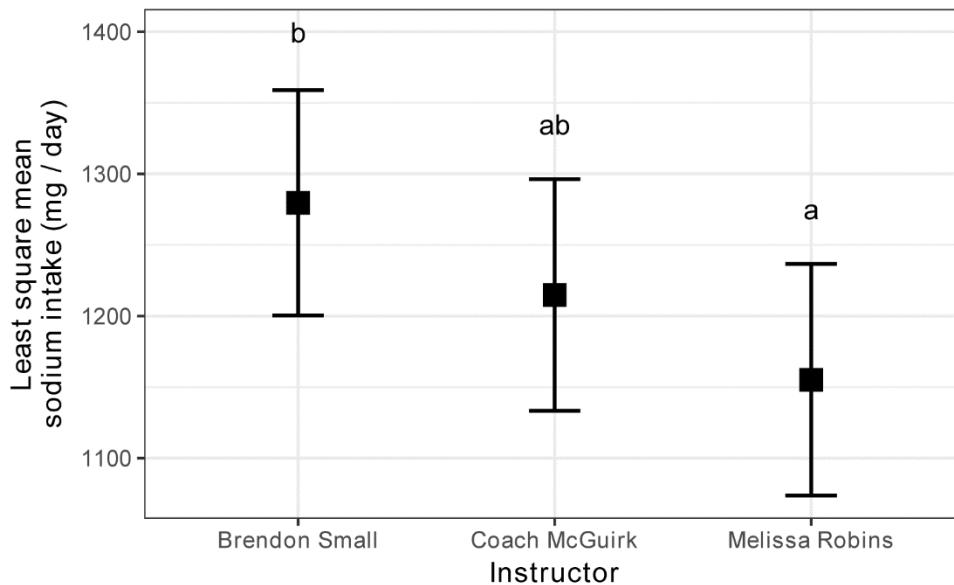
geom_errorbar(aes(ymin  = lower.CL,
                   ymax   = upper.CL),
               width = 0.2,
               size  = 0.7) +

theme_bw() +
theme(axis.title  = element_text(face = "bold"),
      axis.text    = element_text(face = "bold"),
      plot.caption = element_text(hjust = 0)) +

ylab("Estimated marginal mean\nsodium intake (mg / day)") +

geom_text(nudge_x = c(0, 0, 0),
          nudge_y = c(120, 120, 120),
          color   = "black")

```



Daily intake of sodium from diaries for students in Supplemental Nutrition Assistance Program Education (SNAP-ed) workshops. Boxes represent estimated marginal mean values for each of three classes, following one-way analysis of variance (ANOVA). Error bars indicate 95% confidence intervals of the EM means. Means sharing a letter are not significantly different ($\alpha = 0.05$, Tukey-adjusted).

Exercises S

1. Considering Brendon, Melissa, and McGuirk's data with means adjusted for students' towns,

What was the EM mean sodium intake for students for each instructor?

Does *Instructor* have a significant effect on sodium intake?

Does *Town* have a significant effect on sodium intake?

Are the residuals reasonably normal and homoscedastic?

Which instructors had classes with significantly different mean sodium intake from which others?

Does this result correspond to what you would have thought from looking at the plot of the EM means and confidence intervals?

Is this design balanced or unbalanced?

2. As part of a professional skills program, a 4-H club tests its members for typing proficiency. Dr. Katz, and Laura, and Ben want to compare their students' mean typing speed between their classes. They have also recorded the year or grade of each student. They are not interested in the effect of *Year per se*, but they want to account for the effect statistically.

Instructor	Year	words.per.minute
'Dr. Katz Professional Therapist'	7	35
'Dr. Katz Professional Therapist'	7	50
'Dr. Katz Professional Therapist'	7	55
'Dr. Katz Professional Therapist'	7	60
'Dr. Katz Professional Therapist'	8	65
'Dr. Katz Professional Therapist'	8	60
'Dr. Katz Professional Therapist'	8	70
'Dr. Katz Professional Therapist'	8	55
'Dr. Katz Professional Therapist'	9	45
'Dr. Katz Professional Therapist'	9	55
'Dr. Katz Professional Therapist'	9	60
'Dr. Katz Professional Therapist'	9	45
'Dr. Katz Professional Therapist'	10	65
'Dr. Katz Professional Therapist'	10	55
'Dr. Katz Professional Therapist'	11	50
'Dr. Katz Professional Therapist'	12	60
'Laura the Receptionist'	7	55
'Laura the Receptionist'	7	60
'Laura the Receptionist'	7	75
'Laura the Receptionist'	7	65
'Laura the Receptionist'	8	60
'Laura the Receptionist'	8	70
'Laura the Receptionist'	8	75
'Laura the Receptionist'	8	70
'Laura the Receptionist'	9	65
'Laura the Receptionist'	9	72
'Laura the Receptionist'	9	73
'Laura the Receptionist'	9	65
'Laura the Receptionist'	10	80

'Laura the Receptionist'	10	50
'Laura the Receptionist'	10	60
'Laura the Receptionist'	10	70
'Ben Katz'	7	55
'Ben Katz'	7	55
'Ben Katz'	7	70
'Ben Katz'	7	55
'Ben Katz'	8	65
'Ben Katz'	8	60
'Ben Katz'	8	70
'Ben Katz'	8	60
'Ben Katz'	9	60
'Ben Katz'	9	62
'Ben Katz'	9	63
'Ben Katz'	9	65
'Ben Katz'	10	75
'Ben Katz'	10	50
'Ben Katz'	10	50
'Ben Katz'	10	65

Note that the following code is necessary for R to recognize *Year* as a factor variable.

```
### Convert Year from an integer variable to a factor variable

Data$Year = factor(Data$Year,
                    levels=c("7", "8", "9", "10"))
```

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

What was the EM mean typing speed for each instructor's class?

Does *Instructor* have a significant effect on typing speed?

Does *Year* have a significant effect on typing speed?

Are the residuals reasonably normal and homoscedastic?

Which instructors had classes with significantly different mean typing speed from which others?

Does this result correspond to what you would have thought from looking at the plot of the EM means and standard errors?

Is this design balanced or unbalanced?

One-way ANOVA with Random Blocks

This chapter reproduces the example from the previous chapter. The reader should consult that chapter for an explanation of one-way analysis of variance with blocks. Here, the analysis is done with a mixed effects model, with the treatments treated as a fixed effect and the blocks treated as a random effect.

In analysis of variance, blocking variables are often treated as random variables. This is because the blocking variable represents a random selection of levels of that variable. The analyst wants to take the effects of the blocking variable into account, but the identity of the specific levels of the blocks are not of interest.

In the example in this chapter, the instructors are focused on the effect of their different nutrition education programs, which are the treatments; they are not concerned about the effect of one specific town or another *per se*, but do want to take into account any differences due to the different towns.

For a more complete discussion of random effects, see the *Using Random Effects in Models* chapter.

Packages used in this chapter

The packages used in this chapter include:

- psych
- lme4
- lmerTest
- multcompView
- emmeans
- nlme
- car
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(lme4)){install.packages("lme4")}
if(!require(lmerTest)){install.packages("lmerTest")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(nlme)){install.packages("nlme")}
if(!require(car)){install.packages("car")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

One-way ANOVA with random blocks example

<code>Input = ("</code>		
<code>Instructor</code>	<code>Town</code>	<code>Sodium</code>

'Brendon Small'	Squiggleville	1200
'Brendon Small'	Squiggleville	1400
'Brendon Small'	Squiggleville	1350
'Brendon Small'	Metalocalypse	950
'Brendon Small'	Squiggleville	1400
'Brendon Small'	Squiggleville	1150
'Brendon Small'	Squiggleville	1300
'Brendon Small'	Metalocalypse	1325
'Brendon Small'	Metalocalypse	1425
'Brendon Small'	Squiggleville	1500
'Brendon Small'	Squiggleville	1250
'Brendon Small'	Metalocalypse	1150
'Brendon Small'	Metalocalypse	950
'Brendon Small'	Squiggleville	1150
'Brendon Small'	Metalocalypse	1600
'Brendon Small'	Metalocalypse	1300
'Brendon Small'	Metalocalypse	1050
'Brendon Small'	Metalocalypse	1300
'Brendon Small'	Squiggleville	1700
'Brendon Small'	Squiggleville	1300
'Coach McGuirk'	Squiggleville	1100
'Coach McGuirk'	Squiggleville	1200
'Coach McGuirk'	Squiggleville	1250
'Coach McGuirk'	Metalocalypse	1050
'Coach McGuirk'	Metalocalypse	1200
'Coach McGuirk'	Metalocalypse	1250
'Coach McGuirk'	Squiggleville	1350
'Coach McGuirk'	Squiggleville	1350
'Coach McGuirk'	Squiggleville	1325
'Coach McGuirk'	Squiggleville	1525
'Coach McGuirk'	Squiggleville	1225
'Coach McGuirk'	Squiggleville	1125
'Coach McGuirk'	Metalocalypse	1000
'Coach McGuirk'	Metalocalypse	1125
'Coach McGuirk'	Squiggleville	1400
'Coach McGuirk'	Metalocalypse	1200
'Coach McGuirk'	Squiggleville	1150
'Coach McGuirk'	Squiggleville	1400
'Coach McGuirk'	Squiggleville	1500
'Coach McGuirk'	Squiggleville	1200
'Melissa Robins'	Metalocalypse	900
'Melissa Robins'	Metalocalypse	1100
'Melissa Robins'	Metalocalypse	1150
'Melissa Robins'	Metalocalypse	950
'Melissa Robins'	Metalocalypse	1100
'Melissa Robins'	Metalocalypse	1150
'Melissa Robins'	Squiggleville	1250
'Melissa Robins'	Squiggleville	1250
'Melissa Robins'	Squiggleville	1225
'Melissa Robins'	Squiggleville	1325
'Melissa Robins'	Metalocalypse	1125
'Melissa Robins'	Metalocalypse	1025
'Melissa Robins'	Metalocalypse	950
'Melissa Robins'	Metalocalypse	925
'Melissa Robins'	Squiggleville	1200

```

'Melissa Robins' Metalocalypse    1100
'Melissa Robins' Metalocalypse    950
'Melissa Robins' Metalocalypse   1300
'Melissa Robins' Squiggleville   1400
'Melissa Robins' Metalocalypse   1100
")

Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))

Data$Town      = factor(Data$Town,
                        levels=unique(Data$Town))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Mixed-effects model with lmer

In this first example, the model will be specified with the *lmer* function in the package *lme4*. The term *(1|Town)* in the model formula indicates that *Town* should be treated as a random variable, with each level having its own intercept in the model. The *anova* function in the package *lmerTest* is used to produce *p*-values for the fixed effects. The *rand* function in the package *lmerTest* produces *p*-values for the random effects.

Technical note

lmerTest::anova by default returns *p*-values for Type III sum of squares with a Satterthwaite approximation for the degrees of freedom.

```

library(lme4)

library(lmerTest)

model = lmer(Sodium ~ Instructor + (1|Town),
             data=Data,

```

```
REML=TRUE)
```

```
anova(model)
```

Analysis of Variance Table of type III with Satterthwaite approximation for degrees of freedom

	Sum Sq	Mean Sq	NumDF	DenDF	F.value	Pr(>F)
Instructor	154766	77383	2	56.29	3.7413	0.02982 *

```
rand(model)
```

Analysis of Random effects Table:

	Chi.sq	Chi.DF	p.value
Town	10.5	1	0.001 **

p-value and pseudo R-squared for model

To use the *nagelkerke* function with an *lmer* object, the null model with which to compare the *lmer* model has to be specified explicitly.

One approach is to define the null model as one with no fixed effects except for an intercept, indicated with a 1 on the right side of the ~. And to also include the random effects, in this case (1/Town).

```
model.null = lmer(Sodium ~ 1 + (1|Town),
                   data = Data,
                   REML = TRUE)
```

```
anova(model,
      model.null)
```

	Df	AIC	BIC	logLik	deviance	Chisq	Chi Df	Pr(>Chisq)
.1	3	782.34	788.63	-388.17	776.34			
object	5	778.82	789.29	-384.41	768.82	7.5255	2	0.02322 *

```
library(rcompanion)
```

```
nagelkerke(fit = model,
           null = model.null)
```

	\$Pseudo.R.squared.for.model.vs.null	Pseudo.R.squared
McFadden		0.00971559
Cox and Snell (ML)		0.11818400
Nagelkerke (Cragg and Uhler)		0.11818400

Another approach to determining the *p*-value and pseudo *R-squared* for an *lmer* model is to compare the model to a null model with only an intercept and neither the fixed nor the random effects. To

accomplish this, the null model has to be specified with *lm* and not *lmer*. However, it may not be permissible to compare models specified with different functions in this way. The *anova* function in *lmerTest* does allow these comparisons. The *nagelkerke* function will also allow these comparisons, although I do not know if the results are valid.

```
model.null.2 = lm(Sodium ~ 1,
                   data = Data)

anova(model,
       model.null.2)

      Df     AIC     BIC logLik deviance Chisq Chi Df Pr(>Chisq)
..1    2 792.07 796.26 -394.04    788.07
object 5 778.82 789.29 -384.41    768.82 19.253      3  0.0002424 ***
$Pseudo.R.squared.for.model.vs.null
                                Pseudo.R.squared
McFadden                      0.0239772
Cox and Snell (ML)            0.2701590
Nagelkerke (Cragg and Uhler)  0.2701600
```

Post-hoc analysis

The *emmeans* package is able to handle *lmer* objects. For further details, see *?emmeans::models*. For a review of mean separation tests and estimated marginal means, see the chapters *What are Estimated Marginal Means?*; the chapter *Estimated Marginal Means for Multiple Comparisons*; and the “Post-hoc analysis: mean separation tests” section in the *One-way ANOVA* chapter.

The *lmerTest* package has a function *Difflsmeans* which also compares marginal means for treatments, but doesn’t include output for a compact letter display or adjusted *p*-values.

```
library(multcompView)

library(emmeans)

marginal = emmeans(model,
                    ~ Instructor)

CLD = cld(marginal,
           alpha=0.05,
           Letters=letters,      ### Use lower-case letters for .group
           adjust="tukey")        ### Tukey-adjusted comparisons

CLD

Instructor      emmean      SE   df lower.CL upper.CL .group
Melissa Robins 1153.201 83.01880 1.24 476.7519 1829.650  a
```

```
Coach McGuirk 1216.799 83.01880 1.24 540.3498 1893.248 ab
Brendon Small 1280.137 82.60038 1.22 588.9293 1971.345 b
```

Confidence level used: 0.95

P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05

```
pairs(marginal,
      adjust="tukey")

contrast           estimate      SE   df t.ratio p.value
Brendon Small - Coach McGuirk 63.33828 45.93368 56.11  1.379  0.3587
Brendon Small - Melissa Robins 126.93620 46.73138 56.29  2.716  0.0234
Coach McGuirk - Melissa Robins 63.59792 48.62097 56.62  1.308  0.3967

P value adjustment: tukey method for comparing a family of 3 estimate
```

A similar analysis can be conducted with the *diffMeans* function in the *lmerTest* package.

```
library(lmerTest)

diffMeans(model,
          test.effs="Instructor")

Differences of LSMEANS:
                                         Estimate Standard Error   DF t-value Lower CI Upper CI p-value
Instructor Brendon Small - Coach McGuirk    63.3            45.8 56.1    1.38   -28.5    155  0.172
Instructor Brendon Small - Melissa Robins  126.9            46.5 56.3    2.73   33.9     220  0.008 **
Instructor Coach McGuirk - Melissa Robins  63.6            48.0 56.6    1.33   -32.5    160  0.190
```

The following code extracts a data frame from the *diffMeans* output, and then adds a column of *p*-values adjusted by the *fdr* method. See *?p.adjust* for other adjustment options.

```
comparison = diffMeans(model,
                        test.effs="Instructor")[[1]]

p.value = comparison$p.value

comparison$p.adj = p.adjust(p.value,
                            method = "fdr")

comparison
                                         Estimate Standard Error   DF t-value Lower CI Upper CI p-value p.adj
Instructor Brendon Small - Coach McGuirk    63.3383        45.8366 56.1    1.38   -28.4807 155.1573  0.1725 0.1902
Instructor Brendon Small - Melissa Robins  126.9362        46.4659 56.3    2.73   33.8631 220.0093  0.0084 0.0252
Instructor Coach McGuirk - Melissa Robins  63.5979        47.9651 56.6    1.33   -32.4661 159.6619  0.1902 0.1902
```

Mixed-effects model with nlme

Mixed models can also be specified with the *nlme* package. The formula notation for this package is slightly different than for the *lme4* package. The expression *random=~1/Town* indicates that *Town* should be treated as a random variable, with each level having its own intercept in the model.

The fixed effects in the model can be tested with the *Anova* function in the *car* package.

```
library(nlme)

model = lme(Sodium ~ Instructor, random=~1|Town,
            data=Data,
            method="REML")

library(car)

Anova(model)

Analysis of Deviance Table (Type II tests)

      chisq  df Pr(>Chisq)
Instructor 7.4827  2    0.02372 *
```

The random effects in the model can be tested by specifying a null model with only fixed effects and comparing it to the full model with *anova*. The *nlme* package has a function *gls* that creates model objects without random effects in a manner analogous to those specified with *lme*.

```
model.null = gls(Sodium ~ Instructor,
                  data=Data,
                  method="REML")

anova(model,
       model.null)

      Model  df      AIC      BIC   logLik   Test  L.Ratio p-value
model       1  5 749.9281 760.1434 -369.9641
model.null  2  4 758.4229 766.5951 -375.2114 1 vs 2 10.49476  0.0012
```

p-value and pseudo R-squared for model

By default, the *nagelkerke* function will compare an *lme* object to a null model with the fixed effects removed, but the random effects retained.

```
library(rcompanion)

nagelkerke(model)

$Models

Model: "lme.formula, Sodium ~ Instructor, Data, ~1 | Town, REML"
Null: "lme.formula, Sodium ~ 1, Data, ~1 | Town, REML"

$Pseudo.R.squared.for.model.vs.null
                                Pseudo.R.squared
McFadden                      0.00971559
Cox and Snell (ML)             0.11818400
Nagelkerke (Cragg and Uhler)   0.11818400

$Likelihood.ratio.test
Df.diff LogLik.diff Chisq p.value
```

```
-2      -3.7732 7.5463 0.02298
```

```
$Messages
[1] "Note: For models fit with REML, these statistics are based on refitting with
ML"
```

Another approach to determining the p -value and pseudo R -squared for an *lme* model is to compare the model to a null model with only an intercept and neither the fixed nor the random effects. To accomplish this, the null model has to be specified with the *gls* function.

```
model.null.2 = gls(Sodium ~ 1,
                    data=Data,
                    method="REML")

library(rcompanion)

nagelkerke(fit = model,
           null = model.null.2)

$Pseudo.R.squared.for.model.vs.null
                                Pseudo.R.squared
McFadden                      0.0239772
Cox and Snell (ML)            0.2701590
Nagelkerke (Cragg and Uhler)  0.2701600

$Likelihood.ratio.test
Df.diff LogLik.diff chisq   p.value
-3      -9.4479 18.896 0.00028731
```

Post-hoc analysis

The *emmeans* package is able to handle *lme* objects. For further details, see *?emmeans::models*. For a review of mean separation tests and estimated marginal means, see the chapters *What are Estimated Marginal Means?*; the chapter *Estimated Marginal Means for Multiple Comparisons*; and the “Post-hoc analysis: mean separation tests” section in the *One-way ANOVA* chapter.

```
library(multcompView)

library(emmeans)

marginal = emmeans(model,
                     ~ Instructor)

CLD = cld(marginal,
           alpha    = 0.05,
           Letters = letters,      ### Use lower-case letters for .group
           adjust   = "tukey")     ### Tukey-adjusted comparisons

CLD

Instructor      emmean       SE df lower.CL upper.CL .group
Melissa Robins 1153.201 82.92335 1 99.55992 2206.842  a
```

```
Coach McGuirk 1216.799 82.92335 1 163.15784 2270.440 ab
Brendon Small 1280.137 82.59437 1 230.67625 2329.598 b
```

Confidence level used: 0.95

P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05

```
pairs(marginal,
      adjust="tukey")
```

contrast	estimate	SE	df	t.ratio	p.value
Brendon Small - Coach McGuirk	63.33828	45.83662	56	1.382	0.3572
Brendon Small - Melissa Robins	126.93620	46.46588	56	2.732	0.0225
Coach McGuirk - Melissa Robins	63.59792	47.96514	56	1.326	0.3869

P value adjustment: tukey method for comparing a family of 3 estimate

Comparison of results from one-way ANOVA with blocks

Just for interest, a comparison of results from this chapter and the previous chapter are presented here. There is some variation in the values obtained for these statistics.

	Fixed	lmer	lme
p-value for Instructor	0.034	0.030	0.024
p-value for Town	0.00019	0.001	0.0012
Mean separation for Instructor	a ab b	a ab b	a ab b
Overall p-value	< 0.0001	0.00024	0.00029
R-square or Pseudo R-square (Nagelkerke)	0.348	0.270	0.270

Two-way ANOVA

A two-way anova can investigate the *main effects* of each of two independent factor variables, as well as the effect of the *interaction* of these variables.

For an overview of the concepts in multi-way analysis of variance, review the chapter *Factorial ANOVA: Main Effects, Interaction Effects, and Interaction Plots*.

For a review of mean separation tests and estimated marginal means, see the chapters *What are Estimated Marginal Means?*; the chapter *Estimated Marginal Means for Multiple Comparisons*; and the “Post-hoc analysis: mean separation tests” section in the *One-way ANOVA* chapter.

Appropriate data

- Two-way data. That is, one dependent variable measured across two independent factor variables
- Dependent variable is interval/ratio, and is continuous
- Independent variables are a factor with two or more levels. That is, two or more groups for each independent variable
- In the general linear model approach, residuals are normally distributed
- In the general linear model approach, groups have the same variance. That is, homoscedasticity
- Observations among groups are independent. That is, not paired or repeated measures data
- Moderate deviation from normally-distributed residuals is permissible

Hypotheses

- Null hypothesis 1: The means of the dependent variable for each group in the first independent variable are equal
- Alternative hypothesis 1 (two-sided): The means of the dependent variable for each group in the first independent variable are not equal
- Similar hypotheses are considered for the other independent variable
- Similar hypotheses are considered for the interaction effect

Interpretation

- Reporting significant results for the effect of one independent variable as “Significant differences were found among means for groups.” is acceptable. Alternatively, “A significant effect for independent variable on dependent variable was found.”
- Reporting significant results for the interaction effect as “A significant effect for the interaction of independent variable 1 and independent variable 2 on dependent variable was found.” Is acceptable.
- Reporting significant results for mean separation post-hoc tests as “Mean of variable Y for group A was different than that for group B.” is acceptable.

Packages used in this chapter

The packages used in this chapter include:

- psych
- FSA
- car
- multcompView
- emmeans
- Rmisc
- ggplot2
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(FSA)){install.packages("FSA")}
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompView")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(Rmisc)){install.packages("Rmisc")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Two-way ANOVA example

This example will revisit the sodium intake data set with Brendon Small and the other instructors. This time, though, they want to test not only the different nutrition education programs (indicated by *Instructor*), but also four supplements to the program, each of which they have used with some students.

```
Input = "
Instructor      Supplement   Sodium
'Brendon Small'    A        1200
'Brendon Small'    A        1400
'Brendon Small'    A        1350
'Brendon Small'    A        950
'Brendon Small'    A        1400
'Brendon Small'    B        1150
'Brendon Small'    B        1300
'Brendon Small'    B        1325
'Brendon Small'    B        1425
'Brendon Small'    B        1500
'Brendon Small'    C        1250
'Brendon Small'    C        1150
'Brendon Small'    C        950
'Brendon Small'    C        1150
'Brendon Small'    C        1600
'Brendon Small'    D        1300
'Brendon Small'    D        1050
'Brendon Small'    D        1300
'Brendon Small'    D        1700
'Brendon Small'    D        1300
'Coach McGuirk'   A        1100
'Coach McGuirk'   A        1200
'Coach McGuirk'   A        1250
'Coach McGuirk'   A        1050
'Coach McGuirk'   A        1200
'Coach McGuirk'   B        1250
'Coach McGuirk'   B        1350
'Coach McGuirk'   B        1350
'Coach McGuirk'   B        1325
'Coach McGuirk'   B        1525
'Coach McGuirk'   C        1225
'Coach McGuirk'   C        1125"
```

```

'Coach McGuirk'  C      1000
'Coach McGuirk'  C      1125
'Coach McGuirk'  C      1400
'Coach McGuirk'  D      1200
'Coach McGuirk'  D      1150
'Coach McGuirk'  D      1400
'Coach McGuirk'  D      1500
'Coach McGuirk'  D      1200
'Melissa Robins' A      900
'Melissa Robins' A      1100
'Melissa Robins' A      1150
'Melissa Robins' A      950
'Melissa Robins' A      1100
'Melissa Robins' B      1150
'Melissa Robins' B      1250
'Melissa Robins' B      1250
'Melissa Robins' B      1225
'Melissa Robins' B      1325
'Melissa Robins' C      1125
'Melissa Robins' C      1025
'Melissa Robins' C      950
'Melissa Robins' C      925
'Melissa Robins' C      1200
'Melissa Robins' D      1100
'Melissa Robins' D      950
'Melissa Robins' D      1300
'Melissa Robins' D      1400
'Melissa Robins' D      1100
")

Data = read.table(textConnection(Input), header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))

Data$Supplement     = factor(Data$Supplement,
                           levels=unique(Data$Supplement))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

```

```
rm(Input)
```

Use Summarize to check if cell sizes are balanced

We can use the *n* or *nvalid* columns in the *Summarize* output to check if cell sizes are balanced. That is, if each combination of treatments have the same number of observations.

```
library(FSA)
```

```
Summarize(Sodium ~ Supplement + Instructor,
          data=Data,
          digits=3)
```

	Supplement	Instructor	n	mean	sd	min	Q1	median	Q3	max
1	A	Brendon	Small	5	1260	191.703	950	1200	1350	1400
2	B	Brendon	Small	5	1340	132.994	1150	1300	1325	1425
3	C	Brendon	Small	5	1220	238.747	950	1150	1150	1250
4	D	Brendon	Small	5	1330	233.452	1050	1300	1300	1700
5	A	Coach McGuirk	5	1160	82.158	1050	1100	1200	1200	1250
6	B	Coach McGuirk	5	1360	100.933	1250	1325	1350	1350	1525
7	C	Coach McGuirk	5	1175	148.955	1000	1125	1125	1225	1400
8	D	Coach McGuirk	5	1290	151.658	1150	1200	1200	1400	1500
9	A	Melissa Robins	5	1040	108.397	900	950	1100	1100	1150
10	B	Melissa Robins	5	1240	62.750	1150	1225	1250	1250	1325
11	C	Melissa Robins	5	1045	116.458	925	950	1025	1125	1200
12	D	Melissa Robins	5	1170	178.885	950	1100	1100	1300	1400

Define linear model

Note that the interaction of *Instruction* and *Supplement* is indicated in the model definition by *Instructor:Supplement*.

```
model = lm(Sodium ~ Instructor + Supplement + Instructor:Supplement,
           data = Data)

summary(model)    ### will show overall p-value and r-square

Multiple R-squared:  0.3491,  Adjusted R-squared:   0.2
F-statistic: 2.341 on 11 and 48 DF,  p-value: 0.02121
```

Conduct analysis of variance

```
library(car)

Anova(model,
      type = "II")  ### Type II sum of squares

Anova Table (Type II tests)

Response: Sodium
            Sum Sq Df F value    Pr(>F)
```

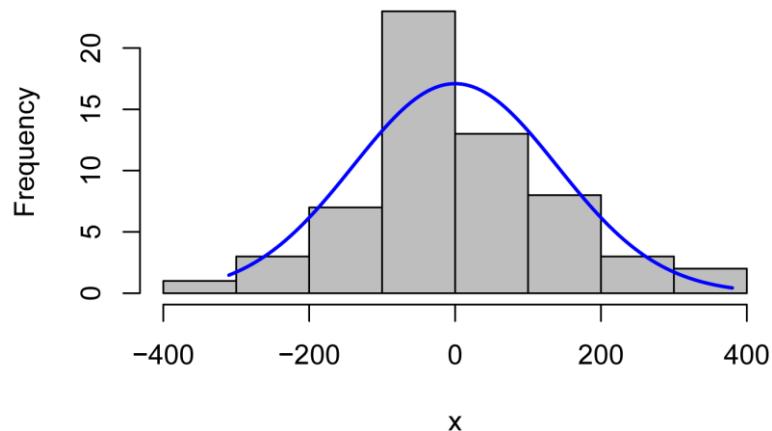
Instructor	290146	2	6.0173	0.004658	**
Supplement	306125	3	4.2324	0.009841	**
Instructor:Supplement	24438	6	0.1689	0.983880	
Residuals	1157250	48			

Histogram of residuals

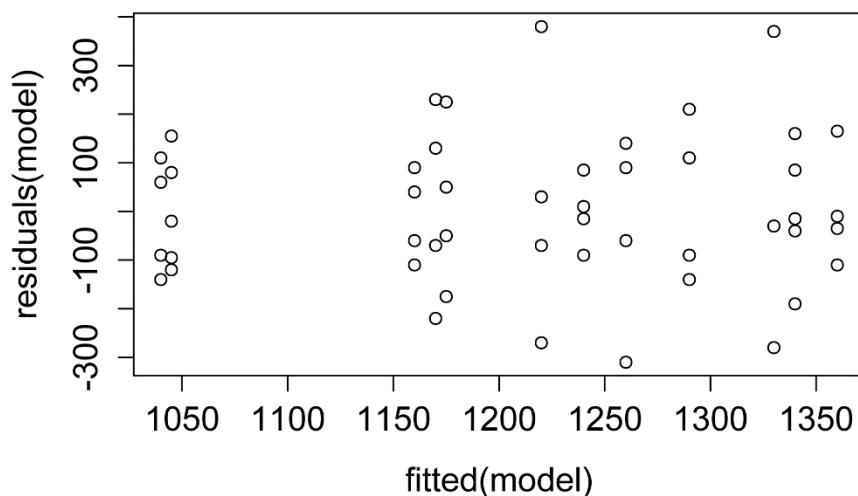
```
x = residuals(model)
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```



```
plot(fitted(model),  
      residuals(model))
```



```
plot(model) ### Additional model checking plots
```

Post-hoc analysis: mean separation tests

For an explanation of using estimated marginal means for multiple comparisons, see the section “Post-hoc analysis: mean separation tests” in the *One-way ANOVA* chapter.

EM means for main effects

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
~ Instructor)

pairs(marginal,
adjust="tukey")

contrast           estimate      SE df t.ratio p.value
Brendon Small - Coach McGuirk    41.25 49.1013 48   0.840  0.6802
Brendon Small - Melissa Robins  163.75 49.1013 48   3.335  0.0046
Coach McGuirk - Melissa Robins 122.50 49.1013 48   2.495  0.0418

Results are averaged over the levels of: Supplement
P value adjustment: tukey method for comparing a family of 3 estimates

CLD = cld(marginal,
alpha = 0.05,
Letters = letters,          ### Use lowercase letters for .group
adjust = "tukey")           ### Tukey-adjusted comparisons

CLD

Instructor      emmean      SE df lower.CL upper.CL .group
Melissa Robins 1123.75 34.71986 48 1053.941 1193.559 a
Coach McGuirk  1246.25 34.71986 48 1176.441 1316.059 b
Brendon Small   1287.50 34.71986 48 1217.691 1357.309 b

Results are averaged over the levels of: Supplement
Confidence level used: 0.95
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05
```

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
~ Supplement)

pairs(marginal,
adjust="tukey")
```

contrast	estimate	SE	df	t.ratio	p.value
A - B	-160.00000	56.6973	48	-2.822	0.0338
A - C	6.666667	56.6973	48	0.118	0.9994
A - D	-110.00000	56.6973	48	-1.940	0.2253
B - C	166.66667	56.6973	48	2.940	0.0251
B - D	50.000000	56.6973	48	0.882	0.8142
C - D	-116.666667	56.6973	48	-2.058	0.1818

Results are averaged over the levels of: Instructor
 P value adjustment: tukey method for comparing a family of 4 estimates

```
CLD = cld(marginal,
           alpha = 0.05,
           Letters = letters,      ### Use lowercase letters for .group
           adjust = "tukey")       ### Tukey-adjusted comparisons
```

CLD

Supplement	emmmean	SE	df	lower.CL	upper.CL	.group
C	1146.667	40.09104	48	1066.058	1227.275	a
A	1153.333	40.09104	48	1072.725	1233.942	a
D	1263.333	40.09104	48	1182.725	1343.942	ab
B	1313.333	40.09104	48	1232.725	1393.942	b

Results are averaged over the levels of: Instructor
 Confidence level used: 0.95
 P value adjustment: tukey method for comparing a family of 4 estimates
 significance level used: alpha = 0.05

Plot of means and confidence intervals for main effects

Because the interaction term was not significant, we would probably want to present means for the main effects only. Some authors would suggest re-doing the analysis without the interaction included.

For different data, the *nudge_y* parameter will need to be adjusted manually.

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
                    ~ Instructor)

CLD = cld(marginal,
           alpha = 0.05,
           Letters = letters,      ### Use lowercase letters for .group
           adjust = "tukey")       ### Tukey-adjusted comparisons

### Order the levels for printing
```

```
CLD$Instructor = factor(CLD$Instructor,
                         levels=c("Brendon Small",
                                  "Coach McGuirk",
                                  "Melissa Robins"))

### Remove spaces in .group

CLD$.group=gsub(" ", "", CLD$.group)

### Plot

library(ggplot2)

ggplot(CLD,
       aes(x      = Instructor,
           y      = emmean,
           label = .group)) +

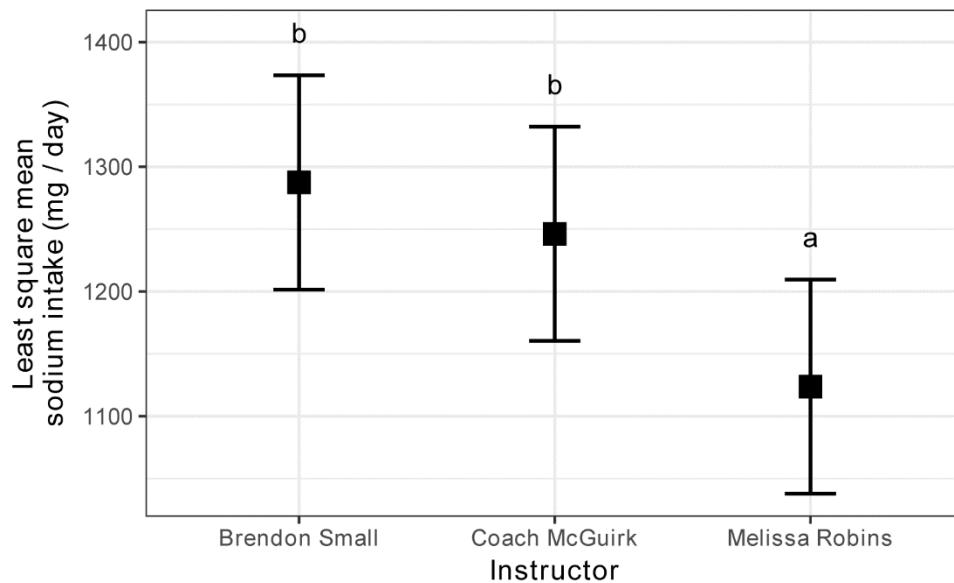
  geom_point(shape  = 15,
             size   = 4) +

  geom_errorbar(aes(ymin  = lower.CL,
                     ymax   = upper.CL),
                width = 0.2,
                size  = 0.7) +

  theme_bw() +
  theme(axis.title  = element_text(face = "bold"),
        axis.text    = element_text(face = "bold"),
        plot.caption = element_text(hjust = 0)) +

  ylab("Estimated marginal mean\nsodium intake (mg / day)") +

  geom_text(nudge_x = c(0, 0, 0),
            nudge_y = c(120, 120, 120),
            color   = "black")
```



Estimated marginal mean daily sodium intake for students in each of three classes with four different supplemental programs. Error bars indicate confidence intervals for the estimated marginal means. Groups sharing the same letter are not significantly different (alpha = 0.05, Tukey-adjusted).

```

library(multcompView)
library(emmeans)

marginal = emmeans(model,
                    ~ Supplement)

CLD = cld(marginal,
            alpha    = 0.05,
            Letters = letters,           ### Use lowercase letters for .group
            adjust   = "tukey")          ### Tukey-adjusted comparisons

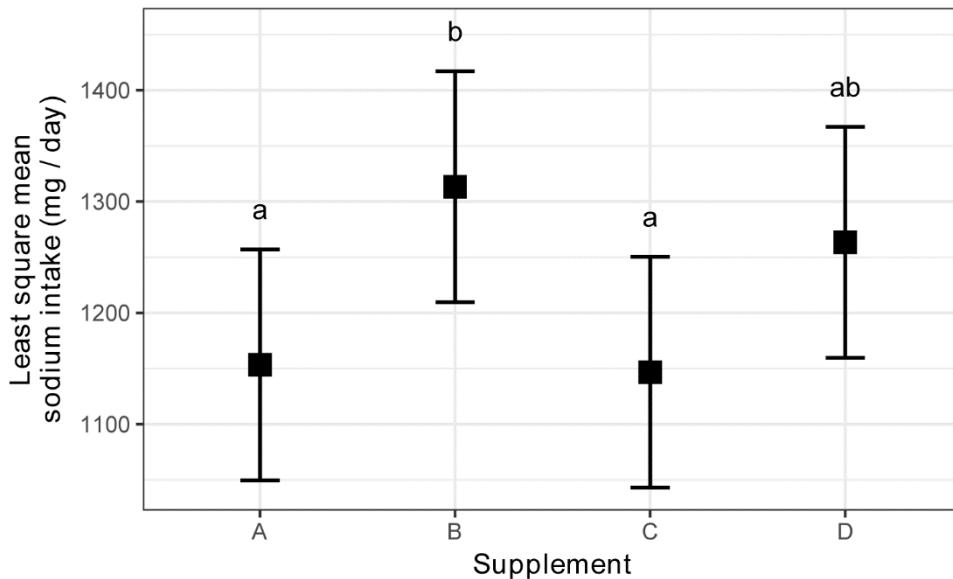
### Order the levels for printing
CLD$Supplement = factor(CLD$Supplement,
                         levels=c("A", "B", "C", "D"))

### Remove spaces in .group
CLD$.group=gsub(" ", "", CLD$.group)

### Plot
library(ggplot2)

```

```
ggplot(CLD,
  aes(x      = Supplement,
      y      = emmean,
      label = .group)) +
  geom_point(shape = 15,
             size  = 4) +
  geom_errorbar(aes(ymin = lower.CL,
                     ymax  = upper.CL),
                width = 0.2,
                size  = 0.7) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold"),
        axis.text   = element_text(face = "bold"),
        plot.caption = element_text(hjust = 0)) +
  ylab("Estimated marginal mean\nsodium intake (mg / day)") +
  geom_text(nudge_x = c(0, 0, 0, 0),
            nudge_y = c(140, 140, 140, 140),
            color   = "black")
```



Estimated marginal mean daily sodium intake for students in each of three classes with four different supplemental programs. Error bars indicate confidence intervals for the estimated marginal means. Groups sharing the same letter are not significantly different (alpha = 0.05, Tukey-adjusted).

Plot of EM means and confidence intervals for interaction

If the interaction effect were significant, it would make sense to present an interaction plot of the means for both main effects together on one plot. We can indicate mean separations with letters on the plot.

For different data, the *nudge_y* parameter will need to be adjusted manually.

```

library(multcompView)

library(emmeans)

marginal = emmeans(model,
                    ~ Instructor + Supplement)

CLD = cld(marginal,
            alpha    = 0.05,
            Letters = letters,      ##### Use lowercase letters for .group
            adjust   = "tukey")      ##### Tukey-adjusted comparisons

##### Order the levels for printing

CLD$Instructor = factor(CLD$Instructor,
                         levels=c("Brendon Small",
                                  "Coach McGuirk",
                                  "Melissa Robins"))

CLD$Supplement = factor(CLD$Supplement,
                         levels=c("A", "B", "C", "D"))

##### Remove spaces in .group

CLD$.group=gsub(" ", "", CLD$.group)

CLD

##### Plot

library(ggplot2)

pd = position_dodge(0.6)      ##### How much to jitter the points on the plot

ggplot(CLD,
       aes(x      = Instructor,
           y      = emmean,
           color = Supplement,
           label = .group)) +

  geom_point(shape  = 15,
             size   = 4,
             position = pd) +

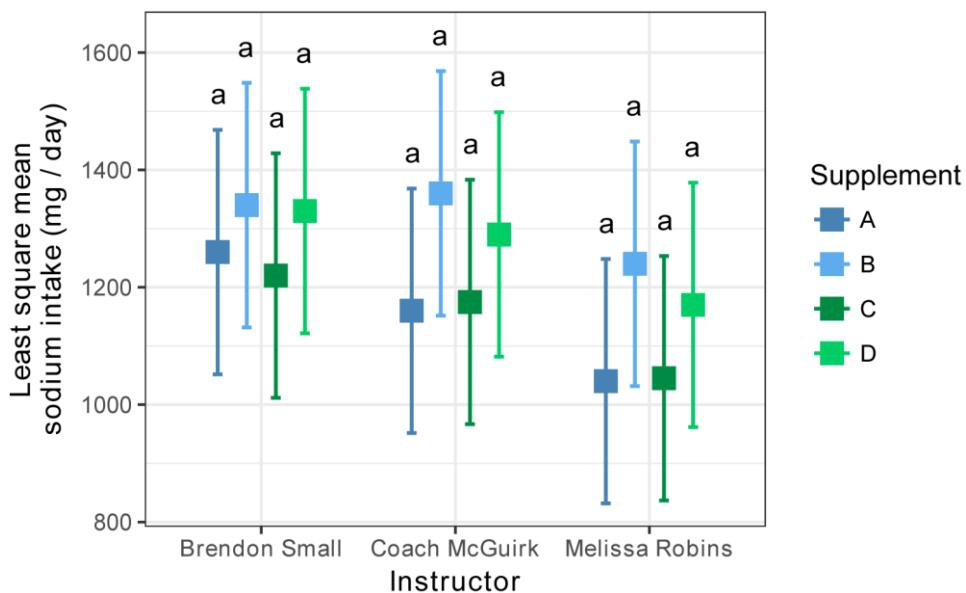
  geom_errorbar(aes(ymin  = lower.CL,
                    ymax   = upper.CL),
                width = 0.2,
                size  = 0.7,

```

```

position = pd) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold"),
        axis.text = element_text(face = "bold"),
        plot.caption = element_text(hjust = 0)) +
  ylab("Estimated marginal mean\nsodium intake (mg / day)") +
  geom_text(nudge_x = c(-0.22, 0.08, -0.22, 0.22,
                        0.08, 0.08, -0.08, -0.22,
                        0.22, 0.22, -0.08, -0.08),
            nudge_y = rep(270, 12),
            color = "black") +
  scale_color_manual(values = c("steelblue", "steelblue2",
                                "springgreen4", "springgreen3"))

```



Estimated marginal mean daily sodium intake for students in each of three classes with four different supplemental programs. Error bars indicate confidence intervals for the estimated marginal means. Groups sharing the same letter are not significantly different (alpha = 0.05, Tukey-adjusted).

Exercises T

1. Considering Brendon, Melissa, and McGuirk's data with for instructional programs and supplements,
 - a. Is the experimental design balanced?
 - b. Does *Instructor* have a significant effect on sodium intake?
 - c. Does *Supplement* have a significant effect on sodium intake?

- d. Does the interaction of *Instructor* and *Supplement* have a significant effect on sodium intake?
- e. Are the residuals from the analysis reasonably normal and homoscedastic?
- f. Which instructors had classes with significantly different mean sodium intake from which others?
- g. Does this result correspond to what you would have thought from looking at the plot of the EM means and confidence intervals?
- h. How would you summarize the effect of the supplements?
2. As part of a professional skills program, a 4-H club tests its members for typing proficiency. Dr. Katz, and Laura, and Ben want to compare their students' mean typing speed between their classes, as well as for the software they used to teach typing.

Instructor	Software	words.per.minute
'Dr. Katz Professional Therapist'	A	35
'Dr. Katz Professional Therapist'	A	50
'Dr. Katz Professional Therapist'	A	55
'Dr. Katz Professional Therapist'	A	60
'Dr. Katz Professional Therapist'	B	65
'Dr. Katz Professional Therapist'	B	60
'Dr. Katz Professional Therapist'	B	70
'Dr. Katz Professional Therapist'	B	55
'Dr. Katz Professional Therapist'	C	45
'Dr. Katz Professional Therapist'	C	55
'Dr. Katz Professional Therapist'	C	60
'Dr. Katz Professional Therapist'	C	45
'Laura the Receptionist'	A	55
'Laura the Receptionist'	A	60
'Laura the Receptionist'	A	75
'Laura the Receptionist'	A	65
'Laura the Receptionist'	B	60
'Laura the Receptionist'	B	70
'Laura the Receptionist'	B	75
'Laura the Receptionist'	B	70
'Laura the Receptionist'	C	65
'Laura the Receptionist'	C	72
'Laura the Receptionist'	C	73
'Laura the Receptionist'	C	65
'Ben Katz'	A	55
'Ben Katz'	A	55
'Ben Katz'	A	70
'Ben Katz'	A	55
'Ben Katz'	B	65
'Ben Katz'	B	60
'Ben Katz'	B	70
'Ben Katz'	B	60
'Ben Katz'	C	60
'Ben Katz'	C	62

'Ben Katz'	C	63
'Ben Katz'	C	65

For each of the following, answer the question, and **show the output from the analyses you used to answer the question.**

- a. What was the EM mean typing speed for each instructor's class?
- b. Does *Instructor* have a significant effect on typing speed?
- c. Does *Software* have a significant effect on typing speed?
- d. Are the residuals reasonably normal and homoscedastic?
- e. Which instructors had classes with significantly different mean typing speed from which others?
- f. How do you summarize the results of the effect of *Software*.
- g. What is your summary of the analysis and practical conclusions?

Repeated Measures ANOVA

When an experimental design takes measurements on the same experimental unit over time, the analysis of the data must take into account the probability that measurements for a given experimental unit will be correlated in some way. For example, if we were measuring calorie intake for students, we would expect that if one student had a higher intake at *Time 1*, that that student would have a higher intake others at *Time 2*, and so on.

In previous chapters, our approach to deal with non-independent observations was to treat *Student* as a blocking variable or as a random (blocking) variable.

The approach in this chapter is to include an autocorrelation structure in the model using the *nlme* package.

Packages used in this chapter

The packages used in this chapter include:

- psych
- nlme
- car
- multcompView
- emmeans
- ggplot2

- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(nlme)){install.packages("nlme")}
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Repeated measures ANOVA example

In this example, students were asked to document their daily caloric intake once a month for six months. Students were divided into three groups with each receiving instruction in nutrition education using one of three curricula.

There are different ways we might approach this problem. If we simply wanted to see if one curriculum was better at decreasing caloric intake in students, we might do a simple analysis of variance on the difference between each student's final and initial intake.

In the approach here we will use a repeated measures analysis with all the measurements, treating *Student* as a random variable to take into account native differences among students, and including an autocorrelation structure.

```
Input = "
Instruction      Student Month   Calories.per.day
'Curriculum A'    a       1       2000
'Curriculum A'    a       2       1978
'Curriculum A'    a       3       1962
'Curriculum A'    a       4       1873
'Curriculum A'    a       5       1782
'Curriculum A'    a       6       1737
'Curriculum A'    b       1       1900
'Curriculum A'    b       2       1826
'Curriculum A'    b       3       1782
'Curriculum A'    b       4       1718
'Curriculum A'    b       5       1639
'Curriculum A'    b       6       1644
'Curriculum A'    c       1       2100
'Curriculum A'    c       2       2067
'Curriculum A'    c       3       2065
'Curriculum A'    c       4       2015
'Curriculum A'    c       5       1994
'Curriculum A'    c       6       1919
'Curriculum A'    d       1       2000
'Curriculum A'    d       2       1981
'Curriculum A'    d       3       1987
'Curriculum A'    d       4       2016
'Curriculum A'    d       5       2010
```

```
'Curriculum A'    d    6    1946
'Curriculum B'    e    1    2100
'Curriculum B'    e    2    2004
'Curriculum B'    e    3    2027
'Curriculum B'    e    4    2109
'Curriculum B'    e    5    2197
'Curriculum B'    e    6    2294
'Curriculum B'    f    1    2000
'Curriculum B'    f    2    2011
'Curriculum B'    f    3    2089
'Curriculum B'    f    4    2124
'Curriculum B'    f    5    2199
'Curriculum B'    f    6    2234
'Curriculum B'    g    1    2000
'Curriculum B'    g    2    2074
'Curriculum B'    g    3    2141
'Curriculum B'    g    4    2199
'Curriculum B'    g    5    2265
'Curriculum B'    g    6    2254
'Curriculum B'    h    1    2000
'Curriculum B'    h    2    1970
'Curriculum B'    h    3    1951
'Curriculum B'    h    4    1981
'Curriculum B'    h    5    1987
'Curriculum B'    h    6    1969
'Curriculum C'    i    1    1950
'Curriculum C'    i    2    2007
'Curriculum C'    i    3    1978
'Curriculum C'    i    4    1965
'Curriculum C'    i    5    1984
'Curriculum C'    i    6    2020
'Curriculum C'    j    1    2000
'Curriculum C'    j    2    2029
'Curriculum C'    j    3    2033
'Curriculum C'    j    4    2050
'Curriculum C'    j    5    2001
'Curriculum C'    j    6    1988
'Curriculum C'    k    1    2000
'Curriculum C'    k    2    1976
'Curriculum C'    k    3    2025
'Curriculum C'    k    4    2047
'Curriculum C'    k    5    2033
'Curriculum C'    k    6    1984
'Curriculum C'    l    1    2000
'Curriculum C'    l    2    2020
'Curriculum C'    l    3    2009
'Curriculum C'    l    4    2017
'Curriculum C'    l    5    1989
'Curriculum C'    l    6    2020
")
```

```
Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them
```

```

Data$Instruction = factor(Data$Instruction,
                          levels=unique(Data$Instruction))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

Define model and conduct analysis of deviance

This example will use a mixed effects model to describe the repeated measures analysis, using the *lme* function in the *nlme* package. *Student* is treated as a random variable in the model.

The autocorrelation structure is described with the *correlation* statement. In this case, *corAR1* is used to indicate a temporal autocorrelation structure of order one, often abbreviated as AR(1). This statement takes the form:

```
correlation = Structure(form = ~ time | subjvar)
```

where:

- *Structure* is the autocorrelation structure. Options are listed in *library(nlme); ?corClasses*
- *time* is the variable indicating time. In this case, *Month*. For the corAR1 structure, the time variable must be an integer variable.
- *subjvar* indicates the variable for experimental units, in this case *Student*. Autocorrelation is modeled within levels of the *subjvar*, and not between them.

For the *corAR1* structure, a value for the first order correlation can be specified. In this case, the value of 0.429 is found using the *ACF* function in the “Optional analysis: determining autocorrelation in residuals” section below.

Autocorrelation structures can be chosen by either of two methods. The first is to choose a structure based on theoretical expectations of how the data should be correlated. The second is to try various autocorrelation structures and compare the resulting models with a criterion like AIC, AICc, or BIC to choose the structure that best models the data.

Optional technical note:

In this case, it isn't necessary to use a mixed effects model. That is, it's not necessary to include *Student* as a random variable. A similar analysis could be conducted by using the *gls* function in the *nlme* package, and including the *correlation* option, but excluding the *random* option, as follows in black.

```
library(nlme)

model = gls(Calories.per.day ~ Instruction + Month + Instruction*Month,
            correlation = corAR1(form = ~ Month | Student,
                                  value = 0.8990),
            data=Data,
            method="REML")

library(car)

Anova(model)
```

Code for analysis

```
library(nlme)

model = lme(Calories.per.day ~ Instruction + Month + Instruction*Month,
            random = ~1|student,
            correlation = corAR1(form = ~ Month | Student,
                                  value = 0.4287),
            data=Data,
            method="REML")
```

```
library(car)
```

```
Anova(model)
```

Analysis of Deviance Table (Type II tests)

```
Response: Calories.per.day
          Chisq Df Pr(>chisq)
Instruction      10.4221  2   0.005456 ***
Month           0.0198  1   0.888045
Instruction:Month 38.6045  2   4.141e-09 ***
```

Test the random effects in the model

The random effects in the model can be tested by comparing the model to a model fitted with just the fixed effects and excluding the random effects. Because there are not random effects in this second model, the *gls* function in the *nlme* package is used to fit this model.

```
model.fixed = gls(Calories.per.day ~ Instruction + Month + Instruction*Month,
                   data=Data,
                   method="REML")
```

```
anova(model,
      model.fixed)
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
model		1	9	716.9693	736.6762	-349.4847		
model.fixed		2	7	813.6213	828.9489	-399.8106	1 vs 2	100.652 <.0001

p-value and pseudo R-squared for model

The *nagelkerke* function can be used to calculate a *p*-value and pseudo *R-squared* value for the model.

One approach is to define the null model as one with no fixed effects except for an intercept, indicated with a 1 on the right side of the ~. And to also include the random effects, in this case 1/*Student*.

```
library(rcompanion)

model.null = lme(Calories.per.day ~ 1,
                  random = ~1|Student,
                  data = Data)

nagelkerke(model,
           model.null)

$Pseudo.R.squared.for.model.vs.null
                                Pseudo.R.squared
McFadden                           0.123969
Cox and Snell (ML)                 0.768652
Nagelkerke (Cragg and Uhler)       0.768658

$Likelihood.ratio.test
Df.diff LogLik.diff Chisq   p.value
-6        -52.698 105.4 1.8733e-20
```

Another approach to determining the *p*-value and pseudo *R-squared* for an *lme* model is to compare the model to a null model with only an intercept and neither the fixed nor the random effects. To accomplish this, the null model has to be specified with the *gls* function.

```
library(rcompanion)

model.null.2 = gls(Calories.per.day ~ 1,
                   data = Data)

nagelkerke(model,
           model.null.2)

$Pseudo.R.squared.for.model.vs.null
                                Pseudo.R.squared
McFadden                           0.168971
Cox and Snell (ML)                 0.877943
Nagelkerke (Cragg and Uhler)       0.877946

$Likelihood.ratio.test
```

```
Df.diff LogLik.diff Chisq p.value
-7      -75.717 151.43 2.0282e-29
```

Post-hoc analysis

The *emmeans* package is able to handle *lme* objects. For further details, see *?emmeans::models*. For a review of mean separation tests and estimated marginal means, see the chapters *What are Estimated Marginal Means?*; the chapter *Estimated Marginal Means for Multiple Comparisons*; and the “Post-hoc analysis: mean separation tests” section in the *One-way ANOVA* chapter.

Because *Month* is an integer variable, not a factor variable, it is listed in the *emmeans* *cld* table as its average only.

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
~ Instruction:Month)

cld(marginal,
alpha = 0.05,
Letters = letters,     ### Use lower-case letters for .group
adjust = "tukey")     ### Tukey-adjusted comparisons

Instruction Month emmean      SE df lower.CL upper.CL .group
Curriculum A   3.5 1907.601 42.83711 11 1787.206 2027.996 a
Curriculum C   3.5 1997.429 42.83711  9 1872.222 2122.637 ab
Curriculum B   3.5 2102.965 42.83711  9 1977.758 2228.172 b

Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.0505
```

Interaction plot

For this plot, we will use the *groupwiseMean* function to calculate the natural mean of each *Instruction* x *Month* combination, along with the confidence interval of each mean with the percentile method.

```
library(rcompanion)

Sum = groupwiseMean(Calories.per.day ~ Instruction + Month,
                     data      = Data,
                     conf      = 0.95,
                     digits    = 3,
                     traditional = FALSE,
                     percentile = TRUE)

Sum

  Instruction Month n Mean Conf.level Percentile.lower Percentile.upper
1 Curriculum A     1 4 2000        0.95          1920          2080
```

2	Curriculum A	2	4	1960	0.95	1860	2040
3	Curriculum A	3	4	1950	0.95	1830	2040
4	Curriculum A	4	4	1910	0.95	1760	2020
5	Curriculum A	5	4	1860	0.95	1710	2000
6	Curriculum A	6	4	1810	0.95	1690	1930
7	Curriculum B	1	4	2020	0.95	2000	2080
8	Curriculum B	2	4	2010	0.95	1980	2060
9	Curriculum B	3	4	2050	0.95	1990	2120
10	Curriculum B	4	4	2100	0.95	2020	2180
11	Curriculum B	5	4	2160	0.95	2040	2250
12	Curriculum B	6	4	2190	0.95	2040	2280
13	Curriculum C	1	4	1990	0.95	1960	2000
14	Curriculum C	2	4	2010	0.95	1990	2020
15	Curriculum C	3	4	2010	0.95	1990	2030
16	Curriculum C	4	4	2020	0.95	1990	2050
17	Curriculum C	5	4	2000	0.95	1990	2020
18	Curriculum C	6	4	2000	0.95	1990	2020

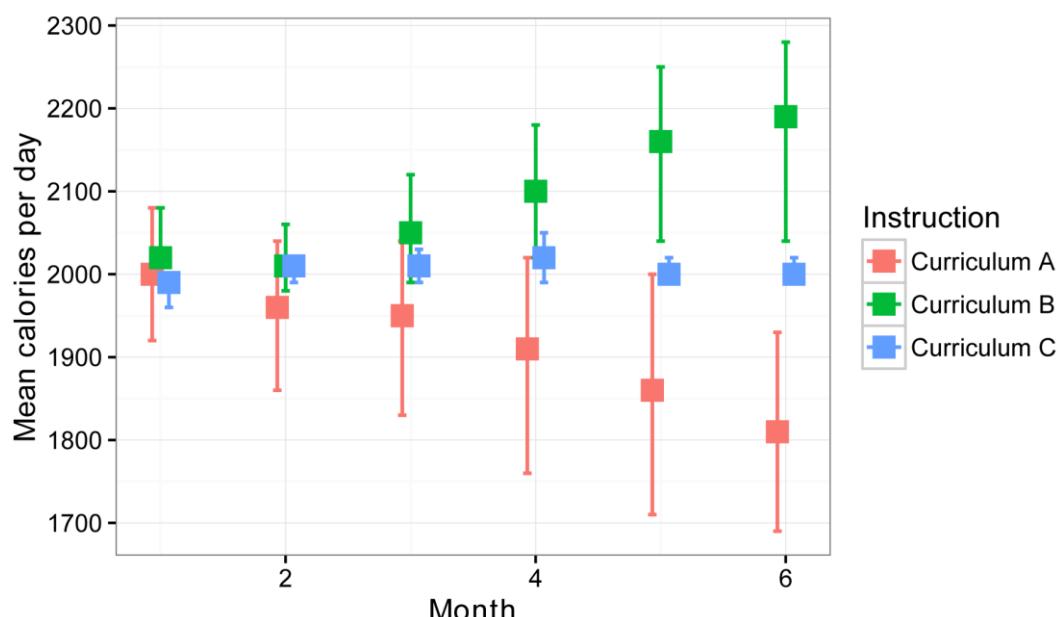
```

library(ggplot2)

pd = position_dodge(.2)

ggplot(Sum, aes(x = Month,
                 y = Mean,
                 color = Instruction)) +
  geom_errorbar(aes(ymin=Percentile.lower,
                     ymax=Percentile.upper),
                width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Mean calories per day")

```



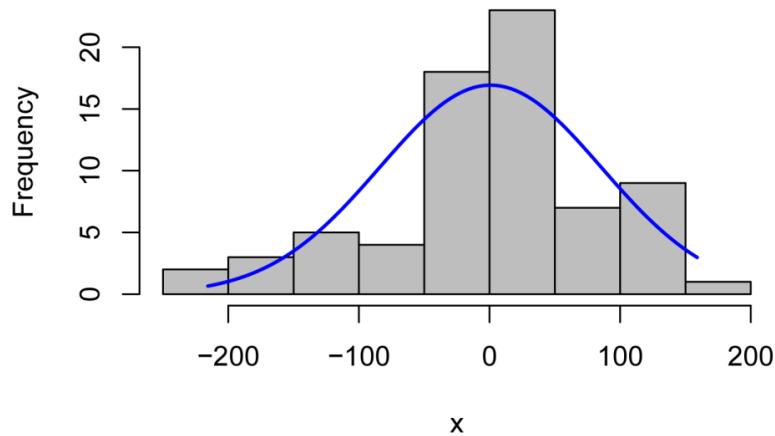
Histogram of residuals

Residuals from a mixed model fit with *nlme* should be normally distributed. Plotting residuals vs. fitted values, to check for homoscedasticity and independence, is probably also advisable.

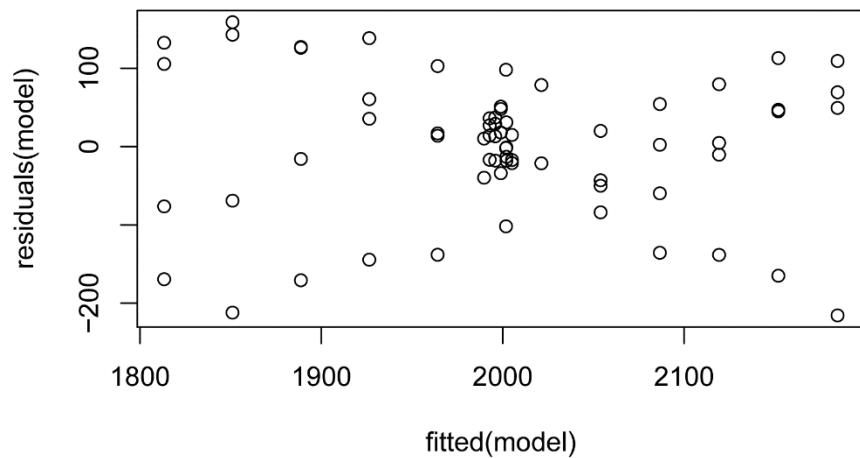
```
x = residuals(model)
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```



```
plot(fitted(model),  
      residuals(model))
```

**Optional analysis: determining autocorrelation in residuals**

The *ACF* function in the *nlme* package will indicate the autocorrelation for lags in the time variable. Note that for a *gls* model, the *form* of the autocorrelation structure can be specified. For an *lme* model, the function uses the innermost group level and assumes equally spaced intervals.

```

library(nlme)

model.a = gls(calories.per.day ~ Instruction + Month + Instruction*Month,
              data=Data)

ACF(model.a,
     form = ~ Month | Student)

  lag      ACF
1   0 1.0000000
2   1 0.8989822
3   2 0.7462712
4   3 0.6249217
5   4 0.5365430
6   5 0.4564673

library(nlme)

model.b = lme(calories.per.day ~ Instruction + Month + Instruction*Month,
               random = ~1|Student,
               data=Data)

ACF(model.b)

  lag      ACF
1   0 1.0000000
2   1 0.4286522
3   2 -0.1750233
4   3 -0.6283754
5   4 -0.8651088
6   5 -0.6957774

```

Optional discussion on specifying formulae for repeated measures analysis

Specifying random effects in models

The general schema for specifying random effects in R is:

```
Formula.for.random.effects | unit.for.which.these.apply
```

So that, in the *nlme* package:

- `random = ~ 1 | subject`
Indicates that each *subject* will have its own intercept
- `random = ~ rep | subject`
Indicates that each *subject* will have its own intercept and its own slope for *rep*
- `random = ~ 1 | rep/subject`

Indicates that each *subject-within-rep* unit will have its own intercept

Indicating time and subject variables

A simple repeated analysis statement in *proc mixed* in SAS could be specified with:

```
repeated date / subject = id type = AR(1)
```

A similar specification in with the *gls* function in *nlme* package in R would be:

```
correlation = corAR1(form = ~ date | id)
```

Likewise, a simple mixed effects repeated analysis statement in *proc mixed* in SAS could be specified with:

```
random id
repeated date / subject = id type = AR(1)
```

A similar specification in with the *lme* function in *nlme* package in R would be:

```
random = ~1 | id,
correlation = corAR1(form = ~ date | id)
```

Specifying nested effects

In repeated measures analysis, it is common to used nested effects. For example, if our subject variable is *treatment* within *block*,

In SAS:

```
treatment(block)
```

is equivalent to

```
block block*treatment
```

In R:

```
block/treatment
```

is equivalent to

```
block + block:treatment
```

Correlation and Linear Regression

Correlation and linear regression each explore the relationship between two quantitative variables. Both are very common analyses.

Correlation determines if one variable varies systematically as another variable changes. It does not specify that one variable is the dependent variable and the other is the independent variable. Often, it is useful to look at which variables are correlated to others in a data set, and it is especially useful to see which variables correlate to a particular variable of interest.

The three forms of correlation presented here are Pearson, Kendall, and Spearman. The test determining the p -value for Pearson correlation is a parametric test that assumes that data are bivariate normal. Kendall and Spearman correlation use nonparametric tests.

In contrast, linear regression specifies one variable as the independent variable and another as the dependent variable. The resultant model relates the variables with a linear relationship.

The tests associated with linear regression are parametric and assume normality, homoscedasticity, and independence of residuals, as well as a linear relationship between the two variables.

Appropriate data

- For Pearson correlation, two interval/ratio variables. Together the data in the variables are bivariate normal. The relationship between the two variables is linear. Outliers can detrimentally affect results.
- For Kendall correlation, two variables of interval/ratio or ordinal type.
- For Spearman correlation, two variables of interval/ratio or ordinal type.
- For linear regression, two interval/ratio variables. The relationship between the two variables is linear. Residuals are normal, independent, and homoscedastic. Outliers can affect the results unless robust methods are used.

Hypotheses

- For correlation, null hypothesis: The correlation coefficient (r , τ , or ρ) for the sampled population is zero. Or, there is no correlation between the two variables.
- For linear regression, null hypothesis: The slope of the fit line for the sampled population is zero. Or, there is no linear relationship between the two variables.

Interpretation

- For correlation, reporting significant results as “Variable A was significantly correlated to Variable B” is acceptable. Alternatively, “A significant correlation between Variable A and Variable B was found.” Or, “Variables A, B, and C were significantly correlated.”

- For linear regression, reporting significant results as “Variable A was significantly linearly related to Variable B” is acceptable. Alternatively, “A significant linear regression between Variable A and Variable B was found.”

Packages used in this chapter

The packages used in this chapter include:

- psych
- PerformanceAnalytics
- ggplot2
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(PerformanceAnalytics)){install.packages("PerformanceAnalytics")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Examples for correlation and linear regression

Brendon Small and company recorded several measurements for students in their classes related to their nutrition education program: *Grade*, *Weight* in kilograms, intake of *Calories* per day, daily *Sodium* intake in milligrams, and *Score* on the assessment of knowledge gain.

```
Input = "
Instructor      Grade    Weight   Calories  Sodium  Score
'Brendon Small'    6        43     2069    1287    77
'Brendon Small'    6        41     1990    1164    76
'Brendon Small'    6        40     1975    1177    76
'Brendon Small'    6        44     2116    1262    84
'Brendon Small'    6        45     2161    1271    86
'Brendon Small'    6        44     2091    1222    87
'Brendon Small'    6        48     2236    1377    90
'Brendon Small'    6        47     2198    1288    78
'Brendon Small'    6        46     2190    1284    89
'Jason Penopolis'  7        45     2134    1262    76
'Jason Penopolis'  7        45     2128    1281    80
'Jason Penopolis'  7        46     2190    1305    84
'Jason Penopolis'  7        43     2070    1199    68
'Jason Penopolis'  7        48     2266    1368    85
'Jason Penopolis'  7        47     2216    1340    76
'Jason Penopolis'  7        47     2203    1273    69
'Jason Penopolis'  7        43     2040    1277    86
'Jason Penopolis'  7        48     2248    1329    81
'Melissa Robins'   8        48     2265    1361    67
'Melissa Robins'   8        46     2184    1268    68
'Melissa Robins'   8        53     2441    1380    66
'Melissa Robins'   8        48     2234    1386    65
'Melissa Robins'   8        52     2403    1408    70
'Melissa Robins'   8        53     2438    1380    83"
```

```
'Melissa Robins'    8      52     2360    1378    74
'Melissa Robins'    8      51     2344    1413    65
'Melissa Robins'    8      51     2351    1400    68
'Paula Small'       9      52     2390    1412    78
'Paula Small'       9      54     2470    1422    62
'Paula Small'       9      49     2280    1382    61
'Paula Small'       9      50     2308    1410    72
'Paula Small'       9      55     2505    1410    80
'Paula Small'       9      52     2409    1382    60
'Paula Small'       9      53     2431    1422    70
'Paula Small'       9      56     2523    1388    79
'Paula Small'       9      50     2315    1404    71
'Coach McGuirk'    10     52     2406    1420    68
'Coach McGuirk'    10     58     2699    1405    65
'Coach McGuirk'    10     57     2571    1400    64
'Coach McGuirk'    10     52     2394    1420    69
'Coach McGuirk'    10     55     2518    1379    70
'Coach McGuirk'    10     52     2379    1393    61
'Coach McGuirk'    10     59     2636    1417    70
'Coach McGuirk'    10     54     2465    1414    59
'Coach McGuirk'    10     54     2479    1383    61
")
```

```
Data = read.table(textConnection(Input), header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))
```

Check the data frame

```
library(psych)
headTail(Data)
str(Data)
summary(Data)
```

Remove unnecessary objects

```
rm(Input)
```

Visualizing correlated variables

Correlation does not have independent and dependent variables

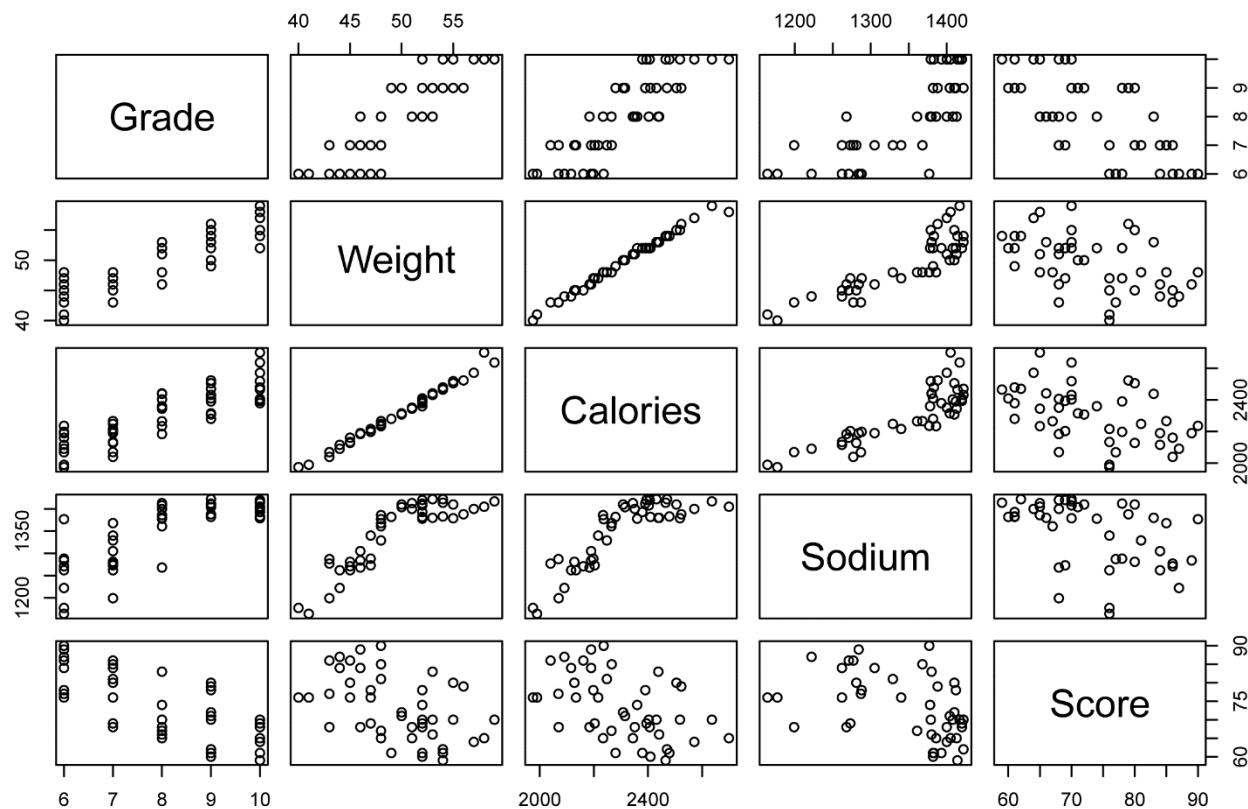
Notice in the formulae for the *pairs* and *cor.test* functions, that all variables are named to the right of the *~*. One reason for this is that correlation is measured between two variables without assuming

that one variable is the dependent variable and one is the independent variable. By necessity, one variable must be plotted on the x -axis and one on the y -axis, but their placement is arbitrary.

Multiple correlation

The *pairs* function can plot multiple numeric or integer variables on a single plot to look for correlations among the variables.

```
pairs(data=Data,
~ Grade + Weight + Calories + Sodium + Score)
```



The *corr.test* function in the *psych* package can be used in a similar manner, with the output being a table of correlation coefficients and a table of p -values. p -values can be adjusted with the *adjust*= option. Options for correlation methods are “pearson”, “kendall”, and “spearman”. The function can produce confidence intervals for the correlation coefficients, but I recommend using the *cor.ci* function in the *psych* package for this task (not shown here).

The *corr.test* function requires that the data frame contain only numeric or integer variables, so we will first create a new data frame called *Data.num* containing only the numeric and integer variables.

```
Data.num = Data[c("Grade", "Weight", "Calories", "Sodium", "Score")]
```

```
library(psych)
```

```
corr.test(Data.num,
```

```
use      = "pairwise",
method  = "pearson",
adjust   = "none")
```

Correlation matrix

	Grade	Weight	Calories	Sodium	Score
Grade	1.00	0.85	0.85	0.79	-0.70
Weight	0.85	1.00	0.99	0.87	-0.48
Calories	0.85	0.99	1.00	0.85	-0.48
Sodium	0.79	0.87	0.85	1.00	-0.45
Score	-0.70	-0.48	-0.48	-0.45	1.00

Sample size
[1] 45

A useful plot of histograms, correlations, and correlation coefficients can be produced with the *chart.Correlation* function in the *PerformanceAnalytics* package.

In the output, the numbers represent the correlation coefficients (*r*, *tau*, or *rho*, depending on whether Pearson, Kendall, or Spearman correlation is selected, respectively). The stars represent the *p*-value of the correlation:

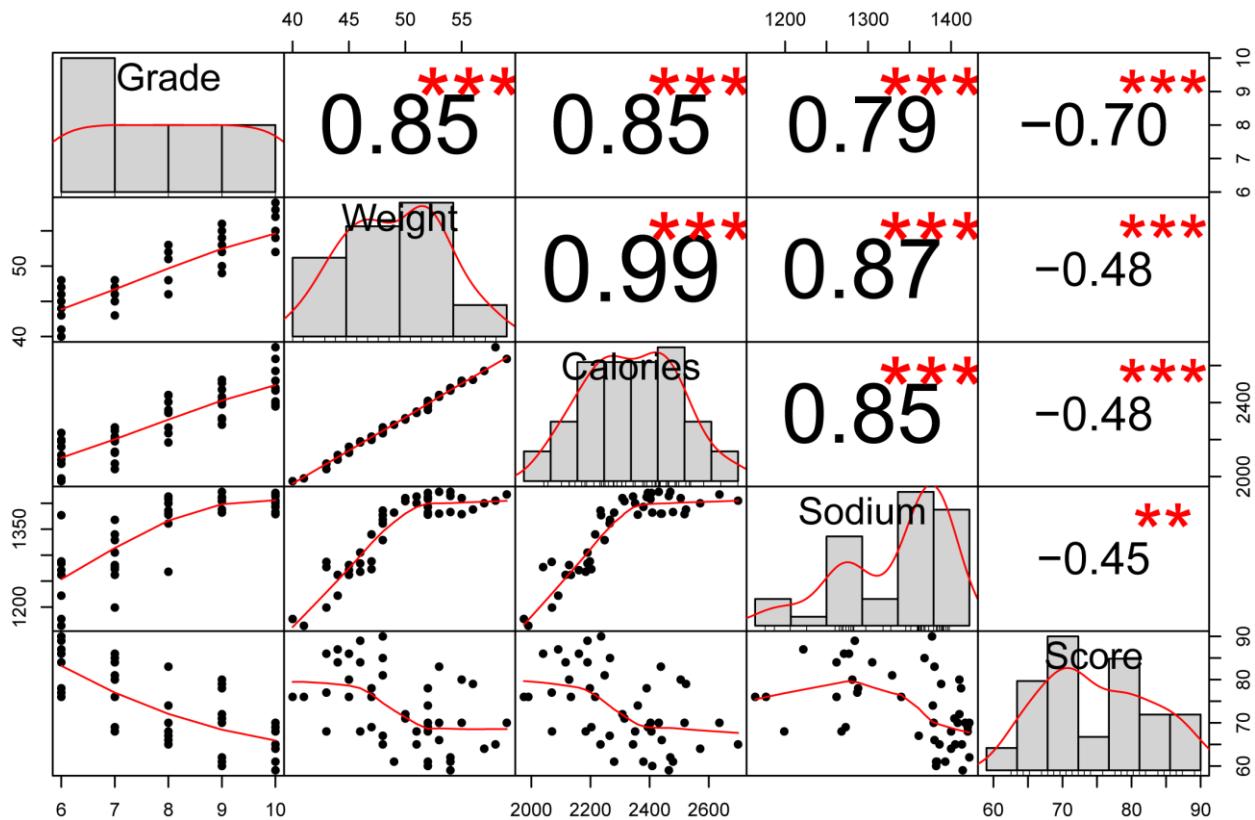
*	<i>p</i> < 0.05
**	<i>p</i> < 0.01
***	<i>p</i> < 0.001

The function requires that the data frame contain only numeric or integer variables, so we will first create a new data frame called *Data.num* containing only the numeric and integer variables.

```
Data.num = Data[c("Grade", "Weight", "Calories", "Sodium", "Score")]

library(PerformanceAnalytics)

chart.Correlation(Data.num,
                  method="pearson",
                  histogram=TRUE,
                  pch=16)
```



Effect size statistics

r, rho, and tau

The statistics r , ρ , and τ are used as effect sizes for Pearson, Spearman, and Kendall regression, respectively. These statistics vary from -1 to 1 , with 0 indicating no correlation, 1 indicating a perfect positive correlation, and -1 indicating a perfect negative correlation. Like other effect size statistics, these statistics are not affected by sample size.

Interpretation of effect sizes necessarily varies by discipline and the expectations of the experiment. They should not be considered universal. An interpretation of r is given by Cohen (1988). It is probably reasonable to use similar interpretations for ρ and τ .

	<u>small</u>	<u>medium</u>	<u>large</u>
r	$0.10 - < 0.30$	$0.30 - < 0.50$	≥ 0.50

Adapted from Cohen (1988).

r-squared

For linear regression, r -squared is used as an effect size statistic. It indicates the proportion of the variability in the dependent variable that is explained by model. That is, an r -squared of 0.60 indicates that 60% of the variability in the dependent variable is explained by the model.

Pearson correlation

The test for Pearson correlation is a parametric analysis that requires that the relationship between the variables is linear, and that the data be bivariate normal. Variables should be interval/ratio. The test is sensitive to outliers.

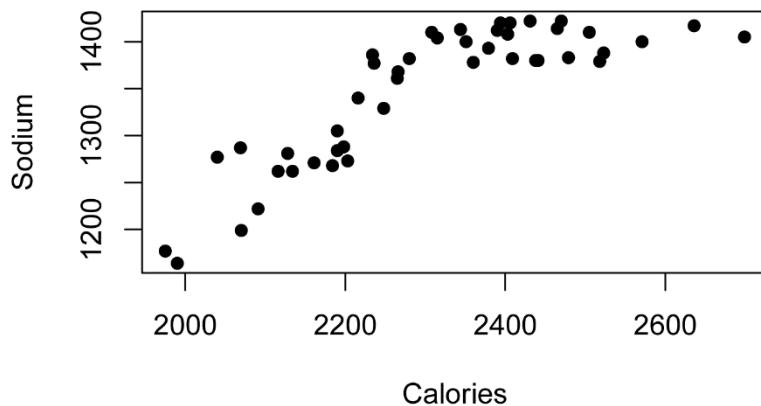
The correlation coefficient, r , can range from +1 to -1, with +1 being a perfect positive correlation and -1 being a perfect negative correlation. An r of 0 represents no correlation whatsoever. The hypothesis test determines if the r value is significantly different from 0.

As an example, we'll plot *Sodium* vs. *Calories*, and use the *cor.test* function to test the correlation of these two variables.

Simple plot of the data

Note that the relationship between the y and x variables is not particularly linear.

```
plot(Sodium ~ Calories,
      data=Data,
      pch=16,
      xlab = "Calories",
      ylab = "Sodium")
```

Pearson correlation

Note that the results report the p -value for the hypothesis test as well as the r value, written as *cor*, 0.849.

```
cor.test(~ Sodium + Calories,
        data=Data,
        method = "pearson")
```

Pearson's product-moment correlation

```
t = 10.534, df = 43, p-value = 1.737e-13
alternative hypothesis: true correlation is not equal to 0
```

```
95 percent confidence interval:
0.7397691 0.9145785
```

```
sample estimates:
  cor
0.8489548
```

Plot residuals

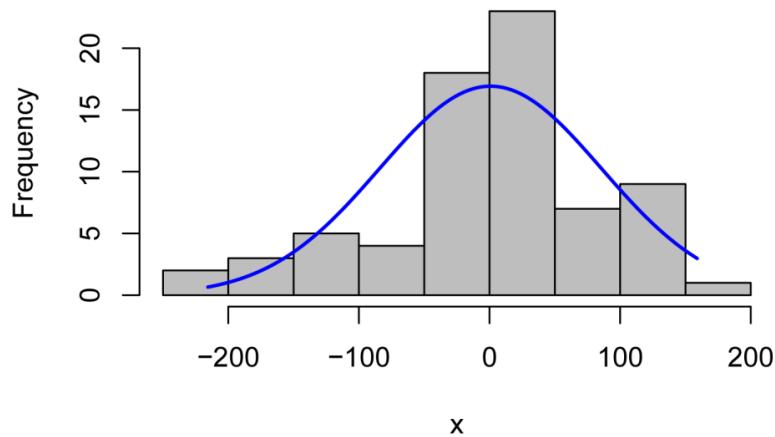
It's not a bad idea to look at the residuals from Pearson correlation to be sure the data meet the assumption of bivariate normality. Unfortunately, the *cor.test* function doesn't supply residuals. One solution is to use the *lm* function, which actually redoing the analysis as a linear regression.

```
model = lm(Sodium ~ Calories,
           data = Data)
```

```
x = residuals(model)
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```



Kendall correlation

Kendall correlation is considered a nonparametric analysis. It is a rank-based test that does not require assumptions about the distribution of the data. Variables can be interval/ratio or ordinal.

The correlation coefficient from the test is *tau*, which can range from +1 to -1, with +1 being a perfect positive correlation and -1 being a perfect negative correlation. A *tau* of 0 represents no correlation whatsoever. The hypothesis test determines if the *tau* value is significantly different from 0.

As a technical note, the *cor.test* function in R calculates *tau-b*, which handles ties in ranks well.

The test is relatively robust to outliers in the data. The test is sometimes cited for being reliable when there are small number of samples or when there are many ties in ranks.

Simple plot of the data

(See Pearson correlation above.)

Kendall correlation

Note that the results report the *p*-value for the hypothesis test as well as the *tau* value, written as *tau*, 0.649.

```
cor.test( ~ Sodium + Calories,
          data=Data,
          method = "kendall")

Kendall's rank correlation tau

z = 6.2631, p-value = 3.774e-10
alternative hypothesis: true tau is not equal to 0

sample estimates:
      tau
0.6490902
```

Spearman correlation

Spearman correlation is considered a nonparametric analysis. It is a rank-based test that does not require assumptions about the distribution of the data. Variables can be interval/ratio or ordinal.

The correlation coefficient from the test, *rho*, can range from +1 to -1, with +1 being a perfect positive correlation and -1 being a perfect negative correlation. A *rho* of 0 represents no correlation whatsoever. The hypothesis test determines if the *rho* value is significantly different from 0.

Spearman correlation is probably most often used with ordinal data. It tests for a monotonic relationship between the variables. It is relatively robust to outliers in the data.

Simple plot of the data

(See Pearson correlation above.)

Spearman correlation

Note that the results report the *p*-value for the hypothesis test as well as the *rho* value, written as *rho*, 0.820.

```
cor.test( ~ Sodium + Calories,
          data=Data,
          method = "spearman")

Spearman's rank correlation rho

s = 2729.7, p-value = 5.443e-12
alternative hypothesis: true rho is not equal to 0

sample estimates:
      rho
0.8201766
```

Linear regression

Linear regression is a very common approach to model the relationship between two interval/ratio variables. The method assumes that there is a linear relationship between the dependent variable and the independent variable, and finds a best fit model for this relationship.

Dependent and Independent variables

When plotted, the dependent variable is usually placed on the y -axis, and the independent variable is usually placed in the x -axis.

Interpretation of coefficients

The outcome of linear regression includes estimating the intercept and the slope of the linear model. Linear regression can then be used as a predictive model, whereby the model can be used to predict a y value for any given x . In practice, the model shouldn't be used to predict values beyond the range of the x values used to develop the model.

Multiple, nominal, and ordinal independent variables

If there are multiple independent variables of interval/ratio type in the model, then linear regression expands to multiple regression. The polynomial regression example in this chapter is a form of multiple regression.

If the independent variable were of nominal type, then the linear regression would become a one-way analysis of variance.

Handling independent variables of ordinal type can be complicated. Often they are treated as either nominal type or interval/ratio type, although there are drawbacks to each approach.

Assumptions

Linear regression assumes a linear relationship between the two variables, normality of the residuals, independence of the residuals, and homoscedasticity of residuals.

Note on writing r-squared

For bivariate linear regression, the *r-squared* value often uses a lower case r ; however, some authors prefer to use a capital R . For multiple regression, the R in the *R-squared* value is usually capitalized. The name of the statistic may be written out as "r-squared" for convenience, or as r^2 .

Define model, and produce model coefficients, p-value, and r-squared value

Linear regression can be performed with the *lm* function, which was the same function we used for analysis of variance.

The *summary* function for *lm* model objects includes estimates for model parameters (intercept and slope), as well as an *r-squared* value for the model and *p*-value for the model.

Note that even for bivariate regression, the output calls the *r-squared* value "Multiple R-squared".

```
model = lm(Sodium ~ Calories,
           data = Data)
```

```
summary(model)
```

Coefficients:

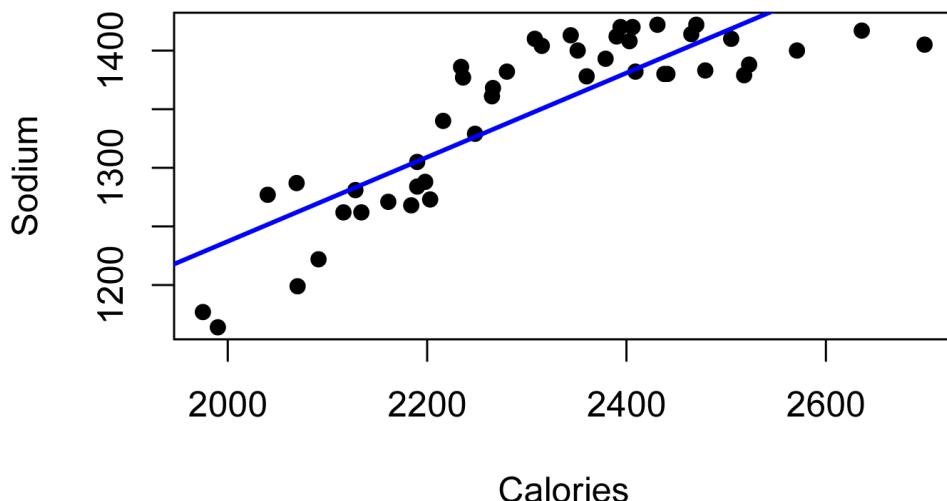
	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	519.07547	78.78211	6.589	5.09e-08 ***
Calories	0.35909	0.03409	10.534	1.74e-13 ***

Residual standard error: 38.89 on 43 degrees of freedom
 Multiple R-squared: 0.7207, Adjusted R-squared: 0.7142
 F-statistic: 111 on 1 and 43 DF, p-value: 1.737e-13

Plot data with best fit line

```
plot(Sodium ~ Calories,
      data=Data,
      pch=16,
      xlab = "Calories",
      ylab = "Sodium")

abline(model,
       col = "blue",
       lwd = 2)
```



There is a clear nonlinearity to the data, suggesting that the simple linear model is not the best fit. This nonlinearity is apparent in the plot of residuals vs. fitted values as well.

The next chapter will include fitting linear plateau, quadratic plateau, and a curvilinear models with this data.

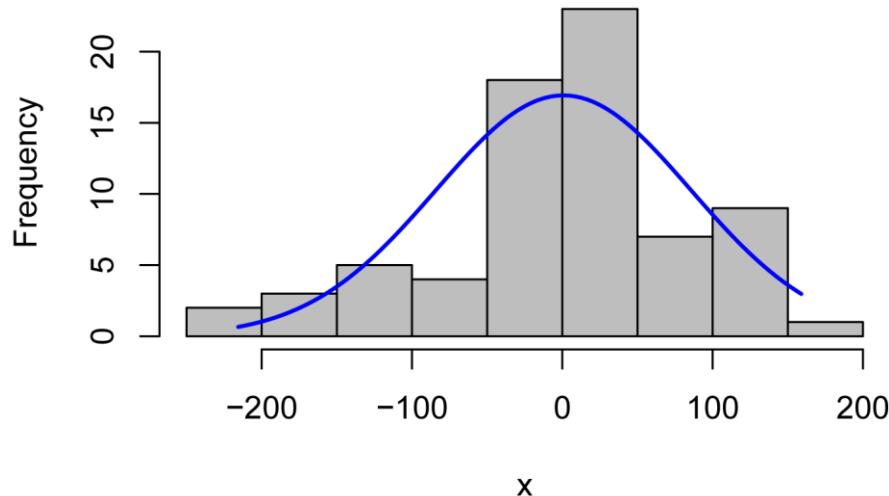
The following section will attempt to improve the model fit by adding polynomial terms.

Plots of residuals

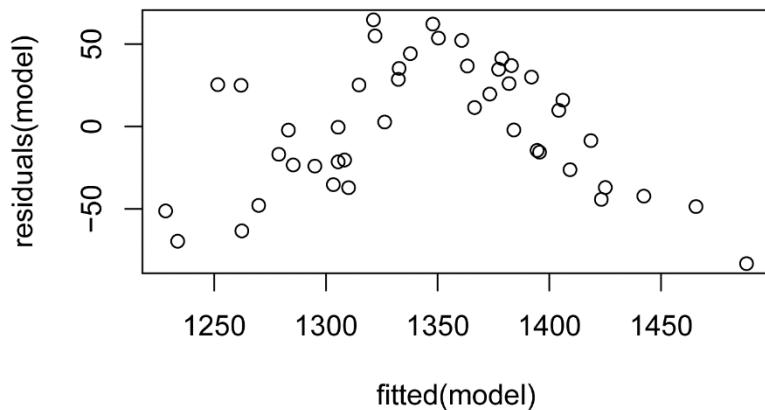
```
x = residuals(model)
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```



```
plot(fitted(model),  
      residuals(model))
```

***Polynomial regression***

Polynomial regression adds additional terms to the model, so that the terms include some set of the linear, quadratic, cubic, and quartic, etc., forms of the independent variable. These terms are the independent variable, the square of the independent variable, the cube of the independent variable, and so on.

Create polynomial terms in the data frame

Because the variable *Calories* is an integer variable, we will need to convert it to a numeric variable. Otherwise R will produce errors when we try to square and cube the values of the variable.

Our new variables will be *Calories2* for the square of *Calories*, *Calories3* for the cube of *Calories*, and so on.

```
Data$Calories = as.numeric(Data$Calories)

Data$Calories2 = Data$Calories * Data$Calories
Data$Calories3 = Data$Calories * Data$Calories * Data$Calories
Data$Calories4 = Data$Calories * Data$Calories * Data$Calories * Data$Calories
```

Define models and determine best model

Chances are that we will not need all of the polynomial terms to adequately model our data. One approach to choosing the best model is to construct several models with increasing numbers of polynomial terms, and then use a model selection criterion like AIC, AICc, or BIC to choose the best one. These criteria balance the goodness-of-fit of each model versus its complexity. That is, the goal is to find a model which adequately explains the data without having too many terms.

We can use the *compareLM* function to list AIC, AICc, and BIC for a series of models.

```
model.1 = lm(Sodium ~ Calories,
             data = Data)

model.2 = lm(Sodium ~ Calories + Calories2,
             data = Data)

model.3 = lm(Sodium ~ Calories + Calories2 + Calories3,
             data = Data)

model.4 = lm(Sodium ~ Calories + Calories2 + Calories3 + Calories4,
             data = Data)

library(rcompanion)

compareLM(model.1, model.2, model.3, model.4)

$Models
  Formula
1 "Sodium ~ Calories"
2 "Sodium ~ Calories + Calories2"
3 "Sodium ~ Calories + Calories2 + Calories3"
4 "Sodium ~ Calories + Calories2 + Calories3 + Calories4"

$Fit.criteria
  Rank Df.res   AIC   AICc     BIC R.squared Adj.R.sq    p.value Shapiro.W Shapiro.p
1     2      43 461.1 461.7 466.5     0.7207    0.7142 1.737e-13    0.9696    0.2800
2     3      42 431.4 432.4 438.6     0.8621    0.8556 8.487e-19    0.9791    0.5839
3     4      41 433.2 434.7 442.2     0.8627    0.8526 1.025e-17    0.9749    0.4288
```

4	5	40	428.5	430.8	439.4	0.8815	0.8696	5.566e-18	0.9625	0.1518
---	---	----	-------	-------	-------	--------	--------	-----------	--------	--------

If we use BIC as our model fit criterion, we would choose *model.2* as the best model for these data, since it had the lowest BIC. AIC or AICc could have been used as criteria instead.

BIC tends to penalize models more than the other criteria for having additional parameters, so it will tend to choose models with fewer terms.

There is not generally accepted advice as to which model fitting criterion to use. My advice might be to use BIC when a more parsimonious model (one with fewer terms) is a priority, and in other cases use AICc.

For final model, produce model coefficients, p-value, and R-squared value

```
summary(model.2)
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -3.707e+03  6.463e+02 -5.736 9.53e-07 ***
Calories     4.034e+00  5.604e-01   7.198 7.58e-09 ***
Calories2    -7.946e-04  1.211e-04  -6.563 6.14e-08 ***
Residual standard error: 27.65 on 42 degrees of freedom
Multiple R-squared:  0.8621, Adjusted R-squared:  0.8556 
F-statistic: 131.3 on 2 and 42 DF,  p-value: < 2.2e-16
```

Note on retaining lower-order effects

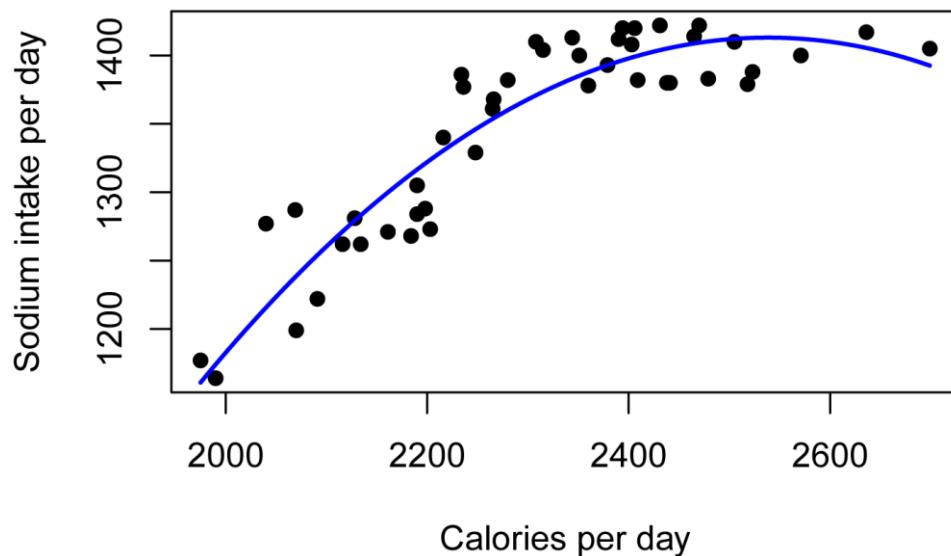
In polynomial regression, we typically keep all lower order effects of any higher order effects we include. So, in this case, if the final model included *Calories2*, but the effect of *Calories* were not significant, we would still include *Calories* in the model since we are including *Calories2*.

Plot data with best fit line

For bivariate data, the function *plotPredy* will plot the data and the predicted line for the model. It also works for polynomial functions, if the *order* option is changed.

```
library(rcompanion)

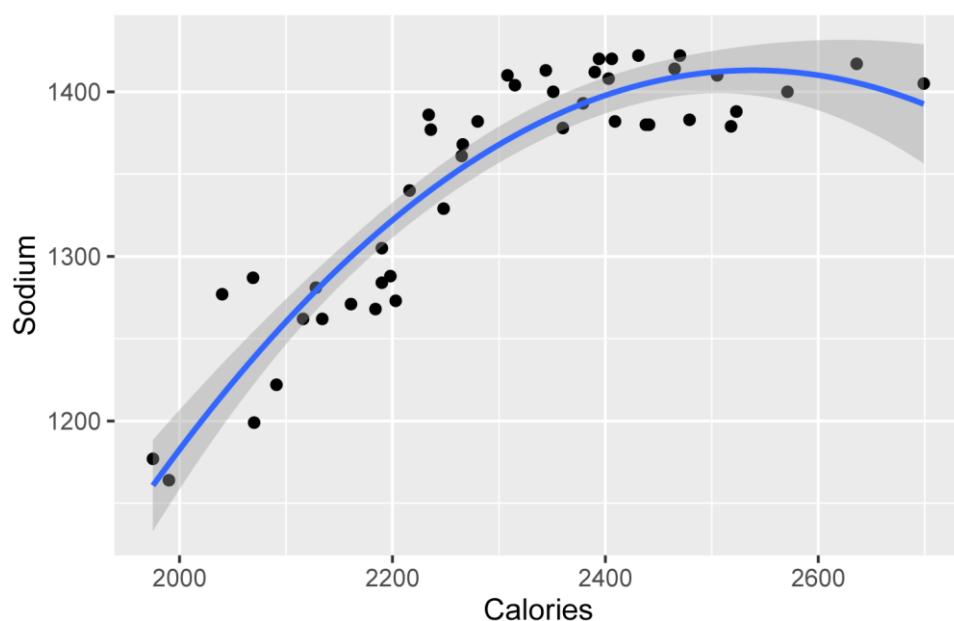
plotPredy(data = Data,
          y = Sodium,
          x = Calories,
          x2 = Calories2,
          model = model.2,
          order = 2,
          xlab = "Calories per day",
          ylab = "Sodium intake per day")
```

Plot of best fit line with confidence interval

For polynomial regression, the *ggplot* function can plot the best-fit curve with the confidence interval of the curve shaded.

```
library(ggplot2)

ggplot(Data,
       aes(x = Calories,
           y = Sodium)) +
  geom_point() +
  geom_smooth(method = "lm",
              formula = y ~ poly(x, 2, raw=TRUE), ### polynomial of order 2
              se      = TRUE)
```

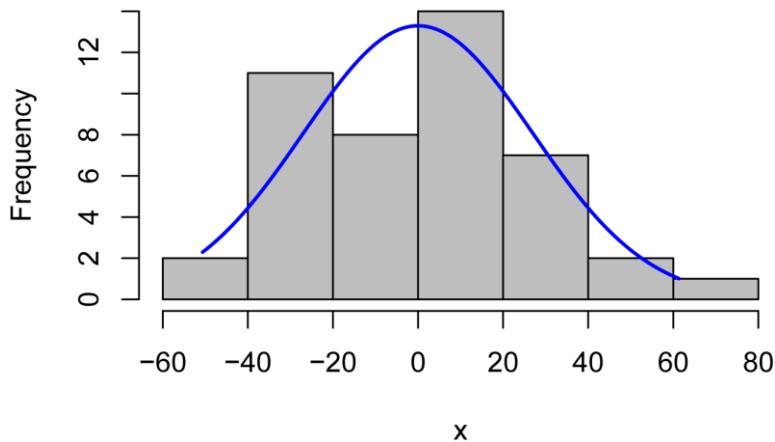


Plots of residuals

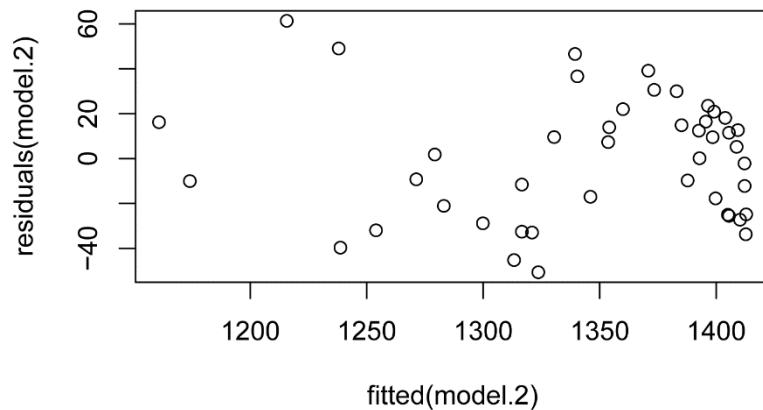
```
x = residuals(model.2)
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```



```
plot(fitted(model.2),  
      residuals(model.2))
```

**A few of xkcd comics***Correlation*

xkcd.com/552/

Extrapolating

xkcd.com/605/

Cat proximity
xkcd.com/231/

References

Cohen, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Edition. Routledge.

“Correlation and Linear Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_01.html.

“Multiple Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_05.html.

Exercises U

1. Consider the data from Brendon, Jason, Melissa, Paula, and McGuirk. Report for each answer, indicate **how you know**, when appropriate, by reporting the values of the statistic you are using or other information you used.
 - a. Which two variables are the most strongly correlated?
 - b. Which two variables are the least strongly correlated?
 - c. Are there any pairs of variables that are statistically uncorrelated? Which?
 - d. Name a pair of variables that is positively correlated.
 - e. Name a pair of variables that is negatively correlated.
 - f. Is *Sodium* **significantly** correlated with *Calories*?
 - g. By linear regression, is there a significant linear relationship of *Sodium* vs. *Calories*?
 - h. Does the quadratic polynomial model fit the *Sodium* vs. *Calories* data better than the linear model? Consider the *p*-value, the *r*-squared value, the range of values for each of *Sodium* and *Calories*, and your practical conclusions.
2. As part of a professional skills program, a 4-H club tests its members for typing proficiency (*Words.per.minute*), *Proofreading* skill, proficiency with using a *Spreadsheet*, and acumen in *Statistics*.

Instructor	Grade	words.per.minute	Proofreading	Spreadsheet	Statistics
'Dr. Katz'	6	35	53	75	61
'Dr. Katz'	6	50	77	24	51
'Dr. Katz'	6	55	71	62	55
'Dr. Katz'	6	60	78	27	91
'Dr. Katz'	6	65	84	44	95
'Dr. Katz'	6	60	79	38	50

'Dr. Katz'	6	70	96	12	94
'Dr. Katz'	6	55	61	55	76
'Dr. Katz'	6	45	73	59	75
'Dr. Katz'	6	55	75	55	80
'Dr. Katz'	6	60	85	35	84
'Dr. Katz'	6	45	61	49	80
'Laura'	7	55	59	79	57
'Laura'	7	60	60	60	60
'Laura'	7	75	90	19	64
'Laura'	7	65	87	32	65
'Laura'	7	60	70	33	94
'Laura'	7	70	84	27	54
'Laura'	7	75	87	24	59
'Laura'	7	70	97	38	74
'Laura'	7	65	86	30	52
'Laura'	7	72	91	36	66
'Laura'	7	73	88	20	57
'Laura'	7	65	86	19	71
'Ben Katz'	8	55	84	20	76
'Ben Katz'	8	55	63	44	94
'Ben Katz'	8	70	95	31	88
'Ben Katz'	8	55	63	69	93
'Ben Katz'	8	65	65	47	70
'Ben Katz'	8	60	61	63	92
'Ben Katz'	8	70	80	35	60
'Ben Katz'	8	60	88	38	58
'Ben Katz'	8	60	71	65	99
'Ben Katz'	8	62	78	46	54
'Ben Katz'	8	63	89	17	60
'Ben Katz'	8	65	75	33	77

For each of the following, answer the question, and **show the output** from the analyses you used to answer the question. Where relevant, indicate **how you know**.

- a. Which two variables are the most strongly correlated?
- b. Name a pair of variables that are statistically uncorrelated.
- c. Name a pair of variables that is positively correlated.
- d. Name a pair of variables that is negatively correlated.
- e. Consider the correlation between *Spreadsheet* and *Proofreading*.
 - i. What is the value of the correlation coefficient r for this correlation?
 - ii. What is the value of τ ?
 - iii. What is the value of ρ ?

f. Conduct a linear regression of Proofreading vs. Words.per.minute.

- i. What is the p-value for this model?
- ii. What is the r-squared value?
- iii. Do the residuals suggest that the linear regression model is an appropriate model?
- iv. What can you conclude about the results of the linear regression? Consider the *p*-value, the *r*-squared value, the range of values for each of *Proofreading* and *Words.per.minute*, and your practical conclusions.

Advanced Parametric Methods

Packages used in this chapter

The packages used in this chapter include:

- robustbase
- psych
- minpack.lm
- car
- rcompanion
- mblm

The following commands will install these packages if they are not already installed:

```
if(!require(robustbase)){install.packages("robustbase")}
if(!require(psych)){install.packages("psych")}
if(!require(nlstools)){install.packages("nlstools")}
if(!require(minpack.lm)){install.packages("minpack.lm")}
if(!require(car)){install.packages("car")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(mblm)){install.packages("mblm")}
```

More complex experimental designs

Except for *t*-tests, the approach of this book for parametric statistics has been to develop linear models (with the *lm* function) or mixed effects models (with the *nlme* or *lme4* packages) and then to apply analysis of variance, model checking, and post-hoc testing.

These relatively simple models can be expanded to include more independent variables of either continuous or factor type. Likewise, more complex designs can include nested and crossed factors. Some traditional experimental designs include Latin square, split plot, and incomplete block.

Analysis of co-variance

Analysis of covariance combines continuous independent variables with factor independent variables.

A related design is the paired watershed design which, while it was developed for watershed studies, can be adapted to various situations in a variety of fields.

“Analysis of Covariance” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_04.html.

Clausen, J.C. and J. Spooner. 1993. Paired Watershed Study Design. 841-F-93-009. United States Environmental Protection Agency, Office of Water. Washington, DC.
nepis.epa.gov/Exe/ZyPURL.cgi?Dockey=20004PR6.TXT.

Dressing, S.A. and D.W. Meals. 2005. Designing water quality monitoring programs for watershed projects. U.S. Environmental Protection Agency. Fairfax, VA.
www.epa.gov/sites/production/files/2015-10/documents/technote2_wq_monitoring.pdf.

Nonlinear regression and curvilinear regression

Nonlinear and curvilinear regression are similar to linear regression, except that the relationship between the dependent variable and the independent variable is curved in some way. Specific methods include polynomial regression, spline regression, and nonlinear regression.

Polynomial regression was covered briefly in the previous chapter, while some examples of curvilinear regression are shown below in the “Linear plateau and quadratic plateau models” section in this chapter.

For further discussion, see:

“Curvilinear Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_03.html.

Multiple regression

Multiple regression is similar to linear regression, except that the model contains multiple independent variables.

The polynomial regression example in the previous chapter is one example of multiple regression.

For further discussion, see:

“Multiple Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_05.html.

Logistic regression

Logistic regression is similar to linear regression, except that the dependent variable is categorical. In standard logistic regression, the dependent variable has only two levels, and in multinomial logistic regression, the dependent variable can have more than two levels.

For examples of logistic regression, see the chapter *Models for Nominal Data* in this book.

Analysis of count data

When the dependent variable is a counted quantity and not a measured quantity, it is appropriate to use Poisson regression or related techniques.

For a discussion of when standard parametric approaches may or may not be used with count data, see the “Count data may not be appropriate for common parametric tests” section in the *Introduction to Parametric Tests* chapter.

For examples of appropriate ways to model count data, see the *Regression for Count Data* chapter in this book.

Robust techniques

Standard parametric analyses can be sensitive to outliers and other deviations from assumptions of the analyses. The example below shows that a single point can affect the predicted line from standard linear regression. It is said that this point has high *leverage*, as so has high *influence* in the model.

One approach in these cases is to use nonparametric techniques.

Another approach is to adjust the model to better fit the data.

A third way is to determine if certain data points are true *outliers*, and then give them further examination.

A fourth way is to use a robust technique such as robust regression. The references below discuss robust regression and robust techniques appropriate for designs lending themselves to an analysis of variance approach.

“One-way Analysis with Permutation Test” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_06a.html.

“Two-way Anova with Robust Estimation” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/d_08a.html.

Search for *Robust regression* in the following:

“Correlation and Linear Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_01.html.

Example of robust linear regression

Note that this example does not include checking assumptions of the model, but is intended only to compare the results of ordinary linear regression and robust regression when there is a data point with high influence.

```
Input = "
X   Y
0   0
1   2
2   5
3   6
4   9
5   10
6   11
7   14
8   16
9   20
10  44
")

Data = read.table(textConnection(Input), header=TRUE)
```

Linear regression

```
model1 = lm(Y ~ X,
            data = Data)

summary(model1)

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -3.1364    3.6614  -0.857 0.413888
X             3.1182    0.6189   5.038 0.000701 ***
Residual standard error: 6.491 on 9 degrees of freedom
Multiple R-squared:  0.7383, Adjusted R-squared:  0.7092
F-statistic: 25.39 on 1 and 9 DF,  p-value: 0.0007013
```

Calculate statistics and plot

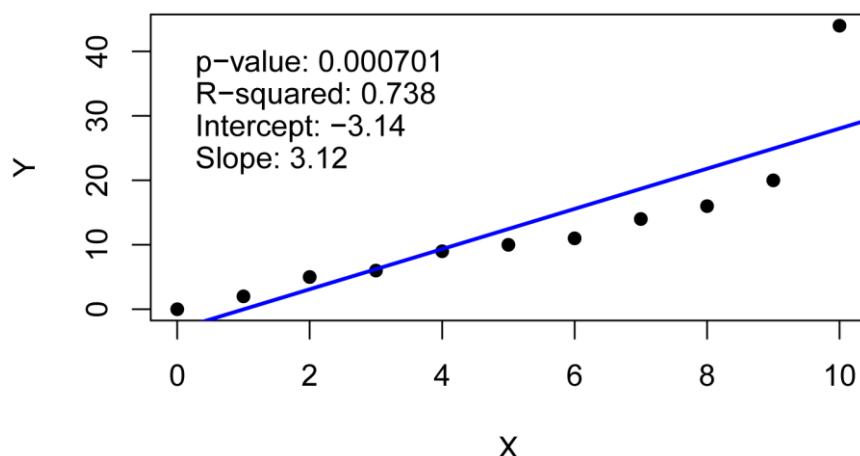
```
Pvalue = pf(summary(model1)$fstatistic[1],
            summary(model1)$fstatistic[2],
            summary(model1)$fstatistic[3],
            lower.tail = FALSE)

R2      = summary(model1)$r.squared

t1      = paste0("p-value: ", signif(Pvalue, digits=3))
```

```
t2      = paste0("R-squared: ", signif(R2, digits=3))
t3      = paste0("Intercept: ", signif(coef(model)[1], digits=3))
t4      = paste0("Slope: ", signif(coef(model)[2], digits=3))

plot(Y ~ X,
      data = Data,
      pch = 16)
abline(model1,
      col="blue",
      lwd=2)
text(0, 38, labels = t1, pos=4)
text(0, 33, labels = t2, pos=4)
text(0, 28, labels = t3, pos=4)
text(0, 23, labels = t4, pos=4)
```

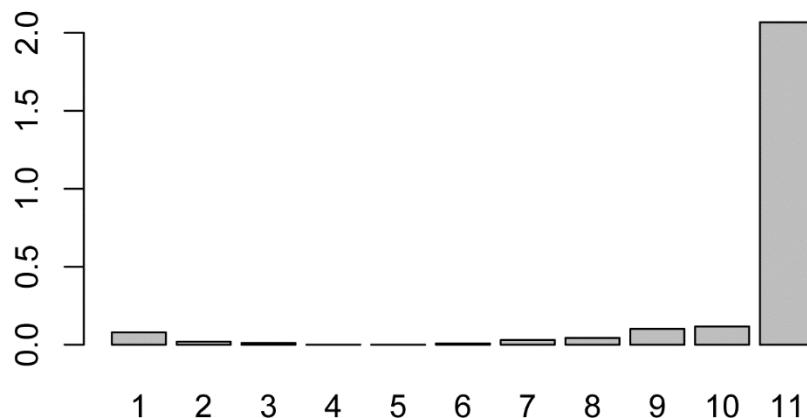


Cook's distance

Cook's distance is a measure of influence of each data point, specifically determining how much predicted values would change if the observation were deleted. It is sometimes recommended that a Cook's distance of 1 or more merits further examination of the data point. However, there are other recommendations for critical values for Cook's distance.

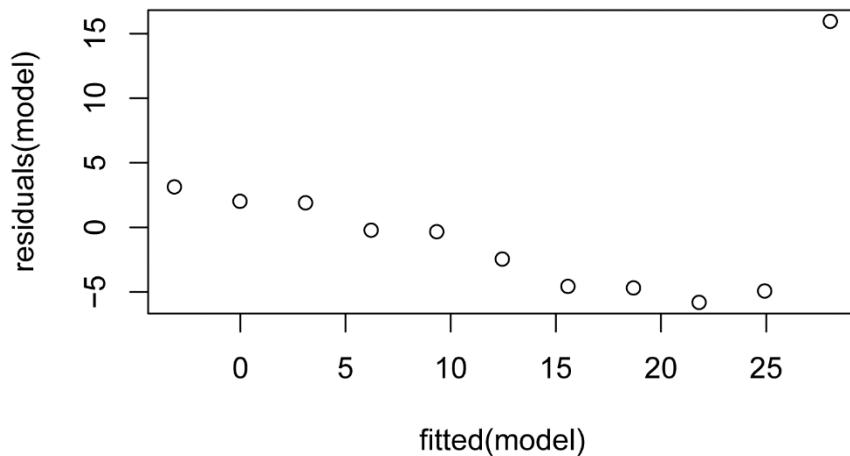
In this case, the high Cook's distance for observation 11 makes it a candidate for further evaluation.

```
barplot(cooks.distance(model1))
```

*Plot of residuals*

This plot suggests that the residuals are not independent of the fitted values. We would probably want to adjust the model by adding additional terms, using a nonlinear or curvilinear approach, or using a robust analysis to minimize the influence of certain data points.

```
plot(fitted(model),
      residuals(model))
```

Robust regression

```
library(robustbase)

model.r = lmrob(Y ~ X,
                  data = Data)

summary(model.r)

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.09015   0.32804   0.275    0.79    
X            2.04249   0.10970  18.619  1.7e-08 ***
```

Robust residual standard error: 0.9378
 Multiple R-squared: 0.9817, Adjusted R-squared: 0.9796

Calculate statistics and plot

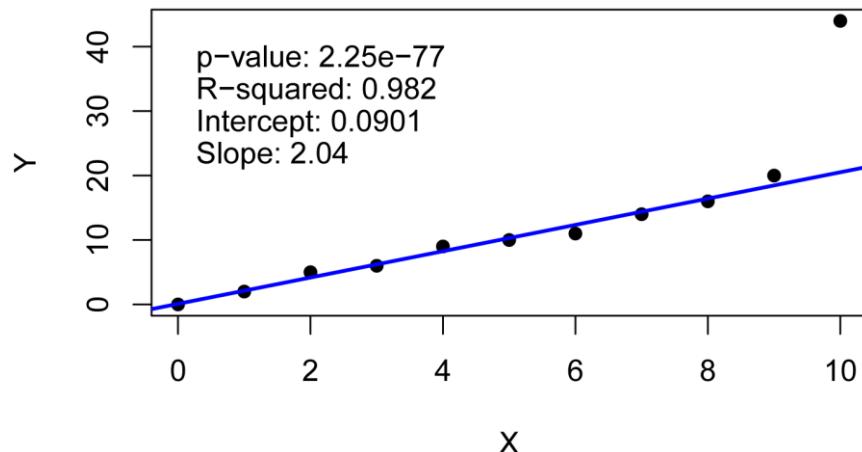
```
model.null = lmrob(Y ~ 1,
                     data = Data)

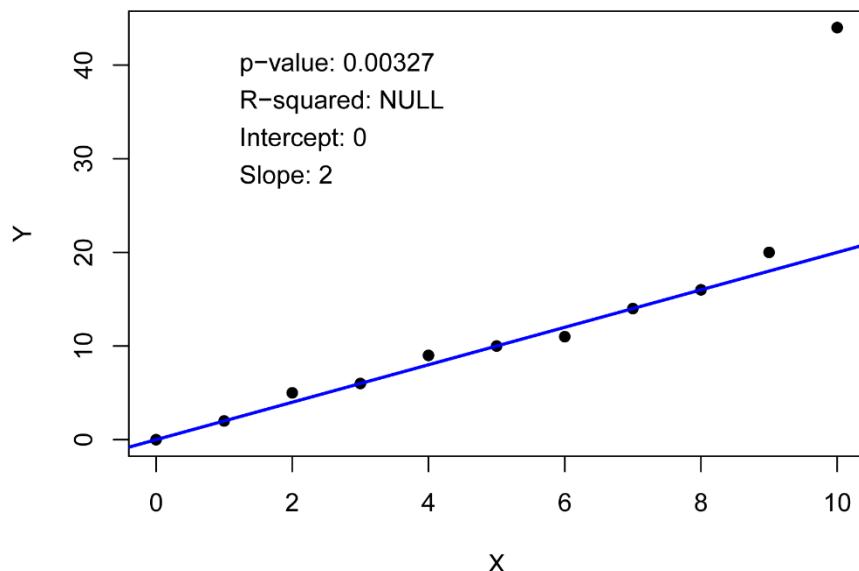
Pvalue = anova(model.r, model.null)[4][2,1]

R2      = summary(model.r)$r.squared

t1      = paste0("p-value: ", signif(Pvalue, digits=3))
t2      = paste0("R-squared: ", signif(R2, digits=3))
t3      = paste0("Intercept: ", signif(coefficients(model.r)[1], digits=3))
t4      = paste0("Slope: ", signif(coefficients(model.r)[2], digits=3))

plot(Y ~ X,
      data = Data,
      pch  = 16)
abline(model.r,
       col="blue",
       lwd=2)
text(0, 38, labels = t1, pos=4)
text(0, 33, labels = t2, pos=4)
text(0, 28, labels = t3, pos=4)
text(0, 23, labels = t4, pos=4)
```





Contrasts in Linear Models

Contrasts can be used in linear models to compare groups of treatments to one another. For example, if you were testing four curricula and wanted to compare the effect of *One* and *Two* vs. *Three* and *Four*, you could accomplish this with a single-degree-of freedom contrast.

“Contrasts in Linear Models” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/h_01.html.

Linear plateau and quadratic plateau models

Linear plateau and quadratic plateau models are segmented models for bivariate data, typically where the *y*-value increases with an increase in *x* to some critical value, beyond which the *y*-value ceases to increase. The statistics of interest include the *critical value* (the *x*-value above which there is no further increase in *y*), and the *plateau value* (the statistically highest value that *y* reaches).

For another example of these models, see the “Use the best model” section of the *Basic Plots* chapter.

Also included in this section is an example of curvilinear regression, with a Mitscherlich–Bray model fit to the same data.

Examples of linear plateau and quadratic plateau models

```
Input = "
Instructor      Grade    weight  Calories Sodium  Score
'Brendon Small'   6        43     2069    1287    77
'Brendon Small'   6        41     1990    1164    76
'Brendon Small'   6        40     1975    1177    76
'Brendon Small'   6        44     2116    1262    84
'Brendon Small'   6        45     2161    1271    86
'Brendon Small'   6        44     2091    1222    87
```

```
'Brendon Small'    6     48    2236    1377    90
'Brendon Small'    6     47    2198    1288    78
'Brendon Small'    6     46    2190    1284    89
'Jason Penopolis'  7     45    2134    1262    76
'Jason Penopolis'  7     45    2128    1281    80
'Jason Penopolis'  7     46    2190    1305    84
'Jason Penopolis'  7     43    2070    1199    68
'Jason Penopolis'  7     48    2266    1368    85
'Jason Penopolis'  7     47    2216    1340    76
'Jason Penopolis'  7     47    2203    1273    69
'Jason Penopolis'  7     43    2040    1277    86
'Jason Penopolis'  7     48    2248    1329    81
'Melissa Robins'   8     48    2265    1361    67
'Melissa Robins'   8     46    2184    1268    68
'Melissa Robins'   8     53    2441    1380    66
'Melissa Robins'   8     48    2234    1386    65
'Melissa Robins'   8     52    2403    1408    70
'Melissa Robins'   8     53    2438    1380    83
'Melissa Robins'   8     52    2360    1378    74
'Melissa Robins'   8     51    2344    1413    65
'Melissa Robins'   8     51    2351    1400    68
'Paula Small'       9     52    2390    1412    78
'Paula Small'       9     54    2470    1422    62
'Paula Small'       9     49    2280    1382    61
'Paula Small'       9     50    2308    1410    72
'Paula Small'       9     55    2505    1410    80
'Paula Small'       9     52    2409    1382    60
'Paula Small'       9     53    2431    1422    70
'Paula Small'       9     56    2523    1388    79
'Paula Small'       9     50    2315    1404    71
'Coach McGuirk'   10     52    2406    1420    68
'Coach McGuirk'   10     58    2699    1405    65
'Coach McGuirk'   10     57    2571    1400    64
'Coach McGuirk'   10     52    2394    1420    69
'Coach McGuirk'   10     55    2518    1379    70
'Coach McGuirk'   10     52    2379    1393    61
'Coach McGuirk'   10     59    2636    1417    70
'Coach McGuirk'   10     54    2465    1414    59
'Coach McGuirk'   10     54    2479    1383    61
")
```

```
Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))
```

```
### Check the data frame
```

```
library(psych)

headTail(Data)
```

```

str(Data)
summary(Data)

### Remove unnecessary objects
rm(Input)

```

Linear plateau model

The *nls* function in the native *stats* package can fit nonlinear and curvilinear functions. To accomplish this, a function—*linplat*, here—will be defined with the *x* and *y* variables (*Calories* and *Sodium*) along with parameters (*a*, *b*, and *clx*). It is the goal of the *nls* function to find the best fit values for these parameters for these data.

The code below starts with an attempt to find reasonable starting values for the parameters. If these starting values don't yield appropriate results, other starting values can be tried.

The fitted parameters *a*, *b*, and *clx* designate the best fit intercept, slope, and critical *x* value. The plateau value is calculated as $a + b * clx$.

```

### Find reasonable initial values for parameters

fit.lm      = lm(Sodium ~ Calories, data=Data)
a.ini       = fit.lm$coefficients[1]
b.ini       = fit.lm$coefficients[2]
clx.ini    = mean(Data$Calories)

### Define linear plateau function

linplat = function(x, a, b, clx)
  {ifelse(x < clx, a + b * x,
         a + b * clx)}

### Find best fit parameters

model = nls(Sodium ~ linplat(Calories, a, b, clx),
            data = Data,
            start = list(a    = a.ini,
                         b    = b.ini,
                         clx = clx.ini),
            trace = FALSE,
            nls.control(maxiter = 1000))

summary(model)

Parameters:
Estimate Std. Error t value Pr(>|t|)
```

```
a    -128.85535 116.10819   -1.11    0.273
b      0.65768  0.05343   12.31  1.6e-15 ***
c1x 2326.51521 16.82086 138.31 < 2e-16 ***
```

p-value and pseudo R-squared

```
### Define null model

nullfunct = function(x, m){m}

m.ini     = mean(Data$Sodium)

null = nls(Sodium ~ nullfunct(Calories, m),
           data = Data,
           start = list(m    = m.ini),
           trace = FALSE,
           nls.control(maxiter = 1000))

### Find p-value and pseudo R-squared

library(rcompanion)

nagelkerke(model,
           null)

$Pseudo.R.squared.for.model.vs.null
                               Pseudo.R.squared
McFadden                      0.195417
Cox and Snell (ML)            0.892004
Nagelkerke (Cragg and Uhler)  0.892014

$Likelihood.ratio.test
Df.diff LogLik.diff Chisq  p.value
-2      -50.077 100.15 1.785e-22
```

Determine confidence intervals for parameter estimates

```
library(nlstools)

confint2(model,
         level = 0.95)

          2.5 %      97.5 %
a    -363.1711583 105.4604611
b      0.5498523  0.7654999
c1x 2292.5693351 2360.4610904
```

```
library(nlstools)
```

```
Boot = nlsBoot(model)
```

```
summary(Boot)
```

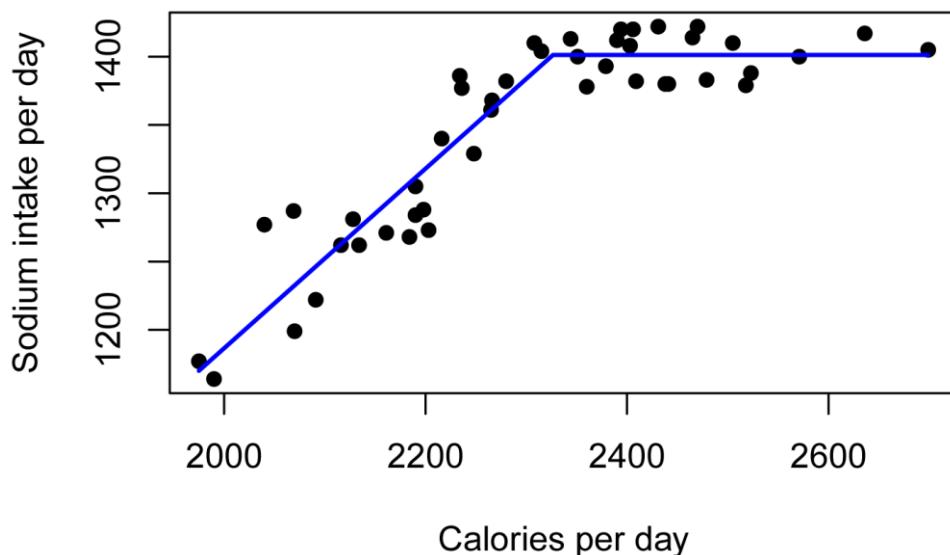
```
-----
Bootstrap statistics
  Estimate Std. error
a -143.5189740 111.1829997
b 0.6643267 0.05132983
c1x 2326.8124177 16.71482729

-----
Median of bootstrap estimates and percentile confidence intervals
  Median      2.5%    97.5%
a -143.5474796 -376.3224461 73.437548
b 0.6638483 0.5652069 0.771089
c1x 2327.1279329 2292.5244830 2363.929178
```

Plot data with best fit line

```
library(rcompanion)

plotPredy(data = Data,
          x = Calories,
          y = Sodium,
          model = model,
          xlab = "Calories per day",
          ylab = "Sodium intake per day")
```



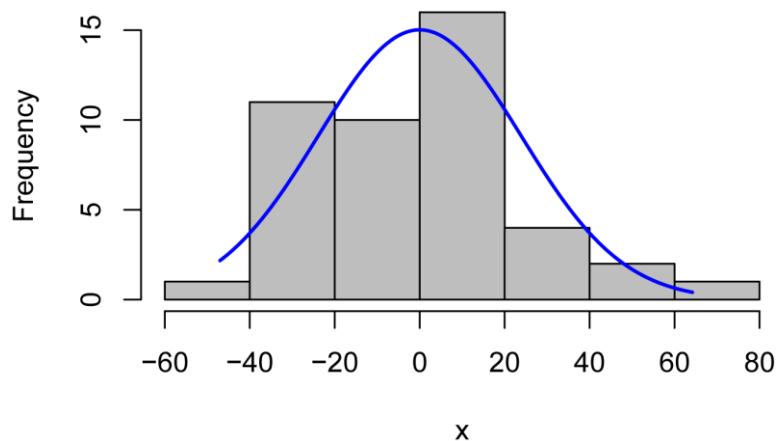
A plot of daily sodium intake vs. daily caloric intake for students. Best fit linear plateau model is shown. Critical x value = 2327 calories, and Plateau y = 1401 mg. p < 0.0001. Pseudo R-squared (Nagelkerke) = 0.892.

Plots of residuals

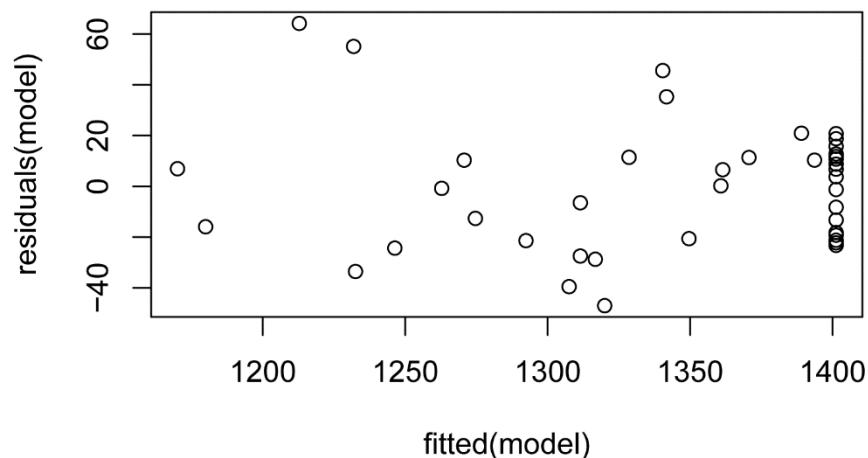
```
x = residuals(model)
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```



```
plot(fitted(model),  
      residuals(model))
```

Quadratic plateau model

A quadratic plateau model is similar to a linear plateau model, except that the linear segment is replaced with a quadratic function.

The fitted parameters a , b , and clx designate the best fit intercept, linear coefficient, and critical x value. The quadratic coefficient is calculated as $-0.5 * b / clx$. The plateau value is calculated as $a + b * clx - 0.5 * b * clx^2$.

```
### Find reasonable initial values for parameters
```

```

fit.lm      = lm(Sodium ~ Calories, data=Data)
a.ini      = fit.lm$coefficients[1]
b.ini      = fit.lm$coefficients[2]
clx.ini    = mean(Data$Calories)

### Define quadratic plateau function

quadplat = function(x, a, b, clx) {
  ifelse(x < clx, a + b * x + (-0.5*b/clx) * x * x,
         a + b * clx + (-0.5*b/clx) * clx * clx)}

### Find best fit parameters

model = nls(Sodium ~ quadplat(Calories, a, b, clx),
            data = Data,
            start = list(a = a.ini,
                         b = b.ini,
                         clx = clx.ini),
            trace = FALSE,
            nls.control(maxiter = 1000))

summary(model)

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a -4217.7196   935.1139 -4.510 5.13e-05 ***
b     4.4920    0.8251  5.444 2.49e-06 ***
clx  2504.4059   48.9679 51.144 < 2e-16 ***
```

p-value and pseudo R-squared

```

### Define null model

nullfunct = function(x, m){m}

m.ini      = mean(Data$Sodium)

null = nls(Sodium ~ nullfunct(Calories, m),
           data = Data,
           start = list(m = m.ini),
           trace = FALSE,
           nls.control(maxiter = 1000))

### Find p-value and pseudo R-squared

library(rcompanion)
```

```
nagelkerke(model,
null)

$Pseudo.R.squared.for.model.vs.null
                           Pseudo.R.squared
McFadden                      0.175609
Cox and Snell (ML)            0.864674
Nagelkerke (Cragg and Uhler)  0.864683

$Likelihood.ratio.test
  Df.diff LogLik.diff Chisq   p.value
-2      -45.001 90.003 2.8583e-20
```

Determine confidence intervals for parameter estimates

```
library(nlstools)

confint2(model,
         level = 0.95)

              2.5 %      97.5 %
a    -6104.855900 -2330.583368
b        2.826818     6.157185
c1x  2405.584577  2603.227200

library(nlstools)

Boot = nlsBoot(model)

summary(Boot)

-----
Bootstrap statistics
      Estimate Std. error
a    -4294.518746 956.6204011
b        4.560459  0.8448001
c1x  2510.154733 53.3134728

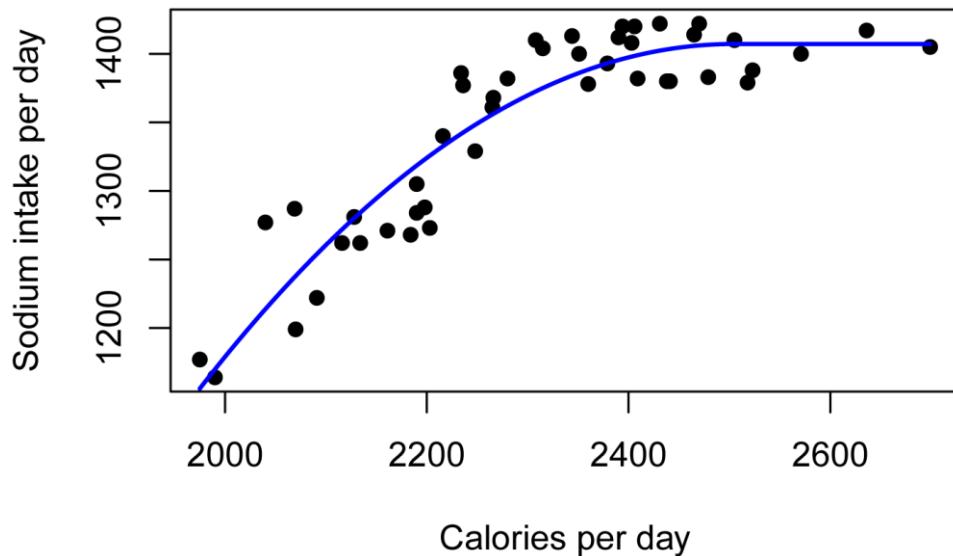
-----
Median of bootstrap estimates and percentile confidence intervals
      Median      2.5%      97.5%
a    -4250.030936 -6311.47965 -2546.971334
b        4.515874    3.03532    6.359601
c1x  2504.795426  2425.75206  2626.813217
```

Plot data with best fit line

```
library(rcompanion)

plotPredy(data = Data,
          x     = Calories,
```

```
y      = Sodium,
model = model,
xlab  = "Calories per day",
ylab  = "Sodium intake per day")
```

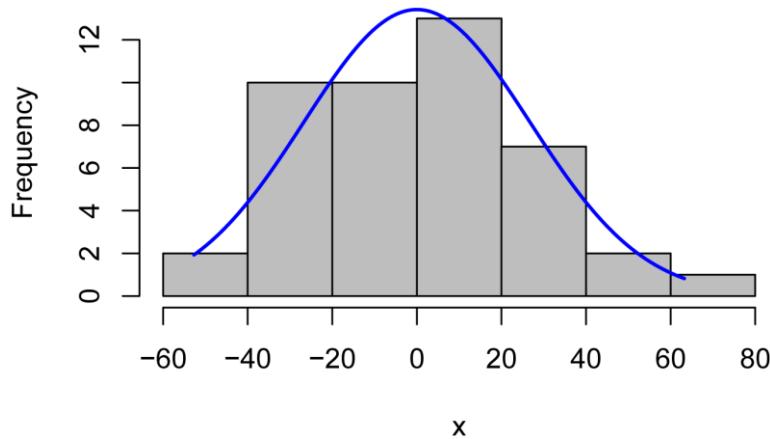


A plot of daily sodium intake vs. daily caloric intake for students. Best fit quadratic plateau model is shown. Critical x value = 2504 calories, and Plateau $y = 1403$ mg. $p < 0.0001$. Pseudo R-squared (Nagelkerke) = 0.865.

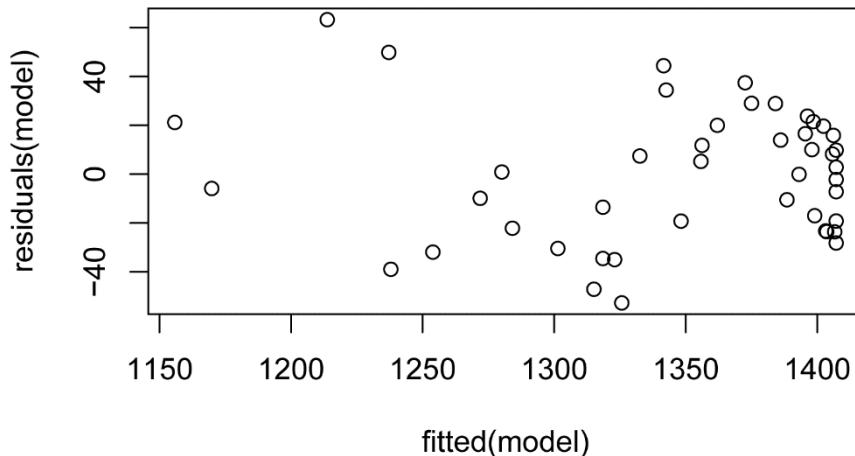
Plots of residuals

```
x = residuals(model)
```

```
library(rcompanion)
plotNormalHistogram(x)
```



```
plot(fitted(model),
      residuals(model))
```



Mitscherlich–Bray model

This model is an example of curvilinear regression, specifically with a general exponential model. The model here is specified as $y = a - be^{-cx}$.

In the function below, the constant D is set to the minimum of *Calories* + 1. This will be subtracted from the x values because the numerically large values for *Calories* make the fitting of the exponential function difficult without doing this subtraction.

```
### Find reasonable initial values for parameters
```

```
a.ini    = 1400
b.ini    = -120
c.ini    = 0.01
```

```
### Set a constant to adjust x values
```

```
D        = 1974
```

```
### Define exponential function
```

```
expon = function(x, a, b, c) {
  a + b * exp(-1 * c * (x - D))}
```

```
### Find best fit parameters
```

```
model = nls(Sodium ~ expon(Calories, a, b, c),
            data = Data,
            start = list(a = a.ini,
                         b = b.ini,
                         c = c.ini),
            trace = FALSE,
            nls.control(maxiter = 10000))
```

```
summary(model)

Parameters:
Estimate Std. Error t value Pr(>|t|)
a 1.450e+03 2.309e+01 62.790 < 2e-16 ***
b -2.966e+02 2.142e+01 -13.846 < 2e-16 ***
c 3.807e-03 7.973e-04 4.776 2.2e-05 ***
```

p-value and pseudo R-squared

```
### Define null model

nullfunct = function(x, m){m}

m.ini      = mean(Data$Sodium)

null = nls(Sodium ~ nullfunct(calories, m),
           data = Data,
           start = list(m = m.ini),
           trace = FALSE,
           nls.control(maxiter = 1000))

### Find p-value and pseudo R-squared

library(rcompanion)

nagelkerke(model,
           null)

$Pseudo.R.squared.for.model.vs.null
                                Pseudo.R.squared
McFadden                      0.162515
Cox and Snell (ML)            0.842908
Nagelkerke (Cragg and Uhler)  0.842918

$Likelihood.ratio.test
Df.diff LogLik.diff Chisq   p.value
-2      -41.646 83.292 8.1931e-19
```

Determine confidence intervals for parameter estimates

```
library(nlstools)

confint2(model,
         level = 0.95)

          2.5 %      97.5 %
a 1.403060e+03 1.496244e+03
b -3.398171e+02 -2.533587e+02
c 2.198512e-03 5.416355e-03
```

```

library(nlstools)

Boot = nlsBoot(model)

summary(Boot)

-----
Bootstrap statistics
      Estimate   Std. error
a 1.453109e+03 2.522906e+01
b -3.005815e+02 2.244473e+01
c  3.850955e-03 7.832523e-04

-----
Median of bootstrap estimates and percentile confidence intervals
      Median    2.5%    97.5%
a 1.449234e+03 1.415464e+03 1.513570e+03
b -2.998719e+02 -3.474264e+02 -2.566514e+02
c  3.827519e-03 2.390379e-03 5.410459e-03

```

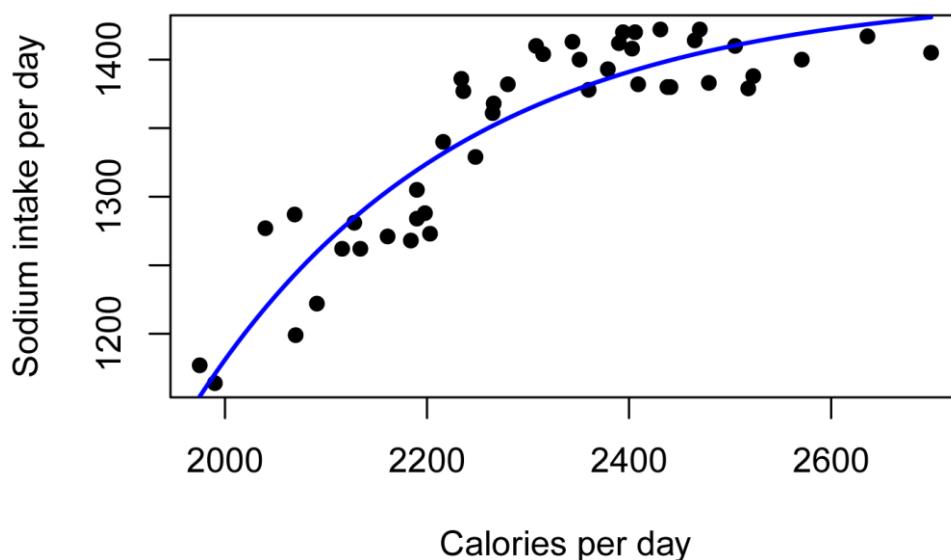
Plot data with best fit line

```

library(rcompanion)

plotPredy(data  = Data,
          x     = Calories,
          y     = Sodium,
          model = model,
          xlab  = "Calories per day",
          ylab  = "Sodium intake per day")

```



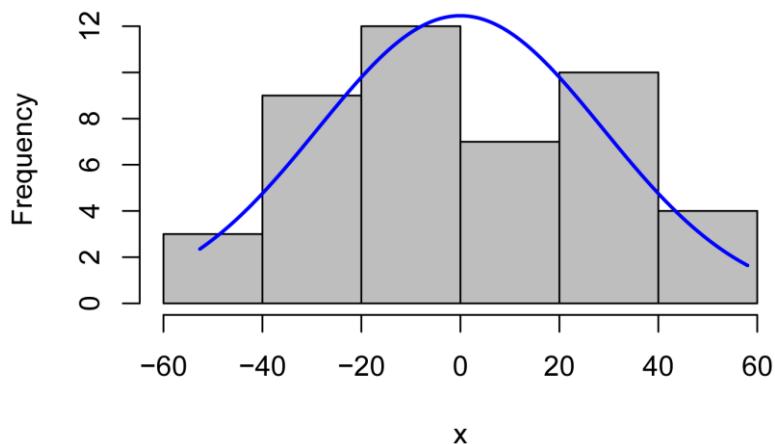
A plot of daily sodium intake vs. daily caloric intake for students. Best fit exponential model is shown. $p < 0.0001$. Pseudo R-squared (Nagelkerke) = 0.843.

Plots of residuals

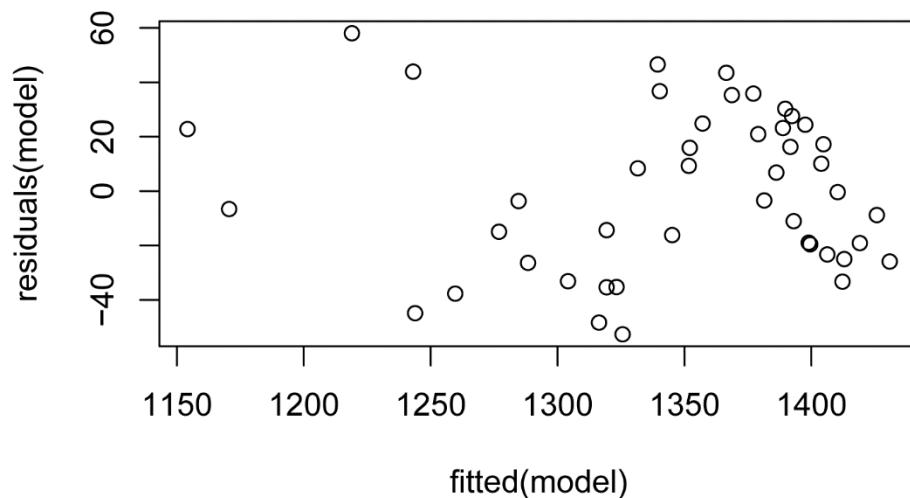
```
x = residuals(model)
```

```
library(rcompanion)
```

```
plotNormalHistogram(x)
```



```
plot(fitted(model),  
      residuals(model))
```

Better functions to fit nonlinear regression

Finding reasonable starting values for some functions can be difficult, and the *nls* function may not find a solution for certain functions if starting values are not close to their best values. One issue is that the Gauss–Newton method that *nls* uses will fail in some situations.

Typically, I will put my data into a spreadsheet and make a plot of the function I am trying to fit. I can then change parameter values and watch the curve change until I get the curve reasonably close to the data.

The package *minpack.lm* has a function *nlsLM*, which uses the Levenberg–Marquardt method. It is more successful at fitting parameters for difficult functions when the initial values for parameters are poor. An example of using this package is shown in the “Fitting curvilinear models with the *minpack.lm* package” below.

(The package *nlmrt* uses the Marquardt–Nash method, which is also tolerant of difficult functions and poor starting values for parameters. However, at the time of writing, this package is not available for newer versions of R.)

Fitting curvilinear models with the *minpack.lm* package

For a plot of the best fit line and plots of residuals, see the previous section of this chapter.

```
library(minpack.lm)

### Define exponential function

expon = function(x, a, b, c) {
  a + b * exp(-1 * c * (x - D))}

### Find best fit parameters

model = nlsLM(Sodium ~ expon(Calories, a, b, c),
               data = Data,
               start = list(a = 1400,
                            b = -1,
                            c = 0.1))

summary(model)

Parameters:
  Estimate Std. Error t value Pr(>|t|)
a  1.450e+03  2.309e+01  62.789 < 2e-16 ***
b -2.966e+02  2.142e+01 -13.846 < 2e-16 ***
c  3.807e-03  7.972e-04   4.776  2.2e-05 ***
```

Cate–Nelson analysis

Cate–Nelson analysis is a useful approach for some bivariate data that don’t fit neatly into linear model, linear–plateau model, or curvilinear model. The analysis divides the data into two populations: those x values that are likely to have high y values, and those x values that are likely to have low y values.

For more discussion and examples, see:

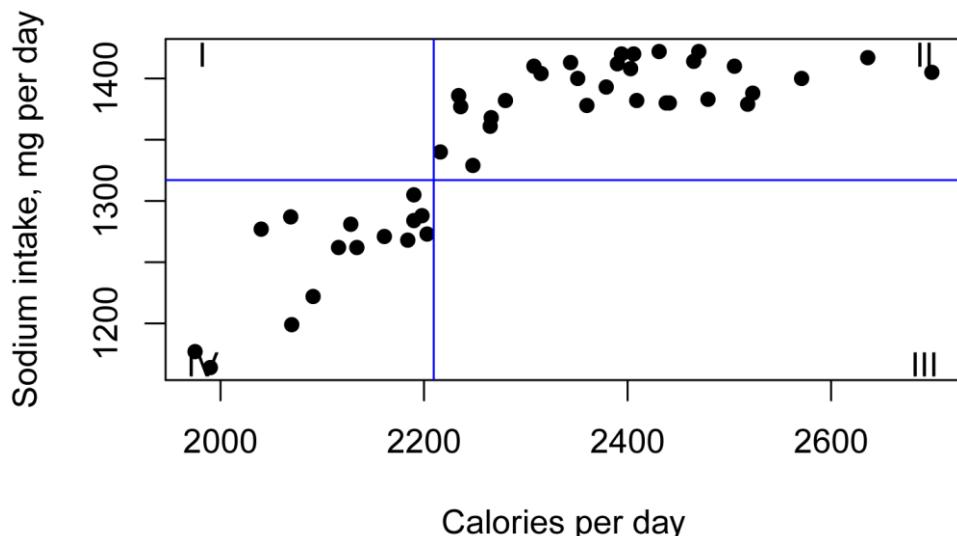
“Cate–Nelson Analysis” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/h_02.html.

```
library(rcompanion)
```

```
cateNelson(x = Data$Calories,
            y = Data$Sodium,
            plotit=TRUE,
            hollow=TRUE,
            xlab="Calories per day",
            ylab="Sodium intake, mg per day",
            trend="positive",
            c1x=1,
            c1y=1,
            xthreshold=0.10,
            ythreshold=0.15)
```

Final model:

n	CLx	SS	CLy	Q.I	Q.II	Q.III	Q.IV	Fisher.p.value
1 45	2209.5	190716.1	1317	0	30	0	15	2.899665e-12



A plot of daily sodium intake vs. daily caloric intake for students, with a Cate–Nelson analysis applied. Students with a Caloric intake greater than 2210 calories tended to have a sodium intake greater than 1317 mg per day.

Transforming Data

Transforming data is one step in addressing data that do not fit model assumptions, and is also used to coerce different variables to have similar distributions. Before transforming data, see the “Steps to handle violations of assumption” section in the *Assessing Model Assumptions* chapter.

Transforming data

Most parametric tests require that residuals be normally distributed and that the residuals be homoscedastic.

One approach when residuals fail to meet these conditions is to transform one or more variables to better follow a normal distribution. Often, just the dependent variable in a model will need to be transformed. However, in complex models and multiple regression, it is sometimes helpful to transform both dependent and independent variables that deviate greatly from a normal distribution.

There is nothing illicit in transforming variables, but you must be careful about how the results from analyses with transformed variables are reported. For example, looking at the turbidity of water across three locations, you might report, “Locations showed a significant difference in log-transformed turbidity.” To present means or other summary statistics, you might present the mean of transformed values, or back transform means to their original units.

Some measurements in nature are naturally normally distributed. Other measurements are naturally log-normally distributed. These include some natural pollutants in water: There may be many low values with fewer high values and even fewer very high values.

For right-skewed data—tail is on the right, positive skew—, common transformations include square root, cube root, and log.

For left-skewed data—tail is on the left, negative skew—, common transformations include square root (constant – x), cube root (constant – x), and log (constant – x).

Because $\log(0)$ is undefined—as is the log of any negative number—, when using a log transformation, a constant should be added to all values to make them all positive before transformation. It is also sometimes helpful to add a constant when using other transformations.

Another approach is to use a general power transformation, such as Tukey’s Ladder of Powers or a Box–Cox transformation. These determine a *lambda* value, which is used as the power coefficient to transform values. $X_{new} = X^{\lambda}$ for Tukey, and $X_{new} = (X^{\lambda} - 1) / \lambda$ for Box–Cox.

The *transformTukey* function in the *rcompanion* package finds the *lambda* which makes a single vector of values—that is, one variable—as normally distributed as possible with a simple power transformation.

The Box–Cox procedure is included in the *MASS* package with the function *boxcox*. It uses a log-likelihood procedure to find the *lambda* to use to transform the dependent variable for a linear model (such as an ANOVA or linear regression). It can also be used on a single vector.

Packages used in this chapter

The packages used in this chapter include:

- car
- MASS
- rcompanion

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("car")}
if(!require(MASS)){install.packages("MASS")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

Example of transforming skewed data

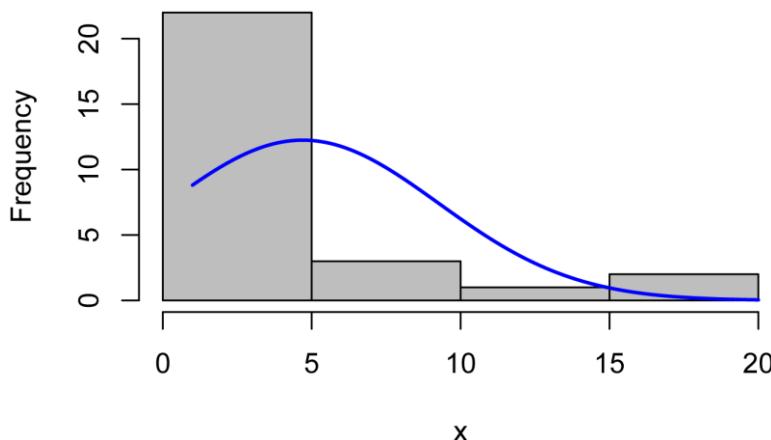
This example uses hypothetical data of river water turbidity. Turbidity is a measure of how cloudy water is due to suspended material in the water. Water quality parameters such as this are often naturally log-normally distributed: values are often low, but are occasionally high or very high.

The first plot is a histogram of the *Turbidity* values, with a normal curve superimposed. Looking at the gray bars, this data is skewed strongly to the right (positive skew), and looks more or less log-normal. The gray bars deviate noticeably from the red normal curve.

The second plot is a normal quantile plot (normal Q-Q plot). If the data were normally distributed, the points would follow the red line fairly closely.

```
Turbidity = c(1.0, 1.2, 1.1, 1.1, 2.4, 2.2, 2.6, 4.1, 5.0, 10.0, 4.0, 4.1, 4.2, 4.1,
5.1, 4.5, 5.0, 15.2, 10.0, 20.0, 1.1, 1.1, 1.2, 1.6, 2.2, 3.0, 4.0, 10.5)
```

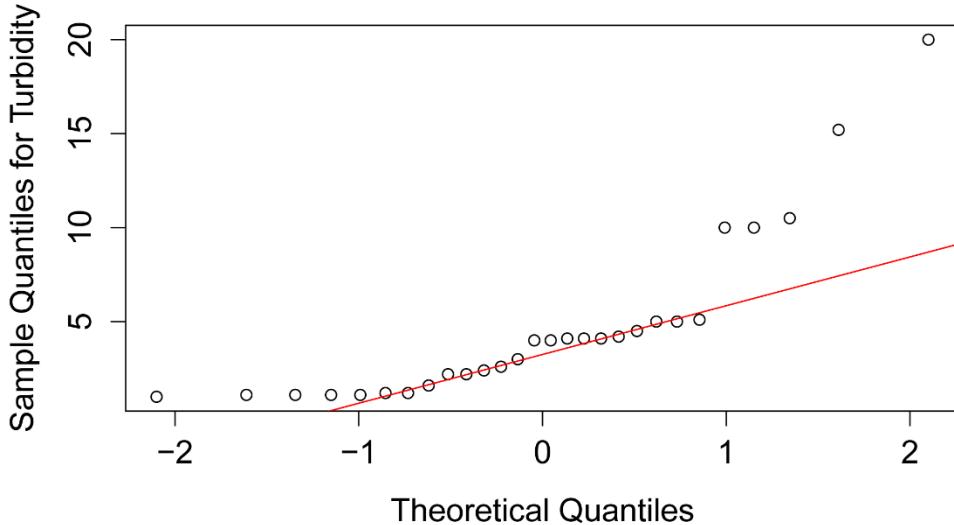
```
library(rcompanion)
plotNormalHistogram (Turbidity)
```



```
qqnorm(Turbidity,
       ylab="Sample Quantiles for Turbidity")
```

```
qqline(Turbidity,
       col="red")
```

Normal Q-Q Plot

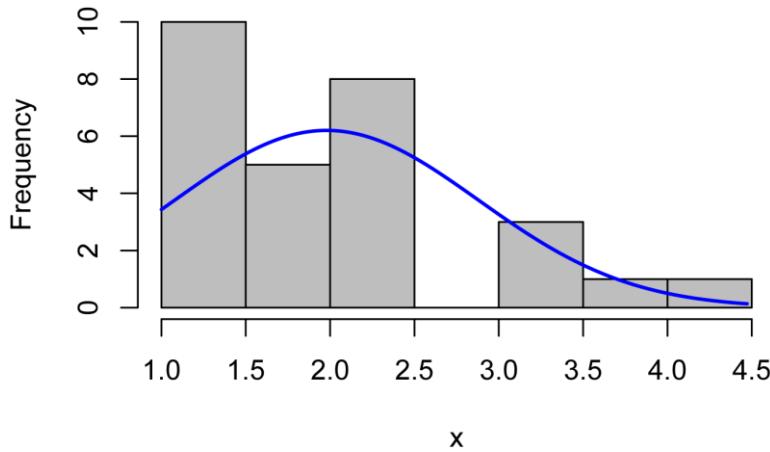
***Square root transformation***

Since the data is right-skewed, we will apply common transformations for right-skewed data: square root, cube root, and log. The square root transformation improves the distribution of the data somewhat.

```
T_sqrt = sqrt(Turbidity)
```

```
library(rcompanion)
```

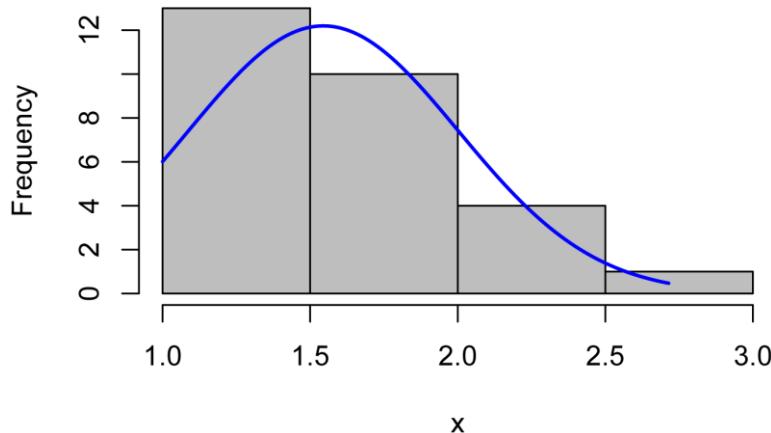
```
plotNormalHistogram(T_sqrt)
```

***Cube root transformation***

The cube root transformation is stronger than the square root transformation.

```
T_cub = sign(Turbidity) * abs(Turbidity)^(1/3)    # Avoid complex numbers
# for some cube roots
```

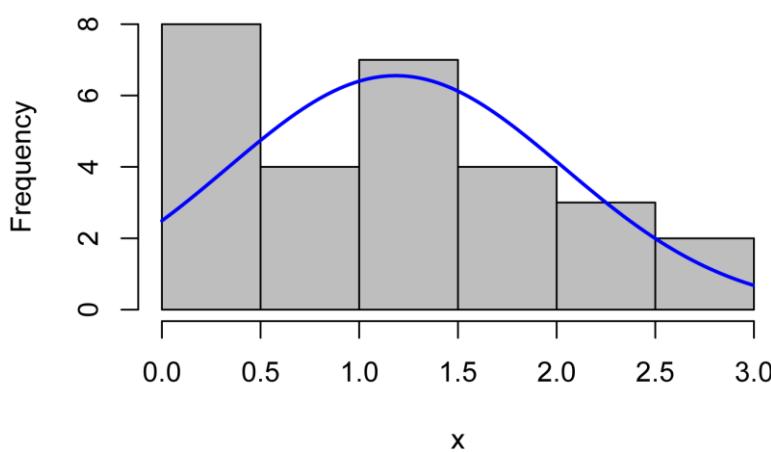
```
library(rcompanion)
plotNormalHistogram(T_cub)
```



Log transformation

The log transformation is a relatively strong transformation. Because certain measurements in nature are naturally log-normal, it is often a successful transformation for certain data sets. While the transformed data here does not follow a normal distribution very well, it is probably about as close as we can get with these particular data.

```
T_log = log(Turbidity)
library(rcompanion)
plotNormalHistogram(T_log)
```



Tukey's Ladder of Powers transformation

The approach of Tukey's Ladder of Powers uses a power transformation on a data set. For example, raising data to a 0.5 power is equivalent to applying a square root transformation; raising data to a 0.33 power is equivalent to applying a cube root transformation.

Here, I use the *transformTukey* function, which performs iterative Shapiro–Wilk tests, and finds the *lambda* value that maximizes the W statistic from those tests. In essence, this finds the power transformation that makes the data fit the normal distribution as closely as possible with this type of transformation.

Left skewed values should be adjusted with (constant – value), to convert the skew to right skewed, and perhaps making all values positive. In some cases of right skewed data, it may be beneficial to add a constant to make all data values positive before transformation. For large values, it may be helpful to scale values to a more reasonable range.

In this example, the resultant *lambda* of -0.1 is slightly stronger than a log transformation, since a log transformation corresponds to a *lambda* of 0.

```
### Some options require NCStats

library(rcompanion)

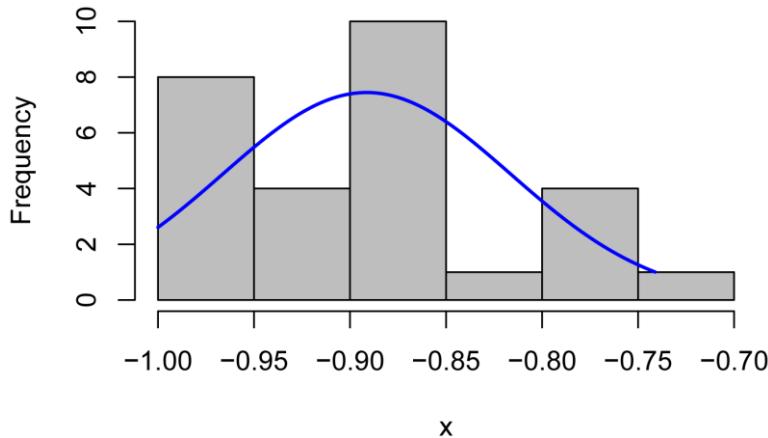
T_tuk =
  transformTukey(Turbidity,
    plotit=FALSE)

  lambda      w Shapiro.p.value
397   -0.1  0.935       0.08248

  if (lambda >  0){TRANS = x ^ lambda}
  if (lambda == 0){TRANS = log(x)}
  if (lambda <  0){TRANS = -1 * x ^ lambda}

library(rcompanion)

plotNormalHistogram(T_tuk)
```



Example of Tukey-transformed data in ANOVA

For an example of how transforming data can improve the distribution of the residuals of a parametric analysis, we will use the same turbidity values, but assign them to three different locations.

Transforming the turbidity values to be more normally distributed, both improves the distribution of the residuals of the analysis and makes a more powerful test, lowering the p -value.

```
Input =""
Location Turbidity
a      1.0
a      1.2
a      1.1
a      1.1
a      2.4
a      2.2
a      2.6
a      4.1
a      5.0
a     10.0
b      4.0
b      4.1
b      4.2
b      4.1
b      5.1
b      4.5
b      5.0
b     15.2
b     10.0
b     20.0
c      1.1
c      1.1
c      1.2
c      1.6
c      2.2
c      3.0
c      4.0
c     10.5
```

")

```
Data = read.table(textConnection(Input), header=TRUE)
```

Attempt ANOVA on un-transformed data

Here, even though the analysis of variance results in a significant p -value ($p = 0.03$), the residuals deviate from the normal distribution enough to make the analysis invalid. The plot of the residuals vs. the fitted values shows that the residuals are somewhat heteroscedastic, though not terribly so.

```
boxplot(Turbidity ~ Location,
       data = Data,
       ylab="Turbidity",
       xlab="Location")
```

```
model = lm(Turbidity ~ Location,
           data=Data)
```

```
library(car)
```

```
Anova(model, type="II")
```

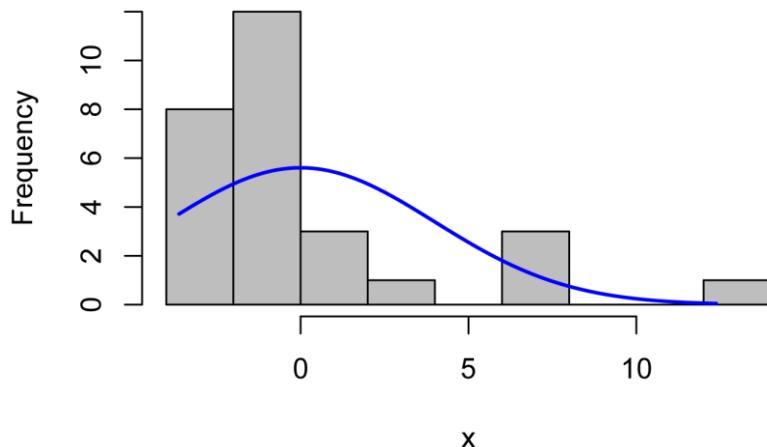
Anova Table (Type II tests)

	Sum Sq	Df	F value	Pr(>F)
Location	132.63	2	3.8651	0.03447 *
Residuals	428.95	25		

```
x = (residuals(model))
```

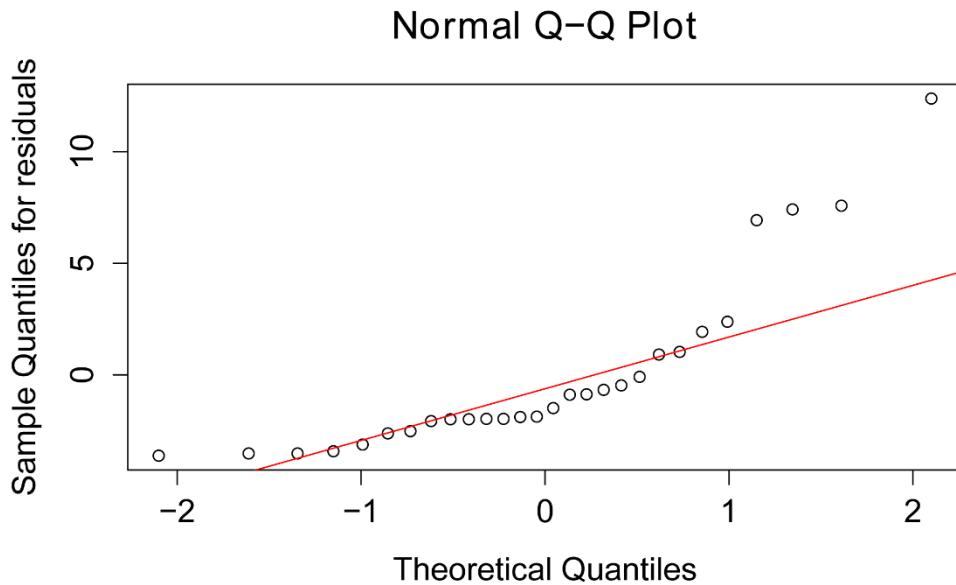
```
library(rcompanion)
```

```
plotNormalHistogram(x)
```

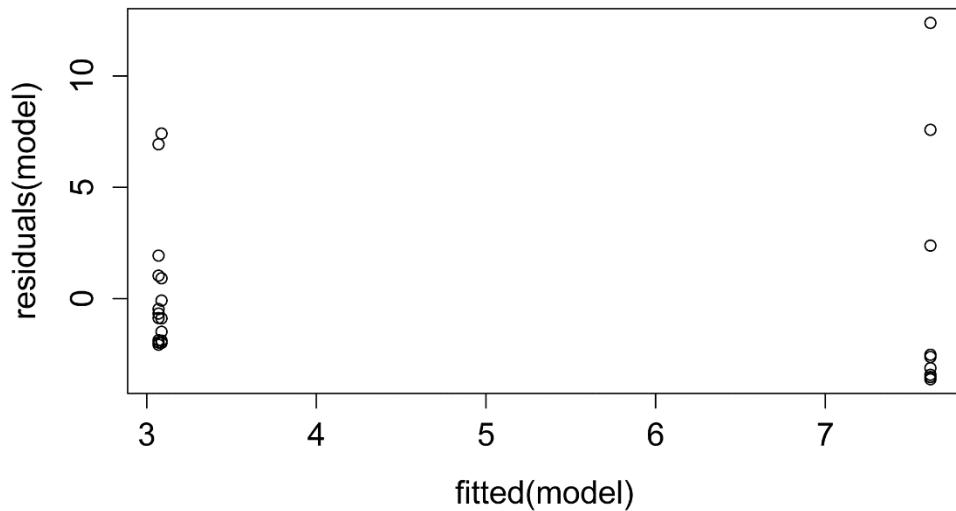


```
qqnorm(residuals(model),
       ylab="Sample Quantiles for residuals")
```

```
qqline(residuals(model),
      col="red")
```



```
plot(fitted(model),
      residuals(model))
```



Transform data

```
### Some options require NCStats
library(rcompanion)

Data$Turbidity_tuk =
  transformTukey(Data$Turbidity,
    plotit=FALSE)
```

```

lambda      w Shapiro.p.value
397   -0.1 0.935           0.08248

if (lambda > 0){TRANS = x ^ lambda}
if (lambda == 0){TRANS = log(x)}
if (lambda < 0){TRANS = -1 * x ^ lambda}

```

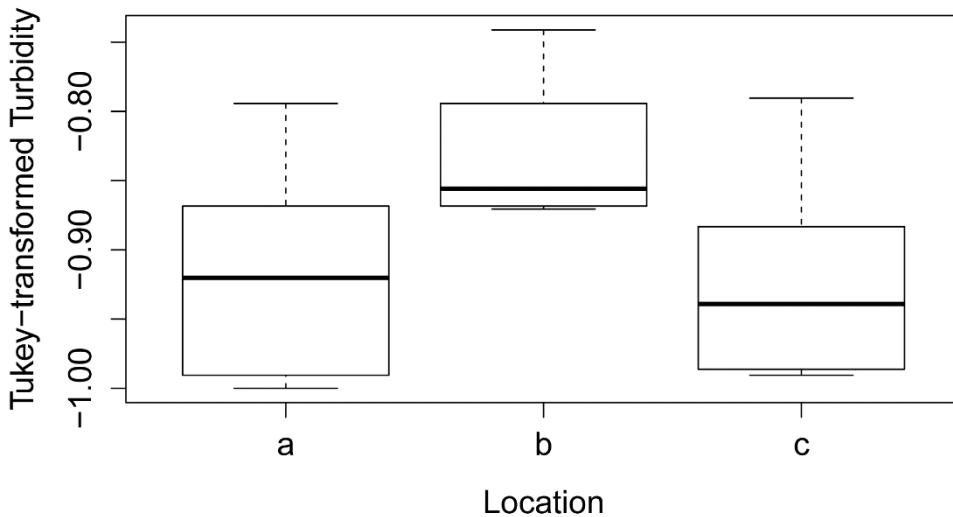
ANOVA with Tukey-transformed data

After transformation, the residuals from the ANOVA are closer to a normal distribution—although not perfectly—, making the F -test more appropriate. In addition, the test is more powerful as indicated by the lower p -value ($p = 0.005$) than with the untransformed data. The plot of the residuals vs. the fitted values shows that the residuals are about as heteroscedastic as they were with the untransformed data.

```

boxplot(Turbidity_tuk ~ Location,
        data = Data,
        ylab="Tukey-transformed Turbidity",
        xlab="Location")

```



```

model = lm(Turbidity_tuk ~ Location,
            data=Data)

library(car)

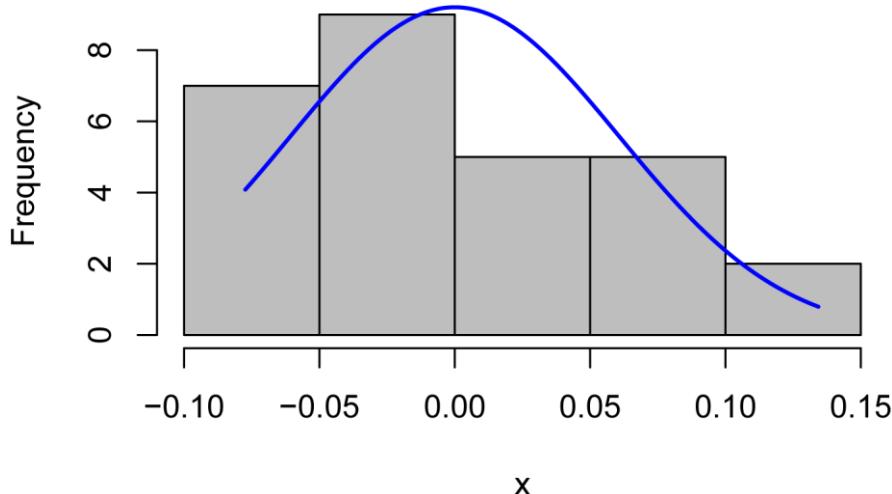
Anova(model, type="II")

Anova Table (Type II tests)

          Sum Sq Df F value    Pr(>F)
Location 0.052506  2 6.6018 0.004988 ***
Residuals 0.099416 25

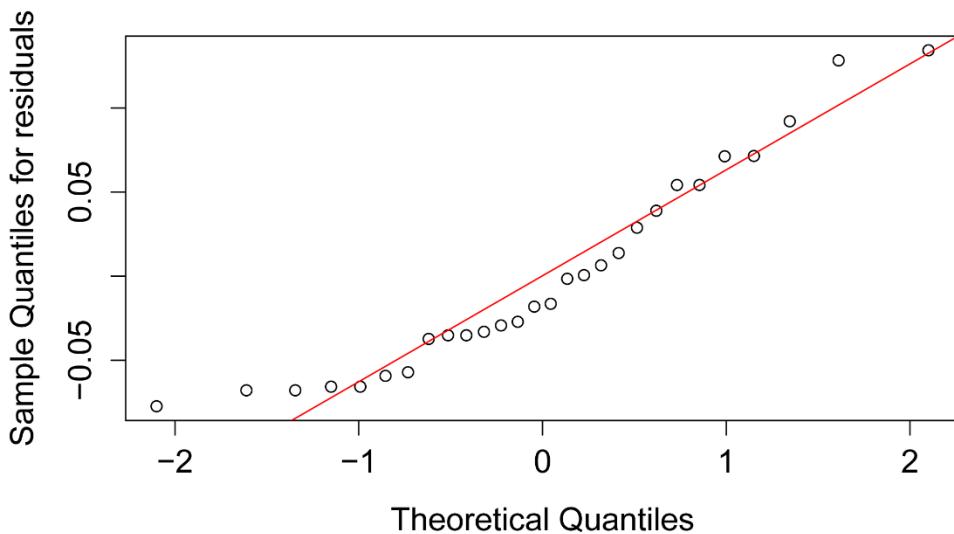
```

```
x = residuals(model)
library(rcompanion)
plotNormalHistogram(x)
```

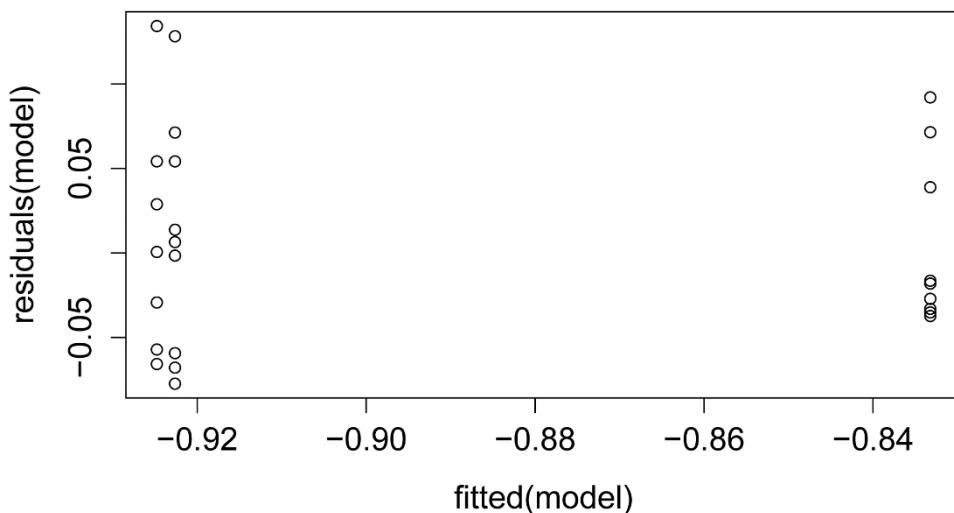


```
qqnorm(residuals(model),
       ylab="Sample Quantiles for residuals")
qqline(residuals(model),
       col="red")
```

Normal Q-Q Plot



```
plot(fitted(model),
      residuals(model))
```



Box-Cox transformation

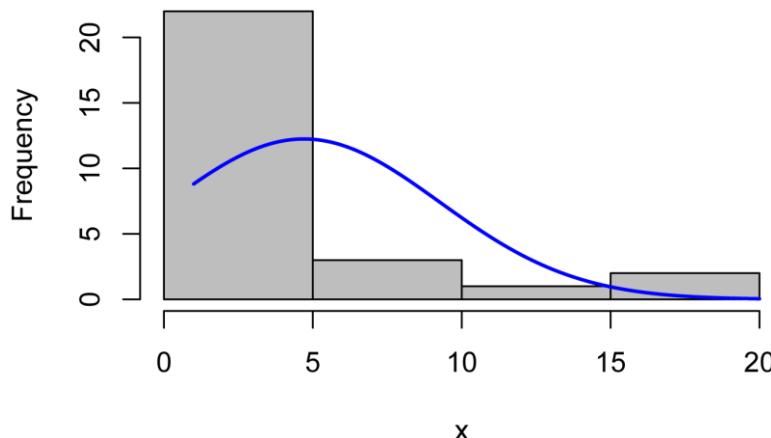
The Box-Cox procedure is similar in concept to the Tukey Ladder of Power procedure described above. However, instead of transforming a single variable, it maximizes a log-likelihood statistic for a linear model (such as ANOVA or linear regression). It will also work on a single variable using a formula of $x \sim 1$.

The Box-Cox procedure is available with the *boxcox* function in the *MASS* package. However, a few steps are needed to extract the *lambda* value and transform the data set.

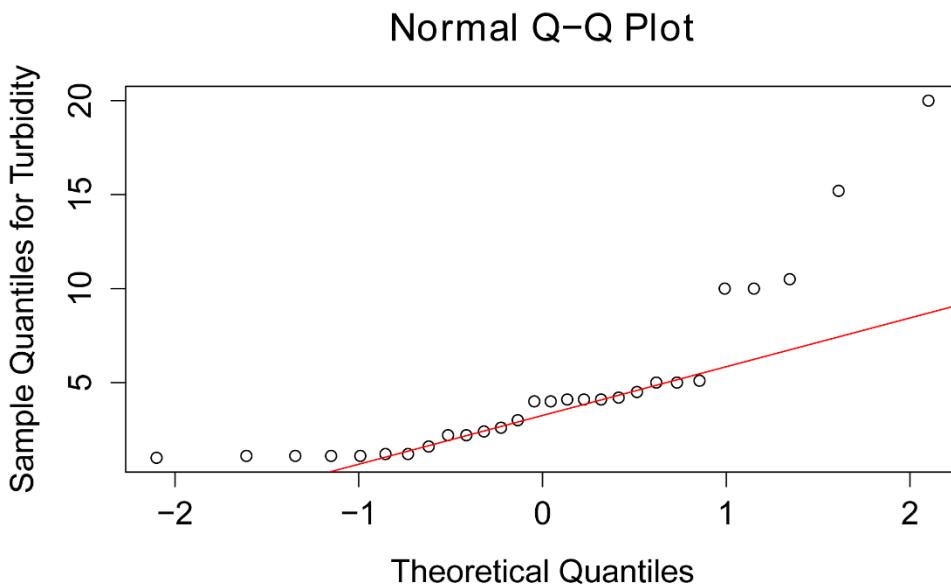
This example uses the same turbidity data.

```
Turbidity = c(1.0, 1.2, 1.1, 1.1, 2.4, 2.2, 2.6, 4.1, 5.0, 10.0, 4.0, 4.1, 4.2, 4.1,
5.1, 4.5, 5.0, 15.2, 10.0, 20.0, 1.1, 1.1, 1.2, 1.6, 2.2, 3.0, 4.0, 10.5)
```

```
library(rcompanion)
plotNormalHistogram(Turbidity)
```



```
qqnorm(Turbidity,
       ylab="Sample Quantiles for Turbidity")
qqline(Turbidity,
       col="red")
```



Box-Cox transformation for a single variable

```
library(MASS)

Box = boxcox(Turbidity ~ 1,
              lambda = seq(-6,6,0.1))          # Transform Turbidity as a single vector
                                                # Try values -6 to 6 by 0.1

Cox = data.frame(Box$x, Box$y)           # Create a data frame with the results

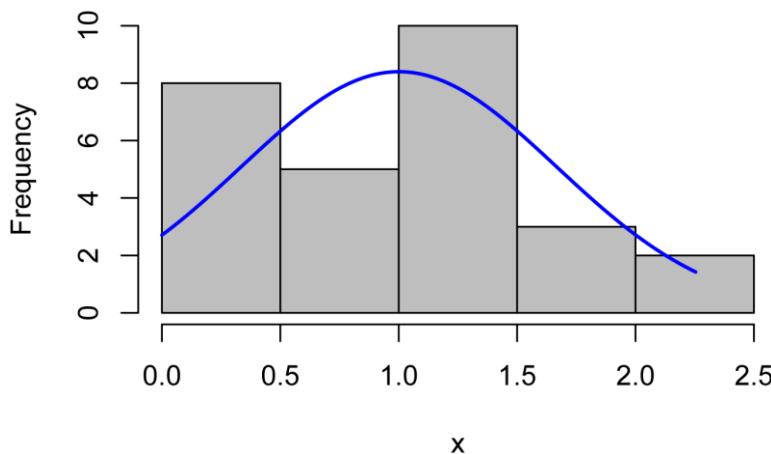
Cox2 = Cox[with(Cox, order(-Cox$Box.y)),] # Order the new data frame by decreasing y

Cox2[1,]                                    # Display the lambda with the greatest
                                            # log likelihood

      Box.x      Box.y
59   -0.2 -41.35829

lambda = Cox2[1, "Box.x"]                  # Extract that lambda
T_box = (Turbidity ^ lambda - 1)/lambda    # Transform the original data

library(rcompanion)
plotNormalHistogram(T_box)
```



Example of Box-Cox transformation for ANOVA model

```

Input =""
Location Turbidity
a      1.0
a      1.2
a      1.1
a      1.1
a      2.4
a      2.2
a      2.6
a      4.1
a      5.0
a     10.0
b      4.0
b      4.1
b      4.2
b      4.1
b      5.1
b      4.5
b      5.0
b     15.2
b     10.0
b     20.0
c      1.1
c      1.1
c      1.2
c      1.6
c      2.2
c      3.0
c      4.0
c     10.5
")

Data = read.table(textConnection(Input), header=TRUE)

```

Attempt ANOVA on un-transformed data

```
model = lm(Turbidity ~ Location,
           data=Data)

library(car)

Anova(model, type="II")

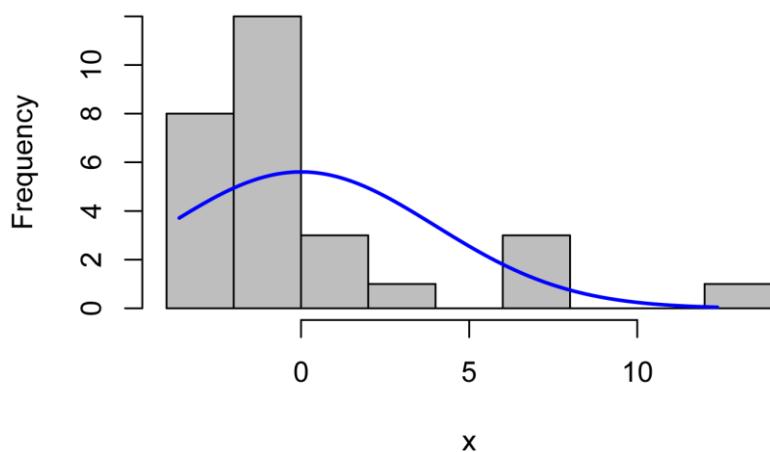
Anova Table (Type II tests)

Sum Sq Df F value Pr(>F)
Location 132.63 2 3.8651 0.03447 *
Residuals 428.95 25
```

```
x = residuals(model)

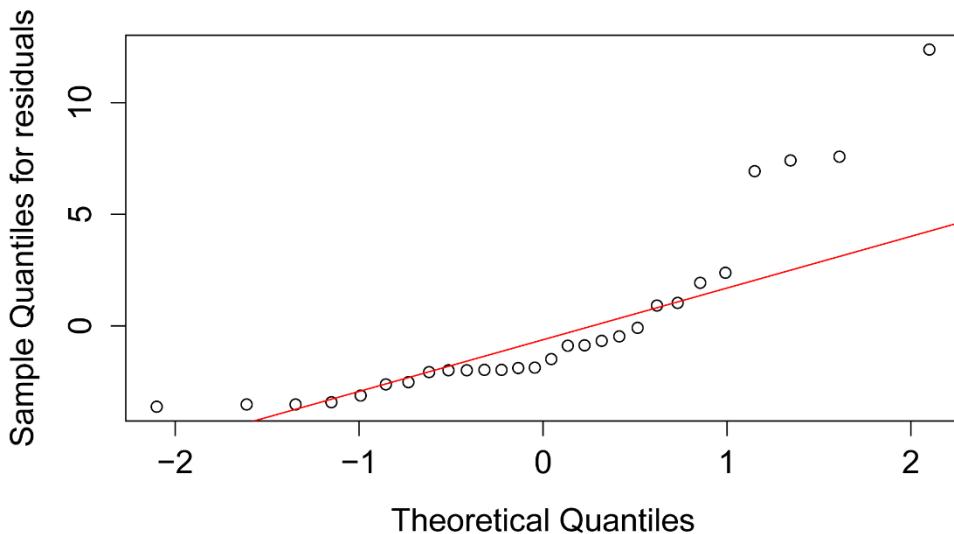
library(rcompanion)

plotNormalHistogram(x)
```

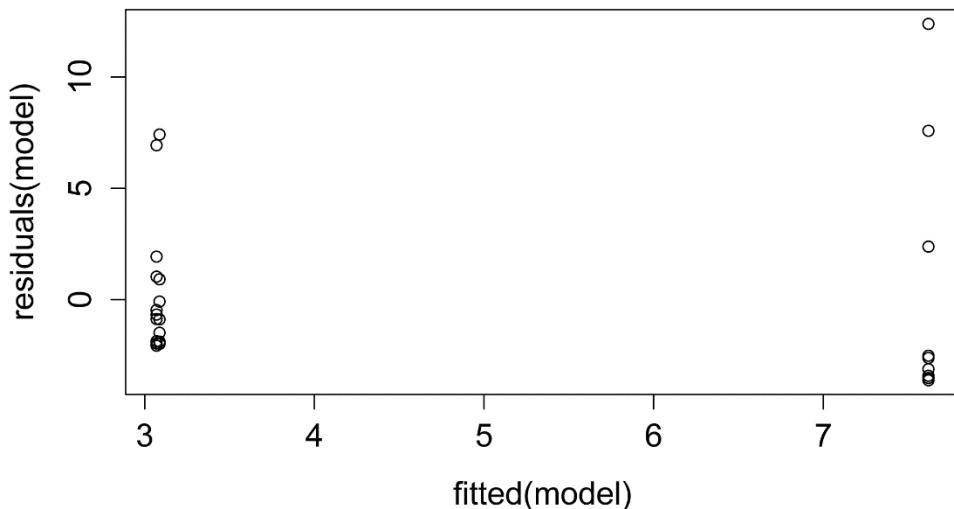


```
qqnorm(residuals(model),
       ylab="Sample Quantiles for residuals")
qqline(residuals(model),
      col="red")
```

Normal Q-Q Plot



```
plot(fitted(model),
      residuals(model))
```



Transform data

```
library(MASS)

Box = boxcox(Turbidity ~ Location,
              data = Data,
              lambda = seq(-6,6,0.1)
            )

Cox = data.frame(Box$x, Box$y)

cox2 = Cox[with(Cox, order(-Cox$Box.y)),]
```

```

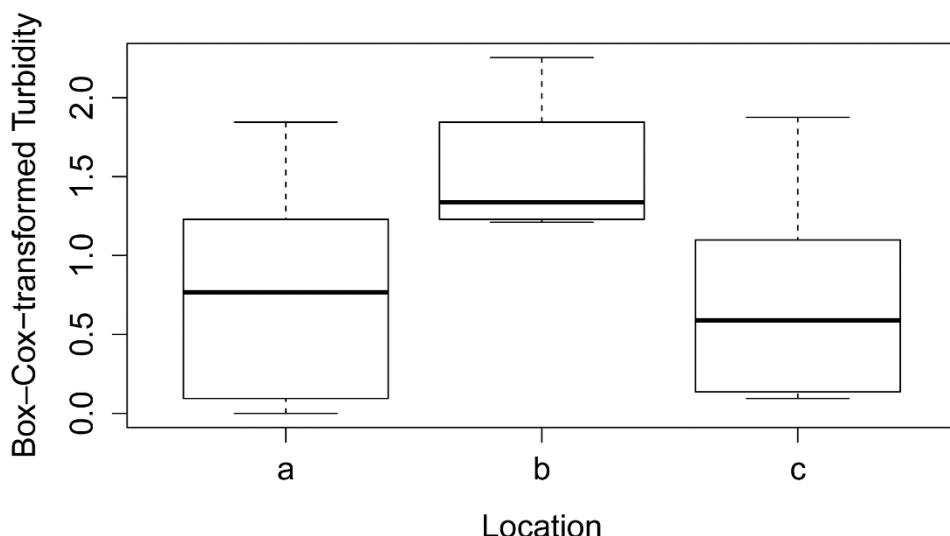
cox2[1,]

lambda = Cox2[1, "Box.x"]

Data$Turbidity_box = (Data$Turbidity ^ lambda - 1)/lambda

boxplot(Turbidity_box ~ Location,
        data = Data,
        ylab="Box-Cox-transformed Turbidity",
        xlab="Location")

```



Perform ANOVA and check residuals

```

model = lm(Turbidity_box ~ Location,
           data=Data)

library(car)

Anova(model, type="II")

      Anova Table (Type II tests)

      Sum Sq Df F value Pr(>F)
Location 0.16657  2 6.6929 0.0047 **
Residuals 0.31110 25

```

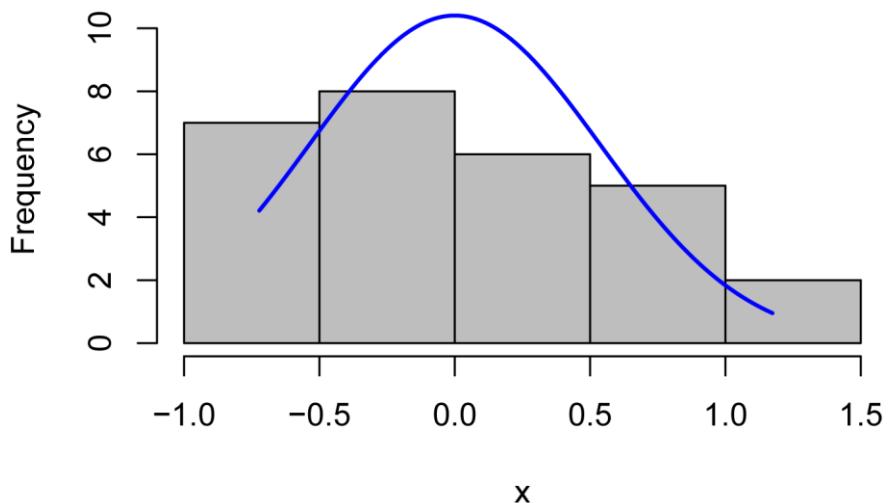
```

x = residuals(model)

library(rcompanion)

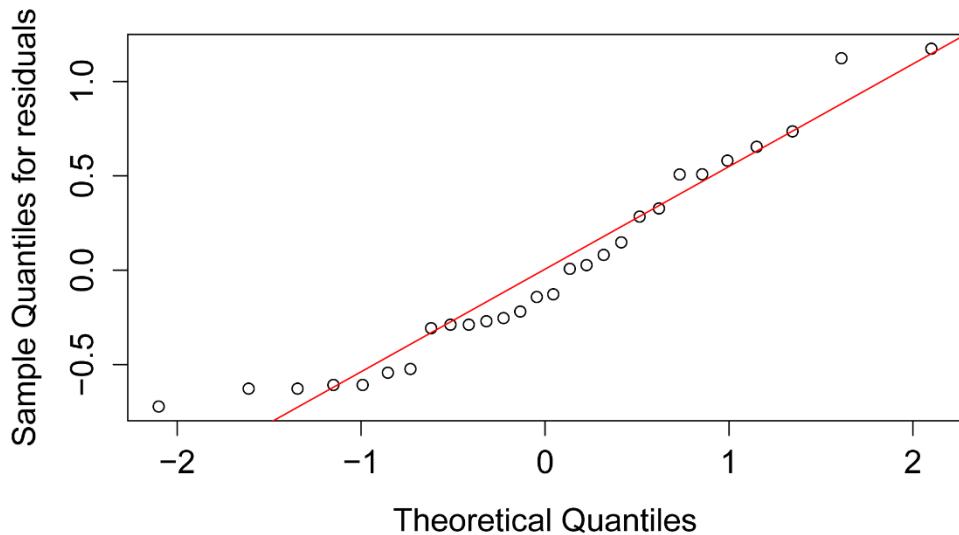
plotNormalHistogram(x)

```

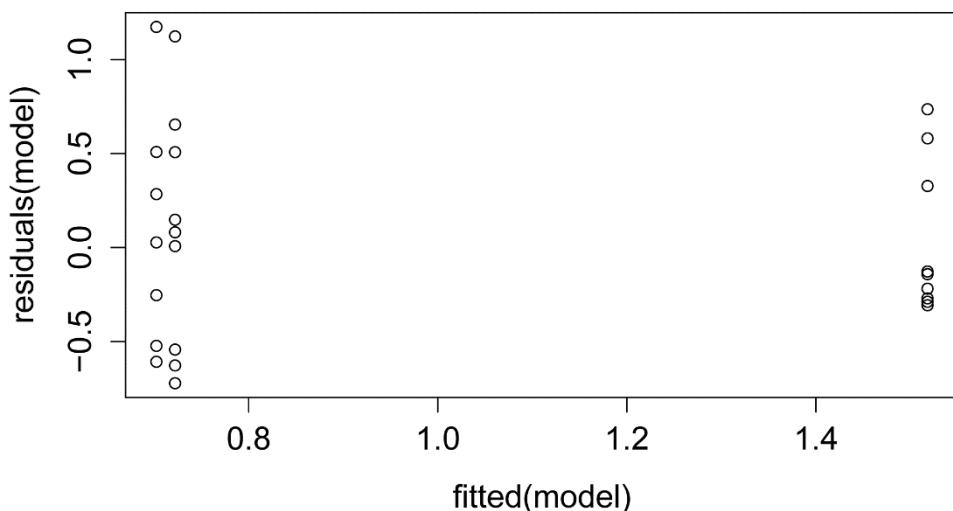


```
qqnorm(residuals(model),  
       ylab="Sample Quantiles for residuals")  
qqline(residuals(model),  
       col="red")
```

Normal Q-Q Plot



```
plot(fitted(model),  
      residuals(model))
```



Conclusions

Both the Tukey's Ladder of Powers principle as implemented by the *transformTukey* function and the Box–Cox procedure were successful at transforming a single variable to follow a more normal distribution. They were also both successful at improving the distribution of residuals from a simple ANOVA.

The Box–Cox procedure has the advantage of dealing with the dependent variable of a linear model, while the *transformTukey* function works only for a single variable without considering other variables. Because of this, the Box–Cox procedure may be advantageous when a relatively simple model is considered. In cases where there are complex models or multiple regression, it may be helpful to transform both dependent and independent variables independently.

Analysis of Count Data and Percentage Data

Regression for Count Data

Introduction

Count data

In general, common parametric tests like t-test and anova shouldn't be used for count data. One reason is technical in nature: that parametric analyses require continuous data. Count data is by its nature discrete and is left-censored at zero. (That is, usually counts can't be less than zero.)

A second reason is more practical in nature. Count data are often highly skewed, and often produce skewed residuals if a parametric approach is attempted. In this case, the hypothesis tests will not be accurate.

For further discussion, see the "Count data may not be appropriate for common parametric tests" section in the *Introduction to Parametric Tests* chapter.

Regression approaches for count data

The most common regression approach for handling count data is probably Poisson regression. However, Poisson regression makes assumptions about the distribution of the data that may not be appropriate in all cases. Hermite regression is a more flexible approach, but at the time of writing doesn't have a complete set of support functions in R. Quasi-Poisson regression is also flexible with data assumptions, but also but at the time of writing doesn't have a complete set of support functions in R. Negative binomial regression allows for overdispersion in data; and zero-inflated regression is useful when there are a high proportion of zero counts in the data.

Cautionary note

Note that model assumptions and pitfalls of these regression techniques are not discussed in depth here. The reader is urged to understand the assumptions of this kind of modeling before proceeding.

Generalized linear regression

Poisson, Hermite, and related regression approaches are a type of *generalized linear model*. This should not be confused with general linear model, which is implemented with the *lm* function. Generalized linear models are implemented with the *glm* function or other functions.

Generalized linear models are used when the dependent variable is count, binary, multinomial, etc. More information on using the *glm* function can be found by using *help(glm)* and *help(family)*. For examples of logistic regression, see the chapter *Models for Nominal Data*; the chapter *Beta Regression for Percent and Proportion Data*; or Mangiafico (2015) in the "References" section. For a table of common uses for family and link function in generalized linear models, see the Wikipedia article in the "References" section for this chapter.

Packages used in this chapter

The packages used in this chapter include:

- psych
- hermite
- lattice
- plyr
- boot
- DescTools
- ggplot2
- car
- multcompView
- emmeans
- MASS
- pscl
- rcompanion
- robust

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(hermite)){install.packages("hermite")}
if(!require(lattice)){install.packages("lattice")}
if(!require(plyr)){install.packages("plyr")}
if(!require(boot)){install.packages("boot")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(MASS)){install.packages("MASS")}
if(!require(pscl)){install.packages("pscl")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(robust)){install.packages("robust")}
```

Count data example

In this example, extension researchers have set up garden plots with different suites of plants, with each suite identified as a level of the variable *Garden* below. In September, they counted the number of monarch butterflies in each garden plot.

```
Input = "
Garden    Monarchs
A          0
A          4
A          2
A          2
A          0
A          6
A          0
A          0
B          5
B          9
```

```

B      7
B      5
B      7
B      5
B      9
B      5
C     10
C     14
C     12
C     12
C     10
C     16
C     10
C     10
")
```

```

Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Garden = factor(Data$Garden,
                      levels=unique(Data$Garden))

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

```

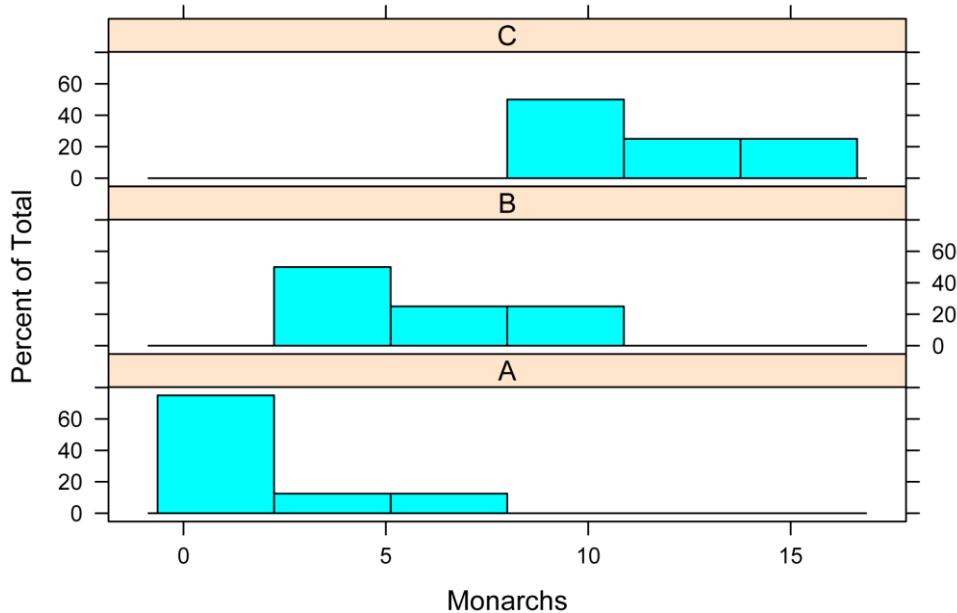
Histograms

```

library(lattice)

histogram(~ Monarchs | Garden,
          data>Data,
          layout=c(1,3)      # columns and rows of individual plots
          )

```



Poisson regression example

Poisson regression makes certain assumptions about the relationship between the mean and the dispersion of the dependent variable. Because this assumption may not be met for all data sets, Poisson regression may not be recommended for routine use. Particularly, classic Poisson regression should be avoided if there is overdispersion in the data or if there are several zero counts in the dependent variable.

An alternate approach for data with overdispersion is negative binomial regression.

An alternative approach for data with many zeros is zero-inflated Poisson regression.

For further discussion, see the “Count data may not be appropriate for common parametric tests” section in the *Introduction to Parametric Tests* chapter.

Note that model assumptions and pitfalls of this approach are not discussed here. The reader is urged to understand the assumptions of this kind of modeling before proceeding.

```
model.p = glm(Monarchs ~ Garden,
               data=Data,
               family="poisson")
```

```
library(car)
```

```
Anova(model.p,
       type="II",
       test="LR")
```

Analysis of Deviance Table (Type II tests)

```

LR Chisq Df Pr(>Chisq)
Garden  66.463  2  3.697e-15 ***

```

```

library(rcompanion)

nagelkerke(model.p)

$Pseudo.R.squared.for.model.vs.null
                               Pseudo.R.squared
McFadden                      0.387929
Cox and Snell (ML)            0.937293
Nagelkerke (Cragg and Uhler)  0.938037

$Likelihood.ratio.test
  Df.diff LogLik.diff  chisq   p.value
    -2      -33.231 66.463 3.6967e-15

```

```

library(multcompView)

library(emmeans)

marginal = emmeans(model.p,
 ~ Garden)

pairs(marginal,
       adjust="tukey")

cld(marginal,
     alpha=0.05,
     Letters=letters, ### Use lower-case letters for .group
     adjust="tukey")  ### Tukey adjustment for multiple comparisons

Garden      emmean        SE df  asymp.LCL asymp.UCL .group
A          0.5596158 0.2672450 NA -0.07849527  1.197727  a
B          1.8718022 0.1386750 NA  1.54068251  2.202922  b
C          2.4638532 0.1031421 NA  2.21757688  2.710130  c

Results are given on the log (not the response) scale.
Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
Tests are performed on the log scale
significance level used: alpha = 0.05

### Note that estimates are on log scale

```

Negative binomial regression example

Negative binomial regression is similar in application to Poisson regression, but allows for overdispersion in the dependent count variable.

This example will use the *glm.nb* function in the *MASS* package. The *Anova* function in the *car* package will be used for an analysis of deviance, and the *nagelkerke* function will be used to determine a *p*-value and pseudo *R-squared* value for the model. Post-hoc analysis can be conducted with the *emmeans* package.

Note that model assumptions and pitfalls of this approach are not discussed here. The reader is urged to understand the assumptions of this kind of modeling before proceeding.

```
library(MASS)

model.nb = glm.nb(Monarchs ~ Garden,
                  data=Data,
                  control = glm.control(maxit=10000))
```

```
library(car)
```

```
Anova(model.nb,
      type="II",
      test="LR")
```

Analysis of Deviance Table (Type II tests)

	LR	Chisq	Df	Pr(>Chisq)
Garden	66.464	2		3.694e-15 ***

```
library(rcompanion)
```

```
nagelkerke(model.nb)
```

	Pseudo.R.squared.for.model1.vs.null	Pseudo.R.squared
McFadden		0.255141
Cox and Snell (ML)		0.776007
Nagelkerke (Cragg and Uhler)		0.778217

	\$Likelihood.ratio.test		
Df.diff	LogLik.diff	Chisq	p.value
-2	-17.954	35.907	1.5952e-08

```
library(multcompView)
```

```
library(emmeans)
```

```
marginal = emmeans(model.nb,
                     ~ Garden)
```

```
pairs(marginal,
       adjust="tukey")
```

```
cld(marginal,
```

```

alpha    = 0.05,
Letters = letters,     ### Use lower-case letters for .group
type    = "response",  ### Report emmeans in original scale
adjust   = "tukey")    ### Tukey adjustment for multiple comparisons

Garden response      SE df asymp.LCL asymp.UCL .group
A          1.75 0.4677072 NA 0.9244706 3.312707 a
B          6.50 0.9013878 NA 4.6677750 9.051422 b
C         11.75 1.2119200 NA 9.1850474 15.031223 c

Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
Intervals are back-transformed from the log scale
P value adjustment: tukey method for comparing a family of 3 estimates
Tests are performed on the log scale
significance level used: alpha = 0.05

```

Zero-inflated regression example

Zero-inflated regression is similar in application to Poisson regression, but allows for an abundance of zeros in the dependent count variable.

This example will use the *zeroinfl* function in the *pscl* package. The *Anova* function in the *car* package will be used for an analysis of deviance, and the *nagelkerke* function will be used to determine a *p*-value and pseudo *R-squared* value for the model. Post-hoc analysis can be conducted with the *emmeans* package.

```

library(pscl)

model.zi = zeroinfl(Monarchs ~ Garden,
                     data = Data,
                     dist = "poisson")

      ### dist = "negbin" may be used

summary(model.zi)

Call:
zeroinfl(formula = Monarchs ~ Garden | Garden, data = Data, dist = "poisson")

Count model coefficients (poisson with log link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 1.2182     0.2847   4.278 1.89e-05 ***
GardenB     0.6536     0.3167   2.064    0.039 *
GardenC     1.2457     0.3029   4.113 3.90e-05 ***

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.046e-02 7.363e-01 -0.096    0.924
GardenB     -2.057e+01 1.071e+04 -0.002    0.998
GardenC     -2.057e+01 1.071e+04 -0.002    0.998

```

```
### Note that there are separate coefficients for the
### Poisson part of the analysis and for the zero-inflation part.
```

```
library(car)

Anova(model.zi,
      type="II",
      test="chisq")

Analysis of Deviance Table (Type II tests)

  Df  Chisq Pr(>Chisq)
Garden  2 23.914 6.414e-06 ***
```

```
library(rcompanion)

nagelkerke(model.zi)

$Pseudo.R.squared.for.model.vs.null
                  Pseudo.R.squared
McFadden           0.284636
Cox and Snell (ML) 0.797356
Nagelkerke (Cragg and Uhler) 0.800291

$Likelihood.ratio.test
  Df.diff LogLik.diff  chisq   p.value
    -4       -19.156 38.311 9.6649e-08
```

```
library(multcompView)

library(emmeans)

marginal = emmeans(model.zi,
                     ~ Garden)

pairs(marginal,
      adjust="tukey")

cld(marginal,
     alpha=0.05,
     Letters=letters, ### Use lower-case letters for .group
     adjust="tukey")  ### Tukey adjustment for multiple comparisons

Garden  emmean        SE df  asymp.LCL asymp.UCL .group
A        1.75 0.7586301 NA -0.06140972  3.561410  a
B        6.50 0.9013877 NA  4.34772229  8.652278  b
C       11.75 1.2119199 NA  8.85625304 14.643747  c

Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
Significance level used: alpha = 0.05
```

```
### Note, emmeans are on the original measurement scale
```

Robust Poisson regression example

Robust Poisson regression is robust to outliers in the dependent variable.

This example uses the *glmRob* function in the *robust* package. The *anova* function can be used to conduct an analysis of deviance. The *p*-value for the model can be found by comparing the model to a null model. However, at the time of writing, I don't know of any way to determine AIC or pseudo *R-squared* for the model.

At the time of writing, the *glmRob* function can only use the Poisson and binomial families of models.

An alternate method is the *glmrob* function in the *robustbase* package.

```
library(robust)

model.rob = glmRob(Monarchs ~ Garden,
                    data = Data,
                    family = "poisson")

anova(model.rob, test="Chisq")
      Df Deviance Resid. Df Resid. Dev      Pr(>Chi)
NULL     NA         NA      23   430.19850               NA
Garden   2  400.9221       21   29.27641 3.567693e-63

model.rob.null = glmRob(Monarchs ~ 1,
                        data = Data,
                        family = "poisson")

anova(model.rob.null, model.rob, test="chisq")
      Terms Resid. Df Resid. Dev Test Df Deviance      Pr(>Chi)
1          1      23   95.12606    NA      NA               NA
2 Garden      21   29.27641      2  65.84965 5.536815e-11
```

Quasi-Poisson regression

Quasi-Poisson regression is useful since it has a variable dispersion parameter, so that it can model over-dispersed data. It may be better than negative binomial regression in some circumstances (Verhoef and Boveng. 2007).

At the time of writing, Quasi-Poisson regression doesn't have complete set of support functions in R. Using the *quasipoisson* family option in the *glm* function, the results will have the same parameter coefficients as with the *poisson* option, but the inference statistics are adjusted in the *summary*

function. The *Anova* function in the *car* package can be used for an analysis of deviance table, and the *emmeans* package can be used for post-hoc comparisons. Since the model doesn't produce a log-likelihood value, I don't know a way to produce a *p*-value for the mode, for a pseudo *R-squared* value for the model.

```
model.qp = glm(Monarchs ~ Garden,
                data=Data,
                family="quasipoisson")

library(car)

Anova(model.qp,
      type="II",
      test="LR")

Analysis of Deviance Table (Type II tests)

Response: Monarchs
          LR Chisq Df Pr(>Chisq)
Garden   52.286  2  4.429e-12 ***

library(multcompView)

library(emmeans)

marginal = emmeans(model.qp,
                     ~ Garden)

pairs(marginal,
      adjust="tukey")

cld(marginal,
     alpha=0.05,
     Letters=letters,
     adjust="tukey")

Garden    emmean        SE df  asymp.LCL asymp.UCL .group
A         0.5596158 0.3013057 NA -0.1598233  1.279055   a
B         1.8718022 0.1563493 NA  1.4984809  2.245123   b
C         2.4638532 0.1162877 NA  2.1861887  2.741518   c

Results are given on the log (not the response) scale.
Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05
```

Hermite regression

The generalized Hermite distribution is a more general distribution that can handle overdispersion or multimodality (Moriña and others, 2015). This makes generalized Hermite regression a powerful and flexible tool for modeling count data. It is implemented with the *hermite* package.

Fitting models with the *hermite* package can be somewhat difficult. One issue is that model fitting may fail without some parameters being specified. Often specifying an appropriate value for the *m* option will help.

A further difficulty with this approach is that, at the time writing, the package isn't supported by the *anova* function to compare models, the *Anova* function to test effects, or other useful functions like *emmeans* for factor effects.

The *hermite* package is used to conduct hermite regression. Here, the *m=3* option is specified. Often the default *m=NULL* can be used. In this case, if the *m* value is not specified, the function cannot complete the model fitting, and errors are produced. Using *m=2* often works. Here, *m=3* was used because it produced a model with a lower AIC than did the *m=2* option.

```
library(hermite)

model = glm.hermite(Monarchs ~ Garden,
                     data = Data,
                     link = "log",
                     m=3)

summary(model)

Coefficients:
              Estimate Std. Error   z value    p-value
(Intercept) 0.5081083  0.3251349 1.5627612 1.181088e-01
GardenB     1.3700567  0.3641379 3.7624662 1.682461e-04
Gardenc     1.9596153  0.3476326 5.6370291 1.730089e-08
dispersion.index 1.0820807  0.2877977 0.1281707 3.601681e-01
order        3.0000000      NA       NA       NA
(Likelihood ratio test against Poisson is reported by *z value* for
*dispersion.index*)

AIC:  112.7762
```

Post-hoc analysis: Medians and confidence intervals

At the time of writing, the *emmeans* package does not support post-hoc analysis of regressions produced with the *hermite* package.

One imperfect approach for post-hoc analysis would be to examine median counts for treatments and the confidence intervals of these medians. We can conclude that groups with non-overlapping 95% confidence intervals for their medians are significantly different.

However, this approach does not represent any information learned from the Hermite regression.

A second issue is that, because the dependent variable is not continuous, the distribution of the bootstrapped confidence intervals is not likely to be continuous, and so is may not be reliable.

To get confidence intervals for the medians for each group, we will use the *groupwiseMedian* function. Here I used the percentile method for confidence intervals.

```
library(rcompanion)

Sum = groupwiseMedian(Monarchs ~ Garden,
                      data=Data,
                      conf=0.95,
                      R=5000,
                      percentile=TRUE,
                      bca=FALSE,
                      digits=3)

Sum

  Garden n Median Conf.level Percentile.lower Percentile.upper
1     A 8     1    0.95          0             4
2     B 8     6    0.95          5             8
3     C 8    11    0.95         10            14

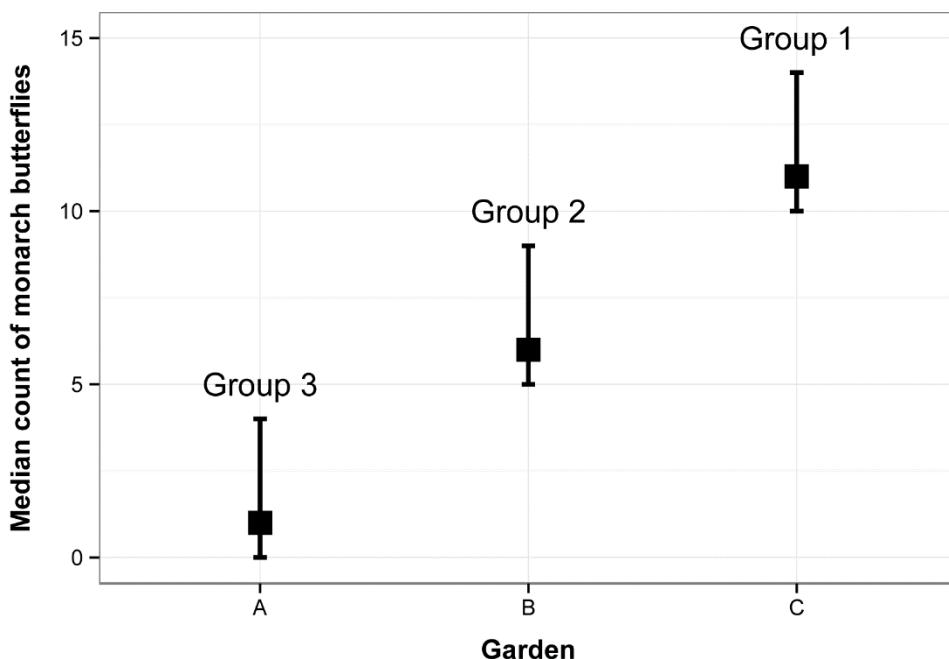
### In this case, none of the confidence intervals overlap.
```

Plot of medians and confidence intervals

The data frame *Sum* created above will be passed to *ggplot* for plotting. At the end of the code, *annotate* is used to add text to the plot to indicate which medians are significantly different from one another.

```
library(ggplot2)

ggplot(Sum,                   ### The data frame to use.
       aes(x = Garden,
           y = Median)) +
  geom_errorbar(aes(ymin = Percentile.lower,
                    ymax = Percentile.upper),
                width = 0.05,
                size = 1) +
  geom_point(shape = 15,
             size = 5) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("Median count of monarch butterflies") +
  annotate("text",
          x = 1:3,
          y = c(5, 10, 15),
          label = c("Group 3", "Group 2", "Group 1"))
```



Optional code for chi-square goodness-of-fit test

An alternative approach to handling count data is to sum up the counts for treatments, and use a chi-square test or related test. Here, a chi-square goodness-of-fit test is used to see if counts differ from “expected” equal proportions.

Omnibus test

```
Tabla = xtabs(Monarchs ~ Garden,
               data = Data)

Tabla
Garden
  A   B   C
14  52  94

chisq.test(Tabla)
Chi-squared test for given probabilities
X-squared = 60.05, df = 2, p-value = 9.127e-14
```

Post-hoc chi-square tests

```
Garden.A = sum(Data$Monarchs[Data$Garden=="A"])
Garden.B = sum(Data$Monarchs[Data$Garden=="B"])
```

```
Garden.C = sum(Data$Monarchs[Data$Garden=="C"])

observed = c(Garden.A, Garden.B)      # observed frequencies
expected = c(1/2, 1/2)                # expected proportions

chisq.test(x = observed,
           p = expected)

Chi-squared test for given probabilities

X-squared = 21.879, df = 1, p-value = 2.904e-06

observed = c(Garden.B, Garden.C)      # observed frequencies
expected = c(1/2, 1/2)                # expected proportions

chisq.test(x = observed,
           p = expected)

Chi-squared test for given probabilities

X-squared = 12.082, df = 1, p-value = 0.0005091

observed = c(Garden.A, Garden.C)      # observed frequencies
expected = c(1/2, 1/2)                # expected proportions

chisq.test(x = observed,
           p = expected)

Chi-squared test for given probabilities

X-squared = 59.259, df = 1, p-value = 1.382e-14
```

Optional analysis: Vuong test to compare Poisson, negative binomial, and zero-inflated models

The Vuong test, implemented by the *pscl* package, can test two non-nested models. It works with *negbin*, *zeroinfl*, and some *glm* model objects which are fitted to the same data.

The null hypothesis is that there is no difference in models. The function produces three tests, a “Raw” test, an AIC-corrected, and a BIC-corrected, any of which could be used.

It has been suggested that the Vuong test not be used to test for zero-inflation (Wilson, 2015).

Define models

```
model.p = glm(Monarchs ~ Garden,
               data=Data,
               family="poisson")

library(MASS)
```

```
model.nb = glm.nb(Monarchs ~ Garden,
                   data=Data,
                   control = glm.control(maxit=10000))

library(pscl)

model.zi = zeroinfl(Monarchs ~ Garden,
                     data = Data,
                     dist = "poisson")
```

Vuong test

```
library(pscl)

vuong(model.p,
      model.nb,
      digits = 4)

Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishable)
-----
          Vuong z-statistic           H_A p-value
Raw            0.03324988 model1 > model2 0.48674
AIC-corrected  0.03324988 model1 > model2 0.48674
BIC-corrected  0.03324988 model1 > model2 0.48674

### Positive Vuong z-statistic suggests that model 1 is superior,
### but, in this case, the difference is not significant,
### and the value of the statistic is probably too tiny to be meaningful.
```

```
vuong(model.p,
      model.zi,
      digits = 4)

Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed N(0,1) under the
null that the models are indistinguishable)
-----
          Vuong z-statistic           H_A p-value
Raw            -1.4424725 model2 > model1 0.074585
AIC-corrected  -0.4335210 model2 > model1 0.332318
BIC-corrected   0.1607786 model1 > model2 0.436134

### Negative Vuong z-statistic suggests that model 2 is superior.
### If the Raw statistic is used, p = 0.07 gives some evidence
### that zi model is superior.
```

```
vuong(model.nb,
      model.zi,
      digits = 4)
```

Vuong Non-Nested Hypothesis Test-Statistic:
(test-statistic is asymptotically distributed $N(0,1)$ under the null that the models are indistinguishable)

	Vuong z-statistic	H_A	p-value
Raw	-1.4424725	model2 > model1	0.074585
AIC-corrected	-0.4335210	model2 > model1	0.332318
BIC-corrected	0.1607786	model1 > model2	0.436134

Negative Vuong z-statistic suggests that model 2 is superior.
If the Raw statistic is used, p = 0.07 gives some evidence
that zi model is superior.

References

Moriña, D., M. Higueras, P. Puig, and M. Oliveira. 2015. Generalized Hermite Distribution Modelling with the R Package *hermite*. The R Journal 7(2):263–274. journal.r-project.org/archive/2015-2/morina-higueras-puig-etal.pdf.

```
help(package="hermite")
library(hermite); ?glm.hermite
library(MASS); ?glm.nb
library(pscl); ?zeroinfl
library(pscl); ?vuong
```

“Simple Logistic Regression” in Mangiafico, S.S. 2015. *An R Companion for the Handbook of Biological Statistics*, version 1.09. rcompanion.org/rcompanion/e_06.html.

“Generalized linear model: Link function”. No date. Wikipedia. Retrieved 31 Jan. 2016. en.wikipedia.org/wiki/Generalized_linear_model#Link_function.

Verhoef, J.M. and P.L. Boveng. 2007. Quasi-Poisson vs. negative binomial regression: How should we model overdispersed count data? Ecology 88(11) 2766–2772. <http://fisher.utstat.toronto.edu/reid/sta2201s/QUASI-POISSON.pdf>.

Wilson, P. 2015. The Misuse of the Vuong Test for Non-Nested Models to Test for Zero-Inflation. *Economic Letters* 127: 51–53. cybermetrics.wlv.ac.uk/paperdata/misusevuong.pdf

References for count data

Grace-Martin, K. No date. "Regression Models for Count Data". *The Analysis Factor*. www.theanalysisfactor.com/regression-models-for-count-data/.

Grace-Martin, K. No date. "Zero-Inflated Poisson Models for Count Outcomes". *The Analysis Factor*. www.theanalysisfactor.com/zero-inflated-poisson-models-for-count-outcomes/.

[IDRE] Institute for Digital Research and Education. 2015. "R Data Analysis Examples: Poisson Regression". UCLA. www.ats.ucla.edu/stat/r/dae/poissonreg.htm.

[IDRE] Institute for Digital Research and Education. 2015. "R Data Analysis Examples: Negative Binomial Regression". UCLA. www.ats.ucla.edu/stat/r/dae/nbreg.htm.

[IDRE] Institute for Digital Research and Education. 2015. "R Data Analysis Examples: Zero-Inflated Poisson Regression". UCLA. www.ats.ucla.edu/stat/r/dae/zipoisson.htm.

[IDRE] Institute for Digital Research and Education. 2015. "R Data Analysis Examples: Zero-Truncated Poisson Regression". UCLA. www.ats.ucla.edu/stat/r/dae/ztp.htm.

Beta Regression for Percent and Proportion Data

Introduction

Proportion data

In general, common parametric tests like t-test and anova shouldn't be used when the dependent variable is proportion data, since proportion data is by its nature bound at 0 and 1, and is often not normally distributed or homoscedastic.

Data of proportions, percentages, and rates can be thought of as falling into a few different categories.

Proportion data of discrete counts

Some proportion data is derived from discrete counts of "successes" and "failures", where the "successes" are divided by the total counts. The example below with passing and failing counts across classes is an example of this. Each observation is a percentage from 0 to 100%, or a proportion from 0 to 1. This kind of data can be analyzed with beta regression or can be analyzed with logistic regression.

Proportion data that is inherently proportional

Other proportion data is inherently proportional, in that it's not possible to count "successes" or "failures", but instead is derived, for example, by dividing one continuous variable by a given denominator value. The sodium intake example below is an example of this. Another case of this kind of proportion data is when a proportion is assessed by subjective measurement. For example, rating a diseased lawn subjectively on the area dead, such as "this plot is 10% dead, and this plot is 20% dead". Each observation is a percentage from 0 to 100%, or a proportion from 0 to 1. This kind of data can be analyzed with beta regression.

Rates with different numerators and denominators

Some rates are expressed with numerators and denominators of different measurements or units. For example, the number of cases of a disease per 100,000 people or the number of televisions per student's home. In these cases, the values are not limited to between 0 and 1, and beta regression is not appropriate.

If the numerator can be considered a count variable, Poisson regression or other methods for count data are usually suggested. As a complication, often the denominator varies in value. For example, if the count is disease occurrence in a city and the denominator is the population in a city. (Each city has a different population.) Or if the numerator is the count of televisions in the home and the denominator is the number of students in the home. In these cases, Poisson regression or related methods are often recommended with an offset for the value in the denominator. These examples are not explored further here, but an example model would be

```
glm(Count_of_television ~ Independent_variable +
  offset(log(Number_of_students_in_home)),
  family="poisson",
  data=Data)
```

Beta regression

Beta regression can be conducted with the *betareg* function in the *betareg* package (Cribari-Neto and Zeileis, 2010). With this function, the dependent variable varies between 0 and 1, but no observation can equal exactly zero or exactly one. The model assumes that the data follow a beta distribution.

p-value and pseudo R-squared for the model

The *summary* function in *betareg* produces a pseudo *R-squared* value for the model, and the recommended test for the *p*-value for the model is the *lrtest* function in the *lmtest* package.

The *nagelkerke* function in the *rcompanion* package also works with beta regression objects. The likelihood ratio test there appears to work fine, but the results for pseudo *R-squared* may be squirrely, and probably should not be relied upon.

Anova-like table

An anova-like table for factor terms can be produced with the *joint_tests* function in the *emmeans* package. It is recommended that the documentation for this function be read before using it.

If this approach doesn't satisfy the needs of the analyst, nested models can be compared with the *lrtest* function.

It is my understanding that the *Anova* function in the *car* package produces incorrect results for *betareg* models.

Final note

Note that model assumptions and pitfalls of this approach are not discussed here. The reader is urged to understand the assumptions of this kind of modeling before proceeding.

Packages used in this chapter

The packages used in this chapter include:

- psych
- betareg
- lmtest
- rcompanion
- multcompView
- emmeans
- ggplot2

The following commands will install these packages if they are not already installed:

```
if(!require(psych)){install.packages("psych")}
if(!require(betareg)){install.packages("betareg")}
if(!require(lmtest)){install.packages("lmtest")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(multcompView)){install.packages("multcompview")}
if(!require(emmeans)){install.packages("emmeans")}
if(!require(ggplot2)){install.packages("ggplot2")}
```

Beta regression example with discrete counts

The following example uses counts of students passing or failing an exam, and asks if there is a relationship between the proportion of passing students and *Grade*. Note that *Grade* is treated as a numeric variable in this analysis.

```
Input = "
Class          Grade  Pass  Fail
'Bully Hill'    12    14     6
'Keuka Lake'    12    15     5
'Heron Hill'    11    18     2
'Castel Grisch' 11    10    10
'Red Newt'      10    17     3
'Finger Lakes'   10     9    11
'Bellview'       9    12     8
'Auburn Road'    9     8    12
'Balic'          8    10    10
'Cape May'       8     8    12
'Hawk Haven'     7    12     8
'Natali'         7     4    16
")
Data = read.table(textConnection(Input),header=TRUE)

### Create the proportion variable

Data$Proportion = Data$Pass / (Data$Pass + Data$Fail)
```

```
### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

      Class Grade Pass Fail Proportion
1     Bully Hill    12   14    6      0.7
2     Keuka Lake    12   15    5      0.75
3     Heron Hill   11   18    2      0.9
4     Castel Grisch 11   10   10      0.5
...      <NA>    ...   ...   ...
9      Balic       8    10   10      0.5
10     Cape May    8     8   12      0.4
11     Hawk Haven   7   12    8      0.6
12     Natali       7     4   16      0.2
```

Beta regression example

```
library(lmtest)

lrtest(model.beta)

Likelihood ratio test

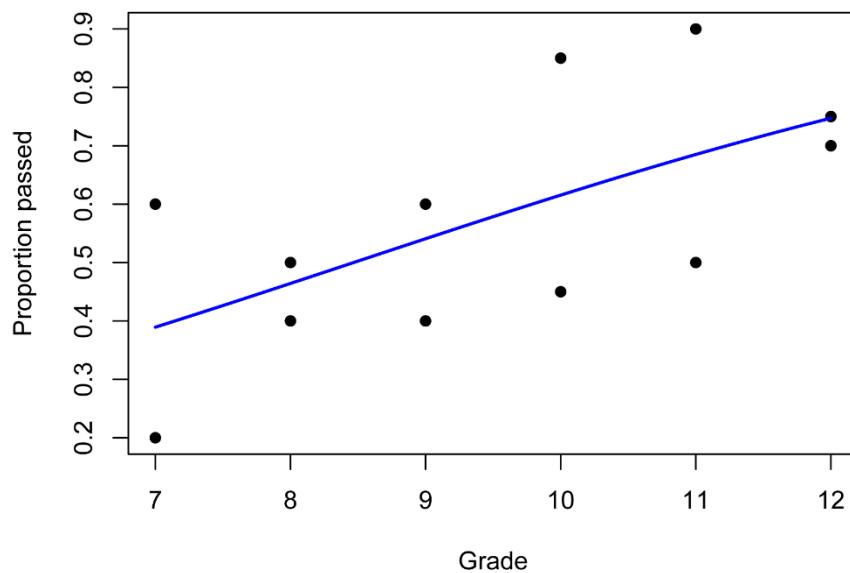
#Df LogLik Df Chisq Pr(>Chisq)
1  3 5.9273
2  2 2.9897 -1 5.8752    0.01536 *
```

summary(model.beta)

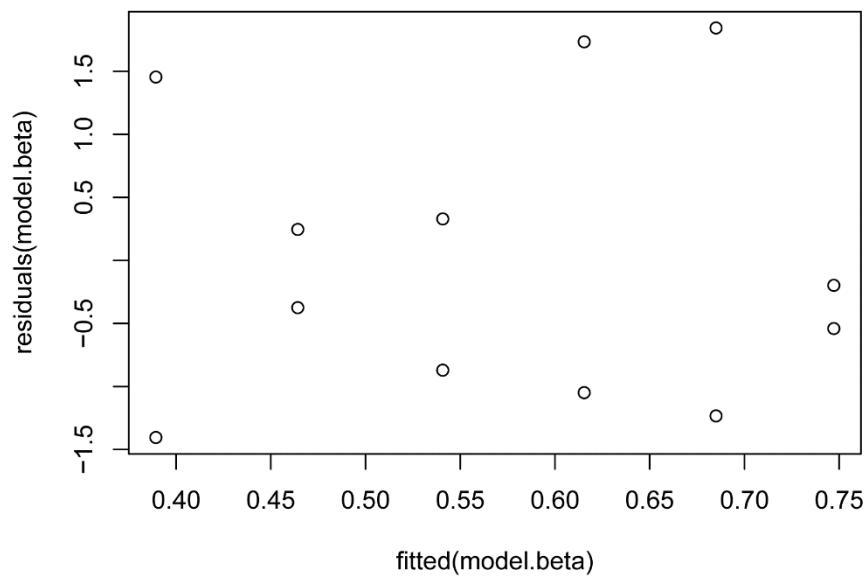
Pseudo R-squared: 0.3785

library(rcompanion)

```
plotPredy(data = Data,
          y = Proportion,
          x = Grade,
          model = model.beta,
          xlab = "Grade",
          ylab = "Proportion passed")
```



```
plot(fitted(model.beta),
      residuals(model.beta))
```



```
### More diagnostic plots: plot(model.beta)
```

Alternate logistic regression

```
Trials = cbind(Data$Pass, Data$Fail)           # Successes, Failures

model.log = glm(Trials ~ Grade,
                data = Data,
                family = binomial(link="logit"))
```

```

library(car)

Anova(model.log,
      type="II",
      test="Wald")

  Analysis of Deviance Table (Type II tests)

    Df  Chisq Pr(>Chisq)
Grade  1 14.305  0.0001554 ***

library(rcompanion)

nagelkerke(model.log)

$Pseudo.R.squared.for.model.vs.null
                           Pseudo.R.squared
McFadden                      0.190924
Cox and Snell (ML)            0.717212
Nagelkerke (Cragg and Uhler)  0.718174

$Likelihood.ratio.test
Df.diff LogLik.diff  Chisq   p.value
-1       -7.5784  15.157 9.8946e-05

library(rcompanion)

plotPredy(data  = Data,
          y     = Percent,
          x     = Grade,
          model = model.log,
          type  = "response",    # Needed for logistic regression
          xlab  = "Grade",
          ylab  = "Proportion passing")

plot(fitted(model.beta),
      residuals(model.beta))

### More diagnostic plots: plot(model.beta)

```

Beta regression example with inherently proportional data

This example revisits the data set from the chapter on two-way analysis of variance. Here, the dependent variable *Proportion* is created by dividing daily student sodium intake by the US FDA “upper safe limit” of 2300 mg. The rest of the analysis is analogous to that of a two-way ANOVA.

Note that for this kind of data, *Proportion* could be greater than 1, if the observed sodium intake were greater than the 2300 mg limit. Because of this, beta regression may not be the best choice of analysis for this kind of data. Usually, beta regression is used for data that are actually bounded at 0 and 1.

```
Input = "
Instructor      Supplement   Sodium
'Brendon Small'    A        1200
'Brendon Small'    A        1400
'Brendon Small'    A        1350
'Brendon Small'    A        950
'Brendon Small'    A        1400
'Brendon Small'    B        1150
'Brendon Small'    B        1300
'Brendon Small'    B        1325
'Brendon Small'    B        1425
'Brendon Small'    B        1500
'Brendon Small'    C        1250
'Brendon Small'    C        1150
'Brendon Small'    C        950
'Brendon Small'    C        1150
'Brendon Small'    C        1600
'Brendon Small'    D        1300
'Brendon Small'    D        1050
'Brendon Small'    D        1300
'Brendon Small'    D        1700
'Brendon Small'    D        1300
'Coach McGuirk'   A        1100
'Coach McGuirk'   A        1200
'Coach McGuirk'   A        1250
'Coach McGuirk'   A        1050
'Coach McGuirk'   A        1200
'Coach McGuirk'   B        1250
'Coach McGuirk'   B        1350
'Coach McGuirk'   B        1350
'Coach McGuirk'   B        1325
'Coach McGuirk'   B        1525
'Coach McGuirk'   C        1225
'Coach McGuirk'   C        1125
'Coach McGuirk'   C        1000
'Coach McGuirk'   C        1125
'Coach McGuirk'   C        1400
'Coach McGuirk'   D        1200
'Coach McGuirk'   D        1150
'Coach McGuirk'   D        1400
'Coach McGuirk'   D        1500
'Coach McGuirk'   D        1200
'Melissa Robins'  A        900
'Melissa Robins'  A        1100
'Melissa Robins'  A        1150
'Melissa Robins'  A        950
'Melissa Robins'  A        1100
'Melissa Robins'  B        1150
'Melissa Robins'  B        1250
'Melissa Robins'  B        1250"
```

```

'Melissa Robins' B      1225
'Melissa Robins' B      1325
'Melissa Robins' C      1125
'Melissa Robins' C      1025
'Melissa Robins' C      950
'Melissa Robins' C      925
'Melissa Robins' C      1200
'Melissa Robins' D      1100
'Melissa Robins' D      950
'Melissa Robins' D      1300
'Melissa Robins' D      1400
'Melissa Robins' D      1100
")

Data = read.table(textConnection(Input),header=TRUE)

### Order factors by the order in data frame
### Otherwise, R will alphabetize them

Data$Instructor = factor(Data$Instructor,
                         levels=unique(Data$Instructor))

Data$Supplement = factor(Data$Supplement,
                         levels=unique(Data$Supplement))

### Create proportion variable

Data$Proportion = Data$Sodium / 2300

### Check the data frame

library(psych)

headTail(Data)

str(Data)

summary(Data)

### Remove unnecessary objects

rm(Input)

      Instructor Supplement Sodium Proportion
1     Brendon Small          A    1200      0.52
2     Brendon Small          A    1400      0.61
3     Brendon Small          A    1350      0.59
4     Brendon Small          A     950      0.41
...           <NA>        <NA>     ...
57   Melissa Robins         D     950      0.41
58   Melissa Robins         D    1300      0.57

```

59	Melissa Robins	D	1400	0.61
60	Melissa Robins	D	1100	0.48

Beta regression

```

library(betareg)

model = betareg(Proportion ~ Instructor + Supplement + Instructor:Supplement,
                 data = Data)

library(emmeans)
joint_tests(model)

model term          df1 df2 F.ratio p.value
Instructor           2 Inf   7.535  0.0005
Supplement           3 Inf   5.169  0.0014
Instructor:Supplement 6 Inf   0.207  0.9748

library(lmtest)
lrtest(model)

Likelihood ratio test

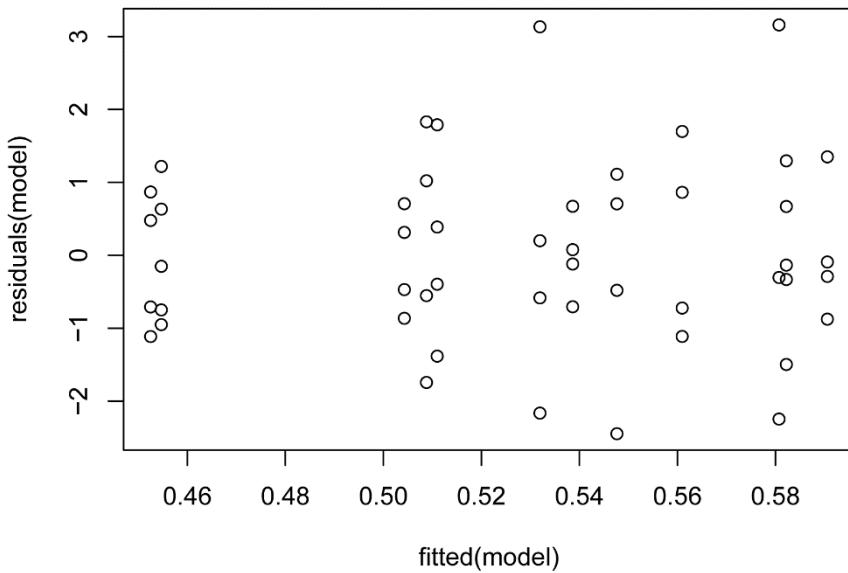
#Df LogLik Df Chisq Pr(>Chisq)
1 13 82.956
2  2 70.260 -11 25.392  0.007984 **

summary(model)

Pseudo R-squared: 0.3431

plot(fitted(model),
      residuals(model))

```



```
### More diagnostic plots: plot(model)
```

```
library(multcompView)
library(emmeans)

marginal = emmeans(model,
                    ~ Instructor)

pairs(marginal,
      adjust="tukey")

Sum = cld(marginal,
          alpha   = 0.05,
          Letters = letters,      ### Use lowercase letters for .group
          adjust   = "tukey")      ### Tukey-adjusted comparisons

Sum

Instructor      emmean      SE df asymp.LCL asymp.UCL .group
Melissa Robins 0.4886613 0.01361908 NA 0.4561425 0.5211801 a
Coach McGuirk  0.5417084 0.01357579 NA 0.5092930 0.5741238 b
Brendon Small   0.5606374 0.01354438 NA 0.5282970 0.5929779 b

Results are averaged over the levels of: Supplement
Confidence level used: 0.95
Conf-level adjustment: sidak method for 3 estimates
P value adjustment: tukey method for comparing a family of 3 estimates
significance level used: alpha = 0.05
```

```

### Order data frame for plotting

Sum = Sum[order(factor(Sum$Instructor,
                         levels=c("Brendon Small",
                                  "Coach McGuirk",
                                  "Melissa Robins"))),]

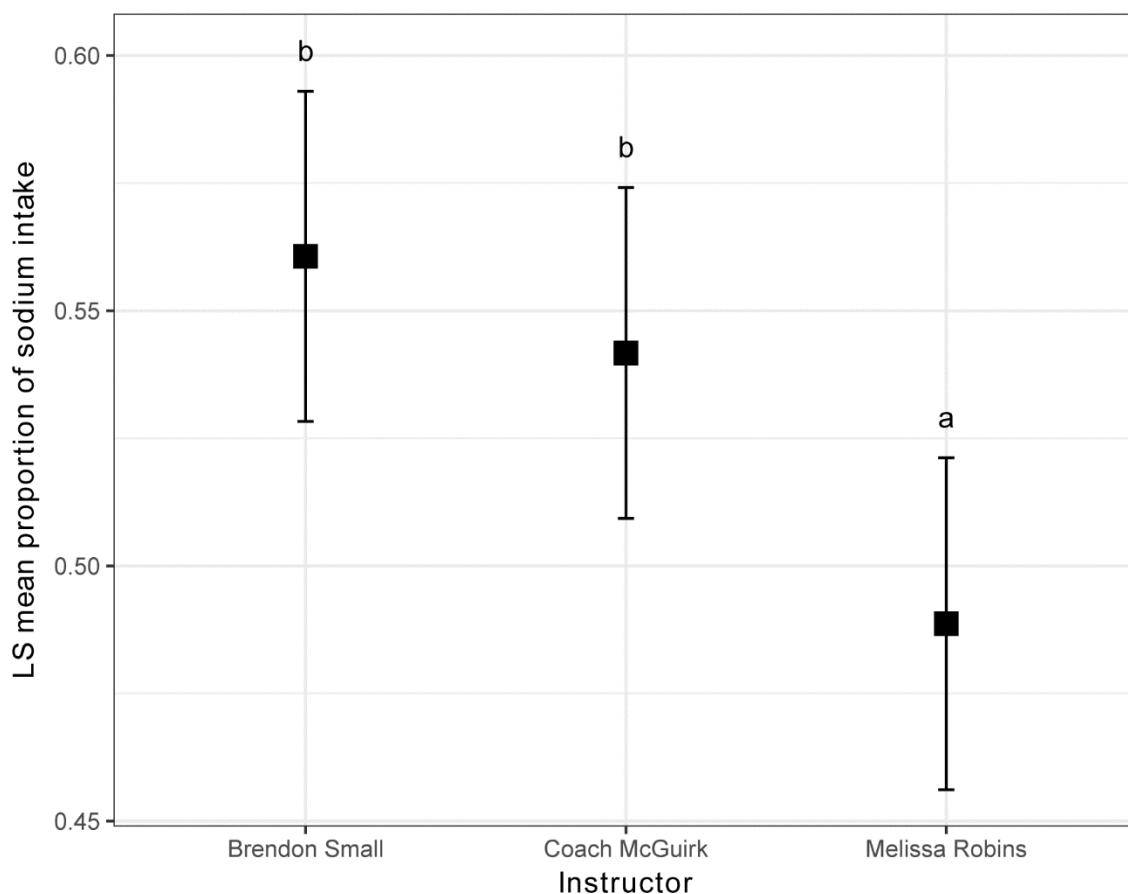
Sum

Instructor      emmean       SE df asymp.LCL asymp.UCL .group
Brendon Small  0.5606374 0.01354438 NA 0.5282970 0.5929779    b
Coach McGuirk 0.5417084 0.01357579 NA 0.5092930 0.5741238    b
Melissa Robins 0.4886613 0.01361908 NA 0.4561425 0.5211801    a

library(ggplot2)

ggplot(Sum,           ### The data frame to use.
       aes(x = Instructor,
            y = emmean)) +
  geom_errorbar(aes(ymin = asymp.LCL,
                     ymax = asymp.UCL),
                width = 0.05,
                size = 0.5) +
  geom_point(shape = 15,
             size = 4) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("EM mean proportion of sodium intake") +
  annotate("text",
           x = 1:3,
           y = Sum$asymp.UCL + 0.008,
           label = gsub(" ", "", Sum$.group))

```



```

marginal = emmeans(model,
                    ~ Supplement)

pairs(marginal,
      adjust="tukey")

Sum = cld(marginal,
           alpha    = 0.05,
           Letters = letters,
           adjust   = "tukey")      ## Use lowercase letters for .group
                                ## Tukey-adjusted comparisons

Sum

### Order data frame for plotting

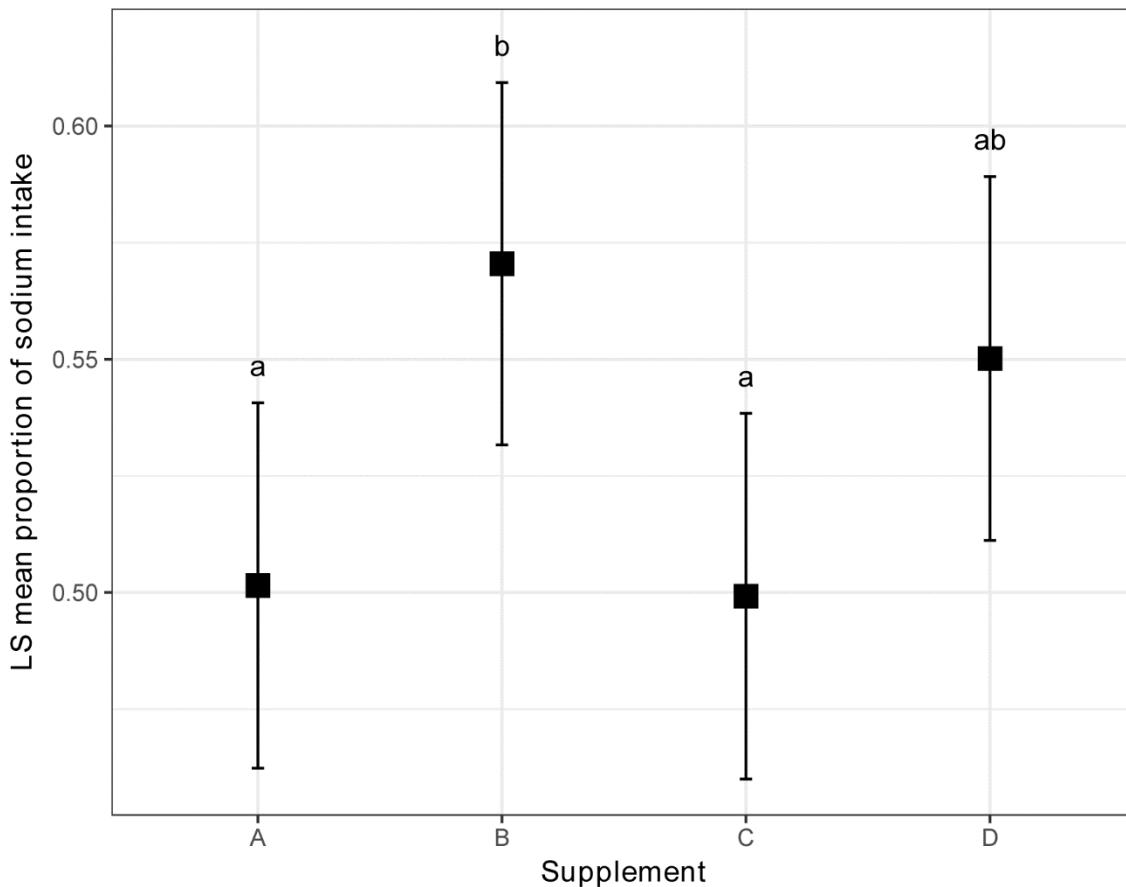
Sum = Sum[order(factor(Sum$Supplement,
                       levels=c("A", "B", "C", "D"))),]

Sum

```

```
library(ggplot2)

ggplot(sum,           ##### The data frame to use.
       aes(x = Supplement,
           y = emmean)) +
  geom_errorbar(aes(ymin = asymp.LCL,
                     ymax = asymp.UCL),
                width = 0.05,
                size = 0.5) +
  geom_point(shape = 15,
             size = 4) +
  theme_bw() +
  theme(axis.title = element_text(face = "bold")) +
  ylab("EM mean proportion of sodium intake") +
  annotate("text",
           x = 1:4,
           y = Sum$asymp.UCL + 0.008,
           label = gsub(" ", "", Sum$.group))
```



References

Cribari-Neto, F. and Zeileis, A. 2010. Beta Regression in R. Journal of Statistical Software 34(2).
www.jstatsoft.org/article/view/v034i02/v34i02.pdf.