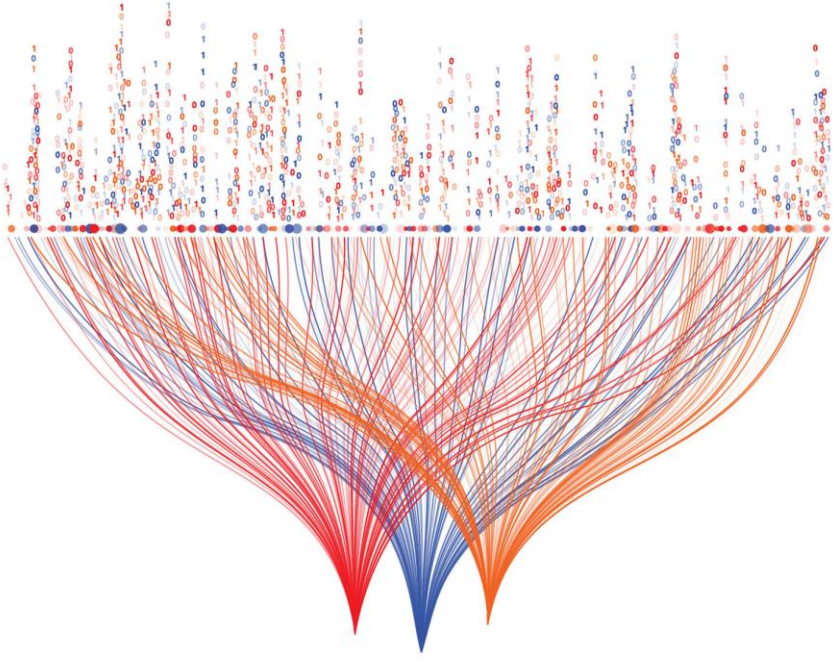
BluΣSky

BLUESKY STATISTICS

7.1 INTRO GUIDE



ROBERT A. MUENCHEN

Robert A. Muenchen

BlueSky Statistics 7.1 Intro Guide

December 15, 2020

Robert A. Muenchen
The University of Tennessee
Greve Hall, Room 517
821 Volunteer Blvd.
Knoxville, TN 37996-3395
USA
muenchen.bob@gmail.com

Typeset using the LaTeXdocument preparation system.

Printed on FSC certified, lead-free, acid-free, buffered paper made from wood-based pulp.

ISBN 978-1-716-44352-7

Preface

This introductory guide for BlueSky Statistics assumes little knowledge of either computing or data analysis. It is a subset of the *BluesSky Statistics 7.1 User Guide* [17], available at Lulu.com. While this guide excludes most of the User Guide's sections on graphics examples and advanced modeling, it does cover many other topics, such as how to:

- Install BlueSky Statistics and determine the settings that will optimize your workflow.
- Read data from a wide variety of sources including delimited text files, Excel files, SAS, SPSS or Stata data sets, and relational databases.
- Manage your data by creating new variables, transforming or recoding existing ones, combining data sets from both the add-cases and add-variables approaches, and pivoting data sets to become wider, or longer, to better enable various graphical and analytic methods.
- Create publication-quality graphs including, bar, pie, scatter, line, box, error bar, and model diagnostic plots.
- Perform all the types of analysis located on BlueSky's Analysis menu, including measures of agreement, clustering, contingency tables, correlation, factor analysis, PCA, market basket analysis, missing value imputation, non-parametrics, reliability, survival, and time series.

Who This Book Is For

This book is written for three main audiences:

1. People who need to analyze data, but who lack the time or inclination to become programmers. BlueSky Statistics is an excellent tool for such analysts.
2. R coders who wish to speed their program development. BlueSky Statistics can automatically generate significant chunks of error-free code to add to their programs.
3. Teams that combine the above two types of analysts, using their programmers to extend the capabilities of BlueSky Statistics for use by their non-programming team members.

This book explains the statistical output and how to interpret it, but it does not replace a thorough book devoted to statistical analysis. While the book covers how to use BlueSky Statistics to speed the generation of R programs, it does not teach the R programming language itself. If you are migrating from SAS or SPSS to the R language, I recommend my book, *R for SAS and SPSS Users* [16]. If you are migrating from Stata, I recommend another of my books, *R for Stata Users* [18].

Acknowledgments

The lead technical reviewer was the consummate Matthew Marler, whose edits greatly improved nearly every section. Thank you, Matthew! I am also grateful to the people who provided advice, caught typos, and suggested improvements, including Ross Dierkhising, Steven Miller, and Frank Thomas.

I am also grateful to the staff of BlueSky Statistics, LLC for the many hours of advice, demonstrations, and feedback.

Many of the examples I present here are modestly enhanced versions from blog posts, the R-help discussion list, and help files. The main benefit I add is the selection, organization, and explanation.

Of course, this book would not have been possible without the efforts of many software developers, including the team at BlueSky Statistics, LLC, the developers of the S language on which R is based, John Chambers, Douglas Bates, Rick Becker, Bill Cleveland, Trevor Hastie, Daryl Pregibon and Allan Wilks[20]; the people who started R itself, Ross Ihaka and Robert Gentleman; the R Development Core Team[21]; Hadley Wickham and tidyverse associates, many other R developers for providing such useful tools for free and all of the R-help participants who have kindly answered so many questions.

Finally, I am grateful to my wife, Carla Foust, and sons Alexander and Conor, who put up with many lost weekends while I wrote this book.

Robert A. Muenchen
muenchen.bob@gmail.com
Knoxville, Tennessee
December 2020

About the Author

Robert A. Muenchen is the author of the *BlueSky Statistics 7.1 User Guide*[17], *R for SAS and SPSS Users* [16] and, with Joseph Hilbe, *R for Stata Users* [18], and with Robert Hoyt, *An Introduction to Biomedical Data Science* [9]. He is also the creator of http://r4stats.com, a popular web site devoted to analyzing trends in data science software, reviewing such software, and helping people learn the R language.

Bob is an ASA Accredited Professional Statistician™ who helps organizations migrate from SAS, SPSS, and Stata to the R Language. He has taught workshops on data science topics for more than 500 organizations and has presented workshops in partnership with the American Statistical Association, RStudio, DataCamp.com, Predictive Analytics World, and Revolution Analytics. Bob has written or co-authored over 70 articles published in scientific journals and conference proceedings and has guided more than 1,000 graduate theses and dissertations at the University of Tennessee.

Bob has served on the advisory boards of BlueSky Statistics, QuestionPro, SAS Institute, SPSS, and the Statistical Graphics Corporation. His contributions have been incorporated into software from those companies as well as JMP, jamovi, and numerous R packages. His research interests include data science software, graphics and visualization, machine learning, and text analytics.

Contents

1

Introduction

1.1 Overview

BlueSky Statistics is an easy-to-use software package for data science. You can use it to enter or open a dataset, transform data in various ways, visualize it, and analyze it. This is all done through the use of menus and dialog boxes. You simply choose a task from the menus, fill in a few details in a dialog box, click OK, and BlueSky will execute that task.

BlueSky Statistics (hereafter "BlueSky") converts your request into the powerful R language behind the scenes. As we will see, you can view and change the R code it writes if you like, but you do not have to. BlueSky can teach you the latest functions from the popular `tidyverse` and `ggplot2` packages if you want to learn the R language. For details about how to run R from within BlueSky, see Chap. 12.

As you can see in Fig. 1.1, there are quite a few Graphical User Interfaces (GUIs) for R. While counting their features is a simplistic way to compare them, that plot does give you at least a rough idea of BlueSky's relative strength. For more information, see my detailed reviews of each of these packages at http://r4stats.com/articles/software-reviews/.

Some of BlueSky's key features:

- A user interface that is very easy to learn and use. BlueSky developers worked on the SPSS development team before starting BlueSky Statistics, LLC, so SPSS users should quickly adapt to the similar BlueSky interface.
- Publication-quality output in true word-processing tables, formatted in the manner that most academic journals prefer (APA style[2]).
- An extensive collection of data "wrangling" items to get your data cleaned up and ready to analyze. These take advantage of several of the popular `tidyverse`[29] packages, such as `dplyr`[26], `broom`[23], and `forcats`[28].
- Advanced missing value imputation methods provided by the `simputation`[25] package.
- The ability to create a wide range of graphics through the use of the popular `ggplot2` package [27].
- The ability to work with multiple data sets at once, changing from one to another with a mouse click.

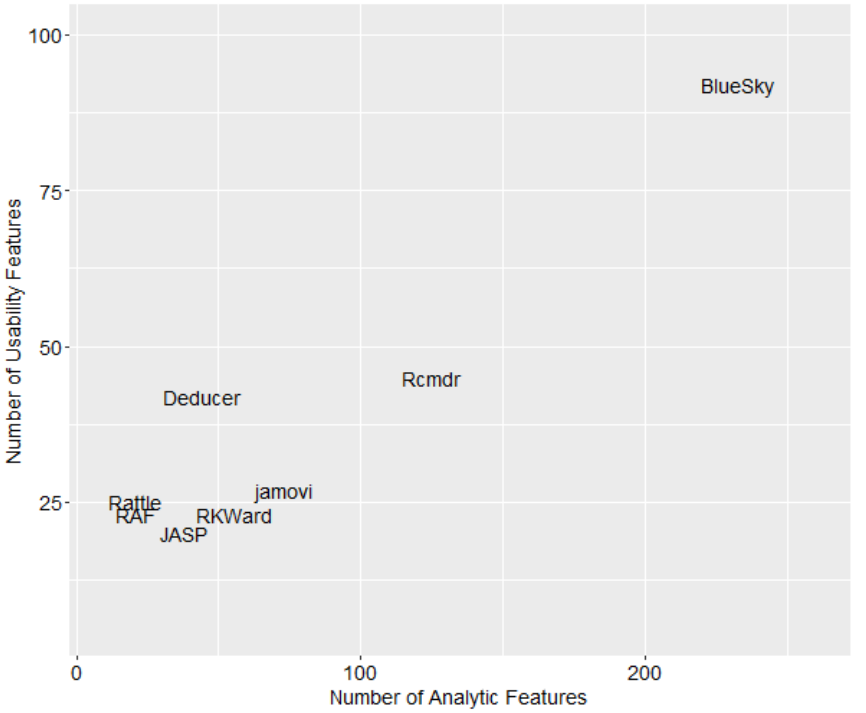


Fig. 1.1: Comparison of R graphical user interfaces. The number of analytic features is displayed on the x-axis, and the number of usability features, including data management, is on the y-axis.

- The ability to control multiple sets of output at once changing the destination with a mouse click.
- The option of manually specifying all modeling details or automating that process through the popular `caret`[12] package.
- The ability to save R models, reload them later, and apply them to score new datasets.

1.2 Choosing a Version

BlueSky is available in a free, Open Source Edition for Microsoft Windows. BlueSky Statistics Commercial Edition also runs on Windows Server. Mac and Linux versions are in development and should be available in early 2021.

The Open Source Edition has all but a handful of menu items pre-installed, plus it can install any five additional items. Of the several hundred menu items available, there are only a few that are not pre-installed in the Open Source Edition:

- Concordance correlation coefficient

- Concordance correlation coefficient, multiple raters
- Cox multiple models
- Linear regression, multiple models
- Logistic regression, multiple models
- Quantile regression

There are several types of Cox, Linear, and Logistic regression menu items pre-installed.

The Commercial Edition has all menu items pre-installed. It can run on any Microsoft Windows desktop or any Windows-based terminal server. Use on a server provides higher performance and controls over which extensions your organization is using. Future plans include local extension repositories to enable groups to share extensions they write more efficiently. The Commercial Edition includes a commercial license and technical support. Pricing is significantly lower than most commercial data science software, especially for universities.

1.3 Installing BlueSky Statistics

To install the BlueSky open source version, double-click on the ".exe" file you downloaded and follow the simple directions. BlueSky includes a copy of the R language in its installation. While you could install a separate copy of R, it is never necessary to do so.

If you purchase the Commercial Edition, the company will send you additional installation instructions.

1.4 Getting Started with BlueSky

You start BlueSky by choosing it from your computer's Start Menu, or by double-clicking on its icon. Two windows will open, the *Analysis Window* and the *Output Viewer*.

The Analysis Window is shown in Fig. 1.2. You use your mouse (or finger on a touch-screen) to choose a menu. That menu will drop-down displaying a set of choices. Choosing an item from the menu will then open a dialog box that you use to control the task.

BlueSky sessions typically begin with the opening of a dataset. You accomplish this using the following steps:

1. Go to the File menu and choose "Open" as shown in Fig. 1.3.
2. Navigate to the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(RData) \Titanic.RData" select it, and click "Open."
3. A message will appear (Fig. 1.4) warning you that you are opening an RData dataset, and that doing so could overwrite one of the same name. The message box also contains a button that provides more information for R programmers and a check-box asking it not to bother you with

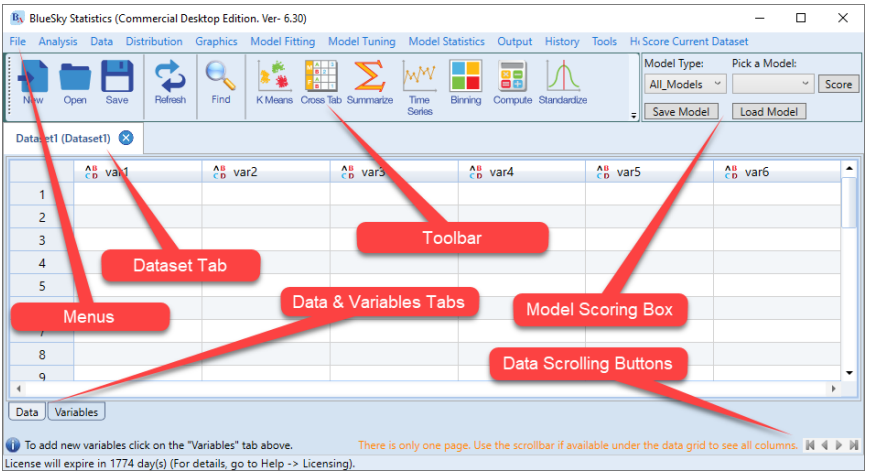


Fig. 1.2: BlueSky Analysis Window with essential parts annotated.

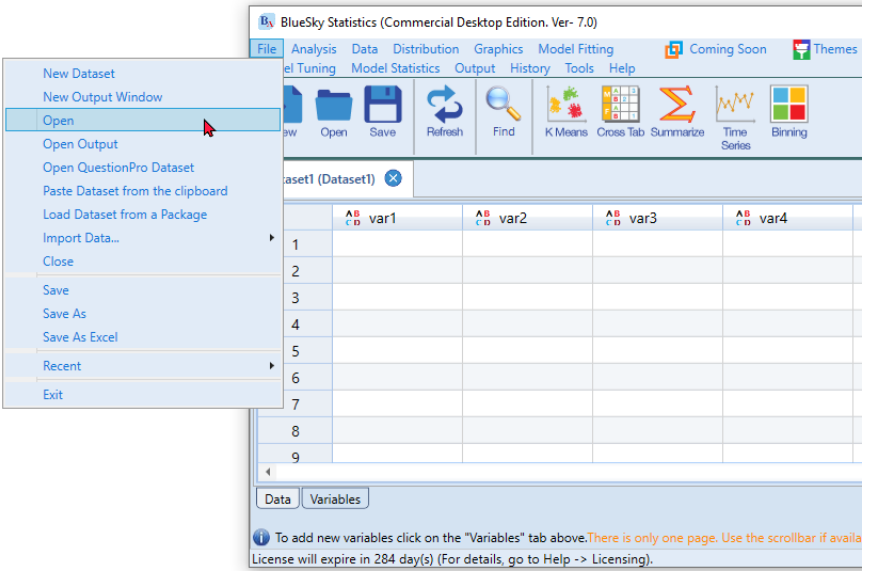


Fig. 1.3: "File> Open" menu. The right side of the image is truncated to make more room for the drop-down menu.

that message in the future. We do not have another dataset of the same name open, so we can click OK.

The data will appear in the Analysis window, as shown in Fig. 1.5. This data contains information regarding the people who sailed on the fateful Titanic voyage.[1] The variable descriptions are shown in Table 1.1.

[1] For details see https://en.wikipedia.org/wiki/RMS_Titanic

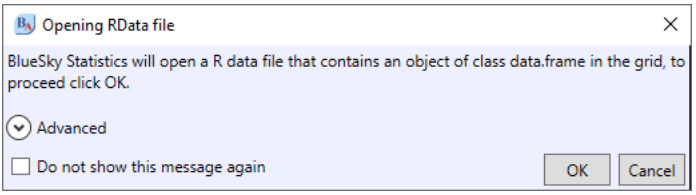
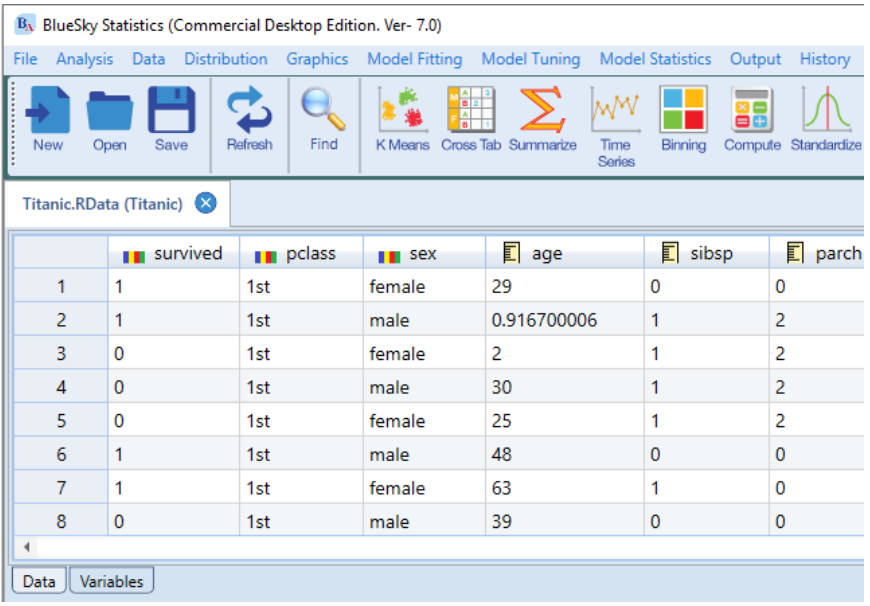


Fig. 1.4: Warning regarding opening RData files. This same message is shown with the "Advanced" button pushed in the next chapter, Fig. 2.11.



	survived	pclass	sex	age	sibsp	parch
1	1	1st	female	29	0	0
2	1	1st	male	0.916700006	1	2
3	0	1st	female	2	1	2
4	0	1st	male	30	1	2
5	0	1st	female	25	1	2
6	1	1st	male	48	0	0
7	1	1st	female	63	1	0
8	0	1st	male	39	0	0

Fig. 1.5: Datagrid showing the Titanic dataset. The right side of this image is truncated to increase the size of the data.

Table 1.1: Titanic variable descriptions.

Variable	Description
survived	A factor indicating survival, yes or no
pclass	A factor indicating passenger class, 1st 2nd 3rd
sex	A factor indicating gender: male female
age	Age in years, min 0.167 max 80.0
sibsp	Number of siblings or spouses aboard, integer: 0...8
parch	Number of parents or children aboard, integer: 0...6

Before we analyze this data, let us examine the variable names closely in Fig. 1.5. Note how the variables survived, pclass (passenger class), and

sex are tagged with a multi-colored icon [2]. That indicates that those are categorical variables called "factors." Age and sibsp are tagged with icons representing numeric variables. This is a crucial distinction for data analysis! Since pclass and sex are both character data, it is rather obvious that their values represent categories and not continuous measurements. However, the survived variable displayed as zeros and ones. Technically that is numeric data even though it represents only the two categories of survived (1) and died (0).

Datasets created by BlueSky (or R) can store variables as factors. But what would have happened if we had read data from some other file format? Reading a comma-separated values file offers a check-box to automatically convert character (string) variables to factors. However, numeric factors need to be manually identified by following the steps in the Data chapter, Sec. 4.6.

Now that we have some data, let us create a graph using these steps:

1. Choose the menu item, "Graphics> Bar Chart, counts."
2. Select the variable "survived" by clicking it in the Source Variables box. Then click the blue arrow to move it to the X variable box. You can also click, hold, and drag the variable from box to box.
3. Follow those steps to move the sex variable to the "Fill" box.
4. Leave the type set to "Stacked bar graph."
5. The dialog box should now look like Fig. 1.6, so click OK.

The graph will appear, as shown in Fig. 1.7. You can see that the majority of survivors are female, while the inverse holds true for the non-survivors. To see if this is a significant relationship, let us create a contingency table to examine the impact of sex on survival:

1. Choose the "Analysis> Contingency Tables> Cross-tab, Multi-way" menu, and you will see the dialog in Fig 1.8. Note that the Row and Column field headings have red asterisks "*" beside them. That indicates that those fields are required. The "OK" button will remain gray and un-clickable until you supply all the required fields.
2. Drag sex into the Row field.
3. Drag survived into the Column field.
4. Clicking the "Options" button will open an additional dialog, as shown in the front of Fig 1.8. Check the boxes for Row percentages and the Chisq statistic, then click OK.

Figure 1.2, shows that the row percentages indicate that 75.26% of females survived while only 20.52% of males did. The Pearson Chi-Square's p-value is much smaller than the traditional 0.05 level, indicating that we can reject the hypothesis that sex and survival are independent. The odds ratio indicates that the odds of survival for males are only 8.49% as high as for females. The fact that the confidence interval for the odds ratio does

[2] The Titanic dataset shipped with BlueSky comes from the `earth` [7] package, where it is named "etitanic" and survived is a numeric variable, rather than a factor.

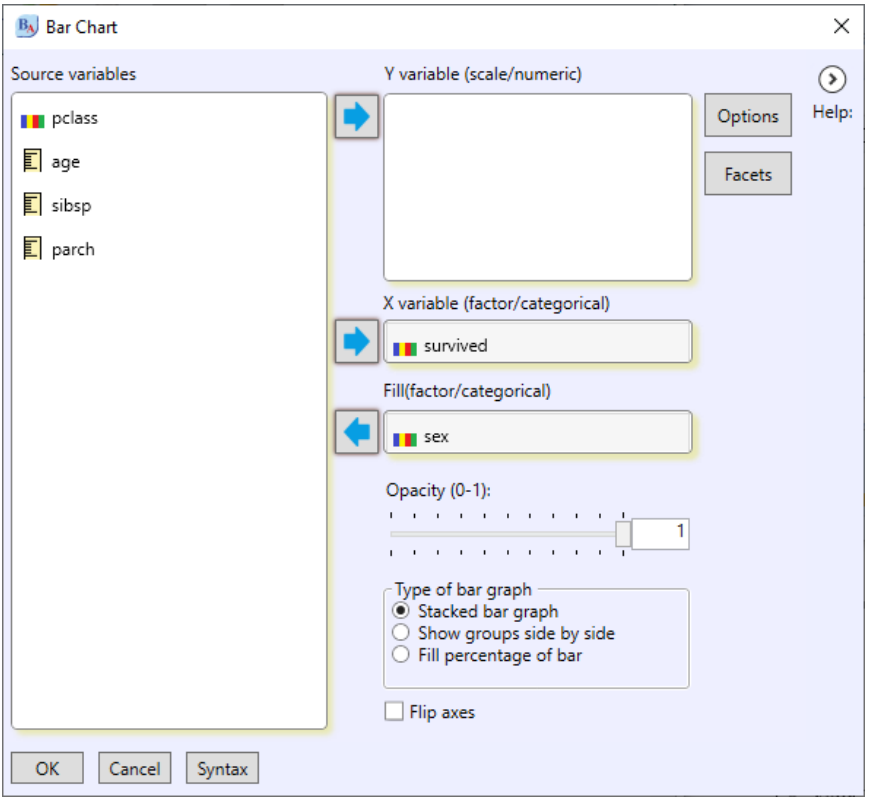


Fig. 1.6: Bar chart dialog.

not include "1" (even odds) indicates that we can reject the hypothesis that the odds of survival are even for males and females.

You will quickly learn how to use many of BlueSky's dialog boxes to do various types of graphs and analyses. Often the dialogs take much space and contain only a few settings, making enumerated lists a much simpler way to describe the steps involved. In those cases, I will not bother to show the dialogs.

1.5 History Menu

Perhaps the most frequent thing you will find yourself doing in BlueSky is recalling and re-running a dialog box. You can open any dialog by following the same steps as you did before, but BlueSky has a shortcut. The History menu, located on each Analysis Window and Output Window, records the actions taken during the current BlueSky session. For example, in Fig. 1.9, you can see that I did the two things described in this chapter: a bar chart of counts, followed by a crosstab. Note that the most recent task is at the top of the menu.

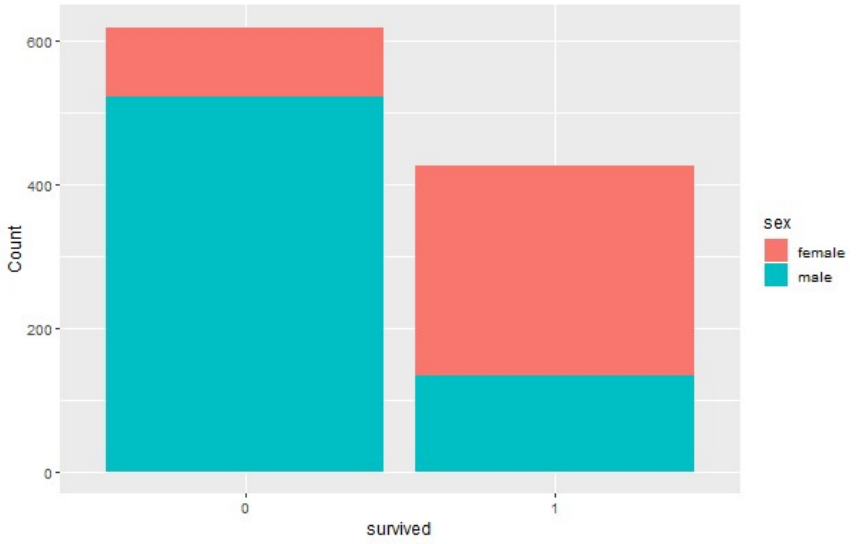


Fig. 1.7: Bar chart showing impact of sex on Titanic survival.

During that session, I could have chosen to re-run either of those steps by selecting that item from the History menu. Its dialog would have reappeared, with all its previous settings in place. I could have then made any changes I like and clicked "OK" to rerun it. That makes iterative work easy. The resulting output would appear at the end of all previous tables and graphs.

Unfortunately, BlueSky does not save the history when you save the Output Window to a file. To get long-term repeatability requires the use of R code, as described in Chap. 12. A future version of BlueSky is in development that will save the history across sessions.

1.6 Getting Help

BlueSky provides simple task-by-task dialog boxes that generate much more complex code. So for a particular task, you might want to get help on 1) the dialog box's settings, 2) the custom functions it uses (if any), and 3) the R functions that the custom functions use.

The level of help that BlueSky provides varies depending on how much support the developers think you need. Each dialog box has a help button in its upper right corner, which opens a Help window off to the dialog box's right. Figure 1.10 shows the dialog box for a bar chart with the help file open. As with many dialog boxes, it provides a summary description of what the task accomplishes, how to use the dialog box, all the GUI settings, and how the accompanying function works should you choose to write your own code.

In the bottom right corner of each dialog box is a "Get R Help" button that takes you to the R help page for the standard R function that performs

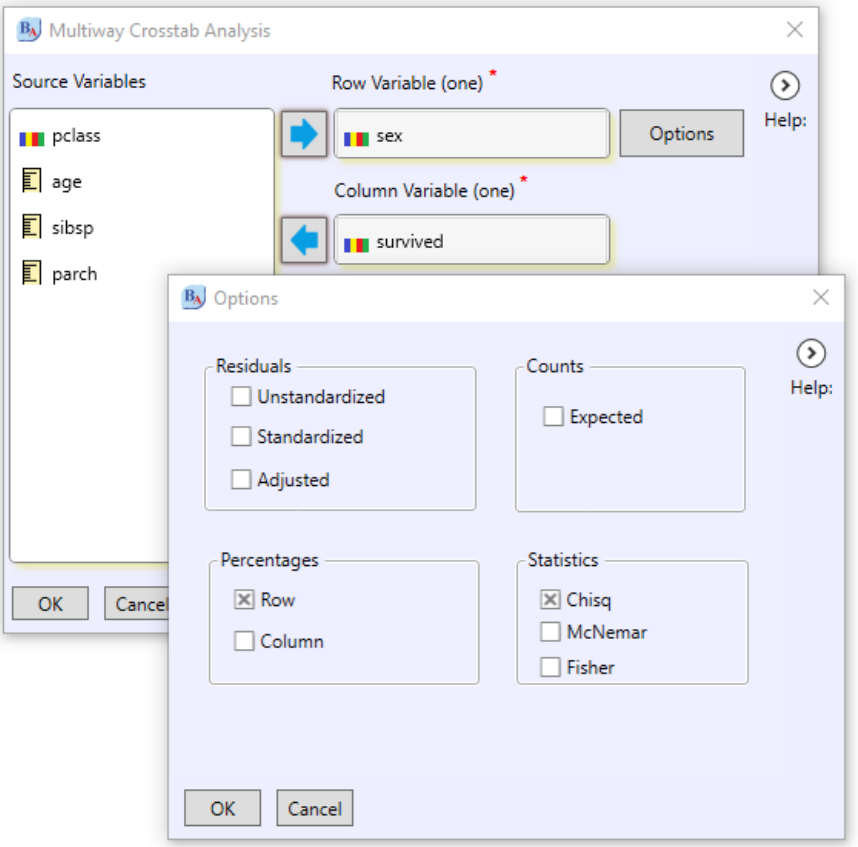


Fig. 1.8: Dialog box from "Analysis> Contingency Tables> Cross-tab, Multi-way." the "Options" button was clicked, bringing that box to the fore.

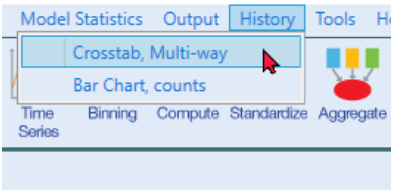


Fig. 1.9: An example history menu showing the two tasks I did during that session.

the calculations (sometimes these are called directly, other times BlueSky calls them from inside its own functions.)

In the bottom right corner of each dialog box is a "Get R Help" button that takes you to the R help page for the standard R function that actually does the calculations (sometimes these are called directly, other times they are used inside BlueSky's functions.) For example, clicking that on the Bar

Table 1.2: Output from contingency table analysis.

Multiway Crosstab Analysis

[etitanic] - etitanic

sex * survived Cross Tabulation

			survived		
			0	1	Total
sex	female	Count	96	292	388
		% within sex	24.7423	75.2577	100
	male	Count	523	135	658
		% within sex	79.4833	20.5167	100
Total		Count	619	427	1046
		% within sex	59.1778	40.8222	100

Statistical Test Results

	Value	df	Asyp. Sig	Odds ratio	95%Confidence interval	
Pearson Chi Square	302.7586	1	0.0000	0.0849	0.063	0.1144

Chart dialog brought up the help file for the R function that does the work, which is the `geom_bar` function from the popular `ggplot2` package (see Fig. 1.11)

For some dialog boxes that simply call an R function (e.g., independent samples t-test), BlueSky will display R's built-in help file. While this variable help approach is well done, I would prefer a more consistent approach. Be aware that there are often things in the R help files that BlueSky does not use. For example, in the case of the t-test, the help file describes how "formula" works, but that concept is not addressable using BlueSky's dialog box (nor is it needed).

1.6.1 Visiting the BlueSky Website

Choosing, "Help> Visit BlueSky Website" is just a quicker alternative to starting your default browser and going to https://BlueSkyStatistics.com. It takes you quickly to the main page of the company website.

1.6.2 About

The "Help> About" menu simply displays the version of the software that you're using and whether it is the open source or commercial version, and

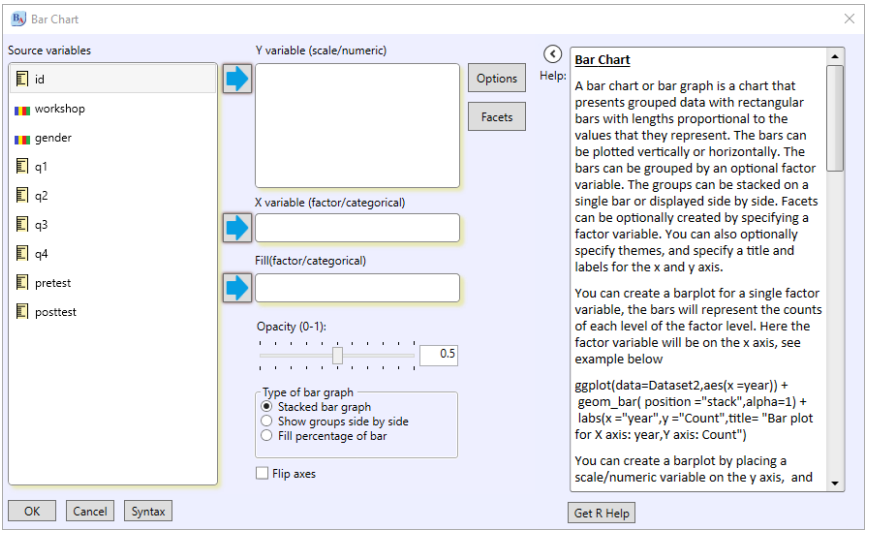


Fig. 1.10: Bar chart dialog box showing the help file on the right-hand side.

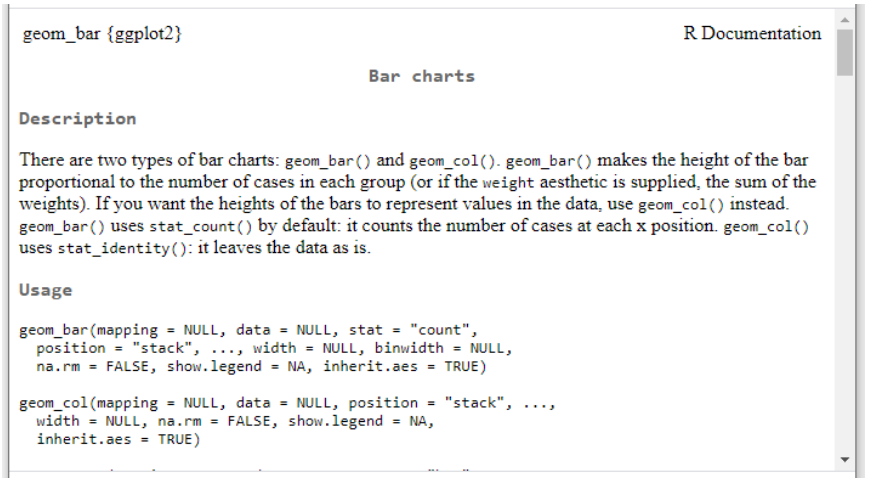


Fig. 1.11: The help file for the geom_bar function which does the work for the bar chart dialog.

whether it is for a server or a desktop computer. You can see in Fig. 1.12 that I'm using the commercial version 7.1.

1.6.3 The Index

I have gone to great lengths to create a detailed, cross-referenced index. You can usually look up a topic more than one way, such as "regression, linear" and "linear regression." If you know a topic and want to find out what R function BlueSky uses for it, you will find it at the end of most sections. If

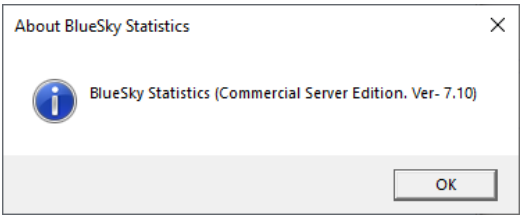


Fig. 1.12: "Help> About" window, showing the version number and commercial status.

you already know an R function, you can see if BlueSky uses it, and if so, where. You can search the index by function name, such as "t.test function" or under its package name, as in "stats package, t.test."

1.6.4 Technical Support

BlueSky is available in two versions, a free open source one, and a commercially licensed and fully supported one. Either way, you can start to get support by going to BlueSkyStatistics.com and clicking on "Support."

Before submitting a question there:

1. Review the links on the right side of the page, i.e. "Getting Started With BlueSky Statistics" and "Top Tech Notes."
2. Check to see if your question is answered on the BlueSky Statistics forum at StackOverflow: https://stackoverflow.com/search?q=BlueSky+Statistics.
3. Commercial customers can enter a support ticket on the company website. Be sure to enter your organization/company name to ensure an expedient response.
4. Open source customers can enter their question on the BlueSky Statistics forum at StackOverflow. If a fellow BlueSky user doesn't answer the question fairly quickly, it is likely that a BlueSky LLC staff member will.

1.7 Your Next Steps

While BlueSky's menus are in alphabetical order, I have organized the initial chapters here in order of importance for learning. The chapters you should read next, in order are:

Chapter 2 covers reading various file types, including creating your own data files using the Datagrid.

Chapter 3 covers many aspects of changing the output format to match your style of work. So far, we have seen tabular output in a pseudo-APA journal style, but it looks much better when exported to Word or PowerPoint. You may prefer your output tables to look like Excel, and BlueSky offers that style as well.

Chapter 4 describes BlueSky's extensive data management features. It is often said data scientists spend 80% of their time cleaning and restructuring their data. That has certainly been the case in my career!

After reading those chapters, you will have a solid foundation to skip around the remaining ones as needs arise.

2

File Menu

As with most menu-based software, you cannot do much with BlueSky until you master its File menu. It controls the various ways to acquire data, including the use of the Datagrid spreadsheet-style editor. It also manages the Analysis and Output windows and their contents, both for opening and for saving in various formats.

2.1 New Dataset

When you first open BlueSky, the Analysis Window opens, showing you an empty Datagrid, as it appears in Fig. 2.1. The other way to obtain a new empty Datagrid is to choose "File> New Dataset." BlueSky is adept at handling multiple datasets. That ability allows you to add a new empty dataset to a new tab in the Application Window. This section contains all you need to know to enter data using the BlueSky Datagrid.

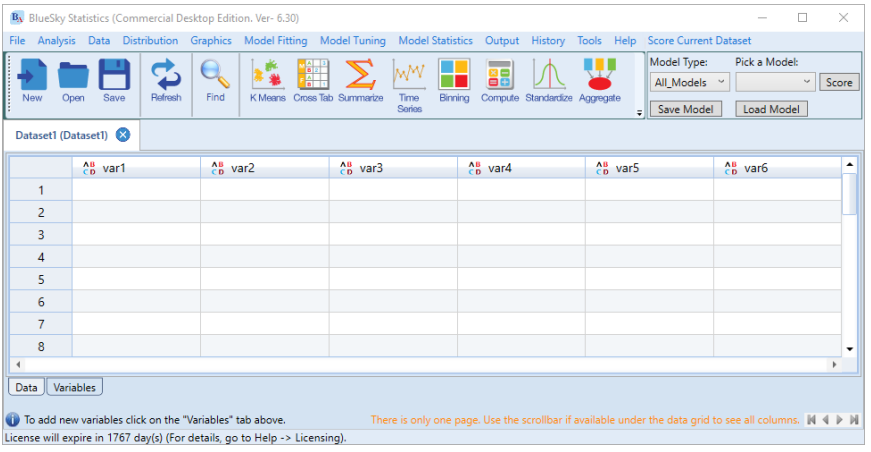


Fig. 2.1: Analysis Window at start-up, showing an empty Datagrid.

While the Datagrid is a handy tool, it does not have the vast array of features that a separate program, like a spreadsheet or database, would have. Those applications have many advanced features such as guessing how to complete a text string you're typing, generating automatic series of numbers or days of the week, or setting data validation limits, which prevent you from typing impossible values. Those are all very helpful features for manual data entry, and if you are entering more than a handful of data values, I strongly recommend using such tools instead of the Datagrid. Where the Datagrid shines is managing data after it is entered or imported from other software.

Let us begin by looking at empty Datagrid as it appears immediately after starting BlueSky (Fig. 2.1). It contains columns of variables, tentatively named var1, var2,...varN across the top. Note that each variable name has a small icon to its left, showing the colorful letters A, B, C, D. Those icons are there to tell you that when you first enter some data, Datagrid will store it as strings, or character variables. They will remain as such until you enter some data *and save it*.

Down the side are numbered rows, ready for you to enter data into. The rows will represent all the measures for a particular case, or observation. With our practice data, a case will represent a person.

Whenever you learn a new program, it is wise to begin by entering a tiny amount of data. Then save that data to a file, exit the program, and restart it to ensure you can open the file that you think you saved. It is easy think you understand these basic steps, only to discover after hours of work that you did not!

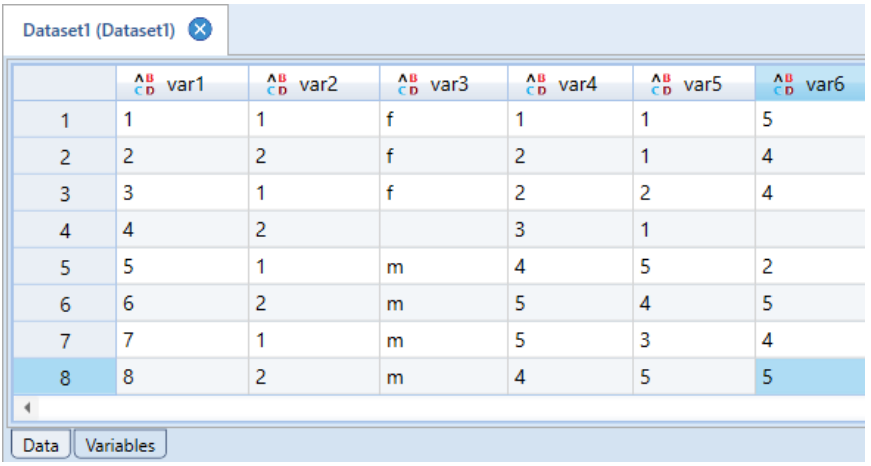
Following this advice, enter the small data set shown in Fig. 2.2. When finished, choose "File> Save as" and save the data to a file called "mydata.RData." The extension ".RData" is what BlueSky and the R language both use as their data file extension.

Next, exit BlueSky, start it back up, and use "File> Open" to open the data. It should appear as in Fig. 2.3. There we see that the character variables have flag-like striped icons next to their names, indicating that the Datagrid now stores them as "factors" or categorical data. That means they will work with the graphics and analysis features in BlueSky that expect factor variables, like a bar chart.

Two tasks that remain in entering our practice data: entering more descriptive variable names and entering more data. At the bottom left of the Datagrid are two tabs labeled "Data" and "Variables." So far, we have been using the Data tab. Clicking on the Variables tab will show you data about your variables, called metadata, shown in Fig. 2.4. To change variable names in the Variables tab, simply click on a name and type a new one. Rename the variables so that they have the names shown in Fig. 2.5. Ignore the fact that there is a new variable named "q4" for now. We will add that in a moment.

You can do many other things on the Variables tab by right-clicking on any row. A menu will pop up, offering several choices:

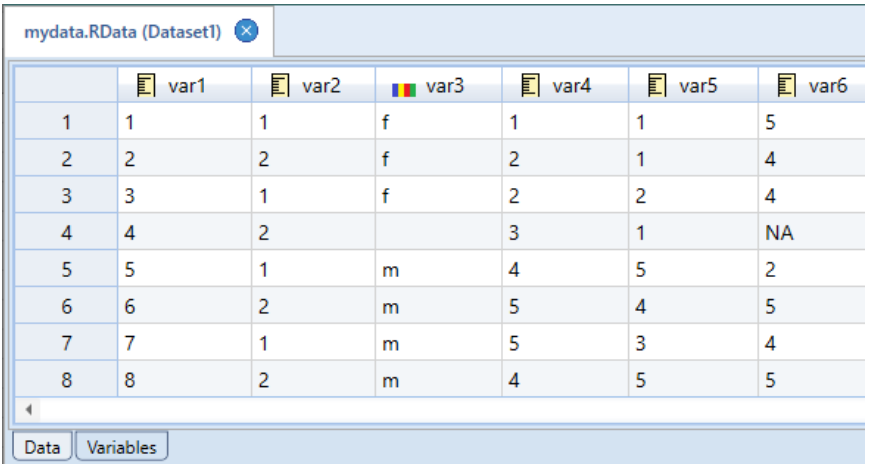
- Add Factor Level.


Dataset1 (Dataset1)

	var1	var2	var3	var4	var5	var6
1	1	1	f	1	1	5
2	2	2	f	2	1	4
3	3	1	f	2	2	4
4	4	2		3	1	
5	5	1	m	4	5	2
6	6	2	m	5	4	5
7	7	1	m	5	3	4
8	8	2	m	4	5	5

Data Variables

Fig. 2.2: The Datagrid showing our practice data entered, but not yet saved. Note that all variables are character data in this screenshot, as indicated by the small A, B, C, D icon at the top of each column, by the tentative variable names.


mydata.RData (Dataset1)

	var1	var2	var3	var4	var5	var6
1	1	1	f	1	1	5
2	2	2	f	2	1	4
3	3	1	f	2	2	4
4	4	2		3	1	NA
5	5	1	m	4	5	2
6	6	2	m	5	4	5
7	7	1	m	5	3	4
8	8	2	m	4	5	5

Data Variables

Fig. 2.3: The Datagrid showing mydata after saving it. Icons by variable names now differentiate numeric from the factor variable for gender.

- Change Label.
- Make Factor.
- Make String.
- Make Numeric.
- Make Date.
- Insert New Numeric Variable.
- Insert New Numeric Variable at End.
- Insert New String Variable.

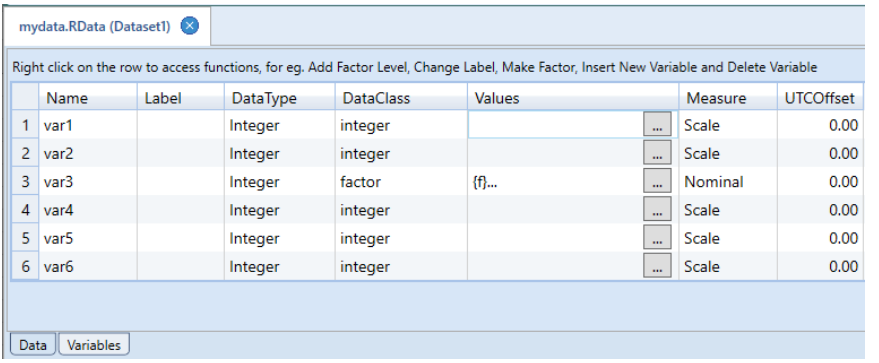


Fig. 2.4: The Datagrid showing information on the Variables tab. Note that "Variables" is chosen in the lower-left corner, and the Name column contains the default variable names.

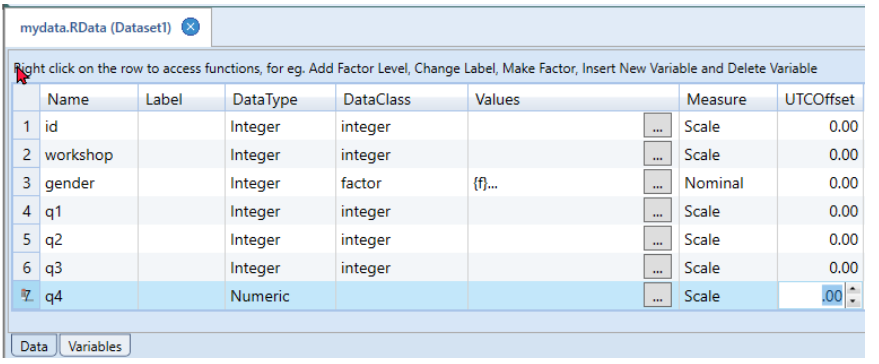


Fig. 2.5: The Datagrid "Variables" tab with new names filled in.

- Insert New String Variable at End.
- Insert New Date Variable.
- Insert New Date Variable at End.
- Insert New Factor Variable.
- Insert New Factor Variable at End.
- Delete Variable.

These are all self-explanatory, but it's a good time to think of the ones you are likely to need with every new dataset. So far, we entered only six variables, but that's because a new empty Datagrid only has room for six! We have a seventh variable, so to add it, right-click on the last variable name (while still on the Variables tab) and choose, "Insert New Numeric Variable at End". Name the new variable "q4", then click on the Data tab and fill in the data values for it.

If you wish to have the gender value labels display as "male" and "female" instead of "m" and "f", go to the Variables tab. On the row for gender, go to the column labeled "Values" and click on the gray box labeled "...". A Values box will open where you can enter the labels in "New Label" boxes.

mydata.RData (mydata)

	id	workshop	gender	q1	q2	q3	q4
1	1	R	female	1	1	5	1
2	2	SAS	female	2	1	4	1
3	3	R	female	2	2	4	3
4	4	SAS	<NA>	3	1	NA	3
5	5	R	male	4	5	2	4
6	6	SAS	male	5	4	5	5
7	7	R	male	5	3	4	4
8	8	SAS	male	4	5	5	5
*	Click here to add a new row						

Fig. 2.6: The Datagrid with all data entered (including q4), the variables renamed, factor labels set for workshop and gender, and the icons showing the proper variable type by each name.

You can also label the Workshop values with "1" meaning the participants took a workshop on R, and "2" meaning they took the SAS workshop.

Note that the Variables tab has a column named "Label," which is for variable labels. Unfortunately, the underlying R language has no standard way of displaying variable labels, so that column is best left empty.

To add new rows of data, click on the Data tab, and scroll to the bottom. You will see the label, "Click here to add a new row." Each time you click that, you'll get a new row into which you can enter data.

Warning: When you edit a specific cell in Datagrid, the new value does not get written into memory until you either press the Enter key or click on any other cell!

When finished, your dataset should appear as it is in Fig. 2.6.

2.2 New Output Window

BlueSky can handle multiple datasets in a *single* Output window. Each dataset has its own tab, and the one that you select is the one that will be analyzed by menu choices. However, sometimes you also want to explore multiple datasets simultaneously, but without output going to separate windows. Choosing "File> New Output Window" lets you open additional output windows.

Since each output window also contains its own R program editor, this is also useful for working on multiple R programs at the same time. For details on R programming, see Chap. 12.

2.3 Open

As demonstrated in the section on BlueSky's Datagrid (Sec. 2.1), entering data directly into BlueSky is rather clunky. Luckily, BlueSky has a superb ability to import data from a wide range of sources, including:

- Comma Separated Values (.csv).
- Plain text files (.txt).
- Excel (old and new xls file types).
- dBase's DBF.
- SPSS (.sav).
- SAS binary files (sas7bdat).
- Standard R workspace files (RData) with individual data frame selection.

To open any of these file formats, choose "File> Open."

2.3.1 Opening a Comma Separated Values File

The most widely-used file type is the Comma Separated Values (CSV) file, so let us open one that comes with BlueSky. Here are the steps to follow:

1. Go to the File menu and choose "Open," as shown in Fig. 1.3.
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample CSV Datasets\Titanic.csv" select it, and click "Open."
3. BlueSky will see the extension of ".csv" show you the dialog box displayed in Fig. 2.7. You can change those settings to match your particular file, but you can just click OK with this file.
4. The file will open into the Datagrid shown in Fig. 2.9. Note that passenger class (pclass) and sex are marked with a colorful icon that tells us that BlueSky knows these are factor (categorical) variables. However, survived is not so marked, so we have to tell BlueSky that it is a factor to work properly in future graphs and analyses.
5. Click on the "Variables" tab at the bottom left of the Datagrid. The Variables tab will appear, showing information about each variable, as shown in Fig. 2.10. Right-clicking on the variable name "survived" will drop down the displayed menu, allowing you to choose "Make Factor."

The dataset should be ready to use now. If you have many variables to convert to factors, it is quicker to use the "Data> Convert Variables to Factors item" described in Sec. 4.6. Since we had to do some additional work after reading the CSV file, it would be helpful to use "File> Save As" to save the dataset as an RData file, BlueSky's native data file format. It will then retain the fact that workshop is a factor and any labels you may have assigned to your variables. Saving an RData file is a standard type of save, similar to any other software; Sec. 2.8 covers its details. The next time you start BlueSky, rather than read the CSV file, you would want to use "File> Open" to open the RData version of the dataset.

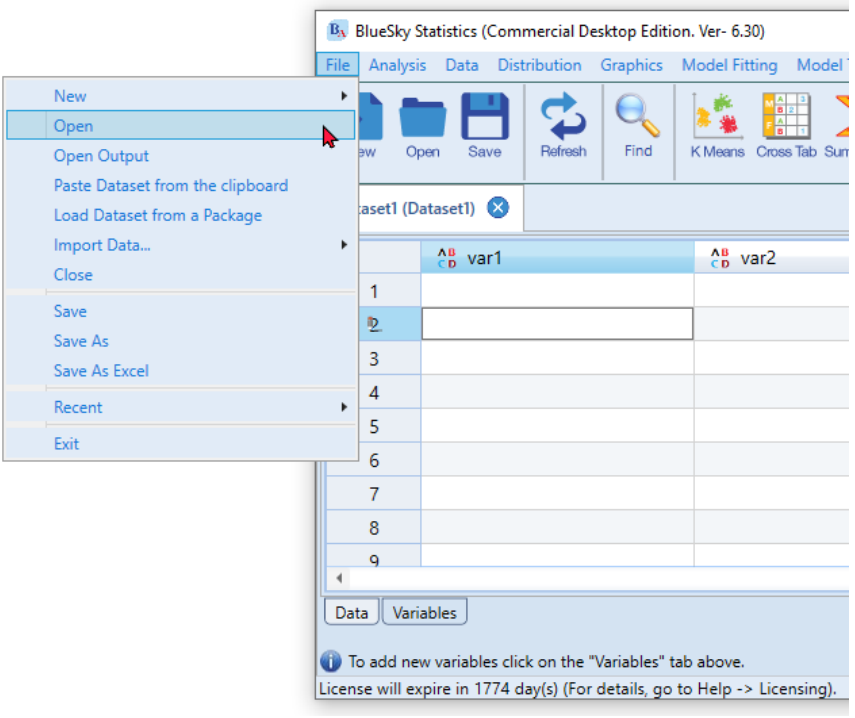


Fig. 2.7: "File> Open" menu.

2.3.2 Opening an R Dataset

R datasets created by BlueSky can only contain a single dataset in a given file. However, the R language can store many datasets, models, etc. in a single file. Therefore, when you open an ".RData" file created outside of BlueSky, there is a chance that it will contain a dataset that will over-write one of your currently open datasets because it shares the same name.

Additionally, if you use the R Syntax Editor to create an object, then you use "File> Open" to open a file that happens to contain a dataset of the same name, BlueSky will overwrite the original dataset.

To warn you of these two possibilities, BlueSky displays the message shown in Fig. 2.11. Once you become aware of this potential problem, you can click on the "Do not show this message again," check-box at the bottom left of the warning dialog. If you turn the warning off and later decide that you want to turn it on again, You can do so using "Tools> Configuration Settings> Output."

2.4 Open Output

As with most file types, you save BlueSky's Output window using "File> Save as" and open it in future sessions using "File> Open Output". An

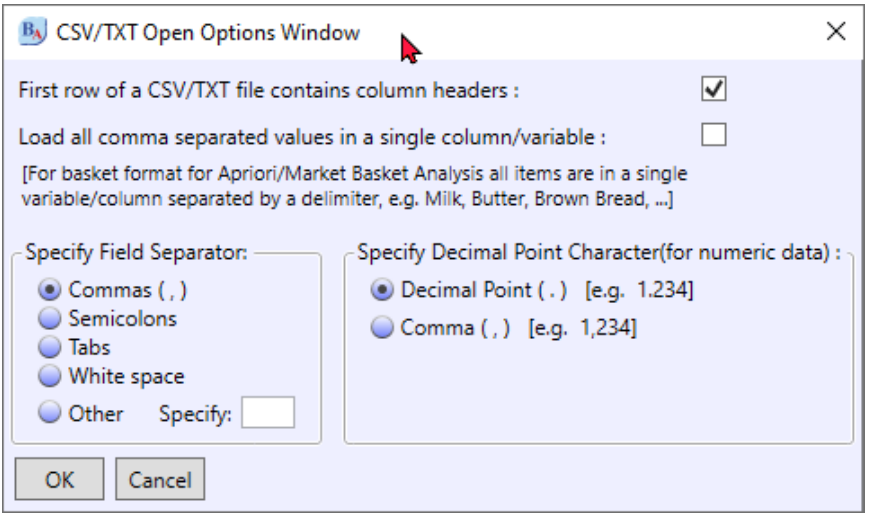
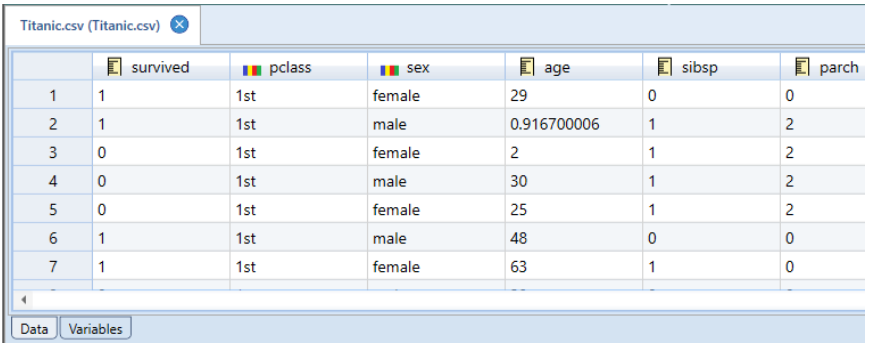


Fig. 2.8: Dialog providing options for opening a comma separated values file.



	survived	pclass	sex	age	sibsp	parch
1	1	1st	female	29	0	0
2	1	1st	male	0.916700006	1	2
3	0	1st	female	2	1	2
4	0	1st	male	30	1	2
5	0	1st	female	25	1	2
6	1	1st	male	48	0	0
7	1	1st	female	63	1	0

Fig. 2.9: Datagrid with CSV file loaded. Note how "survived" lacks the colored icon, which indicates being a factor (as pclass and sex have).

output file contains all your graphs, output tables, messages, and the table of contents. Output files also contain the R code that BlueSky wrote to perform each task. By default, BlueSky will now display that unless you change the default under "Tools> Configuration Settings> Output," by checking, "Display R Syntax in Output window." While this provides a record of precisely what you did, the code is not in a form that is easily re-run. To obtain that, you would have had to have saved your R code from the Syntax Editor, as described in Chap. 12.

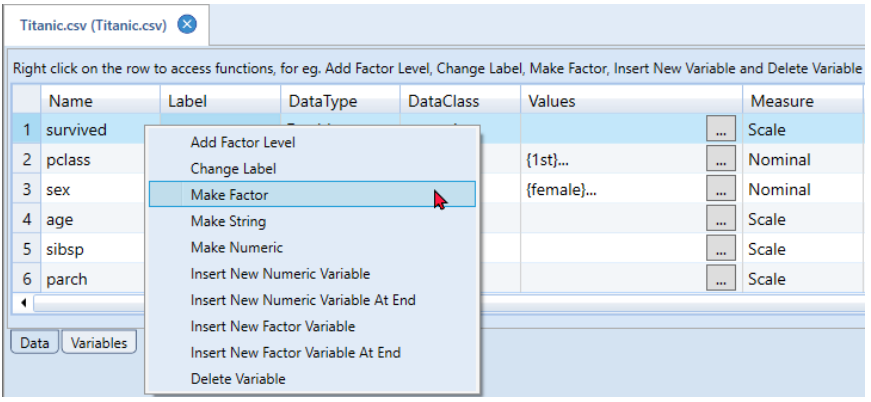


Fig. 2.10: Datagrid Variables tab showing how to make a variable a factor by right-clicking on a variable.

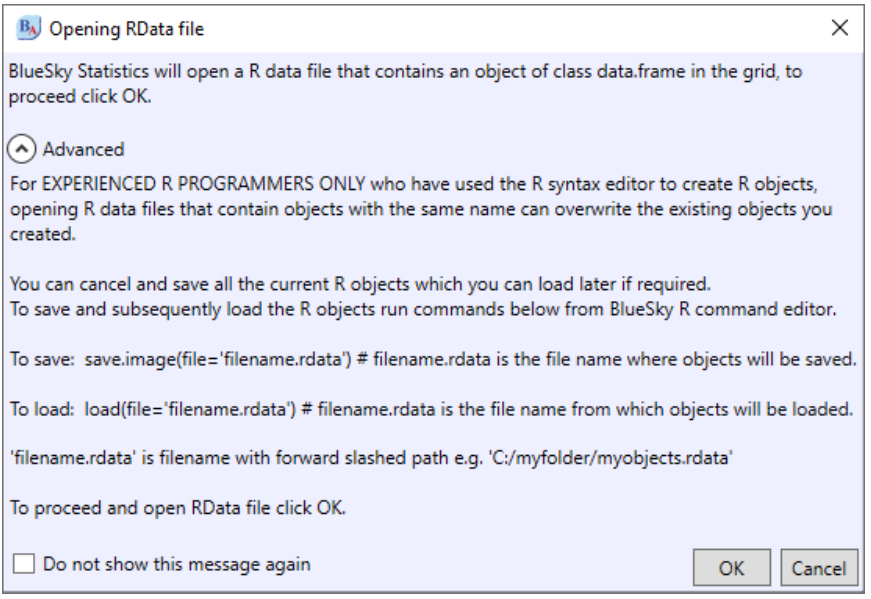


Fig. 2.11: Warning regarding opening RData files. The "Advanced" button has been clicked, displaying more information.

2.5 Paste Dataset from the Clipboard

One of the oddest features of BlueSky's Datagrid is that you cannot paste data into an existing dataset. However, you can paste data into a new dataset by choosing "File> Paste Dataset from Clipboard."

This approach has two significant limitations. First, it will create an entirely new dataset! So where you position the cursor on an existing dataset does not matter as it will not paste into any existing dataset. Second, it expects that the top row will contain the names of your variables. If that

first row contains data, it will still strip off the values and attempt to use them as variable names!

With these limitations, the feature is not particularly useful. You could do all your cutting and pasting using your favorite spreadsheet program, then use this feature when the data meets BlueSky's restrictions. However, at that point, you might as well just save the data to a file and use "File> Open" to read it, which BlueSky does quite well from many formats (see Sec.2.3).

Remember that cutting and pasting data is often a dangerous shortcut to more reliable data management methods. For example, you can join or merge two files side-by-side, adding more variables to an existing dataset. However, that assumes that the rows are perfectly aligned. There are *very* likely errors with that approach, and you are unlikely to find those errors unless you perform a formal merge/join using a matching key such as a case ID number. BlueSky's ability to handle a wide range of such formal equivalents to cutting and pasting is excellent!

2.6 Load Dataset from a Package

BlueSky comes with many example datasets built into it. You can load any one of them by choosing "File> Load Dataset from a Package." In the dialog shown in Fig. 2.12, I chose the "dplyr" package and then clicked the dataset selection drop-down menu to display the list of datasets that package contains. You can see that a dataset of characters from the movie Star Wars is an option. If I had known in advance that this dataset's was "starwars," I could have skipped the step of choosing a package. I could have simply typed the dataset's name into the lower box. However, you would have be very precise about the name, which in this case is all lower-case letters and is a single word.

Occasionally, you may find a conflict that is hard to overcome without knowing the package name. For example, if you try to load the "engel" dataset from the `quantreg` package, if you do not specify the package name, entering only the dataset name will result in an offer to load the "Engel95" dataset from the `np` package. If you run into such a conflict and do not know the name of the package the data resides in, you can search the Internet for "dataset_name R package," and there is a good chance you will find the package's name. Filling that in will then limit the drop-down menu to only those datasets contained within the specified package.

2.7 Import Data

At the time of writing, there is only one menu item on the "File> Import Data" menu, the one used to access SQL databases. Note that BlueSky is not viewing Excel, SAS, SPSS files as databases. Those, and others, it opens directly using "File> Open" as described in Sec. 2.3.

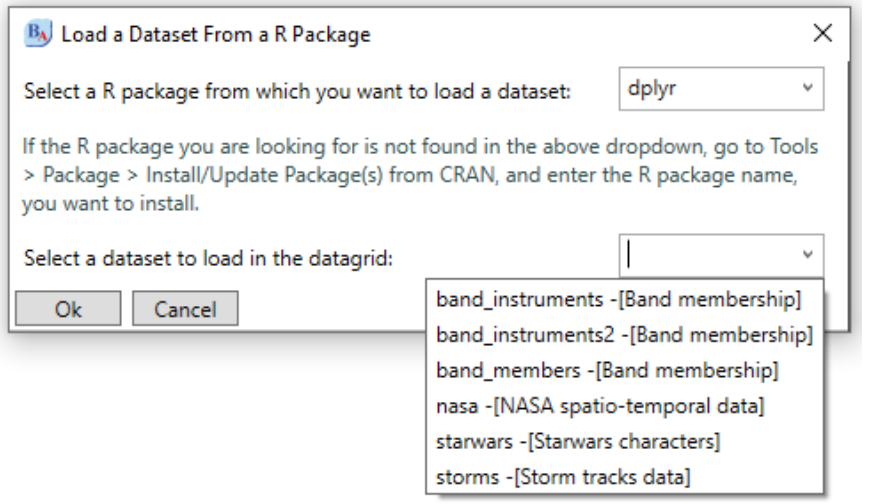


Fig. 2.12: Loading an example dataset from an R package. This shows the "`dplyr`" package selected in the top window, and the choices it offers in the bottom one.

2.7.1 SQL Database

BlueSky can read data from a variety of popular relational databases, including:

- Microsoft Access
- Microsoft SQL Server
- MySQL
- PostgreSQL
- SQLite

To do so choose, "File> Import Data," and BlueSky will open the dialog in Fig. 2.13. Simply fill in the parameters and click OK.

2.8 Save & Save As

BlueSky places the "File> Save" and "File> Save as" items on the Analysis, Output and the Syntax Editor windows. These work the same as with most GUI-based applications, letting you save an existing file with its current name or under a new name or location, respectively.

2.9 Save as PDF

Located on the Output window only, the "File> Save as PDF" item lets you convert the entire contents of that window to a PDF file. To save just

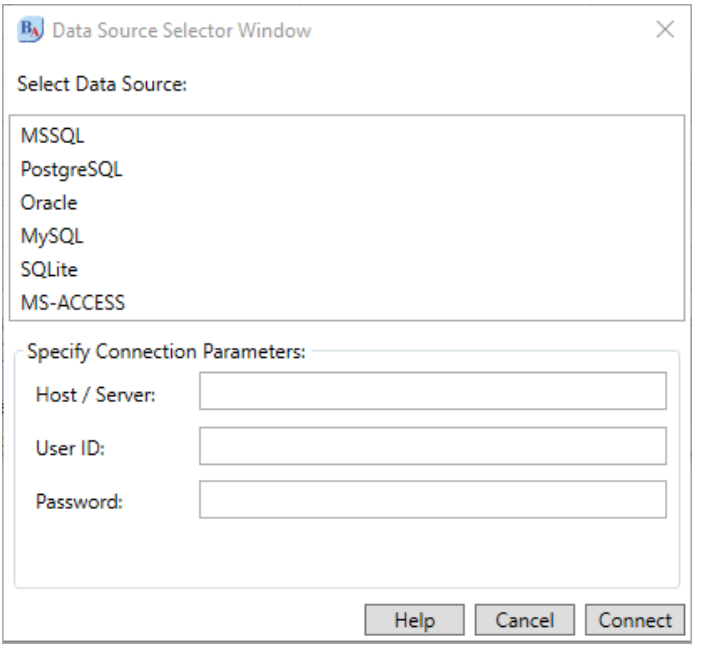


Fig. 2.13: Dialog to select SQL data source.

a subset of the output window, use the trash can icon to delete what you do not want before saving. Details of the PDF file, such as font size, are determined by the configuration settings described in Sec. 11.4.9.

2.10 Recent

Located on the Analysis window only, "File> Recent" opens a list of the most recent files you have used. Choosing one from the list is a quicker way to open a file since you do not have to browse to the exact path; it remembers that for you.

While the list of recent files is updated automatically as you open files, BlueSky stores that list in the folder "C:\Users\your_name\AppData\Roaming \BlueSky\Config" in the file, "BlueSky.exe.config." You can change the list of recent files by editing that file using a text editor if you wish. If you use a word processor to edit that list, be sure to save it back as text only, with none of the formatting commands that word processors usually store in their files.

3

Output

This chapter covers the various ways to manage output including:

- How the Output window works
- Exporting tables to Word and Excel
- Managing multiple Output windows

3.1 Using the Output Window

When you first install BlueSky, it formats its output tables using the style required by many academic journals, "APA style." You can see an approximation of that in Tab. 3.1. When you right-click on any output table, a menu pops up offering to:

- Export to Word
- Export to Excel
- Export to PDF
- Copy to Clipboard
- Delete

Choosing "Export to Word" will start up your copy of Microsoft Word and paste the table into it as a proper word processing table, as shown in Tab. 3.2. In the previous figure, I used a screen capture tool called Snagit to chop off part of the table that I did not want to show. However, once exported to Word, I selected the columns and deleted them using Word's superior table editing capabilities. If you look carefully at the horizontal lines, you will notice that there is now an additional line underneath the names of the various statistics. That is a part of the APA style that BlueSky added that while exporting the table to Word.

A third table format is the spreadsheet style shown in Table. 3.3. That is how tables look if you turn off APA style. See Sec. 3.2 for details.

Finally, I created Tab. 3.4 using the LaTeXtool used to format this entire manual. Now we see the full 300 dots-per-inch (dpi) resolution, and the font style and size are a perfect match for the body of the manual. As nice as it looks, this used the "Copy to Clipboard" feature, followed by slow manual

Table 3.1: The default look of BlueSky tabular output. This is a pasted version of a screen capture, as are most of this manual's examples.

	vars	n	mean	sd	median	trimmed	mad	min	max
age	1	1046	29.8811	14.4135	28	29.3908	11.8608	0.1667	80
sibsp	2	1046	0.5029	0.9122	0	0.3067	0	0	8
parch	3	1046	0.4207	0.8398	0	0.2243	0	0	6

Table 3.2: The look of a table exported to Word. While in Word this is a true word processing table, this manual is formatted in LaTeX, limiting it to a lower resolution screen capture.

	vars	n	mean	sd	median	trimmed	mad	min	max
age	1	1046	29.8811	14.4135	28	29.3908	11.8608	0.1667	80
sibsp	2	1046	0.5029	0.9122	0	0.3067	0	0	8
parch	3	1046	0.4207	0.8398	0	0.2243	0	0	6

Table 3.3: The look of a BlueSky table when APA-style formatting is turned off.

	vars	n	mean	sd	median	trimmed	mad	min	max
age	1	1046	29.8811	14.4135	28	29.3908	11.8608	0.1667	80
sibsp	2	1046	0.5029	0.9122	0	0.3067	0	0	8
parch	3	1046	0.4207	0.8398	0	0.2243	0	0	6

Table 3.4: BlueSky table formatted by LaTeX. The values for this table were copied to the clipboard and manually formatted to achieve full resolution.

	vars	n	mean	sd	median	trimmed	mad	min	max
age	1	1046	29.8811	14.4135	28	29.3908	11.8608	0.1667	80
sibsp	2	1046	0.5029	0.9122	0	0.3067	0	0	8
parch	3	1046	0.4207	0.8398	0	0.2243	0	0	6

formatting. The BlueSky developers plan to add automatic generation of LaTeXtables in a future version.

3.2 Controlling Output Options

You can control several aspects of BlueSky's output by choosing "Tools> Configuration Settings Menu," and clicking on the "Output Settings" tab. Figure 3.1 shows that settings tab. Note that I have un-selected the setting, "Tables in APA (American Psychological Association) Style." Once turned *off*, all output tables will appear like tiny spreadsheets, as Fig. 3.3 shows.

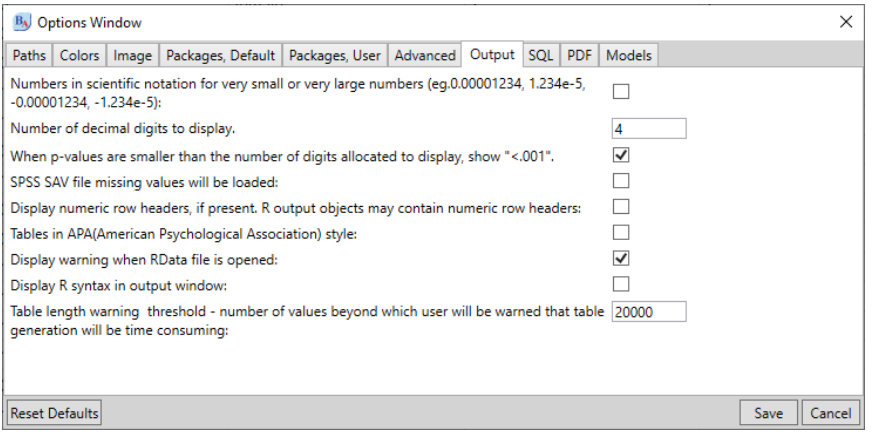


Fig. 3.1: The Output Settings tab from the "Tools> Configuration Settings Menu"

People who often right-click on their output tables and choose "Export to Excel," may prefer this look.

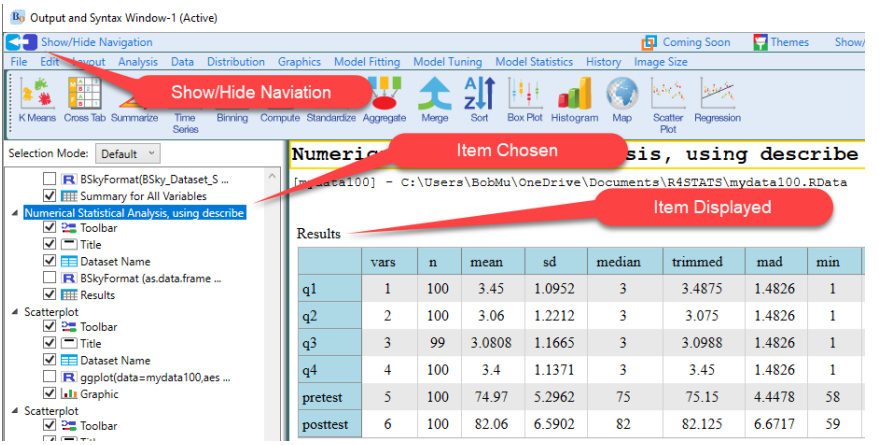
The top option in Fig. 3.1 controls whether or not very large or very small values are displayed using scientific notation. BlueSky does not check this by default, so a very tiny number like 0.00001234 might print as just 0.000000 or even a single 0. Checking this box would make it appear as 1.234e-5, which means 1.234×10^{-5}, or moving the decimal 5 places to the left. This is particularly useful when you have done multiple statistical tests, and you want to correct their p-values by multiplying them by the number of tests done (called the Bonferroni correction).

Other options include the ability to specify how many decimal digits to display and whether you want tiny p-values shown as, say, 0.000000 or as "<0.001". Of course, if scientific notation is used, it will override this setting. BlueSky has not been completely successful at gaining control over the display of R's p-values. Even if you choose to display small numbers as "<0.001" you will still occasionally see "0" appear instead. The choice of scientific notation is more universal in the output created by R, and once you get used to it, it is more informative.

If you want to see the R programming code that BlueSky writes for each task you assign it, then check the "Display R syntax in output window" box. For details on using R programming code, see Chap. 12.

3.3 Using the Output Navigator

Every Output window has a table of contents called the Output Navigator. To see it, click on the "Show/Hide Output Navigator" icon on the upper left of any Output window. It will pop out of the left-hand side of the Output window, as shown in Fig.3.2. You can scroll through the list of all tasks you have completed, and clicking on any one of them will cause the Output



	vars	n	mean	sd	median	trimmed	mad	min
q1	1	100	3.45	1.0952	3	3.4875	1.4826	1
q2	2	100	3.06	1.2212	3	3.075	1.4826	1
q3	3	99	3.0808	1.1665	3	3.0988	1.4826	1
q4	4	100	3.4	1.1371	3	3.45	1.4826	1
pretest	5	100	74.97	5.2962	75	75.15	4.4478	58
posttest	6	100	82.06	6.5902	82	82.125	6.6717	59

Fig. 3.2: The Output window's Output Navigator panel.

window to jump directly to that bit of output. In the figure, I clicked on "Numerical Statistical Analysis, using describe," and the table that step created appeared. You can tell by the spreadsheet style of the output that I had the APA style turned off at the time.

3.4 Managing Multiple Output Windows

Each time you start BlueSky, the Analysis window and a single Output window will open. You can only have one Analysis window open in a given session, but it can contain multiple datasets. Clicking on a dataset tab makes that the active one and all analyses that follow will use that dataset. But how do you choose where the output will go?

Each time BlueSky starts, the Output window appears and is named, "Output and Syntax Window-1 (Active)." Unless you open another Output window, that is where all your output will go. When you first work with BlueSky, one is probably all you will need. However, as you gain experience, you may find it helpful to open additional Output windows using "File> New> New Output Window." When you do, the second Output window will be named: "Output and Syntax Window-2 (Active)" and all future output will go to that window. The first Output window will no longer have "Active" as part of its name. If you decide to route output back to the other window, go to the Analysis window, drop down the Output menu and choose the other window. The Output menu marks which window is active by putting an "A" next to its name, A for Active.

Your control over the Analysis and Output windows are entirely independent. You can change datasets to analyze in the Analysis window, and that will have no impact on which Output window is active. Likewise, you can change which Output window is active without changing the dataset you are analyzing. However, to keep your sanity, I recommend mentally tying one dataset to one output window!

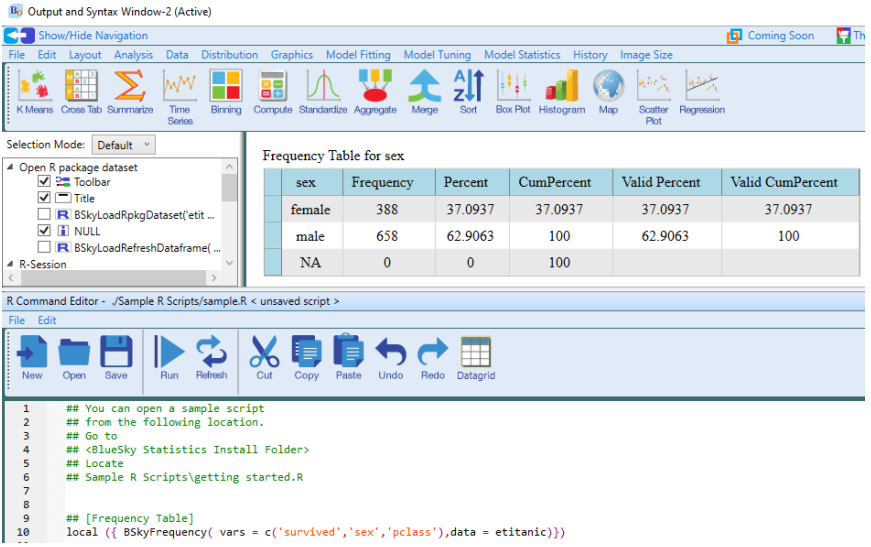


Fig. 3.3: Output window in horizontal layout.

3.5 The Layout Menu

Each Output window has a "Layout" menu that controls how the various panes are displayed.

3.5.1 Horizontal Layout

Choosing "Layout> Horizontal" puts the Output window on top of the Syntax panel, as shown in Fig. 3.3. Note that while the Datagrid and Syntax panel are now wide in the horizontal direction, the Navigation tree does not change position relative to the Datagrid. The Show/Hide buttons still work in the usual way, showing or hiding the Navigation tree or the Syntax panel.

3.5.2 Vertical Layout

Choosing "Layout> Vertical" returns the windows to their taller, side-by-side positions. That is BlueSky's default layout.

3.5.3 Show / Hide Output Navigator Menu Item

Choosing, "Layout> Show Output Navigator" and "Layout> Hide Output Navigator" has the same impact as clicking on the "Show/Hide Output Navigator" icon on the top left-hand side of the Output window. It pops the Navigation tree back out of sight. To get it back out again, you have to click the button since there is only one menu item, which hides it.

3.6 The Edit Menu

BlueSky locates the “File> Edit” item on the Output and Syntax editor windows. As with most edit menus, it allows you to copy, cut, and paste text, as well as undo and redo your edits. Unique to the Syntax editor window is the ability to find and replace text strings. That window also has a toolbar which speeds access to those functions.

4

Data Menu

The data menu offers an extensive collection of items to manage your data and get it into whatever form your analysis might require.

The top half of the menu lists items that affect variables, while the bottom half contains items that affect entire datasets.

4.1 Add ID

A common data management task is adding an identifying counter variable that just goes 1, 2, 3.... You can be use this as a subject variable in a repeated-measures design. You can also use it when restructuring data from "long" to "wide" format.

The dialog for "Data> Add ID" is shown in Fig. 4.1. Let us see how the options affect the resulting variable. You can enter this tiny dataset or just study the results shown in the following two tables.

Table 4.1 has a treatment variable, followed by two ID variables. The "ID No Sort, No Group" column shows what happens when you choose "Data>

Table 4.1: ID variables created with and without grouping by Treatment. Note that it did not sort the data.

Treatment	ID No Sort, No Group	ID Grouped by Treatment
A	1	1
A	2	2
B	3	1
B	4	2
A	5	1
A	6	2
B	7	1
B	8	2

Table 4.2: ID variable created when sorting and grouping by Treatment.

Treatment	ID Grouped & Sorted by Treatment
A	1
A	2
A	3
A	4
B	1
B	2
B	3
B	4

Add ID" and give the dialog nothing on which to sort or group. It simply counts from top to bottom without interruption.

The "ID Grouped by Treatment" column shows how entering a "group by" variable, but no sorting variable, affects the creating of the ID variable. It starts counting over every time the value of the group variable changes. It does not care that group A reappears after two B values since the data are not sorted; it simply begins counting anew.

Table 4.2 shows what happens if you add a variable (Treatment in this case) to the "Variables to sort by first" selection box. That is what Fig. 4.1 shows.

4.2 Bin Numeric Variables

The menu item, "Data> Bin Numeric Variables," is used to divide continuous numeric variables into a relatively small number of integers, with the lowest being just two. An example is dividing the mean score of the tests in a class into grades, such as 90-100 get an "A," 80-89 get a "B," and so on. Section 4.12 covers that type of binning at precise cut-points. The method in this section lets the software choose the cut-points for the bins.

Researchers bin variables for several reasons. They may classify people into low-, medium-, and high-risk categories to help with diagnosis or prognosis, treatment recommendations, and so on. They may also convert numeric to binary to avoid statistical assumptions such as linearity or to ease calculations of effect measures such as the odds ratio or relative risk. The approach used here would classify membership in just two bins using the values 1 and 2. To convert that into the binary 0/1 values that are useful in linear models would require subtracting 1 using a compute covered in Sec. 4.4.1, or a recode covered in Sec. 4.12.

Binning often comes at a high price. When calculating the sample size needed to find a significant effect of, a drug to treat hypertension, you might need five times as many subjects to find a significant dichotomous effect compared to a continuous one. That is due to the great loss of information

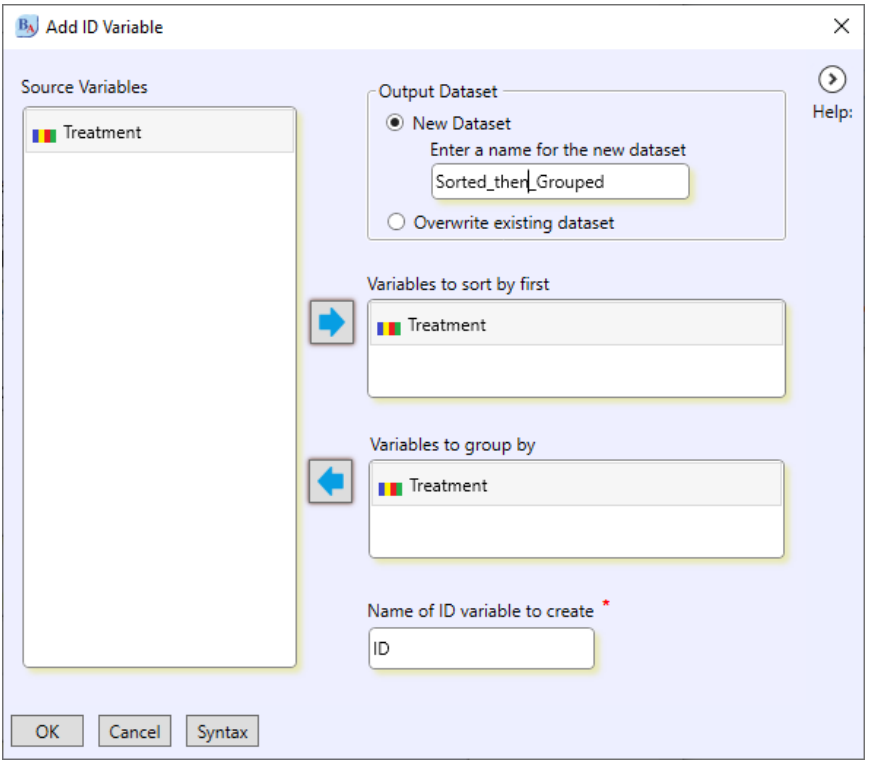


Fig. 4.1: The Add ID dialog.

that could place a person with a systolic blood pressure of 139 into the "normal" category and one with 140 into the "hypertension" category when in reality, the people have nearly identical health on that measure.

Let us see what happens when we apply the bin function to the variable age.

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Bin Numeric Variable(s)."
4. Move the variable age to the "Variable to Bin" box.
5. Enter "age_binned" into the "New variable Name" box.
6. Enter 10 in the "No. of Bins" box.
7. In the "Level Names" box, choose Ranges.
8. In the "Binning Method" box, choose "Natural Breaks (K Means clustering)." The dialog should now look like Fig. 4.2. Click OK.

The "Level Names" part of the dialog offers three choices. If you choose to specify names, you simply enter the names of the new levels separated by commas. For example, if you told it to divide age into just three bins,

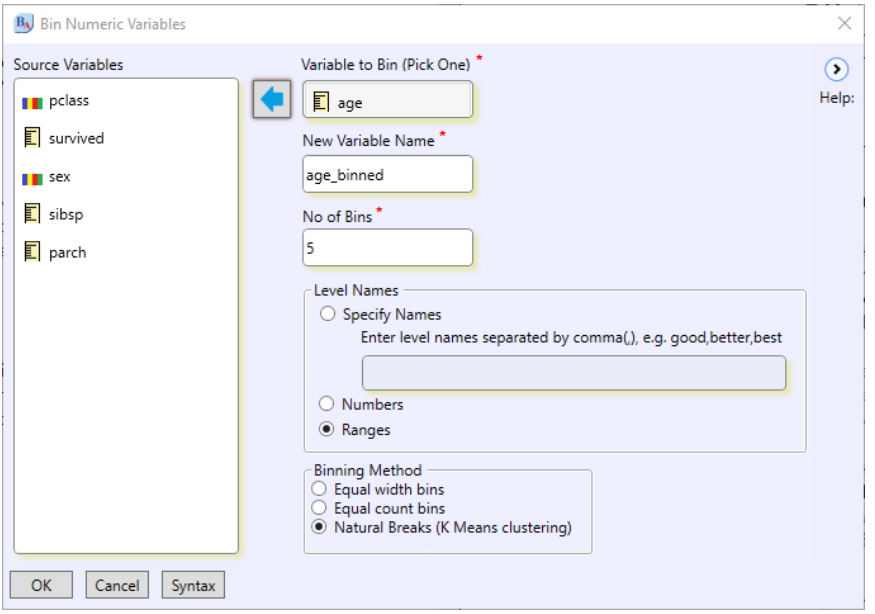


Fig. 4.2: Dialog for binning the variable age.

you could enter "Young, Middle-Aged, Old". If you choose "Numbers" it will use the integers 1, 2, 3.... By choosing "Ranges" we asked it to use the age ranges as the labels themselves. You can see what it chose in the bar chart shown in Fig. 4.3. It put people from negative infinity to 12 in its first category. Apparently, starting at zero wasn't part of its algorithm! Then it put those from 12 to 25 in its second bin. The parenthesis at the beginning of each age range is exclusive, so in the display "(12-25]" the range starts at just *over* 12 years old. The square bracket is inclusive, so it contains the exact age of 25.

The binning method also offers three choices. Equal width bins will divide the range of numbers by the number of bins and put the same number of values in each. In the case of age, each bin would be approximately 80/5 or 16 years wide. You can also do that type of binning using "Data> Rank Variables(s)," where it is called the "ntile" method. See Sec. 4.11 for details.

Equal count bins change each bin's width until there are 1046/5 people in each bin (there are 1046 people in the dataset). That is not a popular choice since all variables will have a rectangular distribution regardless of the original distribution (in our case, it was approximately normal).

The natural breaks method that we chose applied the K-means algorithm to search for natural groups within the variable. For details about that algorithm, see Sec. 6.3.2.

BlueSky does binning using the `RcmdrMisc`[6] package's `bin.var` function.

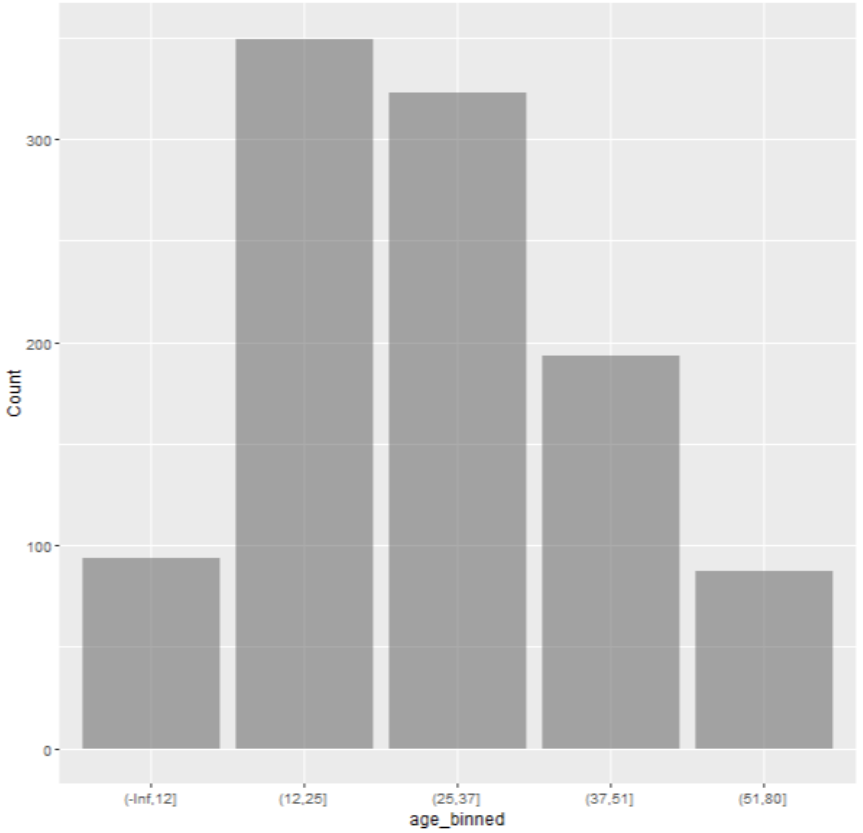


Fig. 4.3: Bar chart of counts for the binned version of age.

4.3 Compute Dummy Variables

Dummy (a.k.a. indicator) variables are used to convert factors into a series of binary variables for which "1" indicates membership in a category and "0" indicates non-membership. For example, if sex were a factor, it would be stored by default as the values 1 and 2, in alphabetical order (i.e. 1=female, 2=male). If those values were stored as a numeric variable, it would imply that males were somehow twice as "much" as females. To avoid such nonsense BlueSky, and its underlying R language, understand the role factors should play in statistical models and so they create a dummy variable on the fly. They focus on the first value, and in this case it essentially creates a variable "female" where the value 1 indicates being female and 0 indicates being male.

Note that for K levels of a factor, only K-1 dummy variables are needed to represent them; if the observation is none of the alternatives, it must be in the only category that did not get a dummy variable created for it. The category that does not have a dummy created for it is called the reference level. While many heuristic models accept K dummy variables, a statistical

model such as linear or logistic regression require K-1 of them, otherwise they will give you the warning:
`prediction from a rank-deficient fit may be misleading`.

Let us create dummy variables for the passenger class on the Titanic. The ship had classes of 1st, 2nd, and 3rd. So we need two dummies to represent the three classes. Here are the steps to follow:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Create Dummy Variables"
4. Move the variable pclass into the "Variables to be dummied" box.
5. Choose the "First Value" as the level to treat as reference.
6. Un-check the other options, and click OK.

The variables "pclass_2nd" and "pclass_3rd" will be added to the Datagrid's right side. Remember that you may have to use the scroll-bar and or arrows at the Datagrid's bottom to see the newly created variables. You can use those variables in any model you wish to build. If we instead chose "Most frequent value" as the level to treat as the reference, it would have created "pclass_1st" and "pclass_2nd" since 3rd class was the most popular one.

Creating K-1 dummies for a factor with K levels is best for statistical models because creating all K of them would create a redundancy that would cause problems. However, machine learning models usually are not bothered by K dummies, and it can make model interpretation easier. Also, some models require K dummies for the target variable when they are predicting class membership. To create K dummies choose, "Keep all levels." That is called "one-hot encoding," which gets its odd name from electrical circuitry terminology.

If you check the box, "Create dummy variables for missing values," it will create another dummy where 1 indicates a missing value for the original factor.

Whenever you create new variables, it is wise to check the Datagrid to ensure that BlueSky made the variables the way you expected.

BlueSky creates dummy variables using the `fastDummies` [11] package's `dummy_cols` function.

4.4 Compute New Variables

One of the most essential features of any data science tool is its ability to compute new variables. You can compute variables uniformly down a column, or you might do so conditioned on the values of other variables, such as body mass index calculated differently for males and females. You can also calculate them by applying an R function across rows, perhaps to get the mean of several variables.

The items in this section compute one variable at a time, but they can use complex formulas. If you wish to apply a single function, like a logarithm or square root to one or even many variables at once, use the Transform menu described in Sec. 4.14.

4.4.1 Compute

The item "Data> Compute New Variables> Compute" can create one variable at a time using formulas that can involve other variables and various functions. It affects every observation exactly the same way. If, instead, you need different formulas for subsets of your data, e.g., males and females, then see the conditional compute approach used in Sec. 4.4.3 and Sec. 4.4.4.

To demonstrate this, let us start with the variable age from the Titanic dataset, subtract its mean, and divide by its standard deviation. This popular computation converts variables into the same scale, with a mean of zero and a standard deviation of one. This is called a Z-score. You can perform this computation more quickly done using, "Data> Standardize Variables," but it makes a good example. Here are the steps involved:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Compute New Variables> Compute."
4. Enter the new variable name, "age_Z" in the "New/Existing Variable name" field.
5. In the field, "Construct the appropriate compute command" enter this formula: (age-mean(age))/sd(age). As you can see in Fig. 4.4, there is a formula builder under that box. If you did not know the names of the functions you needed, you could find them in the formula builder area. To find the statistical functions, you must click the arrows at the right-hand side of the function category names several times. Then clicking on a function name once will cause a description to appear in the Help box, telling you what it does. Double-clicking a function name adds it to the formula.
6. Click OK.

Your dialog should have ended up looking like Fig. 4.4. If it did not, you could use the History menu to recall it for modification.

BlueSky uses a wide selection of functions from R's Base and Stats packages to perform the calculations.

4.4.2 Compute, Apply a Function Across All Rows

A common way to create new variables is to apply a function across each row. For example, the personality test data described in Sec. 6.6 includes five questions for each of the five personality traits. Each of these would be summed or averaged to create a single measure of each trait.

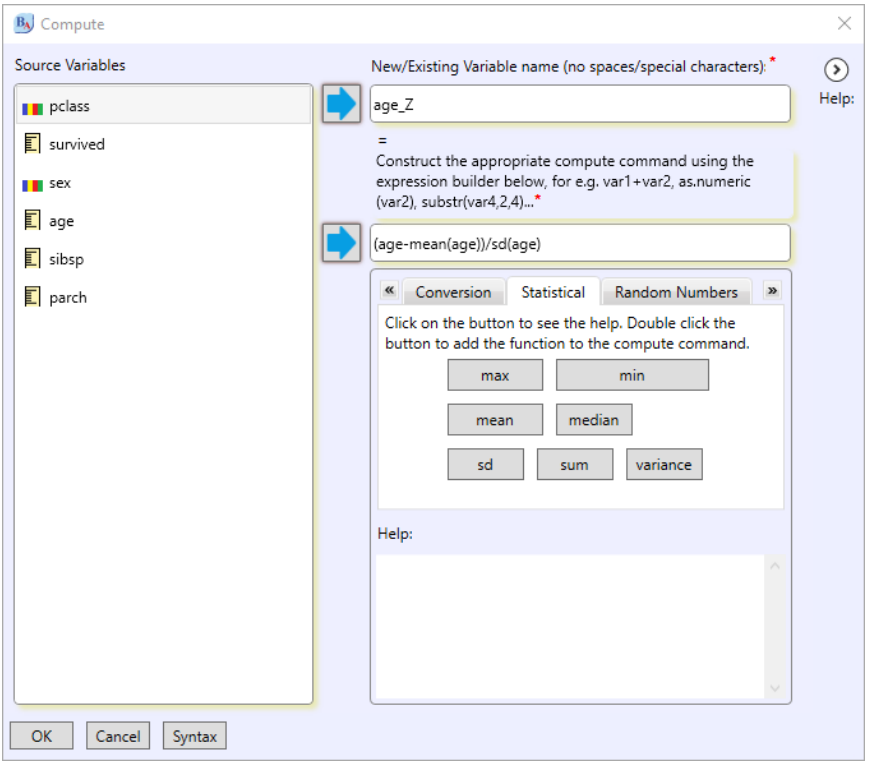


Fig. 4.4: Dialog for computing a Z-score for the variable age.

Let us create a new variable, extroversion, which averages the five items that make up that score.

1. Go to the Analysis window and choose "File> Load dataset from a package."
2. Enter the name "bfi" in the dialog box's lower right. That will load the dataset from the "psych" [22] package, but you do not need to fill in the package name unless you have installed another package that has a dataset of the same name.
3. Choose "Data> Compute New Variables> Compute, apply a function across all rows."
4. Fill in extroversion as the new variable name.
5. Move the variables e1 through e5 to the "Select Variables" box.
6. Choose the mean as the operation to perform, then click OK.

BlueSky will then add the variable to the Datagrid's far right side. You will probably have to move the bottom scroll bar to see it. Note that the Datagrid only displays 40 variables at a time, so depending on how many variables you have in your own data, you may not see the new variable until you use the scroll arrows on the Datagrid's bottom right.

BlueSky creates conditional variables using the `Base` package's `apply` function.

4.4.3 Conditional Compute, if/then

The item "Data> Compute New Variables> Conditional Compute, if/then" can create one variable at a time using formulas that include logical conditions (i.e., selections) to focus their work. When the data meet those conditions, the formula will execute. If not, then the observation will receive a missing value of "NA." If you need to use two formulas, one for when the logic is true and another when it is false, then see the approach described in Sec. 4.4.4. If you need every observation to be treated the same way, i.e., without logical conditions, see Sec. 4.4.

To demonstrate a conditional computation, let us try a variation on the example used in Sec. 4.4. There we used the variable age from the Titanic dataset, subtracted its mean, and divided by its standard deviation. This time we will perform this task only for the females. The mean age of females was 28.69 and their standard deviation was 14.58. Here are the steps involved:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(.RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Compute New Variables> Compute if/then."
4. Enter the new variable name, "age_Z" in the "New/Existing Variable name" field.
5. In the "If Conditional Expression" box, enter "`sex == "female"`". Note that BlueSky (and R) use the double-equals sign for logical equivalence.
6. In the field, "Value if condition is TRUE" enter this formula: `(age-28.69)/14.58`.
7. Click OK.

As you can see in Fig. 4.5, there is a formula builder under both the condition and formula boxes. If you did not know the names of the functions you needed, you could find them in the formula builder area. To find the set of functions you need, you must click the arrows at the function category's right-hand side. Clicking the tabs also helps to find the required set of functions. Then clicking on a function name once will cause a description to appear in the Help box, telling you what it does. Double-clicking a function name adds it to the formula.

Your dialog should have ended up looking like Fig. 4.5. If it did not, you could use the History menu to recall it for modification.

BlueSky uses a wide selection of functions from R's Base and Stats packages to perform the calculations. It creates conditional variables using the `Base` package's `ifelse` function.

4.4.4 Conditional Compute, if/then/else

The item "Data> Compute New Variables> Conditional Compute, if/then/else" can create one variable at a time using formulas that include logical conditions (i.e., selections) to focus their work. If the data meet that condition,

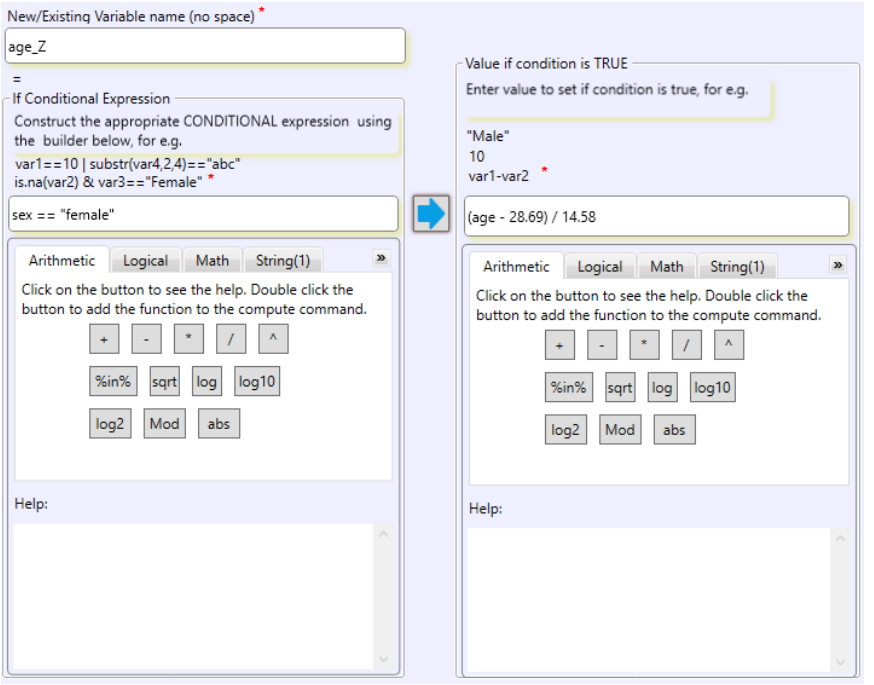


Fig. 4.5: Dialog for computing a Z-score for the variable age, females only. The center of the dialog has been selected to increase its size and legibility.

the TRUE formula will execute. If not, then the FALSE formula will execute. If the logical condition cannot be determined to be TRUE or FALSE, or if the formula components are missing, the observation will receive a missing value of "NA."

If instead, you need to treat every observation the same way, i.e., without logical conditions, see Sec. 4.4.

To demonstrate an if/then/else computation, let us try a variation on the example used in Sec. 4.4.3. There we used the variable age from the Titanic dataset, focused on just the females, subtracted its mean, and divided by its standard deviation. The non-females, i.e., males and missing, received "NA" values in that example. This time we will provide separate formulas for females and males. The mean age of females was 28.69, and their standard deviation was 14.58. The mean age of males was 30.59, and their standard deviation was 14.28. Here are the steps involved:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Compute New Variables> Compute if/then/else."
4. Enter the new variable name, "age_Z" in the "New/Existing Variable name" field.

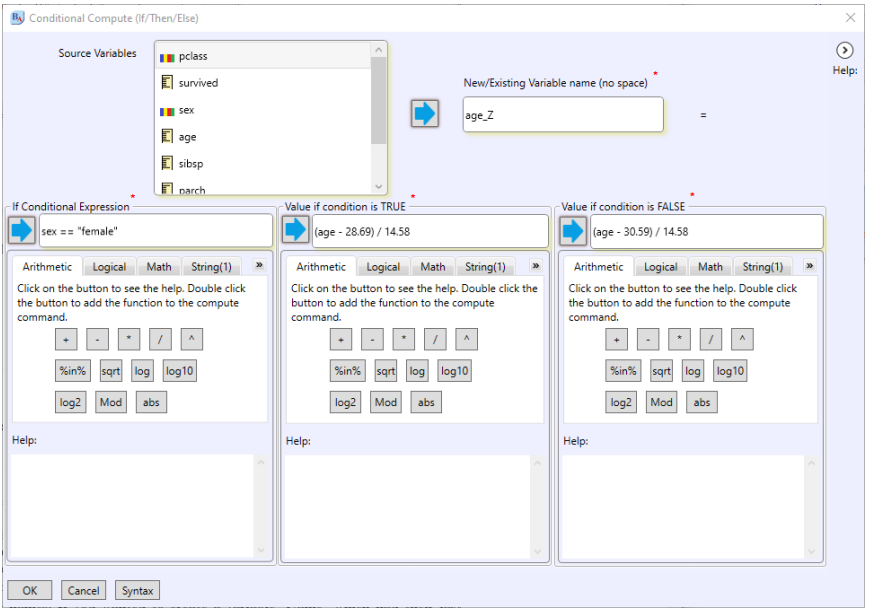


Fig. 4.6: Dialog for computing a Z-score for age, calculated separately for males and females.

5. In the "If Conditional Expression" box, enter "`sex == "female"`". Note that BlueSky (and R) use the double-equals sign for logical equivalence.
6. In the field, "Value if condition is TRUE" enter this formula: (age - 28.69) / 14.58.
7. In the field, "Value if condition is FALSE" enter this formula: (age - 30.59) / 14.58.
8. Click OK.

Your dialog should have ended up looking like Fig. 4.6. If you did not know the names of the functions you needed, you could find them in the formula builder areas. To find the set of functions you need, you must click the arrows at the function category's right-hand side. Clicking the tabs also helps to find the required set of functions. Then clicking on a function name once will cause a description to appear in the Help box, telling you what it does. Double-clicking a function name adds it to the formula.

BlueSky uses a wide selection of functions from R's Base and Stats packages to perform the calculations. It performs if/then/else computations using the `base` package's `ifelse` function.

4.5 Concatenate Multiple Variables

The item "Data> Concatenate Multiple Variables..." combines the values of different variables into a single value of a new variable. This is most widely used to combine character string names. For example, if you have a variable

"First" that contained people's first names and "Last" which contained their last names. If you wanted to create a variable "Name" which had their last name, a comma, then their first name, all in a single field this menu item would do the trick. The main thing know about it is that the *order* in which you select the variables matters. So, in this case, you would first move the variable "Last" into the selection box before you move the variable "First."

You can also use this approach dates if the day, month, and year are stored in separate variables. You could combine them into a single value with a slash in between the parts. The result would be a character variable, so an additional transformation would be required to convert it to a proper date variable. For details, see Sec. 4.7.2 for how to do that.

Another use for this approach is when variables store steps taken or decisions made, and you desire a unique path variable. For example, you might have variables named Page1, Page2, and Page3, which store the first, second, and third web page that a person visited. Combining these three would create values showing the path they took and the order they took it in. That could be further analyzed to determine if you should be encouraging visitors to take a particular path.

BlueSky creates conditional variables using the `base` package's `paste` function.

4.6 Convert Variable(s) to factors

The item "Data> Convert Variable(s) to factors," is one of the most frequently used menus in BlueSky. For BlueSky functions to do the right type of analysis, you must tell it when a variable is a factor (categorical). When you first read data in, it is easy to have BlueSky convert character data to factors automatically. However, with numeric variables, you have to tell it using this step.

Let us use the Titanic data for an example. It contains a variable named "survived" with 1 indicating the passenger survived and 0 indicating he or she died. To use this variable in many situations, we need to tell BlueSky that it is a factor. To do so:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Convert Variable(s) to Factor."
4. Move the variable survived to the "Variables to be converted to factor" box, then click OK. Note that you can convert many variables at once in that box.

If you wish to add value labels to the factor, use the Datagrid Variables tab described in Sec. 2.1.

BlueSky converts variables to factors using the `base` package's `as.factor` function.

4.7 Dates

Date variables are quite unusual in that they are stored in one format but displayed in another.

4.7.1 Convert Dates to Strings

The item "Data> Dates> Convert Dates to Strings," prompts you for the names of your date or date/time variables and whether you want to apply a prefix or suffix to the variable names. For example, if you had variables named x and y, and you specified "_character" as the suffix, the new date variables would be named x_character and y_character.

The tricky part of using this dialog is choosing the date format. There is a seemingly endless array of date and time formats on the drop-down menu. Using "Help> Help on Function" for the function "strptime" will bring up extensive documentation on the options. Two of the more popular formats are:

- For the format, "01/31/2020", choose this: "%m/%d/%Y".
- For the format, "2020-01-31", choose this: "%Y-%m-%d".
- If your date variable includes time information, choose one of the formats which ends with this: "%H:%M:%S".

BlueSky converts date variables to strings using the `base` package's `strptime` function.

4.7.2 Convert Strings to Dates

The item "Data> Dates> Convert String to Date," prompts you for the names of your string (character) variables and whether you want to apply a prefix or suffix to each variable name. For example, if you had variables named x and y, and you specified "_date" as the suffix, the new date variables would be named x_date and y_date.

Using "Help> Help on Function" for the function "strptime" will bring up extensive documentation on the options. Two of the more popular formats are:

- For the format, "01/31/2020", choose this: "%m/%d/%Y."
- For the format, "2020-01-31", choose this: "%Y-%m-%d."
- If your date variable includes time information, choose one of the formats which ends with this: "%H:%M:%S."

If you have dates in one variable and times in another, first concatenate the two with a single space between them (date first) by using the approach shown in Sec. 4.5.

BlueSky converts string variables to dates using the `base` package's `strptime` function.

Table 4.3: Date-out-of-order error report.

row	ID	ID2	date1	date2	date3
4	456	DEF	2015-01-15	2014-12-15	NA

4.7.3 Date Order Check

When collecting data across time, the variables that contain the dates often go from left to right, indicating the passage of time. With such data, it is important for the dates within a row to appear in order. Otherwise a data entry error has occurred. The item, "Dates> Date Order Check" will look across the rows of your data and notify you of any errors that it finds.

Let us look at an example:

1. Choose the item, "File> Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample Excel Datasets\Date_Order.XLS" select it, and click "Open."
3. Choose "sheet 1" when it asks you.
4. Choose "Data> Dates> Date Order Check."
5. Choose date1, date2, date3, *in that order* for the Date Variables.
6. Leave Comparison set to less than: "<".
7. Choose either (or both) of the ID variables as Row Identification.
8. You could check the "Create data set with date error variable" if you want to keep a copy of all the data and add a variable that indicates which rows contain errors. I will skip this step, but if you choose to use it, you can fill in the data set name and the variable's name, which indicates which rows had errors. Note that this new data set will immediately become the active data set in the Datagrid. You can return to the other one by clicking on its Datagrid tab.

Table 4.3 show the results. We see that on row 4, for ID 456 date1 appears before date2, which is never supposed to be the case. Hopefully we can now follow the data collection chain back to its source and fix the error.

To find dates out of order, BlueSky uses a variety of functions from the `dplyr` package.

4.8 Delete Variable(s)

The menu item "Data> Delete," variables removes columns from the Datagrid. To use it, I recommend first saving a copy of the data as shown in Sec. 2.8 as you cannot undo the deletion of columns. Then simply move the names of the variables to delete to the "Variables to be deleted" box and click OK.

4.9 Factor Levels

BlueSky, and the underlying R language, store categorical variables as factors. The values that factors contain are called levels. The order of the levels determines their order in output tables and graphs. The items which appear under "Data> Factor Levels" let you manage factor levels by displaying, adding, dropping, and rearranging them.

Any time you change factor levels, it is wise to run either frequencies (Sec. 6.13.3) or create a contingency table using the original factor and the new one to verify that the changes were done as you were expecting.

4.9.1 Add New Levels

The item "Data> Factor Levels> Add Factor Levels" lets you add new factor levels to one or more existing variables. Using it requires these steps:

1. Choose "Data> Factor Levels> Add Factor Levels" from the menu.
2. Move the factor variable(s) to the box labeled, "Factor variables to add new levels to."
3. Enter the new labels, separated by commas in the next box.
4. It is best *not* to overwrite existing variables, so choose to specify a prefix or suffix and enter it in that box. Then click OK.

BlueSky adds factor levels using the `forcats` package's `fct_expand` function.

4.9.2 Display Levels

The item "Data> Factor Levels> Display Levels," simply lists the name of each factor you choose, followed by the levels for each. Let us try it out using the Titanic dataset:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Factor Levels> Display Levels" to the "Factor variables to display levels for" and click OK.

The output will appear showing the levels for passenger class (pclass) and sex:

```
$pclass
[1] "1st" "2nd" "3rd"

$sex
[1] "female" "male"
```

Remember that the order shown determines the order in all tables and graphs. If you wish to change their order, see the instructions in Sections 4.9.6 through 4.9.9.

BlueSky displays factor levels using the `base` package's `levels` function.

4.9.3 Drop Unused Levels

Oddly enough, factors can have levels which are not used. This is more practical than it seems at first. Imagine that you are collecting patient diagnoses at several hospitals. The patients may have many possible illnesses, but it is unlikely that all illnesses are represented at the time you took your sample. The result would be a very large frequency table, which has many zero counts for the levels of diagnosis that were never used. In such a case, you may wish to delete the unused levels.

To delete unused levels, follow these steps:

1. Choose "Data> Factor Levels> Drop Unused Levels."
2. Move the factor names to the "Factor variables to drop levels" box.
3. Either choose "Drop all unused levels," which is the easiest, or choose to "Enter levels to drop separated by commas" and type them into that box.
4. You cannot undo this process, so it is best *not* to overwrite the original variables. Choose to specify a suffix or prefix for the variable names and click OK.

The new variables will appear on the right side of the Datagrid window. You may have to use the scroll-bar and/or the arrows at the Datagrid's bottom to see the new variables.

BlueSky drops factor levels using the `forcats` package's `fct_drop` function.

4.9.4 Label "NA" as Missing

In BlueSky, missing values are coded as "NA" for Not Available. Some tables and graphs will display counts of NAs, while others will not. You can use the item "Data> Factor Levels> Label "NA" as Missing," to convert the missing to a valid value such as "Missing" or any other label you choose. In other words, the NA values will no longer be missing but will instead have a valid value, which will tell the reader of tables and plots that it means missing.

To use this feature, fill in the factor variables you would like to change, fill in the label you want to use, and specify a suffix (or prefix) label, then click OK.

BlueSky drops factor levels using the `forcats` package's `fct_explicit_na` function.

4.9.5 Lumping Into "Other"

Factors that have many levels often have a problem because tables and plots that display their counts have a relatively small number of levels which account for 80%-90% of the data. That leaves many values with tiny or even zero counts. An example is the diagnoses of patients on a given day. There are thousands of diagnoses, but perhaps 20 of them account for 90% of the patients. A frequency plot would be left with thousands of zeros to display.

Some functions have special features to automatically deal with sparsely populated levels, e.g., the summary statistics demonstrated in Sec. 6.13.11. However, the item "Data> Factor Levels> Lumping into Other" provides this functionality across all items that deal with factors.

The steps used to do this are:

1. Choose "Data> Factor Levels> Lumping into Other."
2. Move the factors you choose to the selection box.
3. Choose a label, or accept the default of "Other."
4. Choose a lumping method based on frequency, the ratio of most common to least common, or proportion. The default usually works quite well.
5. If you have a variable indicating that each case represents multiple cases, enter its name in the "Variable Weights" box.
6. You can choose a method for handling ties in counting levels, but the default is usually sufficient.
7. It is best *not* to overwrite the existing variables since this step cannot be undone. Instead, specify a suffix like "lumped," and it will append that to the original variable name and use the composite name for the new variable(s).
8. Then click OK.

You should follow this step with a frequency table to see if the default settings did what you expected. See Sec. 6.13.3 for details.

BlueSky lumps factor levels using the `forcats` package's `fct_lump` function.

4.9.6 Reorder by Count

The item "Data> Factor Levels> Reorder by Count" is a popular feature which sorts the levels of a factor by the count of each level. When creating bar plots of counts, people often want to see the bars not in alphabetical order, but in order by count. That puts the bars in order from highest to lowest or vice versa.

1. Choose "Data> Factor Levels> Reorder by Count."
2. Move the factors you choose to the selection box.
3. Choose a label, or accept the default of "other".
4. It is best *not* to over-write the existing variables since this step cannot be un-done. Instead, specify a suffix like "lumped," and it will append that to the original variable name and use the composite name for the new variable(s).

5. Choose the order of descending or ascending, then click OK.

One other option is to make the factor an *ordered* one. The resulting appearance in basic tables or plots would be the same. However, creating an ordered factor will change how some analyses are run, which is why this is not a popular approach.

BlueSky reorders factor levels by count using the `forcats` package's `fct_infreq` function.

4.9.7 Reorder by Occurrence in Dataset

You reorder values using the item "Data> Factor Levels> Occurrence," when the data are stored in groups that are pre-sorted by the factor of interest. For example, if all males appear first followed by all females, and that was the desired order of the two levels in tables and graphs, this item keeps that order. Otherwise, factor levels are set in alphabetical order, so females would appear first. To set the order as entered:

1. Choose "Data> Factor Levels> Reorder by Occurrence in Dataset."
2. Move the factors you choose to the selection box.
3. It is best *not* to overwrite the existing variables since this step cannot be undone. Instead, specify a suffix like "lumped," and it will append that to the original variable name and use the composite name for the new variable(s).
4. Choose the order, or reverse order, then click OK.

One other option is to check the box to make the factor an *ordered* one. The resulting appearance in basic tables or plots would be the same. However, creating an ordered factor will change how some analyses are run, which is why this is not a popular approach.

BlueSky reorders factor levels by appearance in the dataset using the `forcats` package's `fct_inorder` function.

4.9.8 Reorder by One Other Variable

The item "Data> Factor Levels> Reorder by One Other Variable" is a popular feature which sorts the levels of a factor by a function of another variable. When creating bar plots of means, people often want to see the bars not in alphabetical order, but in order by those means. That puts the bars in order from highest to lowest or vice versa.

1. Choose "Data> Factor Levels> Reorder by One Other Variable."
2. Move the factor you choose to the reorder selection box.
3. Move the variable to get the mean (e.g.) on to the "Variables to reorder by" box.
4. Choose the function that determines the order.
5. It is best *not* to overwrite the existing variables since this step cannot be undone. Instead, specify a suffix like "lumped" and it will append that to the original variable name and use the composite name for the new variable(s).

6. Choose the order of descending or ascending, then click OK.

BlueSky reorders factor levels by another variable using the `forcats` package's `fct_reorder` function.

4.9.9 Reorder Levels Manually

When creating bar plots of counts, people often want to see the bars not in the default alphabetical order, but in an order of their own choosing. The item "Data> Factor Levels> Reorder Levels Manually," allows you to set the order of factor levels manually.

Keep in mind that there are other approaches in this section which cover how to set that order in a more automated fashion by count (Sec. 4.9.6) or by a function of another variable, such as the mean (Sec. 4.9.8.) However, sometimes you have an order in mind that can only be conveyed manually. To do so, follow these steps:

1. Choose "Data> Factor Levels> Reorder Levels Manually."
2. Move the factor you choose to the reorder box.
3. Specify factor labels in quotes and separated by commas, such as `"Male","Female"`. You only need to specify those that you wish to move, so in this case, merely entering `"Male"` would suffice.
4. Choose to place the labels you entered in first or last place.
5. Enter a new variable name to avoid overwriting the existing name since this step cannot be undone!
6. Click OK.

It is wise to run a frequency table (Sec. 6.13.3) afterward to check that it did what you expected.

BlueSky reorders factor levels manually using the `forcats` package's `fct_relevel` function.

4.9.10 Specify Levels to Keep or Replace by "Other"

Factors that have many levels often have a problem because tables and plots that display their counts have a relatively small number of levels that account for 80%-90% of the data. That leaves many values with tiny or even zero counts. An example is the diagnoses of patients on a given day. There are thousands of diagnoses, but perhaps 20 of them account for 90% of the patients. A frequency plot would be left with thousands of zeros to display.

Some functions have special features to deal with sparsely populated levels automatically, e.g. Sec. 6.13.11. However, the item "Data> Factor Levels> Lumping into Other" provides this functionality across all items that deal with factors.

Previously, in Sec. 4.9.5 we discussed an automated way to deal with sparse levels. However, when there are just a few levels to lump into an "other" category, it is easiest to do so manually. The steps used to do this are:

1. Choose "Data> Factor Levels> Specify Levels to Keep or Replace by Other."
2. Move the factors you choose to the selection box.
3. Choose a label, or accept the default of "other".
4. Choose a lumping method and naming those to keep or those to lump into "other".
5. It is best *not* to overwrite the existing variables since this step cannot be undone. Instead, specify a suffix like "lumped" and it will append that to the original variable name and use the composite name for the new variable(s).
6. Then click OK.

You should follow this step with a frequency table to see if the default settings did what you expected. See Sec. 6.13.3 for details.

BlueSky lumps factor levels using the `forcats` package's `fct_other` function.

4.10 Missing Values

The items under "Data> Missing Values" allow you to replace missing values with a valid value. The value might be a simple mean or median, or it could be the prediction from a sophisticated model.

To determine how many missing values you have, you might first perform an analysis using the items under "Analysis> Missing Values," in Sec. 6.9.

After substituting estimates for the missing values, you might also want to run summary statistics (Sec. 6.13), contingency tables (Sec. 6.4), correlations (Sec. 6.5), or paired mean comparison tests (Sec. 6.8, to compare the variables to themselves, before and after, to see what impact the replacement had.

4.10.1 Remove NAs

The item "Data> Missing Values> Remove NAs" does what is commonly called "listwise deletion." That simply deletes every row in the Datagrid that has any missing values on the variable selected.

You can then use the dataset to create multiple models which would be comparable since they are calculated using the exact same set of rows. Most statistical modeling methods do their own listwise deletion of missing values. However, if left up to them, each model would be done on slightly different cases which then could not be compared with an approach like analysis of variance.

Estimating the missing values instead of deleting the entire row is generally considered an approach that will yield better inferences about the population under study. See Sec. 4.10.5.

To remove NAs, follow these steps:

1. Choose "Data> Missing Values> Remove NAs".
2. Since this cannot be undone, choose the option to create a new dataset, and enter a name for it.

3. Select the variables to include and click OK.

BlueSky removes NAs using the `base` package's `na.omit` function.

4.10.2 Missing Values, basic

The item "Data> Missing Values> Missing Values, basic" performs simple substitutions, replacing missing values with the mean/median/min/max of each variable, or with a value you supply that will be used for all variables.

Imputing the missing values generates a much better estimate than these simple statistical summaries and is considered an approach that will yield better inferences about the population under study. See Sec. 4.10.5.

To substitute values for NAs, follow these steps:

1. Since this step cannot be un-done, create a copy of your data using "File> Save As" as described in Sec. 2.8.
2. Using the copy of your data, choose "Data> Missing Values> Missing Values, basic".
3. Select the variables to replace NAs for.
4. Select either the function to apply or a value to replace, then click OK.

BlueSky replaces NAs using a variety of the `base` package's statistical functions.

4.10.3 Missing Values, formula

The item "Data> Missing Values> Missing Values, formula" performs simple formula-based substitutions. It accepts factors, so if you wished to replace missing values with the most popular factor level, your formula could be simply, `mode(your_var_name)`.

While the formula you use could be something fairly sophisticated such as a linear regression, more automated imputation approaches are described in Sec. 4.10.5.

To substitute formula outcomes for NAs, follow these steps:

1. Since this step cannot be un-done, create a copy of your data using "File> Save As" as described in Sec. 2.8.
2. Using the copy of your data, choose "Data> Missing Values> Missing Values, formula".
3. Enter the formula, perhaps clicking on the formula builder keys to assist.
4. Select either the function to apply, or a value to replace, then click OK.

BlueSky replaces NAs using a variety of the `base` package's statistical functions.

4.10.4 Replace All Missing Values, factor and string variables

The item "Data> Missing Values> Missing Values, factor and string variables" performs simple string (character) substitutions. It will use a single string value for all variables, so they must all be on the same measurement scale for this to be useful.

It is almost always better to estimate what the missing factor level should be for each separate variable. That will yield better inferences about the population under study than will this simplistic approach. See Sec. 4.10.5.

To substitute strings for NAs, follow these steps:

1. Since this step cannot be un-done, create a copy of your data using "File> Save As" as described in Sec. 2.8.
2. Using the copy of your data, choose "Data> Missing Values> Missing Values, factor and string variables".
3. Select the variables to use.
4. Enter the string (do not enclose in quotes) and click OK.

BlueSky does this step using the string replacement features of R's `base` package.

4.10.5 Missing Values, models imputation

Missing value imputation is the best way to handle missing values. This approach uses powerful statistical or machine learning models to derive the best estimates. This approach does require a bit more effort than the simpler value substitution methods described in previous sections.

You have to specify a formula, which is comprised of variables that you think will be good predictors of the missing values. Plus signs, "+" separate these. For example, if you were imputing people's income, the formula might be `assets + debts`. If you were predicting people's systolic blood pressure, it might be `age + cholesterol + bmi`. You can include interactions by placing a colon between two variables, such as `assets + debts + assets:debts`.

You must also choose the type of model to use from Tab. 4.4. While choosing the absolute best for a particular situation may require quite a lot of knowledge, a model that works well in many situation is the "random forest." It can automatically model complex relationships, which include interactions and curves.

Some of the models allow you to specify parameters to improve their prediction. The random forest model usually does well with its default settings.

To use models for missing value imputation, follow these steps:

1. Since this step cannot be un-done, create a copy of your data using "File> Save As" as described in Sec. 2.8.
2. Using the copy of your data, choose "Data> Missing Values> Missing Values, models imputation."
3. Select the single variable to impute. It can only handle one at a time.

Table 4.4: Imputation methods.

Abbreviation	Modeling Method
impute_cart	Classification and Regression Tree
impute_en	Elastic Net (glm/lasso/ridge)
impute_em	Expectation Maximization
impute_knn	K-Nearest Neighbor
impute_mf	Iterative Random Forest
impute_rf	Random Forest
impute_rhd	Random Hot-deck
impute_rlm	Robust Linear Model
impute_shd	Sequential Hot-deck

4. Enter a formula for the model to use (guidance above).
5. Choose an imputation method. When in doubt, I recommend the random forest method: impute_rf.
6. If the model you chose accepts parameters, enter them in the parameter box. Otherwise, leave it empty.
7. Click OK.

BlueSky imputes NAs using several of the `simputation` package's functions, such as `impute_rf` for the random forest model.

4.11 Rank Variables

Ranking can convert values from smallest to largest into 1, 2, 3,... N where N is the sample size. It can also bin variables into ntiles where, for example, the lowest third become 1, the middle third become 2, and the highest third become 3.

However, when values are tied, there must be a decision regarding what number to assign. Unless you have many ties in your data and you care deeply about how they are ranked, you can skip this next discussion.

There are several approaches to breaking ties which are better to demonstrate than describe. Let us take the example where a variable has the scores: 95, 62, 80, 75, 75, 75, NA. Table 4.5 shows the various ranking algorithms and their effects on ties. There, three values have a value of 75, tied for the rank values of 2, 3, and 4.

- row_number gives them the values 2, 3, 4 in order of their appearance.
- min_rank gives them all the smallest rank of 2. This is especially popular in sports rankings.
- dense_rank gives them all the smallest rank of 2, but it also makes the highest value 4 instead of 6.
- ntile(x, 7) breaks them into 7 ntiles, which ends up in this case being the same as row_number.
- ntile(x, 3) divides them into three groups, which arbitrarily puts the first 75 into group 1 and the next two into group 2.

Table 4.5: Ranking methods. The top row shows some example data. Each row below that demonstrates how each ranking function shown on the left would affect the values.

Original Values:	95, 62, 80, 75, 75, 75, NA
row_number(x)	6, 1, 5, 2, 3, 4, NA
min_rank(x)	6, 1, 5, 2, 2, 2, NA
dense_rank(x)	4, 1, 3, 2, 2, 2, NA
ntile(x, 7)	6, 1, 5, 2, 3, 4, NA
ntile(x, 3)	3, 1, 3, 1, 2, 2, NA
percent_rank(x)	1.00, 0.000, 0.800, 0.200, 0.200, 0.200, NA
cume_dist(x)	1.00, 0.167, 0.833, 0.667, 0.667, 0.667, NA

- percent_rank converts them into percentages of the highest value, so the 75s all get a value of 0.20.
- cume_dist takes the percentages of the highest value and accumulates them as it goes.

Let us apply ranking to the numeric variables in the Titanic dataset:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Rank Variable(s)".
4. Fill in a suffix for your new variables, such as "ranked."
5. Move the variables age, sibsp, and parch to the "Variables to rank" box.
6. Leave empty the box for ranking within. You would use that if you wanted to start the ranking over within each level of a factor.
7. Select a ranking function. The default of min_rank is sufficient in most cases. I have chosen to use dense_rank.
8. Since we are not using the ntile method, you can leave the "number of groups" box empty.
9. The dialog should now look like Fig. 4.7. Click OK.

When finished, go to the Datagrid and check the results.

BlueSky does ranking using various functions from the `dplyr` package, such as `min_rank` and `dense_rank`.

4.12 Recode Variables

Recoding variables entails changing values. One of the best-known examples of recoding is converting numeric test scores into letter grades. In that case, the control string would be,
`0:59="F", 60:69="D",70:79="C", 80:89="B", 90:100="A"`. The keywords "lo" and "hi" can be used to represent negative and positive infinity, such

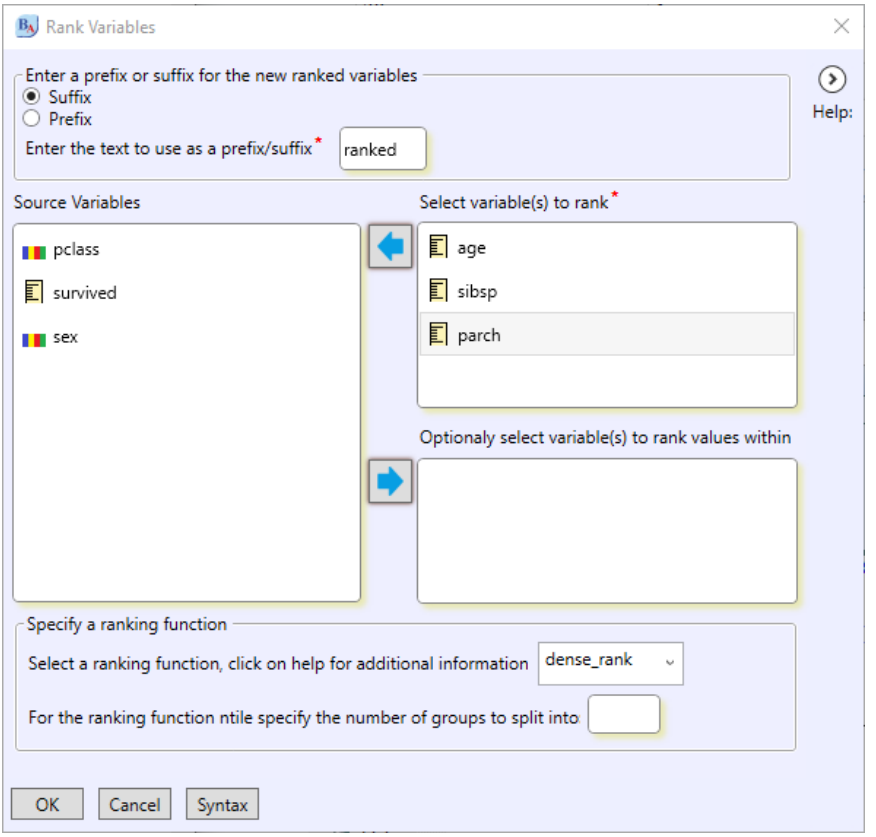


Fig. 4.7: Ranking variables dialog.

as `lo:60="F", 90:hi="A"`. The keyword "else" can be used to combine all unspecified values, as in "else=NA".

This recode would create a character variable or a set of them if you had multiple tests. Those could then be converted to factors using "Data> Convert Variables to Factor(s)" as described in Sec. 4.6. The BlueSky development team plans to add a "Convert to factor" checkbox. That may be included by the time you read this.

Recoding is also often used with tree-based models that would pick their own, perhaps less meaningful cut-points for continuous variables. For example, pre-binning age into 20-29, 30-39, etc. may make more sense when interpreting the model than having a tree model choose 20-28, 29-41, etc. In that case, the control string would be,
`20:29="20-29",30:39="30-39",40:49="40-49"` and so on.

A more automated approach to those first two examples is available with the "Data> Bin Numeric Variables" item, covered in Sec. 4.2. However, that approach picks its own cutoff values, while recoding allows you to specify them.

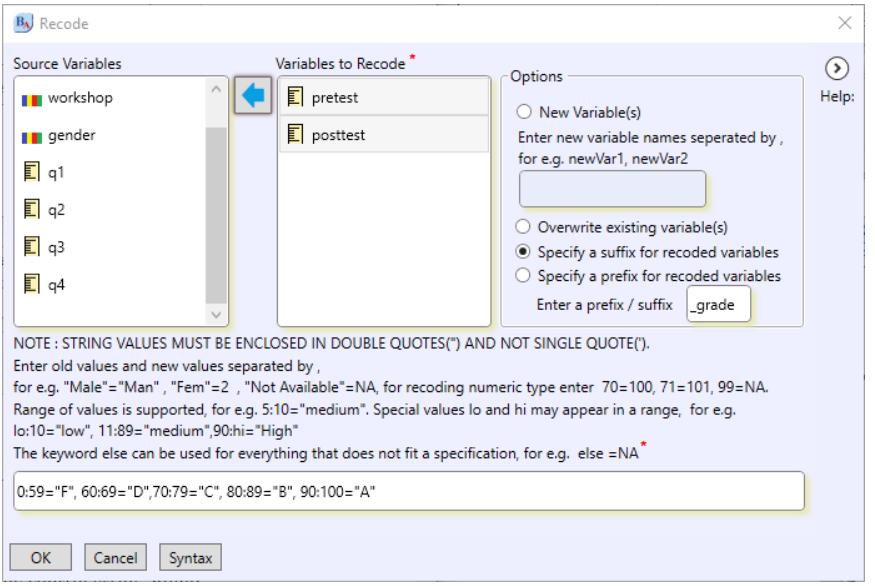


Fig. 4.8: Recode variables dialog. The control string is converting numeric test scores into letter grades.

You can also use recoding to collapse survey items into fewer values. This can simplify presentations in tabular or graphical form. For example, a question might ask how much the respondents agree on a 5-point scale that goes from "1=strongly disagree" to "5=strongly agree." Later, that could be collapsed into a 3-point scale of simply "agree," "neutral," "disagree." The control string to enter for that would be `1=2, 5=4`.

Another common reason to recode has to do with survey item wording. To measure extroversion you might ask one item as, "I like going to parties" and another as "I do not like socializing." Those two should correlate negatively, so if you created an overall score by averaging several such variables, the responses would cancel each other out. Therefore, you would recode them to all be in the same direction beforehand. In that case, the control string would be `1=5, 2=4, 4=2, 5=1`. Although it might appear that these cancel each other out, each value is changed only once.

Using the "Data> Recode Variables" is relatively straightforward. As shown in Fig. 4.8, you choose the variables to recode and decide how to create new variables. You do not want to overwrite the existing ones! I have chosen to add a suffix of "_grade" to my variables. So the grades will be named, "prefix_grade" and "posttest_grade." Then simply enter your control string following the examples above and click OK. Remember that it uses the "old_value=new_value" form.

BlueSky recodes variables using the `BlueSky` package's `BskyRecode` function, which incorporates the `car` package's `recode` function.

4.13 Standardize Variable(s)

Some modeling methods focus on explaining variance. Others focus on measuring distance among observations. In both cases, variables measured on scales that have large means and variances have a greater impact than they deserve. To correct for this, it is common to standardize all the predictors in a model by subtracting each variable's mean and dividing the result by its standard deviation. The result is that each variable will have a mean of zero and a standard deviation of one. The variables will range from roughly -3 to +3. The standardized scores are often called "Z scores."

To standardize your variables, follow these steps:

1. Choose "Data> Standardize Variables."
2. Select your variables.
3. It is best to create new variables rather than replace the existing ones, so choose a suffix for the new variable names such as "_Z". For example, if you have variables height and weight, the new ones would be named height_Z and weight_Z.
4. Check the boxes for center (subtract mean) and scale (divide by standard deviation), then click OK.

BlueSky standardizes variables using the `BlueSky` package's `BSkyStandardizeVars` function. BlueSky package!BSkyStandardizeVars

4.14 Transform Variable(s)

We often need to transform variables turn curved relationships into simpler linear ones, or to make the variable's distribution more bell-shaped. The item "Data> Transform Variable(s)" performs this task easily by limiting its transformations to a single function. For example, if you needed to take the logarithm of a set of variables, it would be quite easy. However, if you needed to apply a more complex formula to the variables, you would instead need to use the methods covered in the Compute Sec. 4.4.1.

To transform your variables, follow these steps:

1. Choose "Data> Transform Variables."
2. Select your variables.
3. It is best to create new variables rather than replace the existing ones, so choose a suffix for the new variable names using the function name that you choose. For example, if you have the variables height and weight and take the logarithm of each, you could use the suffix "_log" and the new ones would be named height_log and weight_log.
4. Choose the function name from the drop-down list, or type it in manually, then click OK.

BlueSky transforms variables using R's `base` package functions.

4.15 Aggregate to Dataset

The item "Data> Aggregate to Dataset," calculates various summary statistics by levels of a factor. You do this when you want to consider a group to be the unit of analysis rather than an individual case.

For example, let us create a dataset that contains the mean and counts for the people on the Titanic who were in 1st, 2nd, and 3rd class. Follow these steps:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Titanic.RData" select it, and click "Open."
3. Choose "Data> Aggregate to Dataset."
4. Enter the name of the dataset you wish to create, such as "Titanic_means."
5. Select the mean as the summarize option.
6. Move any variable you wish to summarize into the aggregate box. In this case, age.
7. Check the box for "Display counts..."
8. You can enter variable names, but keeping the original ones are usually fine.
9. Finally, move the factor pclass to the "Group by" box and click OK.

The new dataset will be created and added to the Datagrid. It will contain one observation for every combination of the "Group by" variables. It will become the default dataset, so any further analyses will use it. If instead, you wish to continue to analyze the original dataset, you will have to click its tab in the Datagrid.

BlueSky aggregates variables using various functions from the `dplyr` package.

4.16 Aggregate to Output

The item "Data> Aggregate to Output" does the same thing that "Aggregate to Dataset" does (Sec. 4.15), but instead of putting the result in a dataset, it writes it to the Output window. Its dialog is identical except for the box that lets you name an output dataset.

4.17 Expand Dataset by Weights

There are several ways you can use to weight each row in a dataset to represent more, or less, than a single row. In this section we briefly discuss two such methods: expansion and model-based weighting.

The expansion approach multiplies the dataset by the weight variable. For example, you might see a two-by-two table of counts in a journal article

and wish you had the raw data. Here is such a table that shows how many people had a disease and how many tested positive (1) and negative (0) for it:

```
Disease, Test,    Count
   1,      1,      670
   0,      1,      202
   1,      0,       74
   0,      0,      640
```

To expand such a table into the complete original dataset, follow these steps:

1. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets(RData)\Diagnostic Testing \epiR_example1_counts.RData" select it, and click "Open."
2. Provide a name for the new expanded dataset, like "Expanded"
3. Choose the item "Data> Expand Data by Weights" and give it Count as your weight variable.
4. Click OK.

Look in the Datagrid for your new dataset. You should see that it has only two variables: disease and test. The variable count is now gone, and if you scroll down to the bottom, you should see the new dataset has 1,586 rows. This expanded dataset is available in the same folder with the name, "epiR_example1.RData". It is used as the dataset in the section on Diagnostic Tests, Sec. 6.2.4.

If you did a crosstabulation on the two variables, it would recreate the original table counts.

Another way to use weights is to choose a method of analysis that incorporates them, such as linear regression, shown in Sec. 8.

4.18 Find Duplicates

Duplicate records often appear in datasets from a variety of sources. In companies, employees transferred from one department to another, may end up with records in both departments for a time. Finding and removing such duplicates is a common data cleansing step. Another topic that can be of great value in eliminating duplicates is selecting the last observation per group. When an ID variable is considered the "group," then you can sort by date and save the most recent record when deleting duplicate IDs. For details, see Sec. 4.25.

The item "Data> Find Duplicates," is relatively easy to use. By default, it finds duplicates based on the values of all variables. If instead you need to focus on particular variables, such as patient within hospital, you list them as "key variables." That is an optional step.

There are three datasets that the dialog (not shown) offers to make:

- "Create data set with all rows associated with the duplicates" – this is essentially creating a report of the duplicates. That makes it easy to study the duplicates, which may give you an idea of their origin. Hopefully, this will lead to improved data collection, preventing duplicates from occurring in your next study. By default, the output dataset name is an appropriate "allduprows," which you can change to whatever you prefer.
- "Create data set with original data and column indicating duplicates" – this leaves the duplicates in place, with the addition of a binary variable showing which are duplicates (1=duplicate, 0=not a duplicate) This is a useful form to return to the source of the data so that their staff can study the duplicates. Then if they choose, they can use the indicator variable to eliminate them. By default, the dataset name is "datadupvar" and the variable which indicates which are duplicates is named "duplicate." You are free to choose any other names, of course.
- "Create data set with all duplicates removed" – this is often the primary goal of this process: toss 'em out! The resulting dataset, named "nodupdata" by default, will contain all the original data, except for the duplicate records.

Let us step through an example.

1. Choose the item "File> Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets (RData)\duplicates.RData" select it, and click "Open."
3. Choose "Data> Find Duplicates."
4. Check the "Create dataset with all rows associated with the duplicates." This dataset will contain the list of duplicates only.
5. Check the box, "Create data set with all duplicates removed," and click OK. This dataset will contain all the data except for the duplicates.

The new data set named "allduprows" will now contain the duplicate rows (not shown). You can study this in the Datagrid to see what types of duplicates exist and perhaps figure out why. The new data set "nodupdata" will contain a copy of your original data set with all the duplicates removed. You can now use that for analysis with no worries of duplicate records mucking up the works!

Warning: finding duplicates creates new data sets, which immediately become the active data set! This means that you cannot use the History menu to run variations on this analysis until you click on the tab to bring your original data back to the front of the Datagrid. If you see the message, "unable to open allduprows" that is what it is trying to tell you.

To find duplicate records, BlueSky uses a variety of functions from the `dplyr` package.

4.19 Merge Datasets

The process of merging or joining datasets adds variables from one dataset to another. The first dataset you choose is called the "left" one, and the

Table 4.6: Band members dataset for merge example.

Name	Band
Mick	Stones
John	Beatles
Paul	Beatles

Table 4.7: Band instruments dataset for merge example.

Name	Instrument
John	guitar
Paul	bass
Keith	guitar

Table 4.8: Bands Merged Dataset.

Name	Band	Plays
Mick	Stones	NA
John	Beatles	guitar
Paul	Beatles	bass
Kieth	Stones	guitar

second is called the "right" one. Those names indicate the order of the variables in the combined set.

The dangerous part of merging is ensuring that the rows match up properly. For example, Tab. 4.6 has some members of the Beatles and the Rolling Stones. Table 4.7 contains the instruments they play. However, the band members' order is not the same in each dataset, so they cannot simply be stuck together side-by-side. You can merge datasets that way, but it is never a good idea since all it takes is one row out of place and all the matches after that will be quite a mess!

Here are the steps to merge these two datasets:

1. Choose the item "File> Load Dataset from a Package."
2. Fill in the name "band_members" in the lower box. The upper box asks for the package name, which is "dplyr," but unless another package uses the name "band_members" it is not necessary to add that information.
3. Choose the item "File> Load Dataset from a Package." This time load the dataset, "band_instruments."
4. Choose "Data> Merge Datasets."
5. Enter "bands_joined," as the name of the merged dataset.
6. Choose "band_members," as the first dataset and "band_instruments" as the second.
7. Under Merge Options, choose "Full Join." The dialog should now look like Fig. 4.9.
8. Click OK.

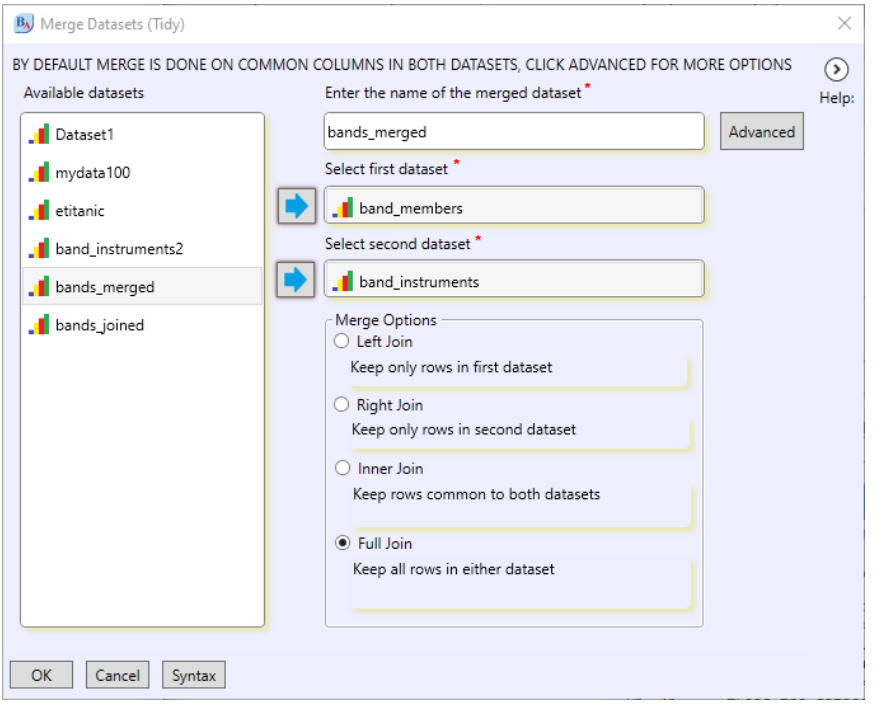


Fig. 4.9: Merge datasets dialog.

The new dataset should appear as shown in Tab. 4.8. There we have all three variables in one dataset. While each of the previous datasets had three observations, the merged one has four since Kieth was not in the first one, and Mick was not in the second. Mick has "NA" filled in as what he plays, and Kieth has an "NA" for his band. BlueSky has filled those in automatically where there was no data.

The `dplyr` package's help file explains the several ways you can merge datasets:

- Left Join – returns all rows from x, and all columns from x and y. Rows in x with no match in y will have NA values in the new columns. If there are multiple matches between x and y, all combinations of the matches are returned.
- Right Join – returns all rows from y, and all columns from x and y. Rows in y with no match in x will have NA values in the new columns. If there are multiple matches between x and y, all combinations of the matches are returned.
- Inner Join – return all rows from x where there are matching values in y, and all columns from x and y. If there are multiple matches between x and y, all combination of the matches are returned.
- Full Join – returns all rows and all columns from both x and y. Where there are not matching values, returns NA for the one missing.

To help protect against mismatched rows, the merge will match the rows on all the variables that the datasets have in common. In our example, that was the band name. In a more complex example with postal addresses, both files might have state, county, city, street, and name. That would help prevent erroneous matches by name alone.

The advanced setting on the merge dialog allows you to control how the columns match:

- If the datasets being merged have common column names – in this box you can fill in a subset of shared variable names to prevent it from matching on them all.
- If column names on which the merge is done are different in each dataset – in this box you can create matches from variables that have different names. For example, a dataset named "band_instruments2" uses the variable "artist" instead of "name" as the performer. these could still be used by filling in `"name" = "artist"` in that control field.
- If there ae common column names in both datasets – and these variables are not being used to match on, then they will be kept with suffixes ".x" and ".y". You can use this control field to use alternatives, such as, "_left" and "_right" to indicate which variable the variable came out of. While that may be fine for debugging purposes, I recommend manually choosing which to keep by deleting the redundancy before attempting the merge.

BlueSky merges datasets using various functions from the `dplyr` package, including `inner_join`, `outer_join`, `left_join`, and `right_join`.

4.20 Refresh Datagrid

The Datagrid automatically shows all changes made through the use of BlueSky dialogs. However, if you use the Syntax Editor to make changes to one of the open datasets, how will the Datagrid know you did that? BlueSky calls that "refreshing the Datagrid." Refreshing it makes BlueSky find the R dataset in memory and re-display it in the Datagrid.

There three ways to refresh the Datagrid.

First, there is a refresh icon at the top of the menu bar, fourth from the left. That will cause the active dataset to refresh immediately.

Second, using menus, the item, "Data> Refresh Datagrid" causes the changes to become visible in the Datagrid and in any other dialogs *immediately*, without using a dialog.

Third, using R code, you can call the R function, `BskyLoadRefreshDataframe`, as described in Sec. 12.7.

In all three approaches refresh the active Datagrid immediately. Therefore, ensure that the dataset you wish to refresh is the active one before making this selection. Remember that the active one is the one whose data is showing at the moment. If it is not, click its tab to bring it to the front.

4.21 Reload Dataset from File

When you have used "File> Open" to read a dataset from a file, and later need to load it again, the item "Data> Reload Dataset from File" is the quickest way to do so. This is particularly useful when you transform the dataset in some way that leads to errors. It works immediately, with no dialog box, so it is important to select the correct dataset tab before using it!

Note that this menu item does not reload datasets that have been loaded using, "File> Load Dataset From a Package."

4.22 Reorder Variables in Dataset Alphabetically

The item "Data> Reorder Variables in Dataset Alphabetically," sorts the variables by their names. Its dialog offers the orders ascending "A-Z," and descending, "Z-A." Both the Datagrid and all dialogs will display the new order.

However, the order is not automatically saved. You can use the item, "File> Save As" to make a copy. That is much safer than using "File> Save" as you may change your mind later, and there is no way to undo the sort to retrieve the original order.

BlueSky sorts variable names using the `base` package's `sort` function then selects them using the `dplyr` package's `select` function.

4.23 Reshape

There are two standard structures for datasets known as "wide" and "long" (or "tall"). Repeating measurements across time can lead to either of these structures.

The wide data structure stores the repeated measures as multiple variables. More repeats across time mean more variables (columns), not more rows. For example, in a six-month dietary study, each participant's weight would be taken and stored as six variables. These would typically be named weight1, weight2,... weight6. This structure is ideal for correlating the weights and doing, say, a paired t-test between weight1 and weight6 to see if a significant change had occurred. However, the wide format is not suitable for plotting all the weights across time, nor is it useful for analysis of variance.

The long data structure stores repeated measures in just two variables, often named "time" and "y". As you add more repeats across time, the dataset gets longer, adding rows instead of columns. In our dietary example, this data structure would be six times longer than the wide format. This format is ideal for plotting time on the x-axis and y on the y-axis (hence the commonly chosen name of "y!") It also makes it easy to do modern repeated measures analysis of variance, which improves power by modeling

Table 4.9: Sleep dataset after converting to wide format.

ID	X1	X2
1	0.7	1.9
2	-1.6	0.8
...	...	...
10	2	3.4

the covariance matrix across time, often with the auto-regressive AR1 structure.

Being able to easily switch between these two formats is essential for performing all the various data analysis methods!

4.23.1 Reshape, long to wide

To demonstrate converting from long to wide format, let us use the sleep study as an example. The study measured ten people before and after taking a sleep enhancement drug. The question is: did the drug help people get extra sleep? The variables are:

- ID – Subject identifier.
- group – A poor name for what is actually time. The same people are in both groups.
- extra – The extra sleep, which can be negative if it was actually less sleep.

To convert this dataset to wide format, follow these steps:

1. Choose the item "File> Load Dataset from a Package."
2. Fill in the name sleep in the lower box. The upper box asks for the package name, which is "datasets," but unless another package uses the name "sleep" it is unnecessary to add that information.
3. Choose the item "Data> Reshape> Reshape, long to wide."
4. Enter a dataset name such as "sleep_wide."
5. Enter group as the repeated factor.
6. Enter extra as the repeated value.
7. Click OK.

The resulting table should look like Tab. 4.9. Note that two new variables are named X1 and X2. I would rename those to be Time1 and Time1, or perhaps Extra1 and Extra2, by typing over their names in the Datagrid's Variables tab.

BlueSky converts from long to wide structures using the `tidyr` package's `spread` function.

Table 4.10: Sleep dataset converted back to long format.

ID	time	y
1	X1	0.7
2	X1	-1.6
...	...	...
10	X2	3.4 (20th line)

4.23.2 Reshape, wide to long

This section continues the example of the previous one, so make sure you do that one first. To convert that wide data structure to a long one, follow these steps:

1. Choose the item "Data> Reshape> Reshape, wide to long".
2. Enter the dataset name that we created in the last section, "sleep_wide".
3. Enter time as the repeated factor.
4. Enter y as the repeated value.
5. Select variables X1 and X2 as variables to reshape into a single column.
6. Click OK.

The sleep_wide dataset should now look like this Tab. 4.10.

BlueSky converts from long to wide structures using the `tidyr` package's `gather` function.

4.24 Sample Dataset

Sampling from a dataset is a common task in data analysis. You might do it to see how the results of an analysis change with different samples. A useful model should predict well on different samples. The item "Data> Sample Dataset" is one way to choose a sample.

Another item exists that will partition the data into training and testing samples. That makes it easier to ensure that you are testing a model on data that were not included in the sample that was used to train the model. For that approach, see Sec. 4.28.

Random sampling to compare training and testing of models is also automated in the validation methods described in Model Tuning, Chap. 9.

To take a random sample, follow these steps:

1. Choose the Datagrid tab of the dataset you wish to sample.
2. Fill in the name of a dataset to save the results to. Do not overwrite the original dataset as there is no way to undo the sample!
3. Choose either a percentage or a number of records to sample.
4. Choose whether to sample with replacement. With replacement means that it might sample same record repeatedly. Otherwise, once it moves a record to the sample dataset, it can never sample that one again during this particular use of the dialog.

5. Set a seed so that you will be able to repeat the sample. Remember that you will not get a new sample so long as you leave the seed set at its default value!

Once finished, the sample will appear as the active dataset in the Datagrid.

BlueSky samples datasets using the `dplyr` package's `sample_n` function.

4.25 Select First or Last Observations Within Groups

Selecting the first or last observations within groups is a very useful task because those cases often contain particularly helpful information. When the data are sorted by group, and date in ascending order within the group, the last observation is the most recent. If you were eliminating duplicate IDs, this would allow you to keep the most up-to-date record. For more on removing duplicates, see Sec. 4.18.

You can use this approach to find, say, the top-selling salesman to whom you would give a bonus. If you sort the data in *descending* order, then the best one will be in the *first* position. You can see that in these cases, choosing first or last often depends upon the sorted order of the dataset. You can set that order by using "Data> Sort," shown in Sec. 4.26.

However, there are also cases where sort order does not matter. When running cumulative totals, perhaps adding employee salaries for each record, then the *last* employee in the group also contains the total salary expense for that department. Nothing about that employee's order is special other than he or she just happened to be the last, and so contains the cumulative total for that department.

Our examples so far have focused on choosing the *single* first or last observation. While this is the most common task of this type, you may also want to select more than one. This can occur when, say, the three top-selling salesmen will get a bonus. If you sort the data in *descending* order, then the best three will be at the top of each group.

The key fields on the dialog (not shown) for selecting first or last observations per group are:

1. Group variables – here, you choose the group variables and the last variable to appear determine first or last status. For example, if the variables "hospital", and "department" appear in that order, then the analysis will choose first or last in each department, *within* each hospital.
2. Position – first or last. This is an easy choice once you understand the examples discussed above. With careful consideration of your goal, and the sort order (if relevant), choose the one that meets your goal.
3. Number of Observations to Select – this is often best left at the default value of 1. However, if you want to pick the best 3, or top 10, etc. of a group, choose that number here.
4. Subset Data Set Name – this is the data set that you are creating. Its default name of "subsetdata" may be fine; if not, choose your own.

Those choices are usually sufficient since they will select out all of the first or last observations into their own data set. That is most often the goal. However, if you wish to make a new data set that contains all the original data, plus a variable which indicates whether or not the record is first or last, then fill in these additional fields:

1. Create Data Set with First or Last Indicator Variable – this check-box activates the creation of a data set, which is a duplicate of the original but which also contains the indicator variable. **Warning:** if you choose to create this data set, it will immediately become the active data set, moving to the front of the Datagrid, covering up the first or last data you may have also created. Clicking on the data set tabs in the Datagrid will show you the various resulting data sets.
2. Data Set Name – is the name of the duplicate data set. By default, this is "originaldata."
3. Indicator Name – this is the name of the variable, which indicates that the record is first or last per group.

Let us look at an example. I would like to find the largest male and female penguins in each species. Here are the steps:

1. Choose the item, "File> Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample R Datasets (RData)\Penguins.RData" select it, and click "Open."
3. Choose "Data> Sort" and sort by species, sex, and body_mass_g. It is essential to leave the sort order ascending order, or "asc" since we want the largest to be the *last* of each species-sex group.
4. Choose "Data> Select First or Last Observation Per Group."
5. Select species, then sex (matching the order in the sorting step above).
6. Set Position to last.
7. "Leave Number of Observations to Select" at 1.
8. For "Subset Data Set Name" use "BigBirds" and click OK.

The resulting data are shown in Fig. 4.10. I manually deleted the other variables to make the figure larger and more legible.

To find first or last observations per group, BlueSky uses various functions from the `dplyr` package.

4.26 Sort Dataset

Sorting a dataset is a helpful way to see the largest or smallest values of one or more variables. Once sorted, you can also look at other the values of other variables that you did not sort on to see if they are near their min or max values.

Unlike some spreadsheet software, each row's contents are locked together, guaranteeing that no statistical analyses will change as a result of the sort. If

species	body_mass_g	sex
Adelie	3900	female
Adelie	4775	male
Chinstrap	4150	female
Chinstrap	4800	male
Gentoo	5200	female
Gentoo	6300	male

Fig. 4.10: Data resulting from selecting the last observation per group. The largest male and female penguins in each species were chosen as "last" after data were sorted by species, sex, and ascending body mass in grams.

you instead needed to sort one column independent of the others (usually a perilous move), you would have to do so using R code in the Syntax Editor.

To sort a dataset, choose it in the Datagrid to make it the active one. Then simply choose "Data> Sort Dataset" and select the variables you want to sort on. You may also set the sort direction to be ascending (abbreviated "asc") or descending ("desc") order. Unfortunately, the choice of direction applies to all the variables, so you cannot sort by ascending X then descending Y in a single step. However, you can sort by Y descending in one step, then by X ascending in a second step to achieve the same result.

BlueSky sorts datasets using the `dplyr` package's `arrange` function.

4.27 Sort to Output

The item "Data> Sort to Output," sorts the dataset by one or more variables. It works the same as "Data> Sort Dataset" except that it displays the result in the Output window and leaves the active dataset in its original order.

To sort a dataset, choose it in the Datagrid to make it the active one. Then simply select "Data> Sort Dataset to Output," and choose the variables you want to sort on. You may also set the sort direction to be ascending (abbreviated "asc") or descending ("desc") order. Unfortunately, the choice of direction applies to all the variables, so you cannot sort by ascending X then descending Y. The resulting data will appear in the Output window.

BlueSky sorts datasets using the `dplyr` package's `arrange` function.

4.28 Split Dataset

Items on this menu split the active dataset in several ways. Splitting for group analysis simply repeats any analysis for each level of a chosen factor (or factors). Splitting for partitioning divides the data into model training and testing datasets. See the following sections for details.

4.28.1 For Group Analysis

Splitting the dataset for group analysis causes every menu selection after than to repeat for each level of one or more factors. This is called "split-file," "by group," or "group by" analysis.

To turn this on, choose "Data> Split Dataset> For Group Analysis> Split." A dialog will then prompt you to select the split factors. In the lower right corner of the active dataset, the Datagrid will display the message, "Split on: var_name" to warn you that you have the split turned on. As long as the split is active, it will repeat every following analysis for every level of the factor(s) named. For example, if you choose to split by gender and then choose a summary analysis, it will summarize once for the females and again for the males. That repetition of analysis will continue until you choose "Data> Split Dataset> For Group Analysis> Remove Split."

BlueSky splits datasets using the `BlueSky` package's `BSkySetDataFrameSplit` function.

4.28.2 For Partitioning

Splitting a dataset for partitioning creates two new datasets, commonly called the training and testing datasets. This section discusses how to do that manually. The chapter on Model Tuning includes model validation methods that automatically perform this type of partitioning, repeatedly trying different splits, if you so choose. See Chap. 9 for details.

Random Split

To manually create one random partitioning split, choose "Data> Split Dataset> For Group Analysis> Random Split". Its dialog will offer the new dataset names of "traindata" and "testdata," which you are free to change. It also offers a split percentage of 80% which will leave 20% for the test dataset. It offers to set a randomization seed of "123" which ensures that you will get the same split every time. Keep in mind that creating a different split requires you to change that seed. Leaving it blank will tell it to use a random seed, which means that you will be unlikely to ever get that exact split again.

Since this item creates two datasets, it is up to you to keep track of which is active! It will create the traindata first, which means that testdata will be the active dataset when it finishes. Make sure you click on the traindata tab to make it the active dataset before developing your model!

Stratified Split

A stratified split creates a random split while maintaining the ratio of a target factor. For example, if you were trying to predict who would get a disease, you could maintain the overall prevalence within the training and testing data so that the sick/ill ratio is the same in each.

To manually create one stratified split, choose "Data> Split Dataset> For Group Analysis> Random Split". Its dialog will offer the new dataset names of "traindata" and "testdata," which you are free to change. It also offers a split percentage of 80%, which will leave 20% for the test dataset. It offers to set a randomization seed of "123" which ensures that you will get the same split every time. Keep in mind that creating a different split requires you to change that seed. Leaving it blank will tell it to use a random seed, which means that you will be unlikely ever to get that exact split again.

It asks one more thing than the random sample split asked: the variable to construct the ransom samples from. That would be a target factor.

BlueSky performs stratified splits using the `caret` package's `createDataPartition` function.

4.29 Stack Datasets

Stacking datasets is a way to add more observations or rows to the bottom of a dataset. The two datasets should have roughly the same variable names. In the case where one dataset has a variable that is lacking in the other when combined, missing values will be created and filled in as "NA" for Not Available.

To stack two datasets, choose the item, "Data> Stack Datasets." A dialog will prompt for the name of the newly combined dataset, and the names two datasets to be stacked. The first one will come out on top.

The main decision to choose one of the following stack options:

- Bind Rows – this will stack the two with no changes, which may end up with duplicate rows.
- Union – this will stack the datasets and remove exact duplicates, considering the values of all variables.
- Intersect – this will keep only the rows that are common to both datasets. This comes in handy when, for example, people are moved from one department to another and you need a list of those who are accidentally in both.
- Difference – this keeps only rows in the first dataset, which are not in the second.

There is also an option hidden under the Advanced button. It prompts you for the name of a variable that will store which dataset each record came from. It is a character variable with the values, "1", "2",.... It can allow you to perform later filtering based on both variable values and which dataset the observations were originally stored in.

BlueSky stacks datasets using the `dplyr` package's `bind_rows` function.

4.30 Subset Dataset

The item "Data> Subset Dataset," allows you to select variables and filter observations. The chosen data can then be routed to a new dataset or

Table 4.11: Example logical selections and their outcomes.

Logic	Selects
`sex == "male"`	Males
`age < 2`	Toddlers
`sex == "male" & age < 2`	Male toddlers
`class=="1st" | class == "2nd"`	1st or 2nd graders
`is.na(age)`	cases with missing values for age

Table 4.12: Logical filtering operators.

Logic	Operator	Notes
Equals	==	Cannot use EQ
Less than	<	Cannot use LT
Greater than	>	Cannot use GT
Less or equal	<=	Cannot use LE
Greater or equal	>=	Cannot use GE
Not equal	!=	Cannot use NE
And	&	Cannot use AND
Or	|	Cannot use OR
Is missing	`is.na(x)`	Can't use `x!=NA`

overwritten to the original dataset. Since you cannot undo this process, I strongly recommend never overwriting the existing data. It is just too easy to make a mistake.

You select variables in the usual way, and the observations are filtered using logic. Table 4.11 shows some common example selections as applied to a dataset of school children. Table 4.12 shows the logical operators you can use to build your own selections.

The checkbox for selecting distinct cases eliminates any duplicate records in your subset. The one for removing unused factor levels deletes factor levels that were excluded by the selection. For example, the selection `class=="1st" | class == "2nd"` would only contain 1st and 2nd graders. However, the values of "3rd", "4th", and so on would remain in the factor. Frequency table or bar plots would still show those unused levels as having zero counts unless you check the box to remove them.

BlueSky subsets datasets using the `dplyr` package's `select` and `filter` function.

4.31 Subset Dataset to Output

The item "Data> Subset Dataset to Output," does the same thing as "Data> Subset Dataset" except that the results are printed to the Output window. The only difference in the dialogs is that the option to remove unused values is unavailable since no further analysis is possible; it just prints the data. For details, see Sec. 4.30.

4.32 Transpose Dataset

Transposing a dataset means flipping it on its side so that rows become columns and vice versa. This is useful when you want to relate observations to one another instead of variables.

For example, consider a dataset in which Olympic judges are in rows and each variable is the athlete's ratings on one of several performances, e.g. 25-meter dash, 50-meter dash, and so on. Correlations would measure the relationship between performances. However, you might want to see how consistent judges are. Transposing the dataset would flip it on its side, making judges the columns to be correlated.

The main problem with transposing is that even a single factor or character variable will cause all the numeric variables to become factors or character variables! That is because the single variable will flip to become a single case and spread its values across all the new columns. To avoid this problem, it is generally best to use "Data> Transpose Dataset> Transpose, select variables" so you can select only numeric variables. If your data contains only numeric variables, then the option, "Data> Transpose Dataset> Transpose, entire dataset" is fine, and a bit quicker since you do not have to choose variables.

BlueSky subsets datasets using the `base` package's `t` function.

4.33 Legacy Data Management

The "Data> Legacy Data Management" section replicates several other items on the Data menu using `base` R functions. The others use functions from the `tidyverse` package.

4.33.1 Aggregate

This item does the same thing as Sec. 4.15, but uses the `base` package's `aggregate` function.

4.33.2 Sort

This item does the same thing as Sec. 4.26, but uses the `base` package's `aggregate` function.

4.33.3 Subset

This item does the same thing as Sec. 4.15 but uses the `base` package's `subset` function.

5

Graphics Menu

BlueSky offers a wide range of high-quality data visualizations in its graphics menu. The dialogs that drive them are all quick and easy to use.

Behind the scenes, BlueSky uses R's popular `ggplot2` graphics package to create most plots. The "gg" in its name refers to the Grammar of Graphics. That grammar is very powerful, but can be challenging to learn because it is a very abstract language. One example of this abstraction is the simple pie chart. The Grammar of Graphics view of a pie is that it is a stacked *bar* plot containing a single bar which is divided by levels of a factor. That bar is then made circular by displaying it on a polar (circular) coordinate system!

BlueSky's menus avoid that complexity by referring to plots and options by common names rather than abstract ones. However, if you wish to learn the code, once you have created the plot you desire, you can then use the History menu to recall the dialog and click the Syntax button to see the ggplot code. That can make learning ggplot code much easier.

5.1 Plots of Counts

One of the most popular measures to plot is counts of categories, and it is good to be aware of the strengths and weaknesses of each type of plot. Scientists have done quite of research into the perceptual accuracy of data graphics, and Elliot provides an excellent summary of their findings [5]. Let us examine four plots using data collected on penguins [8], [13], [14], [15]. This dataset comes from the `palmerpenguins` package, where the dataset is named "penguins" with a lower case "p." I have removed the missing values by using "Data> Missing Values> Remove NAs" (Sec. 4.10.1) and saved it as "Penguins" with a capital "P."

The bar chart shown in Fig. 5.1(a) makes it particularly easy to compare each species' counts. It is easy to look from each bar's height to the y-axis to read off a fairly precise value. It is also easy to compare one bar to another, making it clear that there are more Adelie penguins (red) than Gentoo (blue). You also get a good feel for how many more. However, it is not easy

to figure out the total number of penguins, nor is it easy to get an idea of the percent of each.

One thing that is bad about this bar chart is that the color adds almost nothing to the plot; the bars are clearly labeled! If I had not been comparing these four plots, I would not have used species as a "fill" variable to add that color. It is best to avoid the use of needless color since your audience will waste time trying to figure out what the color means.

The single stacked bar chart shown in Fig 5.1(b) has the opposite set of strengths and weaknesses compared to the bar chart. Estimating the proportion of each species is relatively straightforward, and extracting the total number is now simple. However, reading the number of each species is difficult. For example, to figure out how many Chinstrap birds there are requires reading their value off the y-axis, then subtracting out the number of Gentoos upon which they are stacked.

As I write this, single stacked bar plots are not currently available via menus in BlueSky. However, they are on the to-do list for future versions. In the meantime, you can create one by making a pie chart, pasting the syntax to the program editor, then deleting the line, "`coord_polar("y") +`" (do not forget the trailing plus sign) and then selecting the code and clicking the Run button.

Figure 5.1(c) shows the species counts as a pie chart. Early research showed that people are not very good at estimating the values of angles. From that they concluded that people should avoid using pie charts. However, more recent research has shown that pie charts are good choices if showing proportion is your goal. As with the filled single bar, pies are not a good choice if your readers need to read values for each slice. Even reading the total count of penguins takes your eye in a circular route that we are not used to reading.

Figure 5.1(d) is the Coxcomb plot. It is like a pie chart, except that every angle is the same. It is the radius of each slice that determines the count.

5.2 Setting the Order of Bars, Boxes, Etc.

In many graphics software packages, to change the order of, say, bars in a bar plot, or boxes in a box plot, you would check a box for ascending or descending order on the plot dialog itself. However, BlueSky uses the R language, which stores the orders of factor (category) values inside the variables themselves. Therefore, to determine the plot order, you set the factor levels' order before you do the plot. While this may sound like more work, the order also applies to all output in table form too, which is convenient. When you adjust to this perspective, you set the factor order you prefer when you first read in your dataset and rarely think of it again. If you do change your mind later, BlueSky offers a complete set of factor re-ordering items.

The chart shown in Fig. 5.2(a) has the bars in alphabetical order. That is how factors order their values by default. To create Fig. 5.2(b), I used

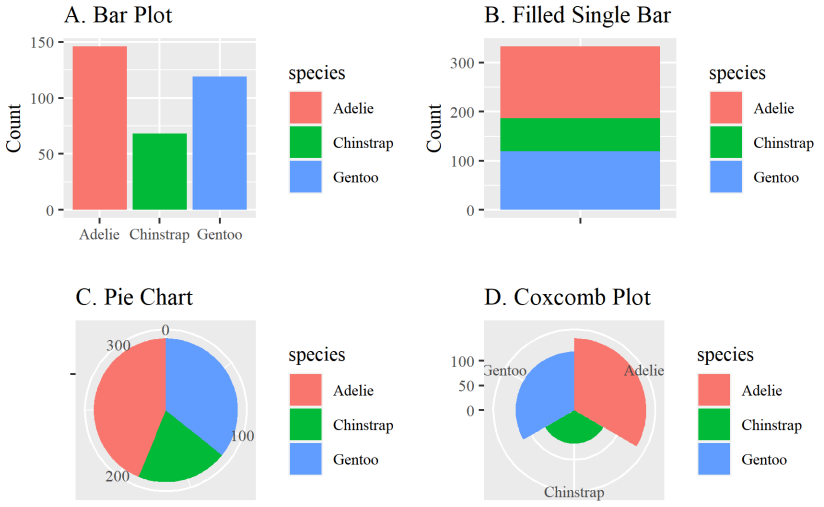


Fig. 5.1: Comparison of bar, filled bar, pie, and bullseye charts. all four show the same counts of penguin species.

"Data> Factor Levels> Reorder by Count" to sort the species levels by their *descending* counts. Similarly, to create Fig. 5.2(c) I used "Data> Factor Levels> Reorder by Count" again, but that time I chose *ascending* counts.

For more details regarding the control of factor levels, see Sec.4.9.

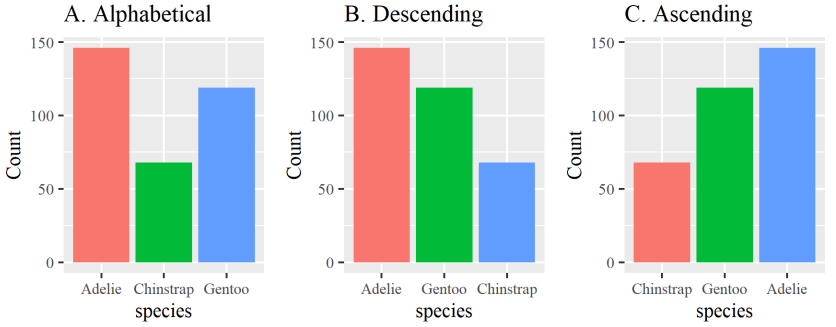


Fig. 5.2: Demonstration of how factor level order determines bar order.

5.3 Graphics Settings & Themes

You can set the sizes of graphs in BlueSky in three ways. The quickest is the menu item "Image Size> Change Graphic Size." It then prompts you for the height and width, which are both 600 by default. The changes will not affect past plots, but you can use the History menu to quickly recall and rerun any previous plot as described in Chap. 1.5. You can return to the original settings by choosing "Image Size> Set Default Graphic Size." That change is immediate (i.e. no dialog prompts you), so you do not have to remember the original settings.

The second way to set the size of plots is by using the item "Tools> Configuration Settings> Image." Section 11.4.3 covers the Settings menu in greater depth.

The third way of setting image size to click on the colorful paintbrush icon labeled "Themes" on the top right of the Application window, next to the "Score Current Dataset" box's title. Figure 5.4 shows the Themes dialog. There you can set image size, but you can also control many other aspects of plot style. While many of those settings apply only to graphs created by the `ggplot2` package, that is what BlueSky uses for nearly all of its plots.

By default, plots use a gray background with white grid lines, as seen in the top left of Fig. 5.3. To change the theme, click on the Themes icon. There you can control the appearance and location of the text used in titles and axis labels. You can also choose the overall style of the graphs by selecting a theme. For example, if you wanted to submit a paper to the Economist magazine, you could choose "theme_economist," and it would set the entire look of the plot to match their specifications.

Figure 5.4 shows four popular themes. The top row shows the default theme, called theme_grey, on the left, and theme_bw on the right. The bottom row shows theme_minimal on the left and theme_hc (hc = horizontal cross-lines) on the right. A comprehensive set of themes available on the website, https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/. You can also create your own custom themes, though that requires the use of R code.

When you have chosen the option settings you like, click the Save button, and these settings will remain in effect for all future graphs.

5.4 Graph Titles, Labels, & Options

Each graphics dialog has an Options button. It opens a dialog with entry fields for the plot's title and for labels of the x- and y-axes. Many plots offer a default title. If you wish to suppress that, simply enter a blank space as the title.

A few plots offer other graphics options, so it is always good to click that button to see what else it might contain.

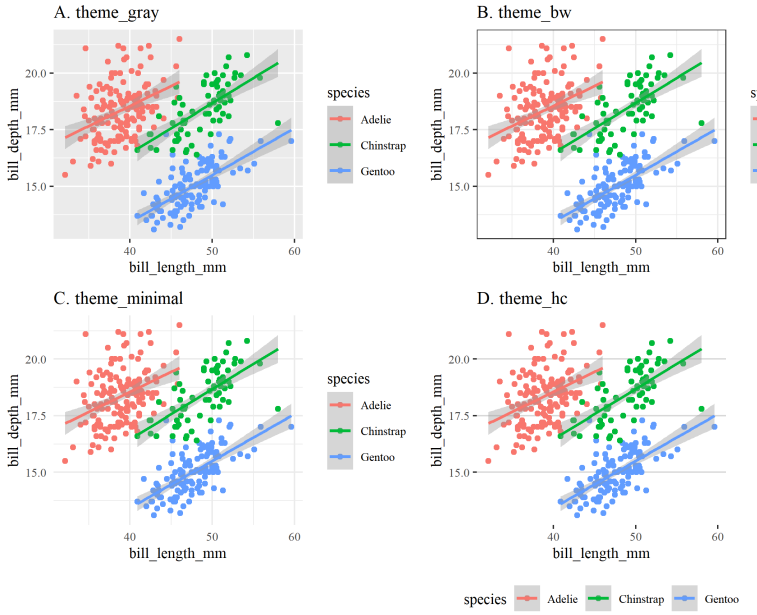


Fig. 5.3: Same plot done with different themes. Note how the legend appears at the bottom of theme hc, matching the overall plot's horizontal look.

5.5 Facets

Most graphics dialogs also have a Facets button. Facets allow you to create "small multiples" of graphs, such as Fig. 5.5. There, the overall plot is one of survival on the x-axis. I used the Facet dialog to add sex as a column facet, and pclass (passenger class) as a row facet. That split the overall plot into the six facets that make up the two-dimensional grid, as shown. Since pages are generally longer than they are wide, it is common practice to put the factor with the greater number of levels in the row position. Of course, it is also possible to use just one factor in the row or column position, creating a one-dimensional grid.

An alternative to row/column faceting is to use a wrapping style of facet, as shown in Fig. 5.6. There, rather than setting a factor in a row or column position, a single factor with many levels is used. Its values are allowed to wrap the small multiple plots into both rows and columns. When the wrapping moves to a new line, it appears to be a two-dimensional grid, but this style of faceting only works with a single factor.

5.6 Axes: to Free or Not to Free?

One of the most useful aspects to facets is that the process fixes the axes; they are the same across all the plots. That makes visual comparisons easy.

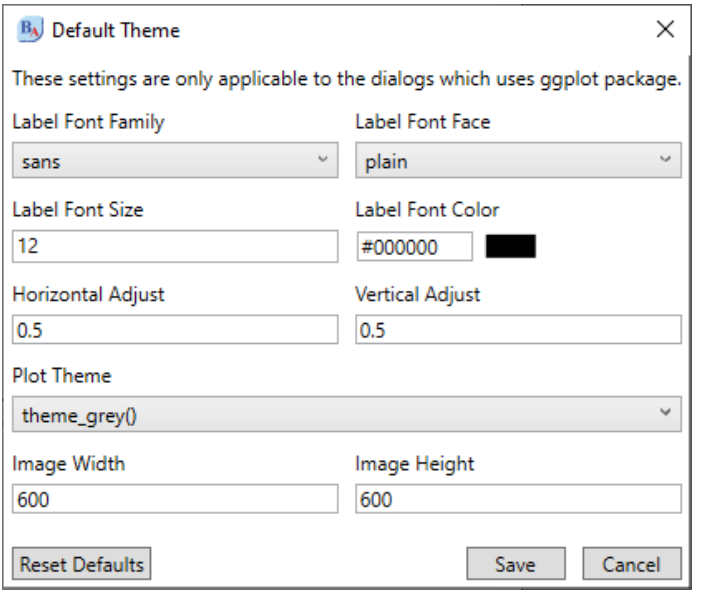


Fig. 5.4: The Themes Dialog.

It also saves room since, as shown in Fig. 5.6, only it only uses one y-axis for each row of facets.

However, there are times when each facet's scales need to be "freed" to be independent. In the Titanic example, imagine that in third class, females' survival advantate had been much greater than in the other classes, but the numbers for both sexes were tiny. With fixed axes, the bars would have been so small that you could not even read their counts. In that case, clicking the Facets button and choosing "Facet Scale: free_y" would allow each row to have an independent y-axis. Figure 5.7 is the same plot as Fig. 5.5, but with the y-axes freed. Note how there is now one bar in each row that completely fills the vertical space on the plot. The axes no longer all stop at 300. This makes it easier to read the values of the smaller bars, but it makes row-to-row comparisons more difficult.

5.7 Missing Values

BlueSky includes missing values in its plots by default. It is generally a good idea to include them in your initial plots to see if there is any particular pattern to them. For example, if you faceted a scatter plot of X and Y by sex and saw that the pattern for the missing values of sex tended to match one or the other genders, that could be useful information.

To eliminate missing values from your plots, you can remove rows containing missing values using the approach described in Sec. 4.10.1. Rather than removing them across all variables, you might first choose to copy the subset of variables that you plan to graph, as shown in Sec. 4.30. That way,

you only lose some of the rows that you are plotting, instead of also losing rows that have missing values on variables irrelevant to the plot at hand. You can also replace the missing with imputed estimations, as shown in Sec. 4.10.5.

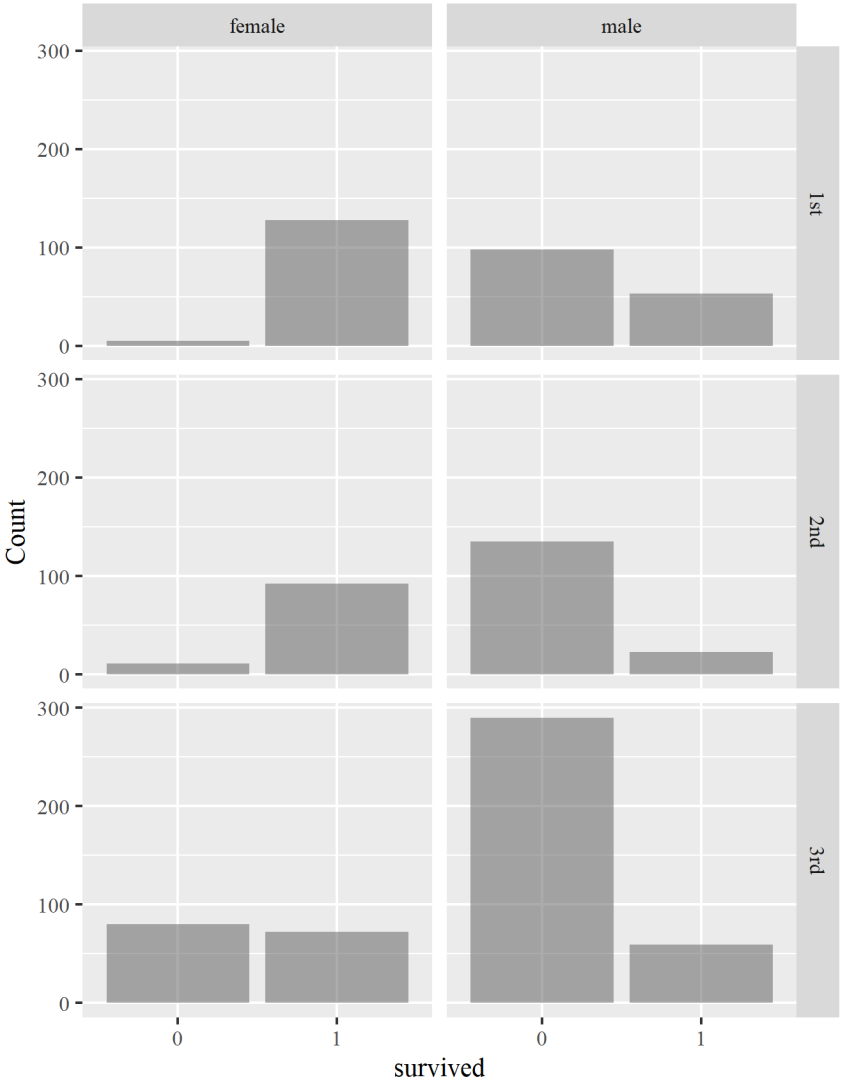


Fig. 5.5: Example of a plot faceted by sex and passenger class. Note how the standardized y-axes make comparisons easy.

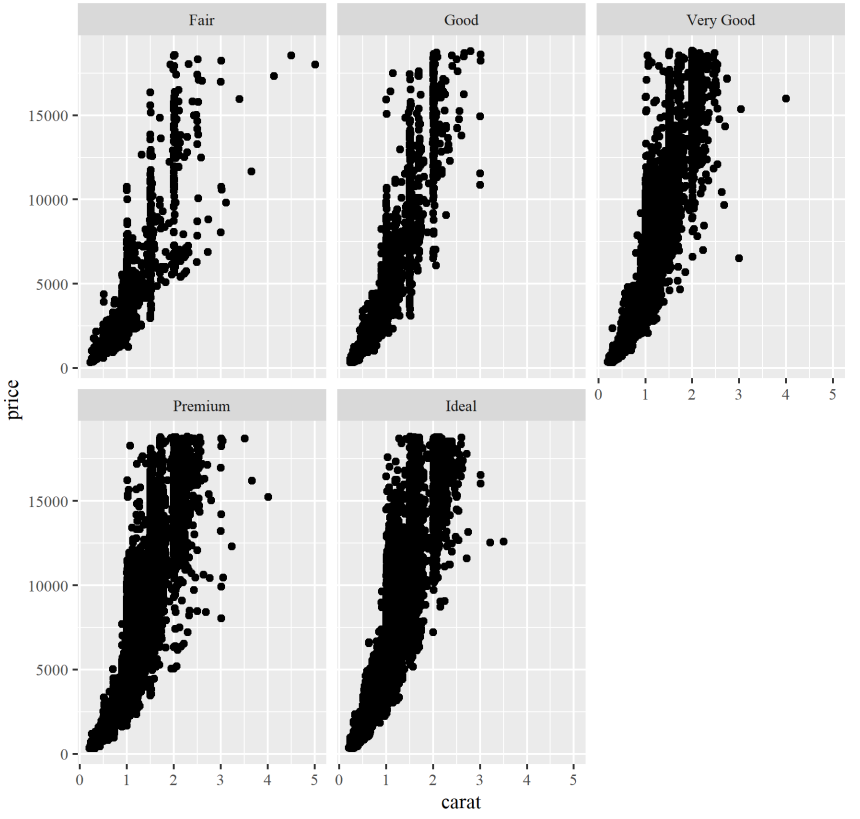


Fig. 5.6: Example of a plot faceted by wrapping. Although there appear to be two dimensions, it uses only one factor, each diamond's cut.

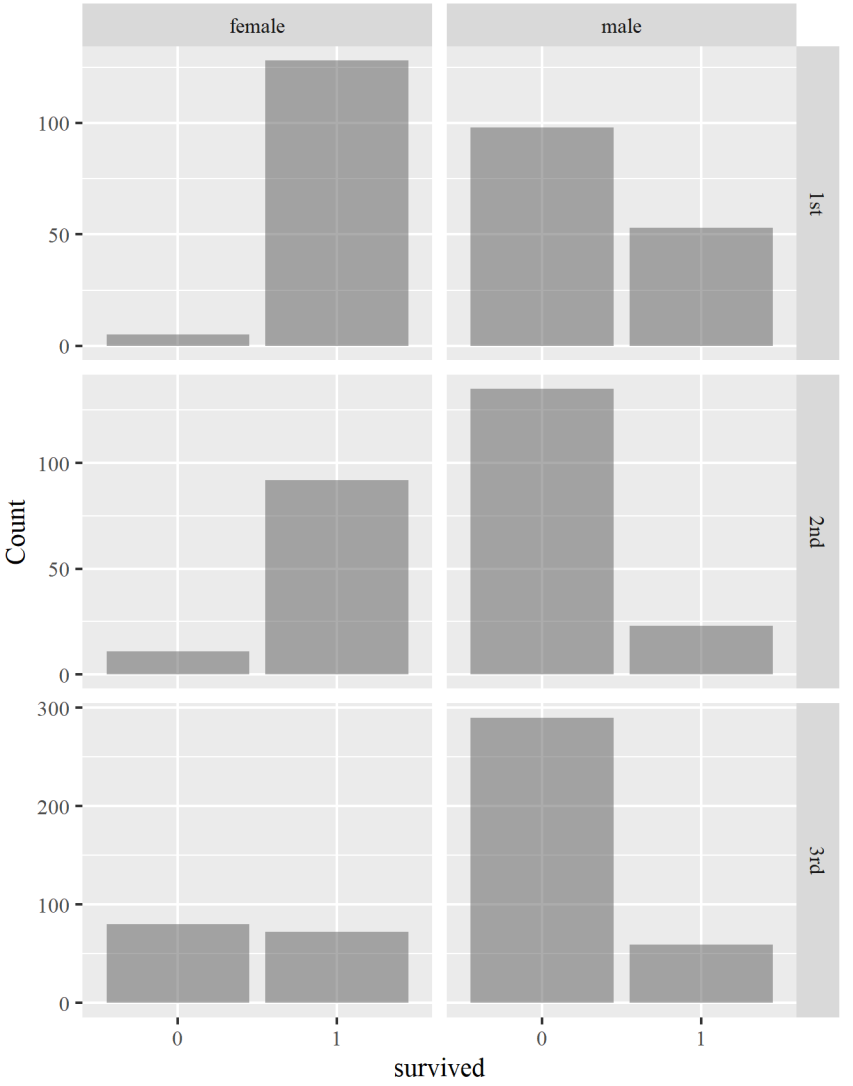


Fig. 5.7: Example of a faceted plot with y-axes scales freed up to be independent. This is the same data shown in Fig. 5.5. Note how there is one bar in each row that completely fills the vertical space. However, now, if you wish to compare rows, you must study each y-axis carefully!

5.8 Controlling Graph Creation

The previous sections in this chapter provide the background that applies to the creation of items on the Graphics menu. After learning that material, creating any particular graph is a simple matter of choosing its dialog and

filling in the parameters. The *BlueSky Statistics 7.1 User Guide*[17] shows an example of each graph type and the steps required to create it. This *Intro Guide* skips those examples to reduce this book's cost.

6

Analysis Menu

6.1 Overview

BlueSky's Analysis menu contains all analytic methods that do not require you to specify model details. If you need to build models which include manually specifying details like interactions, nesting, random vs. fixed effects, and the like, you will find those under the Model Fitting or Model Tuning menus.

6.1.1 One-tailed vs. Two-tailed Tests

Some tests allow you to choose a one-tailed alternative hypothesis that specifies a direction. Rather than the means of two groups being equal or not, they specify that one of the groups has a larger mean. Rather than specify that a correlation is not zero, they might specify that the correlation is positive.

When BlueSky offers the option of a one-tailed test, the resulting p-value will be cut in half, making it much easier to get a significant result. However, if the outcome does not meet the hypothesized direction, your p-value will become insignificant even if it would have been *very* significant had you done a two-tailed test! Some BlueSky analyses offer multiple directions on its p-values, such as "greater than" and "less than," while others offer p-values that are simply labeled "one-tailed." In the latter case, it is up to you to verify that the outcome does indeed meet the assumed direction. If not met, then the result is *not* significant regardless of how small the p-value is!

Because of the temptation to change the directions of hypotheses after the fact, journals may be reluctant to accept results that become significant only when doing a one-tailed test. Of course, if prior results have already demonstrated that direction, they should be fine with a one-tailed test.

6.1.2 Paired vs. Non-Paired Tests

A common mistake is to analyze a paired test as an independent samples one. That almost always leads to a great loss of power, making it much

harder to get a significant result. The opposite error is much less likely. It would entail analyzing independent-samples data as a paired t-test when there is no appropriate way to pair them.

6.1.3 Parametric vs. Non-Parametric Tests

Parametric tests, such as t-tests, analysis of variance, and analysis of covariance assume the variables being averaged are samples from a normal distribution. You can assess that visually using density plots, histograms, Q-Q plots, or P-P plots. If you prefer a hypothesis test of normality, see the Shapiro-Wilk test in Sec. 6.13.7. Those sections all examine the "extra" variable from a sleep study that used drugs to attempt to get people additional sleep at night. That variable is approximately normally distributed.

When variables are not normally distributed, there are several approaches available:

- Transforming Variables – taking the logarithm or square root of the variable may give it a normal distribution.
- Non-Parametric Analysis – these methods assume that the groups you are comparing have distributions that are the same, except for their "location." That is, the distribution for each group can have a different mean or median. Non-parametric tests include tests for independent groups, such as the Wilcoxon-Mann-Whitney (2 groups, Sec. 6.10.4) and the Kruskal-Wallace (3 or more groups, Sec. 6.10.3); and for repeated measures such as the Wilcoxon signed-rank test (2 measures, Sec. 6.10.6), and Friedman test (3 or more measures, Sec. 6.10.2).
- Generalized Linear Models – these tests assume a specific distribution, such as the Poisson or Negative Binomial distributions for count data. An example is the number of fish you might catch in an hour (Poisson is French for fish). Section 8 covers those types of models.

6.2 Agreement Analysis

Agreement between two variables is a broad concept that is measured and visualized in many different ways. Biologists may have instruments that measure with great precision. In contrast, social scientists may have categorical measures derived from raters who "code" the concepts that people discuss during interviews. Luckily, BlueSky covers the whole gamut!

6.2.1 Bland-Altman Plots

The "Analysis> Agreement> Bland-Altman Plots" menu is aimed at comparing two continuous numeric measures of the same underlying value. It assumes that neither is the "true" value but rather that such a value exists somewhere in between the two measures. One of the measures often has a long track record and is often considered a "gold standard." The other may be a new measure, usually one that is quicker or cheaper to apply.

With such a pair of measures, a person's first thought is to calculate a Pearson correlation coefficient, but that is not a very effective approach. Two variables can have a high correlation despite the fact that they yield means that are thousands of units apart!

Bland-Altman plot instead average's the two measures for the x-axis, then plots it against the measures' difference on the y-axis. For example, Bland and Altman's initial paper on this subject [3] compared two methods of measuring Peak Expiatory Flow Rate (PEFR), the maximum rate at which a person can blow air out of their lungs. That rate is indicative of several medical conditions, such as asthma. One measure was taken using a Wright peak flow meter and the other with a mini Wright meter. Let us recreate their plot using the following steps:

1. Use "File> Open" to open the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \bland.altman.PEFR.1986.RData."
2. Choose "Analysis> Agreement> Bland-Altman Plot."
3. Move the variable "WrightFirst" into the "Observer 1 Value" field.
4. Move "MiniWrightFirst" into the Observer 2 Value" field.
5. If you like a gray background, click the "Options" button and choose "theme_grey()", then click OK.

The first table of output contains the mean difference and 95% confidence intervals (Tab. 6.1). Since that output contains zero, we cannot reject the hypothesis that the mean difference is zero.

The second output table, Tab. 6.2 confidence intervals for the mean difference lines. These are useful when computing reference ranges for new diagnostic tests.

Figure 6.1 shows the resulting plot. The line showing zero difference is shown in solid black. The actual mean difference is the dotted line just below the zero line. The wider dashed lines represent a 95% confidence interval around the actual mean difference. Since the points vary randomly around the zero line, we conclude that there is not much difference between the two measurements. Had many of the points fallen outside the confidence intervals, we could have used that as a rough hypothesis test and concluded that there is evidence that the two variables are measuring different things, or perhaps the same thing, but on a different scale.

The Options dialog shown in Fig. 6.2 allows you to control the plot's title and axis labels, as well as the color and style of lines on the plot. You can plot confidence intervals (labeled CIs) for the mean and the standard deviations, though adding them makes the plot quite messy.

The statistics used for this procedure are calculated by the `BlandAltmanLeh` package's `bland.atlman.stats` function. The plot itself is done by the `ggplot2` package's `ggplot` function.

6.2.2 Cohen's Kappa

When measuring the agreement between two raters who are applying a categorical scale, accuracy, i.e., the simple percent of agreement, might be

Mean Difference Lines for WrightFirst and MiniWrightFirst		
lower.limit	mean.diffs	upper.limit
-78.0973	-2.1176	73.862

Table 6.1: Mean difference lines.

lower.limit.ci.lower	lower.limit.ci.upper	mean.diff.ci.lower	mean.diff.ci.upper	upper.limit.ci.lower
-112.6191	-43.5755	-22.0488	17.8135	39.3402

Table 6.2: 95% confidence intervals for mean difference lines.

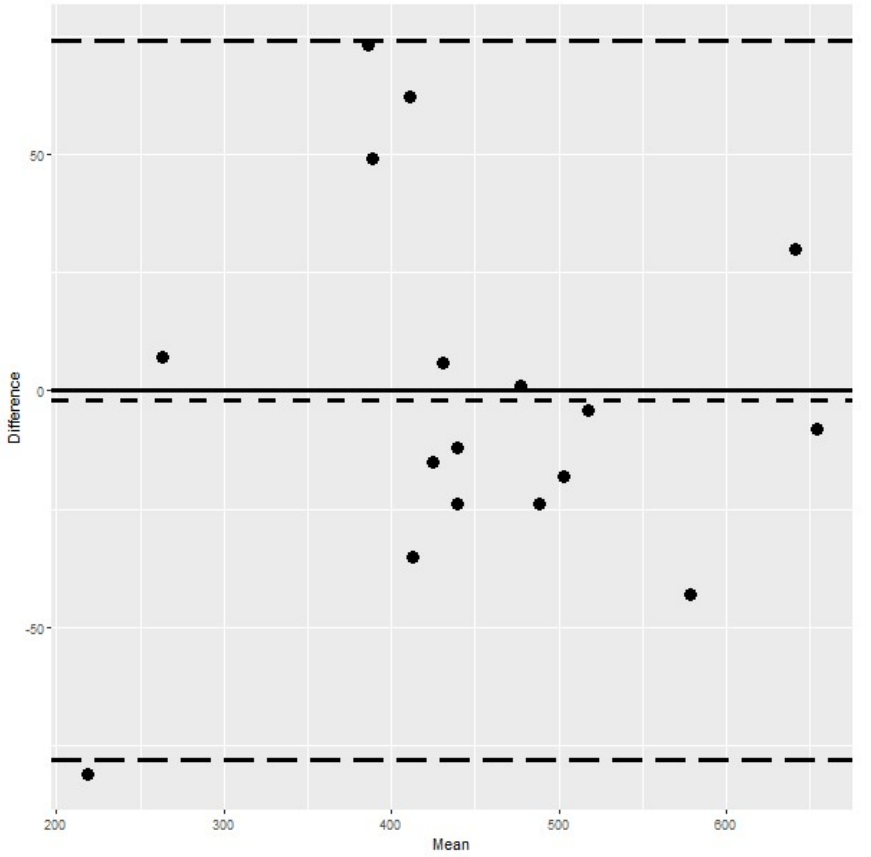


Fig. 6.1: Bland-Altman Plot. The mean of the two measures is on the x-axis, and the difference between the two is on the y-axis.

your first thought. However, that does not take into account the impact of chance. In particular, when assigning ratings to only two categories, when one has many more instances, simply assigning all cases to that large category can yield a very high percentage of agreement. Cohen's Kappa

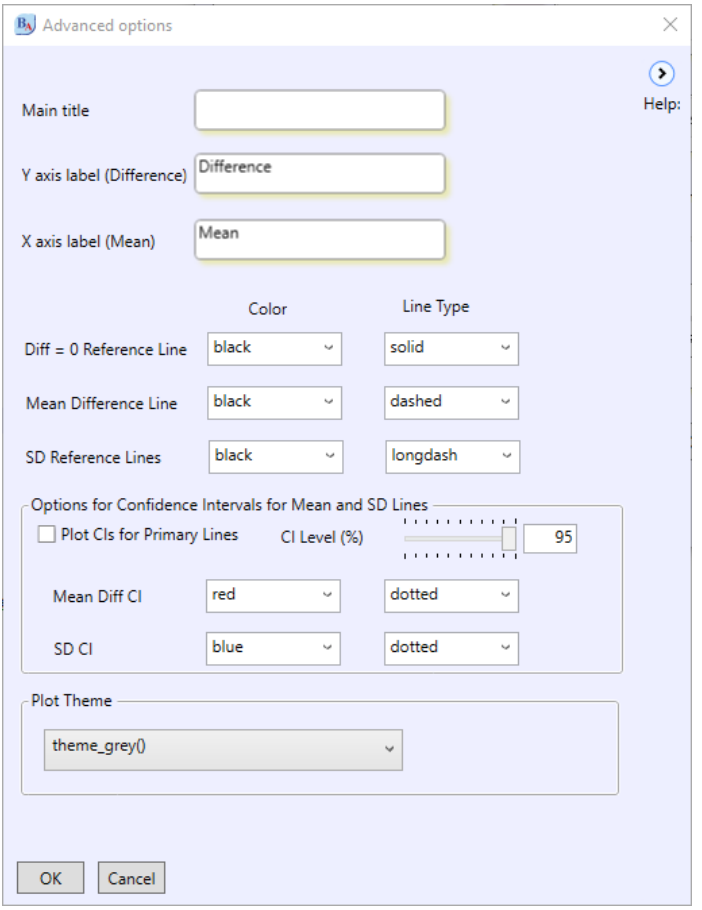


Fig. 6.2: Options dialog from "Analysis> Agreement> Bland-Altman Plot".

Table 6.3: Landis and Koch's recommendations on Cohen's Kappa.

Cohen's Kappa	Interpretation
0.81 to 1.00	Almost perfect agreement
0.61 to 0.80	Substantial agreement
0.41 to 0.60	Moderate agreement
0.21 to 0.50	Fair agreement
0.00 to 0.20	Slight agreement
Less than zero	Poor agreement

corrects the percent of agreement for the effects of chance. Table 6.3 shows the interpretation of Kappa values, as described by Landis and Koch [10].

Let us look at an example where psychologists are diagnosing their patients. How well do they agree? We will use the diagnoses dataset from the `irr` package. Let us perform an analysis using the following steps:

Table 6.4: Crosstabulation of ratings.

Rater 1	Rater 2 Depression	Personality Disorder	Schizo.	Neurosis	Other	Total
Depression	7	1	2	3	0	13
Personality D.	0	8	1	1	0	10
Schizophrenia	0	0	2	0	0	2
Neurosis	0	0	0	1	0	1
Other	0	0	0	0	4	4
Total	7	9	5	5	4	30

Table 6.5: Kappa statistics for rater one and rater two.

Kappa Statistics and CIs (0.95 level): rater1 vs rater2

	Kappa	Std Err	Lower	Upper
Unweighted	0.6512	0.0997	0.4558	0.8465
Weighted (Equal-Spacing)	0.6331	0.1194	0.3991	0.8671

1. In the Analysis window, use “File> Load Dataset from a Package” and enter “irr“ in the package name field, and “diagnoses” into the dataset field.
2. Choose “Analysis> Agreement> Cohen’s Kappa.”
3. Place the variable rater1 in the Observer 1 box.
4. Place the variable rater2 in the Observer 2 box.
5. Check the box for Agreement Plot and Bangdiwala statistics.

The first table of output contains a crosstabulation of the two raters, similar to Tab. 6.4. That particular table was edited to get it to fit on the page. If our two raters agree, we should see a pattern of large numbers on the diagonal cells in the table. For some, such as depression and personality disorder, there is high agreement. For others, such as schizophrenia and depression, the raters do not agree as often.

The Kappa statistic comes next, as shown in Tab. 6.5. The overall measure is 0.65 for unweighted and 0.63 for the weighted values. Referring to Tab. 6.3 we would rate this as substantial agreement.

Table 6.6 shows another measure of agreement, the Bangdiwala statistic. That measure is much less popular than Kappa, but it is particularly helpful in visualizing how the various categories impact the overall agreement. Figure 6.3 displays an agreement plot based on the Bangdiwala statistic. The large squares for depression, personality disorder and other indicate that those categories contribute quite a lot more to agreement than do the schizophrenia and neurosis ratings.

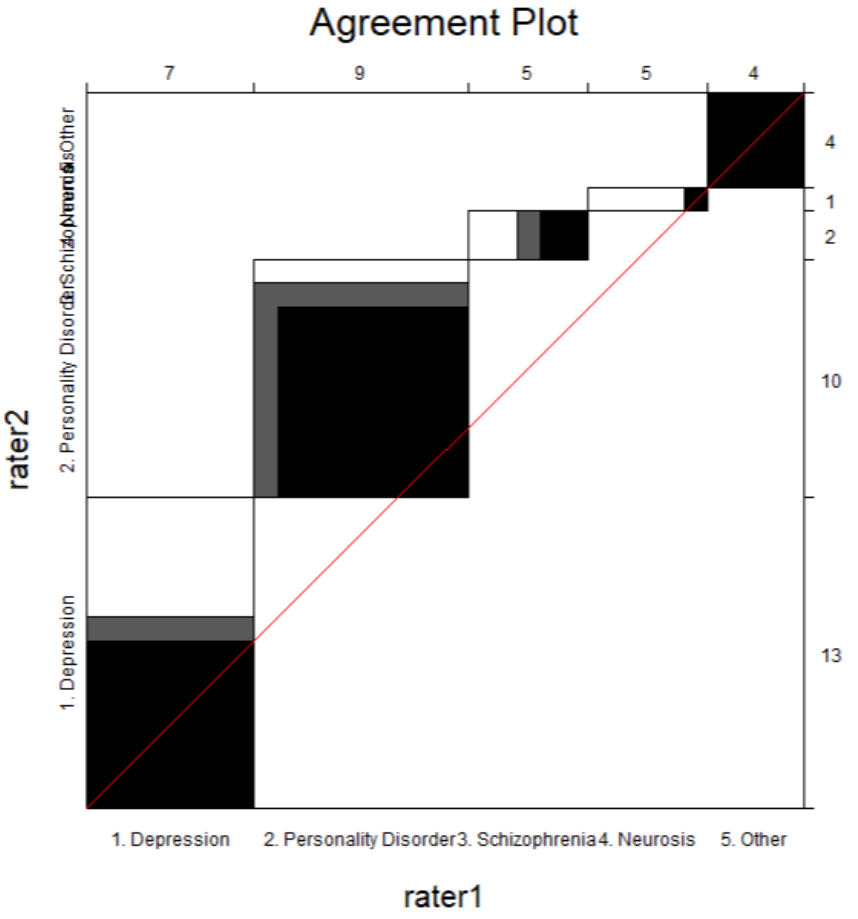
BlueSky performs these calculations using the `vcd` package’s `Kappa` function.

Table 6.6: Bangdiwala statistics for rater 1 and rater 2.

Bangdiwala Statistics

Bangdiwala	Bangdiwala_Weighted	weights1	weights2
0.6321	0.7471	1	0.9375

Fig. 6.3: Agreement plot for rater 1 and rater 2.



6.2.3 Concordance Correlation Coefficient

The Concordance Correlation Coefficient (CCC) is a measure that combines both precision and accuracy to determine how far from perfect agreement two continuous measures are. In this case, perfect agreement is viewed as a 45° diagonal line when plotting the two measures against one another.

As an example, let us examine the same data that we used in the Bland-Altman plot, Sec. 6.2.1.

1. Use "File> Open" to open the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \bland.altman.PEFR.1986.RData."
2. Choose "Analysis> Agreement> Concordance Correlation Coefficient."
3. Move the variable "WrightFirst" into the "Observer 1 Value" field.
4. Move "MiniWrightFirst" into the Observer 2 Value" field.
5. If you like a gray background, click the "Options" button and choose "theme_grey()", then click OK.

The output shows the number of observations and missing values, then gives the CCC estimate of 0.98 with a 95% confidence interval ranging from 0.9522 to 0.9934. That interval does not contain zero so we can reject the hypothesis that CCC is zero.

It also displays a scatterplot of the data with a 45° diagonal line showing perfect agreement (not shown).

BlueSky uses the `epiR` package's `epi.ccc` function to do these calculations.

6.2.4 Diagnostic Testing

The "Analysis> Agreement Analysis> Diagnostic Testing" item lets you compare the effectiveness of a test in various ways. It uses two binary (zero or one) variables, with the outcome being the true diagnosis and the test being the diagnosis. The diagnosis could be a medical test, or it could be the predicted classification from a model. In the case of a model, these measures are also available when scoring a model, as shown in Sec. 10.

Let us calculate these measures:

1. Use "File> Open" to open the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \epiR_example1.RData."
2. Choose "Analysis> Agreement Analysis> Diagnostic Testing."
3. Place the variable disease in the "Outcome Variable" box.
4. Place the variable test in the "Test Variable" box and click OK.

Table 6.7 is the first piece of output. It counts the combinations of the actual disease status and the test outcome. The many measures of test success appear next in Tab. 6.8. Table 6.9 contains the formulas for most of these measures. Additional information is available on Wikipedia here: https://en.wikipedia.org/wiki/Sensitivity_and_specificity and here: https://en.wikipedia.org/wiki/Likelihood_ratios_in_diagnostic_testing.

The `epiR` package's `epi.tests` function calculates the computations in this section.

Table 6.7: Diagnostic crosstabulation. Test=test; Outcome=disease.

	Outcome +	Outcome -	Total
Test +	670	202	872
Test -	74	640	714
Total	744	842	1586

Table 6.8: Diagnostic testing statistics.

	est	lower	upper
Sensitivity	0.9005	0.8767	0.9211
Specificity	0.7601	0.7298	0.7886
Positive Predictive Value (PPV)	0.7683	0.7389	0.796
Negative Predictive Value (NPV)	0.8964	0.8716	0.9177
Diagnostic Accuracy	0.826	0.8064	0.8443
Positive Likelihood Ratio (LR +)	3.7537	3.3207	4.2432
Negative Likelihood Ratio (LR -)	0.1309	0.1051	0.163
Number Needed to Diagnose	1.5137	1.4091	1.6487
Youden's Index	0.6606	0.6065	0.7097
Proportion Outcome Ruled Out	0.4502	0.4255	0.4751
Proportion Outcome Ruled In	0.5498	0.5249	0.5745
Proportion False Positive	0.2399	0.2114	0.2702
Proportion False Negative	0.0995	0.0789	0.1233
Diagnostic Odds Ratio	28.6861	21.5182	38.2417
Apparent Prevalence	0.5498	0.5249	0.5745
True Prevalence	0.4691	0.4443	0.494

6.2.5 Fleiss' Kappa

Fleiss' Kappa is a measure of agreement among multiple judges that who classify observations (e.g., people) into categories (e.g., paranoid, schizophrenic) that corrects for the effects of chance. It is a generalization of Scott's pi, which applies to only two raters.

1. Use "File> Open" to open the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \epiR_example1.RData."
2. Choose "Analysis> Agreement Analysis> Fleiss' Kappa."
3. Place the variables disease and test in the "Rater Variables" box, and click OK.

Table 6.9: Diagnostic testing formulas.

Measure & Aliases	Formula
Proportion of true positives	TP
Proportion of true negatives	TN
Proportion of false positives	FP
Proportion of false negatives	FN
Sensitivity, a.k.a. recall, the hit rate or true positive rate	$TP/(TP+FN)$
Specificity, a.k.a. selectivity or the true negative rate	$TN/(TN+FP)$
Positive Predictive Value	$TP/(TP+FP)$
Negative Predictive Value	$TN/(TN+FN)$
Diagnostic Accuracy	$(TP+TN)/(TP+TN+FP+FN)$
Positive Likelihood Ratio	$Sensitivity/(1-Specificity)$
Negative Likelihood Ratio	$(1-Sensitivity)/(Specificity)$
Number Needed to Diagnose	$1/Youden's Index$
Youden's Index	$Sensitivity+Specificity-1$
Diagnostic Odds Ratio	$(TP/FN)/(FP/TN)$

Table 6.10: Landis and Koch's recommendations on Fleiss' Kappa.

Fleiss' Kappa	Interpretation
0.81 to 1.00	Almost perfect agreement
0.61 to 0.80	Substantial agreement
0.41 to 0.60	Moderate agreement
0.21 to 0.40	Fair agreement
0.01 to 0.20	Slight agreement
Less than zero	Poor agreement

Table 6.11: Fleiss' Kappa measure of agreement.

Raters	N	Fleiss_kappa	std.err	ci.level	lower	upper
2	1586	0.6518	0.0191	0.95	0.6145	0.6892

The output appears in Fig. 6.11. The Kappa value is 0.65, and the 95% confidence interval does not include zero, indicating that we can reject the hypothesis that Kappa is zero. Table 6.10 shows the interpretation of Kappa values, as described by Landis and Koch [10]. By that, we conclude that our test has substantial agreement with the true diagnosis.

The `rel` package's `spi` function perform the calculations in this section.

6.2.6 Intraclass Correlation Coefficient

Intraclass Correlation Coefficients (ICCs) provide measures of reliability that take into account both correlation and agreement among raters. BlueSky can calculate ICCs different ways, depending on whether the rater is fixed (e.g., the Olympic judge is the only one who counts) or randomly chosen, and whether a single rating is used or the average of several judges (more reliable, of course).

1. Use "File> Open" to open the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \sf.RData."
2. Choose "Analysis> Agreement Analysis> IntraClass Correlation Coefficient."
3. Place the variables j1 through j4 in the "Rater Variables" box, check the "model details" box, then click OK.

Table 6.13 shows the six possible ICCs. Choosing the proper one requires careful consideration, as described in the R help file for this function:

> Shrout and Fleiss (1979) consider six cases of reliability of ratings done by k raters on n targets (see the "type" column in the table).
>
> - ICC1 – Each target is rated by a different judge and the judges are selected at random. (This is a one-way ANOVA fixed effects model and is found by (MSB- MSW)/(MSB+ (nr-1)*MSW))

Table 6.12: Intraclass correlation interpretation.

Intraclass Correlation	Reliability Interpretation
.90 or greater	Excellent
0.75 to 0.89	Good
0.50 to 0.74	Moderate
Less than 5.0	Poor

Table 6.13: Intraclass correlation coefficients

	type	ICC	F	df1	df2	p.value	lower bound	upper bound
Single_raters_absolute	ICC1	0.1658	1.7949	5	18	0.1647	-0.1329	0.7226
Single_random_raters	ICC2	0.2898	11.0267	5	15	<.001***	0.0188	0.7611
Single_fixed_raters	ICC3	0.7148	11.0267	5	15	<.001***	0.3424	0.9459
Average_raters_absolute	ICC1k	0.4429	1.7949	5	18	0.1647	-0.8842	0.9124
Average_random_raters	ICC2k	0.6201	11.0267	5	15	<.001***	0.0712	0.9272
Average_fixed_raters	ICC3k	0.9093	11.0267	5	15	<.001***	0.6757	0.9859

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

- ICC2 – A random sample of k judges rates each target. The measure is one of absolute agreement in the ratings. Found as (MSB- MSE)/(MSB + (nr-1)*MSE + nr*(MSJ-MSE)/nc)
- ICC3 – A fixed set of k judges rates each target. There is no generalization to a larger population of judges. (MSB - MSE)/(MSB+ (nr-1)*MSE)

Then, for each of these cases, is reliability to be estimated for a single rating or for the average of k ratings? (The 1 rating case is equivalent to the average inter-correlation, the k rating case to the Spearman Brown adjusted reliability.)

ICC1 is sensitive to differences in means between raters and is a measure of absolute agreement.

ICC2 and ICC3 remove mean differences between judges, but are sensitive to interactions of raters by judges. The difference between ICC2 and ICC3 is whether raters are seen as fixed or random effects.

ICC1k, ICC2k, ICC3K reflect the means of k raters.

Once you have chosen the appropriate ICC type, consider its value and interpret it according to Tab. 6.12.

The `psych` package's `ICC` function performs the calculations for this topic.

6.3 Cluster Analysis

Cluster analysis searches for groups of observations that have similar patterns of measurements across a set of variables. Marketing analysts use this to

reach different types of customers advertisements focused on their attributes. An entomologist might visually inspect a set of beetles that appear to be a single species. Cluster analysis can "see" into high dimensions, and it might find two or more species in the set by focusing on many slight differences simultaneously.

Cluster analysis uses various measures of distance to determine the similarity each pair of observations. Think about just two cases, and only two variables. The differences between the two cases on the two variables form the legs of a triangle. The distance between them is the hypotenuse. You calculate the Euclidean distance between them using the Pythagorean Theorem that we all learned in high school. It is mathematically easy to extend that idea into dozens of dimensions. Some measures can only be integers, which you can envision as the map of city blocks, as seen from above. The distance between two corners is the sum of the lengths of the sides of the blocks that you have to walk to get from one corner to the other. That's called Manhattan or City Block distance.

With distance measures, variables measured using large numbers will affect clustering algorithms than those measured with small numbers. However, a variable with a big mean that does not vary much will not be as helpful in finding clusters as would a variable with a small mean but whose values vary quite a lot. Therefore, when your dataset contains measurements that are on very different scales, it is essential to standardize them. Section 4.13 describes how to do that.

Another weakness of cluster analysis is that variables that are correlated give it trouble. You can check that by calculating correlations, as shown in Sec. 6.5.3. Alternatively, you can use Principal Components Analysis (PCA) to generate an uncorrelated subset of variables. That not only removes correlations, but the resulting component variables are also standardized to have a mean of zero and a standard deviation of one.

Cluster analysis applied to datasets for which the cluster membership is not known. That can make learning about it challenging! Therefore, we will use the iris dataset for which clusters *are* known, to better understand what is happening. Cluster analysis can "see" into many dimensions at once, while we can only easily visualize a couple. Therefore, PCA is good first step to help us visualize clusters. Let us use the principal components from Sec. 6.6.2. Plotting the first two components will let us view the two most important dimensions, as seen in Fig. 6.4. Keep in mind that we usually have no idea of how many clusters might exist, so envision that plot with all black points and it might appear as just two clusters, Setosa and what we know to be a combination of Versicolor and Virginica.

In the sections below, we will use hierarchical clustering and K-means clustering to see how they do in identifying our clusters. While the variables are correlated, we will ignore that problem to make it easier to understand cluster differences.

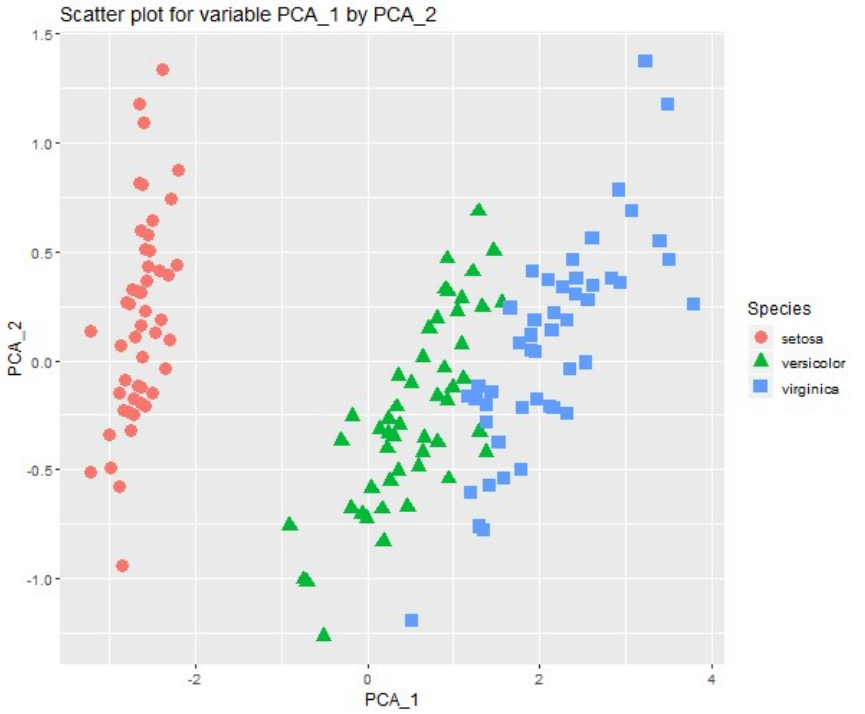


Fig. 6.4: Scatterplot of the first two principal components showing cluster structure.

6.3.1 Hierarchical Cluster

Hierarchical cluster analysis creates a hierarchy of solutions that begins with each observation being in its own cluster. Then, in each following step, it combines each cluster with the one closest to it. As the clusters gain members, the mean of all the member's values is used to compute distance. The position in the space of the mean of those variables is called the cluster's *centroid*. The analysis ends with all the cases in a single cluster. Since there are many solutions to the number of clusters, it is up to you to choose the final number.

Controlling a hierarchical cluster analysis begins with the choice of a distance measure. Euclidean is the most popular and it is BlueSky's default setting. The other parameter to choose is the clustering algorithm. The differences among the various algorithms are beyond our scope, but the default Ward's method is popular due to its effectiveness.

Let us use those defaults and perform an analysis using the following steps:

1. In the Analysis window, use "File> Load Dataset from a Package" and enter the dataset name "iris" into the box on the lower right.
2. Choose "Analysis> Cluster Analysis> Hierarchical Cluster Analysis."
3. Select all four variables except for Species.

4. Set No. of Clusters to "3".
5. Check "Assign clusters to the dataset" and name the cluster variable, "WE3." You often end up trying several algorithms, telling each a different number of clusters to find. This name indicates that the variable stores the cluster membership found by Ward's algorithm (the "W" part), using Euclidean distance (the "E" part) when we asked for 3 clusters.
6. Check "Show cluster biplot" and "Plot cluster dendrogram."
7. Under Options, leave the default methods as "Ward's Method" and "Euclidean" as the distance measure.

When finished, your dialog box should look like the one shown in Fig. 6.5. When you click OK, Tab. 6.14 should appear. The top table shows the number of cases in each cluster. The iris dataset contains 50 of each species, so we know it has not identified species membership completely.

The bottom table in Tab. 6.14 shows the three clusters and their centroids. Studying this table will help us understand the nature of the clusters. We see that cluster 1 has short petals, with a mean of 1.462. Cluster 2 has the narrowest sepals, at 2.76, and cluster 3 has the longest sepals and petals.

The next figure, Fig. 6.7 is called a *(*dendrogram) or tree diagram. It shows the hierarchy from every case being its own cluster at the bottom to all cases being in a single cluster at the top. The y-axis shows the amount of distance between each set of clusters. The 2-cluster solution goes covers a broad range of 50 to 200, while the 3-cluster solution covers only around 20 to 48. This indicates that it found a much stronger case for a 2-cluster solution. The 4-cluster solution covers an even smaller amount of the y-axis.

Figure 6.6 shows its version of the principal component plot that we created previously. The red axis in the middle shows that the most important variables (those on the Comp.1 axis, the first principal component) emphasizes petal width and length. The y-axis is focused on sepal width and a third axis is based on sepal length, though it is not really able to show it on a two-dimensional surface.

Let us compare the cluster membership we saved in our new variable WE3 to the original species variable by choosing "Analysis> Contingency Tables> Crosstab, multi-way." We see in Tab. 6.15 that the analysis correctly identified all 50 of the Setosa and Versicolor flowers. However, the Virginica ones were confused with Versicolor.

Why did it not do better? This dataset contains only lengths and widths of flower parts, not any color or shape information. The analysis would likely have done much better with additional information.

BlueSky does hierarchical cluster analysis using the `BlueSky` package's `BSkyHierClus` function which in turn uses the `stats` packages `hclust` function.

6.3.2 K-Means Cluster

While hierarchical cluster analysis, covered in Sec. 6.3.1, does a good job at identifying groups of related cases in a dataset, it takes a very long time to calculate on large datasets. A popular faster algorithm is K-means cluster.

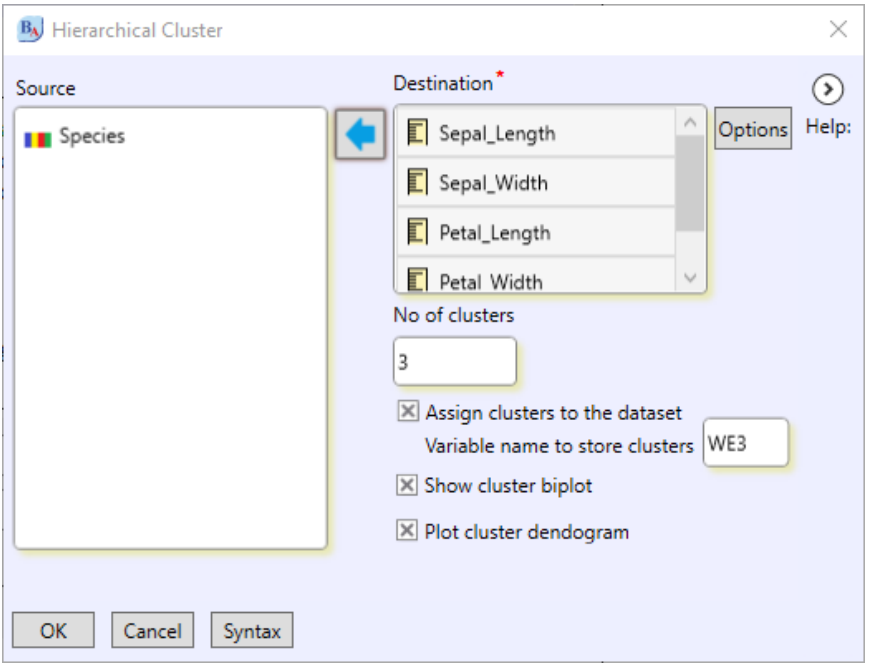


Fig. 6.5: Dialog for hierarchical cluster analysis using Ward's method and euclidean distance.

Table 6.14: Hierarchical Cluster output, first two tables.

Overview

	Cluster 1	Cluster 2	Cluster 3
	50	64	36

Cluster centroids

INDICES	FUN			
	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
1	5.006	3.428	1.462	0.246
2	5.9297	2.7578	4.4109	1.4391
3	6.8528	3.075	5.7861	2.0972

It lets you request a particular number of clusters and does the calculations only on that number.

In our previous example, using the iris dataset, we could ask for a 3-cluster solution, and the K-means algorithm start by randomly positioning three centroid points in space. It would see which cases were closest and then repeatedly adjust the centroids' locations and the cluster memberships, seeking a good solution. When further shifting does not decrease the sum

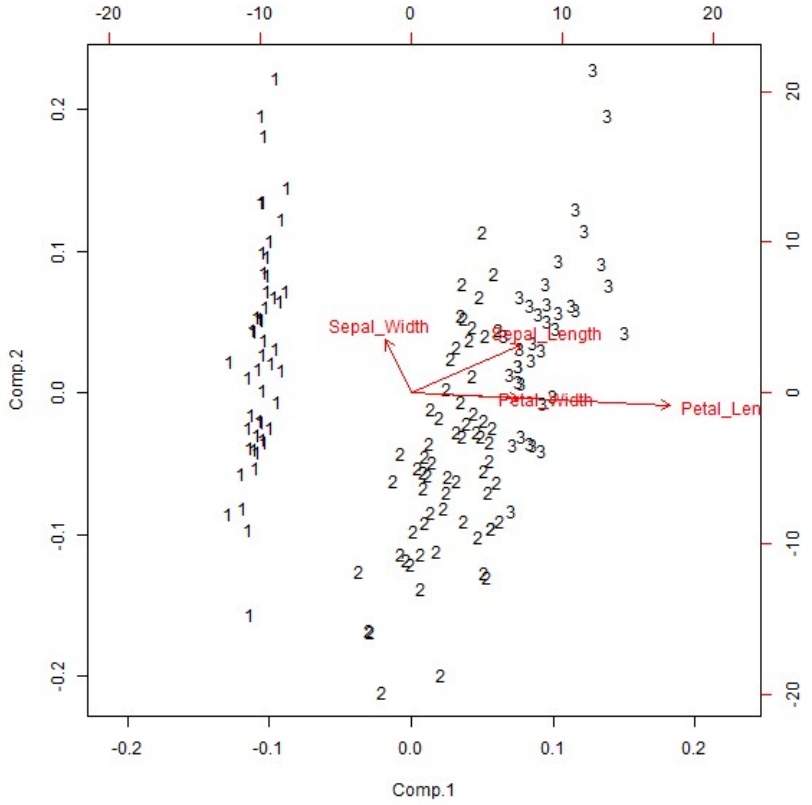


Fig. 6.6: Hierarchical cluster analysis biplot.

Table 6.15: Hierarchical cluster analysis crosstabulation of our hierarchical solution to the original species variable.

		WE3			
		1	2	3	Total
Species	setosa	50	0	0	50
	versicolor	0	50	0	50
	virginica	0	14	36	50
Total		50	64	36	150

of squared distances within each cluster very much, it stops. While there are several variations of this approach, the Hartigan–Wong algorithm is considered the best and is the method used (see the help file for details).

Since it uses random starting points, each time you run it, you can get a different answer (hopefully one that's quite similar). Therefore, the dialog

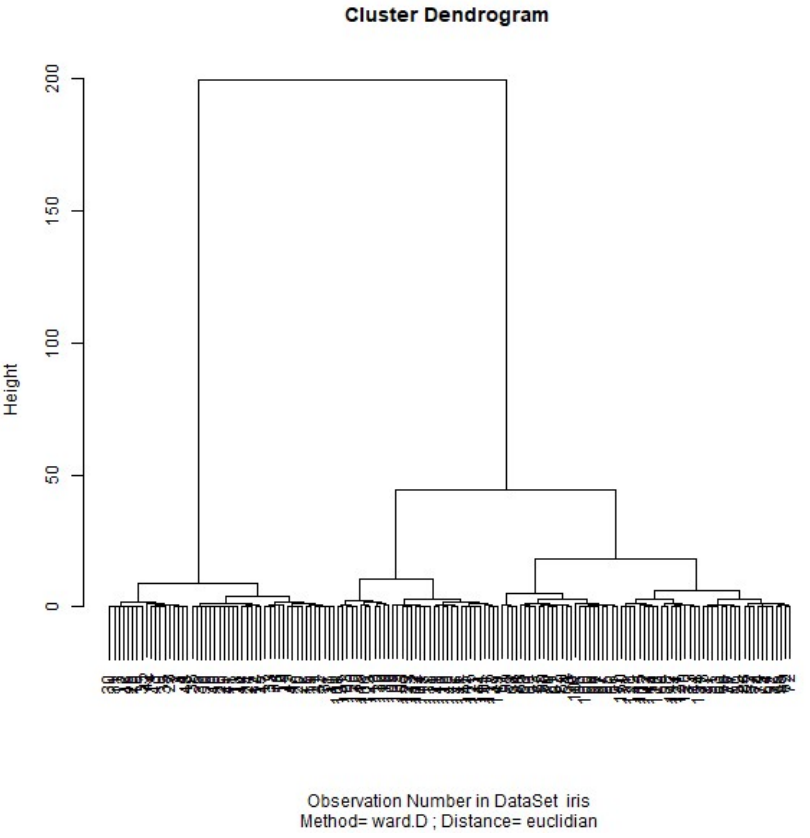


Fig. 6.7: Hierarchical cluster analysis dendrogram. The order of the cases appears on the x-axis but can only be seen with small datasets.

offers a setting for "No. of starting seeds," which is set to 10. It will try that many random starting points, and show the solution that finds the best fit.

Another setting is "Maximum iterations." That too is set to 10, and it lets the algorithm move the centroids and cluster memberships around ten times within each of the starting seed settings. If you get a "failure to converge" message, increase that value until it completes its work. If even a large number of iterations, such as 10,000, does not get it to converge, then there may not be a clear solution for the number of clusters you are requesting.

Here are the steps to follow for the iris dataset we have been using:

1. If you have not already done so, load the iris dataset by choosing "File> Load Dataset from a Package" and entering the dataset name "iris" into the box on the lower right.
2. Choose "Analysis> Cluster Analysis> Hierarchical Cluster Analysis."
3. Select all four variables except for Species.
4. Set No. of Clusters to "3".

Table 6.16: K-means cluster analysis table of centroids, or sets of means, that describes each cluster.

Cluster centroids

Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
5.01	3.43	1.46	0.25
6.85	3.07	5.74	2.07
5.9	2.75	4.39	1.43

Table 6.17: K-Means cluster analysis crosstabulation between species and cluster membership.

		KM3			
		1	2	3	Total
Species	setosa	50	0	0	50
	versicolor	0	48	2	50
	virginica	0	14	36	50
Total		50	62	38	150

5. Check "Assign clusters to the dataset" and name the cluster variable, "KM3." You often try several algorithms, telling each a different number of clusters to find, so this name indicates that the variable stores the cluster membership found by the K-means algorithm when we asked for 3 clusters using Ward's method and Euclidean distance.
6. Check "Show cluster biplot."

The resulting output shows some sums of squares tables) that is not of much interest (not shown).

As with hierarchical cluster analysis, the table of centroids shown in Tab. 6.16 displays each variable's mean for each cluster. The first cluster stands out for having the narrowest petal width. The second has the longest sepal and petal length. The third has the narrowest sepal width.

The cluster biplot it generates is nearly identical to the one shown in Fig. 6.6, and you interpret it the same way; therefore it is not reproduced here.

Let us see how these clusters compare to the known species. Table 6.17 shows that it correctly clustered Setosa, but this time it included the two Versicolors in the cluster with Verginica. The same 14/36 split of Versicolor and Verginica occurred, but this time it reversed the cluster numbers. These numbers are completely arbitrary, so these are the very same clusters with different numeric labels.

BlueSky does hierarchical cluster analysis using the `BlueSky` package's `BSkyKMeans` function, which in turn uses the `stats` package's `kmeans` function.

Table 6.18: List-style contingency table. This output style is especially useful for more than two variables as the list can get much longer without getting too wide to fit on a page.

sex	survived	Freq	cumFreq	freqPercent	cumPercent
female	0	96	96	9.1778	9.1778
	1	292	388	27.9159	37.0937
male	0	523	911	50	87.0937
	1	135	1046	12.9063	100

6.4 Contingency Tables

Contingency tables are basic 2-dimensional tables of counts (as we saw in Tab. 6.17) and, optionally, percents. They are also known as cross-tabulations or simply crosstabs. Excel users might call them pivot tables, though those can be interactive while contingency tables are generally static. Contingency tables are often accompanied by hypothesis tests regarding independence, as well as measures of association such as the odds ratio. For problems that deal with only two values, see also proportion tests in Sec. 6.11.

To demonstrate contingency tables, we will use the Titanic dataset.

1. Go to the File menu and choose "Open."
2. Navigate to the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \Titanic.RData" select it, and click "Open."

6.4.1 Crosstab, list

The "Analysis> Contingency Tables> Crosstab List" item creates tables in the form of a list, as shown in Tab. 6.18. Compare that to the same variables done in a tabular format shown in Tab. 6.19. The list format is especially useful when there are many levels of the factor variables. When there are more than two factors, the list format can still create a single table, while the tabular format ends up with sets of two-way tables.

To create the table shown in Tab. 6.18, prepare the Titanic dataset as shown in Sec. 6.4, then follow these steps:

1. Choose "Analysis> Contingency Tables> Crosstab List."
2. Choose sex and survived as variables for the table.
3. Leave the other settings as they are, and click OK.

Studying the table's frequency column shows that roughly triple the number of females survived as died. The males did not fare so well! Nearly five times as many died as survived. See these figures as percents would have been helpful, but this style of table shows percents and cumulative percents only across all cells of the table.

Other features you can control using the dialog box include:

- Missing Values
  Remove from table – this simply removes all mention of missing values.
  Show and exclude from percentages – this shows the number of missing values but excludes them from percentages.
  Show and include in percentages – this shows the counts and adjusts the percentages to include the number of missing values (not a popular choice).
- Include combinations with zero counts – this shows zeros when there are empty cells, which is more comprehensive but can make some tables much longer.
- Show top N most frequent – this is very helpful when you have many cells with tiny counts that do not interest you. It makes the table focus on just the cells with high counts. An alternative to this is to "lump" sparse factor levels into a category of "other" using the approach shown in Sec. 4.9.5. That would affect all analyses, including contingency tables.

BlueSky creates crosstab lists using the `arsenal` package's `freqlist` function.

6.4.2 Crosstab, multi-way

The most popular way of analyzing the relationship between two factors is the tabular-style contingency table shown in Tab. 6.19. What makes it so popular is the ease with which it allows you to focus on one row (or column) at a time. Looking at the row for females (% within sex), we see that 75% survived, while in the row for males, 79% died. That focus on the larger percent is quicker than looking at the two numbers for survived vs. died and then mentally taking the ratio yourself.

To create the table shown in Tab. 6.18, prepare the Titanic dataset as shown in Sec. 6.4, then follow these steps:

1. Choose "Analysis> Contingency Tables> Crosstab, Multi-way."
2. Choose sex as the row variable.
3. Choose survived as the column variable.
4. The layer variable box would allow factors that would repeat the table above, using passenger class for an example. We will leave this item blank.
5. Click Options, then check row percentages and the Chisq and Fisher boxes; then click OK.

Table 6.20 shows the statistical tests. The Pearson Chi-squared value is 302.76, with 1 degree of freedom, which yields a p-value of 0.0000, or $p < .001$. This is less than the traditional cutoff of 0.05, so we can reject the null hypothesis that sex and survival are independent. Fisher's Exact Test is on the second line, and it yields a comparable p-value. That test is often preferred when there are tiny numbers of counts (e.g., <5) in the cells.

The odds ratio takes the odds of survival for the males (135/523) and divides it by the females' survival odds (292/96) to yield 0.085. That means the odds of survival for males is only 8.5% that of the females. Or you could

Table 6.19: Tabular-style contingency table. This is the more commonly used type.

		survived		
		0	1	Total
female	Count	96	292	388
female	% within sex	24.7423	75.2577	100
male	Count	523	135	658
male	% within sex	79.4833	20.5167	100
	Count	619	427	1046
	% within sex	59.1778	40.8222	100

Table 6.20: Chi-squared test of independence

	Value	df	Asymp. Sig.	Odds ratio	95% Confidence interval	
Pearson Chi Square	302.7586	1	0.0000	0.0849	0.063	0.1144
Fisher's test			0.0000	0.0851	0.0623	0.1156

say that being male decreases the odds of survival by (1-0.085) 91.5%. You can reverse the comparison by inverting the odds ratio: $1/0.085 = 11.8$, so the odds of survival are 11.8 times higher for the females.

If the odds were even for the sexes, then the odds ratio would be 1. The 95% confidence interval for the odds ratio does not include 1, therefore we conclude that we can reject the null hypothesis that the genders had even odds of survival. For more flexibility in odds ratio calculations, see Sec. 6.4.3.

BlueSky creates crosstab lists using the `BlueSky` package's `BSkyCrossTable` function.

6.4.3 Odds Ratios, M by 2 table

The item "Analysis> Contingency Tables> Odds Ratios, M by 2 table" offers more flexibility in calculating odds ratios than the multi-way table discussed in Sec. 6.4.2. To create an example, follow these steps:

1. Choose "Analysis> Contingency Tables> Odds Ratios, M by 2 Table."
2. Choose sex as the Exposure/Predictor variable.
3. Choose survived as the Outcome variable, then click OK.

The first table of output is the simple crosstabulation shown in Tab. 6.21. A glance at that shows most females survived while most males did not, by quite a large margin.

The odds table, shown in Tab. 6.22, starts on the left side by nearly repeating the crosstabulation, which just makes it seem more complicated

Table 6.21: Cross-tabulation from Odds Ratio, M by 2 table.

Predictor	Outcome		
	0	1	Total
female	96	292	388
male	523	135	658
Total	619	427	1046

Table 6.22: Odds ratio from Odds Ratio, M by 2 table.

Predictor	Outcome							
	0	p0	1	p1	oddsratio	lower	upper	p.value
female	96	0.1551	292	0.6838	1	NA	NA	NA
male	523	0.8449	135	0.3162	0.0849	0.063	0.1144	0.0000

than it is. It adds proportions though, which makes it easier to see that 84% of females survived while 68% of males died. The top odds ratio is 1, with missing values (NA) for confidence intervals, which all seems quite odd! This is the female's odds compared to the females, which only makes sense in tables that have more than two rows.

The odds ratio for the males is 0.085, and it is interpreted the same way as described above in Sec. 6.4.2. It also displays the 95% confidence interval, which excludes the even-odds value of 1. The p-value of 0.0000 is well under the usual 0.05 cutoff, so we can reject the null hypothesis that sex and survival are independent.

We have seen all this before in Sec. 6.4.2, so why does BlueSky include this item? The previous method would display the odds ratio only if the table happened to be a two by two one. Let us see what happens when we swap out sex for passenger class, pclass, which has three levels instead of two. Table 6.23 shows the result. The first odds ratio is 1, which is just the odds of 1st class compared to 1st class. The second odds ratio is 0.4482, which is the odds of survival in 2nd class compared to 1st. The third, and final odds ratio is 0.2015, which is the odds of survival in 3rd class compared to 1st. You can change that order by choosing among the various "Category Reversal" options. Those let you set the main comparison category as the Exposure variable or the Outcome variable, or both. You can achieve even greater flexibility in setting the comparison level by changing the factor level's order, as described in Sec. 4.9.

The Options box lets you set the methods of estimation for the odds ratios, including unconditional maximum likelihood (Wald), conditional maximum likelihood (Fisher), median-unbiased method (mid-p), or small-sample adjusted.

The `epitools` package's `epitab` function performs the calculations in this section.

Table 6.23: Odds ratios for a 3 by 2 table.

Predictor	Outcome							
	0	p0	1	p1	oddsratio	lower	upper	p.value
1st	103	0.1664	181	0.4239	1	NA	NA	NA
2nd	146	0.2359	115	0.2693	0.4482	0.3178	0.6322	0.0000
3rd	370	0.5977	131	0.3068	0.2015	0.1473	0.2756	0.0000

6.4.4 Relative Risks, M by 2 table

While odds ratios compare risk by dividing a dichotomous outcome's odds, relative risk divides the outcome's probabilities. Let us examine the same example here as we did immediately before in Sec. 6.4.3. To do so, follow these steps:

1. Choose "Analysis> Contingency Tables> Relative Risks, M by 2 Table."
2. Choose sex as the Exposure/Predictor variable.
3. Choose survived as the Outcome variable, then click OK.

The first table of output is the same simple crosstabulation shown in Tab. 6.24. A glance at that shows most females survived while most males did not, by quite a large margin.

The relative risk table, shown in Tab. 6.24, starts on the left side by nearly repeating the crosstabulation, which just makes it seem more complicated than it is. It adds proportions though, which are the components of relative risk. We see that 75.26% of females survived while only 20.52% of males survived. The relative risk is just the ratio of those two values, or 0.2052. So we can say the probability of survival for males is only 20.1% that of the females. Or we could say that it is around 80% lower for males. The confidence interval does not include 1, which would represent equal risk. The p-value is 0.0000, far less than the usual 0.05 cutoff, so we can reject the null hypothesis that sex and survival are independent. The estimation methods for calculating the risk ratios include unconditional maximum likelihood (Wald) or small-sample adjusted. The confidence intervals are estimated using a normal approximation (Wald).

If we were to swap out sex for passenger class, pclass, which has three levels instead of two, the result would be very similar to what we saw with the odds ratio, only this time with relative risk. See Sec. 6.4.3 for details.

The computations in this section are calculated by the `epitools` package's `epitab` function.

6.4.5 Legacy: Crosstab, Two-way

The item "Analysis> Contingency Tables> Legacy> Crosstab, Two-way" is an older version of The item "Analysis> Contingency Tables> Crosstab, multi-way" covered in Sec. 6.4.2. This version has crude mono-spaced output rather than BlueSky's usual nicely formatted tables.

Let us do the same analysis we did in that earlier section:

Table 6.24: Relative risk of survival by sex.

Predictor	Outcome							
	0	p0	1	p1	riskratio	lower	upper	p.value
female	96	0.2474	292	0.7526	1	NA	NA	NA
male	523	0.7948	135	0.2052	0.2726	0.2321	0.3202	0.0000

1. Choose "Analysis> Contingency Tables> Legacy> Crosstab, Two-way"
2. Choose sex as the row variable.
3. Choose survived as the column variable.
4. Check only the Row Proportion and Pearson's Chi-Squared Test, then click OK. Note that both Fisher's Exact Test and McNemar's test are available under the Options button.

The output is simply mono-spaced text, which is the default in R unless BlueSky or R Markdown formats it. The first bit of output is the key to the numbers in each cell. Although we did not ask for it, the Chi-square contribution is included whenever the Chi-squared test is requested.

```
Cell Contents
|-------------------------|
|                       N |
| Chi-square contribution |
|           N / Row Total |
|-------------------------|
```

Next is the crosstabulation itself.

```
Total Observations in Table:  1046

|               | Titanic$survived
| Titanic$sex   |         0 |         1 | Row Total |
|---------------|-----------|-----------|-----------|
|        female |        96 |       292 |       388 |
|               |    77.748 |   112.707 |           |
|               |     0.247 |     0.753 |     0.371 |
|---------------|-----------|-----------|-----------|
|          male |       523 |       135 |       658 |
|               |    45.845 |    66.459 |           |
|               |     0.795 |     0.205 |     0.629 |
|---------------|-----------|-----------|-----------|
| Column Total  |       619 |       427 |      1046 |
|---------------|-----------|-----------|-----------|
```

Next comes the Chi-squared test, the first of which is the same as we saw in Sec. 6.4.2. It is followed by another Chi-squared test, this one with the

Yates correction, a modification to the statistic that makes it more accurate when cell counts become tiny.

```
Statistics for All Table Factors

Pearson's Chi-squared test
------------------------------------------------------------
Chi^2 =  302.7586     d.f. =  1     p =  8.256155e-68

Pearson's Chi-squared test with Yates' continuity correction
------------------------------------------------------------
Chi^2 =  300.4968     d.f. =  1     p =  2.567603e-67
```

BlueSky uses the `gmodels` package's `CrossTable` function for these calculations.

6.5 Correlation

The Pearson correlation measures the strength of linear association between two continuous numeric variables. Figure 6.8[1] shows the importance of examining that relationship graphically. Plot I demonstrates the type of relationship that correlations are designed to assess. Plot II shows a perfectly curved relationship that could be fit with a quadratic regression model. Plot III looks like it has an extreme outlier, perhaps a data entry error. A robust regression might be the thing to use for it. Plot IV might be the result of an erroneous transformation. However, all four plots have a Pearson correlation of .816, a value high enough that people might be tempted to see it and not bother to look at the plots. You can see what your relationship looks like using a scatterplot.

The Pearson correlation assumes that the data follow a normal distribution. Without meeting that assumption, its p-value will not be accurate. You can assess that through histograms, density plots, or normality tests like the Shapiro-Wilk test (Sec. 6.13.7). If the data are not normally distributed, or if the variable has only an ordinal scale (e.g. low, medium, high) then the Spearman correlation is appropriate.

6.5.1 Correlation Matrix

The item "Analysis> Correlation> Correlation Matrix" computes a set of either Pearson (the default) or Spearman correlations. To create such a matrix, follow this example:

1. In the Analysis window, use "File> Load Dataset from a Package" and enter the dataset name "iris" into the box on the lower right.
2. Choose "Analysis> Correlation> Correlation Matrix"
3. Choose the four petal and sepal measures and click OK.

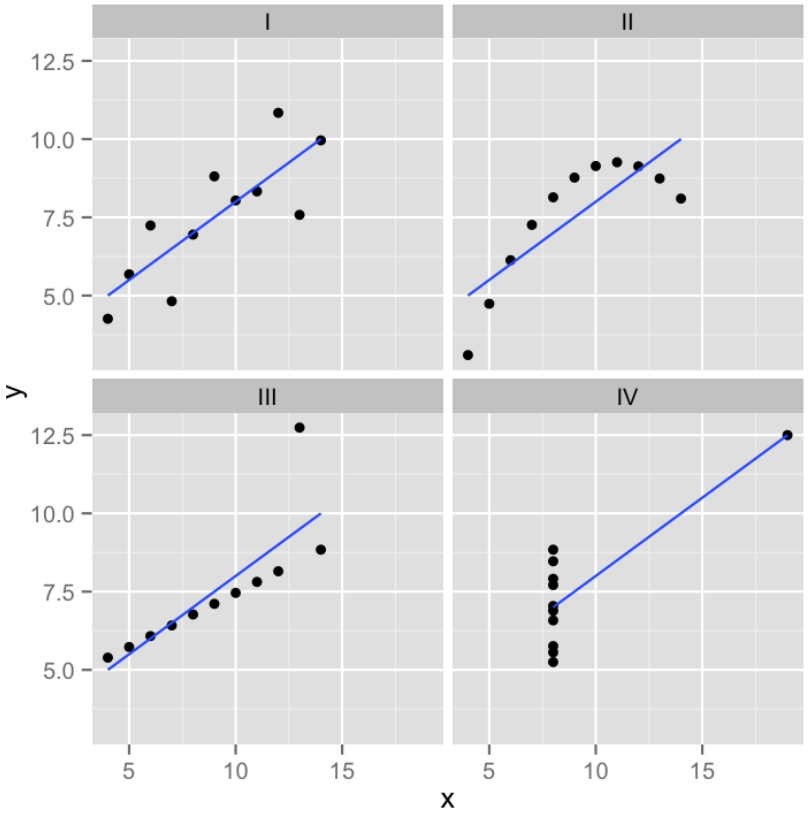


Fig. 6.8: Anscombe's scatterplot examples. All four have the same correlation.

Table 6.25: Correlation matrix of iris data.

	Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
Sepal_Length	1	-0.1176	0.8718	0.8179
Sepal_Width	-0.1176	1	-0.4284	-0.3661
Petal_Length	0.8718	-0.4284	1	0.9629
Petal_Width	0.8179	-0.3661	0.9629	1

Table 6.25 shows the correlation matrix. We see that sepal length is correlated positively with petal length and width and negatively with sepal width. Petal length is very highly correlated with petal width, with r = 0.96. While this matrix is in a nice compact form, it is missing two key elements: the number of cases used in each correlation and the p-value associated with a statistical test for each. We will get to those in the sections that follow.

BlueSky uses the `stats` package's `cor` function to perform these calculations.

Table 6.26: Correlation pair output.

Null Value Considered: 0					
			sample estimate	confidence: 0.95	confidence: 0.95
t	df	p-value	cor	lower	upper
-4.7865	148	0.0000	-0.3661	-0.4972	-0.2187

6.5.2 Correlation Test, one pair

The item "Analysis> Correlation> Correlation Test, one pair" calculates a single correlation and matching p-value. Let us examine one of the more moderate correlations that we previously calculated in Sec. 6.5.1:

1. If you have not already done so, in the Analysis window, use "File> Load Dataset from a Package" and enter the dataset name "iris" into the box on the lower right.
2. Choose "Analysis> Correlation> Correlation Test, one pair"
3. Choose Sepal_Width and Petal_Width and click OK.

Table 6.26 shows the result. We see that the correlation is -0.3661 and its p-value is 0.0000 or $p < .001$. Since the correlation is negative, the wider the sepal width, the narrower the petal width tends to be. Since the p-value is well under the usual 0.05 cutoff, we can reject the null hypothesis that the correlation is zero. The 95% confidence interval also excludes zero, allowing us to draw the same conclusion.

BlueSky uses the `stats` package's `cor.test` function to perform these calculations.

6.5.3 Correlation Test, multi-variable

In the above two sections, we saw a correlation matrix *without* p-values and a single correlation *with* a p-value. Using The item "Analysis> Correlation> Correlation Test, multi-variable" we finally get the complete set of information that most statistics packages provide: correlations, number of subjects used for each calculation, and the associated p-values. We even get something that most other statistics packages lack: p-values adjusted for the number of tests performed. Let us repeat the correlation matrix example from Sec. 6.5.1:

1. If you have not already done so, in the Analysis window, use "File> Load Dataset from a Package" and enter the dataset name "iris" into the box on the lower right.
2. Choose "Analysis> Correlation> Correlation Test, multi-variable."
3. Choose the four petal and sepal measures and move them to the Selected Variables box.

4. Leave the type of correlation on Pearson. However, if you wish to try a Spearman correlation, you can choose that instead.
5. Under "How to handle missing values," choose Pairwise Complete Cases. That will use the maximum number of cases in each correlation, though it will yield slightly different sample sizes on each if there are missing values.
6. Under "Show P Value," choose, "Both P Values." This will provide standard ones and ones that are adjusted for the number of tests performed.
7. Check the "Visualize Correlation Matrix" box.
8. Check the "Visualize WebPlot" box, then click OK.

When you have filled in the dialog box, it should appear as shown in Fig. 6.9. The first piece of output is the correlation matrix itself, as shown in Tab. 6.27. Each cell of that matrix shows the correlation, then below it, the p-value adjusted for the effects of multiple testing (Adj-P). Below that is the unadjusted p-value (Un-adj-P). The final value in each cell is the number of cases used to calculate the correlation.

If you had planned in advance to focus on only one statistical test, then the unadjusted p-value would be the one to use. In addition, if previous research had already shown the p-values in this matrix to be significant – and in the same direction – then the un-adjusted ones would also be fine. However, if you planned in advance to test them all, or you are simply exploring, then you need to correct them for the effects of chance. One way to do this is called the Bonferroni correction. We did five tests, so we could multiply each p-value by five to make the test more conservative. However, that is known to be too conservative, so this item instead uses the Holm method of multiplying the best (smallest) p-value by K, when you have done K tests. Then the second-best p-value is multiplied by K-1, and so on. Once a p-value exceeds 0.05, then all the remaining tests are deemed not significant. This is also called the Sequential Bonferroni test.

The heatmap plot shown in Fig. 6.10 color-codes the correlations to help visualize where they are strongly positive (green) or negative (blue).

Another way to visualize the correlations is the web plot shown in Fig. 6.11. Here the strength of the correlation is indicated by the thickness of the line connecting the pair of variables. Positive ones have blue lines and negative ones are red. The legend uses scientific notation, so the best one of 9.629e-01 is $9.629 \times 10^{-}1$ or .9229.

The `RcmdrMisc` package's `rcorr.adjust` function calculate the correlations in this section. The `DescTools` package's `PlotCorr` and `PlotWeb` functions created the heatmap and web plots, respectively.

6.6 Factor Analysis

While this section is named Factor Analysis, it contains two topics that are mathematically similar but theoretically quite different. Factor analysis

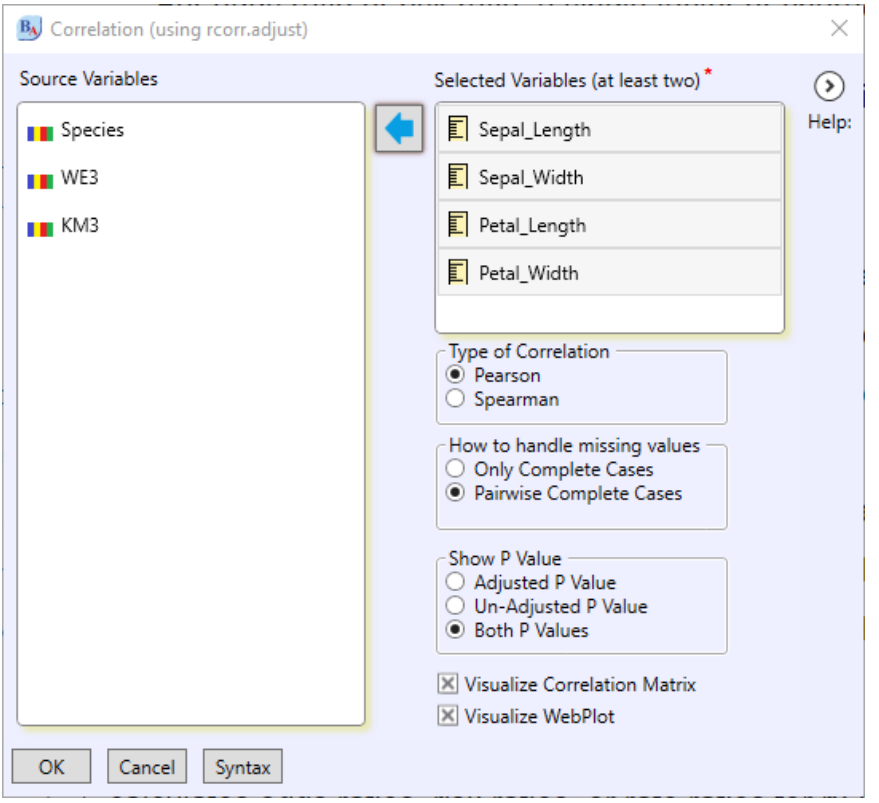


Fig. 6.9: Correlation Test, multi-variable dialog.

aims to measure latent constructs, such as extroversion, which cannot be measured directly.

On the other hand, Principal Components Analysis is conceptually quite simple. It aims to simplify analysis by collapsing many variables into fewer ones that contain most of the original information.

6.6.1 Factor Analysis

Factor analysis (FA) is about estimating the values of "latent" concepts, which cannot easily be measured directly. The personality construct of extroversion is an example. You can ask, "On a scale of 1 to 100, how extroverted are you?" However, you are unlikely to get a reliable answer. Is each person defining extroversion in the same way? I am very comfortable giving talks to large audiences, yet attending a party of mostly strangers makes me a bit nervous. How extroverted is that combination? Getting a reliable and valid measure takes more than one question. In the example we will use, five questions assess each personality trait.

Marketing often uses factor analysis. Surveys of customer satisfaction often have sets of questions that are correlated. Items regarding the product

Table 6.27: Correlation matrix with adjusted p-values.

		Sepal_Length	Sepal_Width	Petal_Length	Petal_Width
Sepal_Length	Correlation	1	-0.1176	0.8718	0.8179
	Adj-P		0.1519	<.0001	<.0001
	Un-adj-P		0.1519	<.0001	<.0001
	n	150	150	150	150
Sepal_Width	Correlation	-0.1176	1	-0.4284	-0.3661
	Adj-P	0.1519		<.0001	<.0001
	Un-adj-P	0.1519		<.0001	<.0001
	n	150	150	150	150
Petal_Length	Correlation	0.8718	-0.4284	1	0.9629
	Adj-P	<.0001	<.0001		<.0001
	Un-adj-P	<.0001	<.0001		<.0001
	n	150	150	150	150
Petal_Width	Correlation	0.8179	-0.3661	0.9629	1
	Adj-P	<.0001	<.0001	<.0001	
	Un-adj-P	<.0001	<.0001	<.0001	
	n	150	150	150	150

may all be answered as high by people who like it and low by those who do not. Unrelated to that may be customer service. Even if people do not like the product, they may appreciate how the company's tech support line helps them use it. Factor analysis can find such relationships.

Let us apply factor analysis to some personality test questions. There are five well-known personality traits: Openness to new ideas, Conscientiousness, Extroversion, Agreeableness, and Neuroticism (a blend of anxiety, worry, fear, etc.) The mnemonic to help remember them is the word OCEAN. The dataset we will use contains five variables for each trait, and they begin with the letter that represents that trait. So the five variables that measure extroversion are E1 through E5. We will use factor analysis to see if it can find these latent constructs. Table 6.28 lists the variable labels.

Here are the steps to follow:

1. Go to the Analysis window and choose "File> Load dataset from a package."
2. Enter the name "bfi" in the lower right of the dialog box. That will load the dataset from the "psych" package, but you probably do not need to fill in the package name.
3. Choose "Analysis> Factor Analysis> Factor Analysis."
4. Choose the variables A1 through O5 and move them to the "Destination Variables" box.
5. Under "Factor Extraction," choose "Automatically extract factors."
6. Check the Screeplot box.
7. Check the "Save factor scores in dataset" box, use Bartlett's method, and enter "Factor_" in the box labeled, "Enter variable name prefix for scores."
8. Under "Rotation Options," choose Promax.
9. The dialog should now appear as shown in Fig. 6.12. Click OK.

Table 6.28: Variable descriptions for factor analysis example.

Variable	Question
A1	Am indifferent to the feelings of others.
A2	Inquire about others' well-being.
A3	Know how to comfort others.
A4	Love children.
A5	Make people feel at ease.
C1	Am exacting in my work.
C2	Continue until everything is perfect.
C3	Do things according to a plan.
C4	Do things in a half-way manner.
C5	Waste my time.
E1	Don't talk a lot.
E2	Find it difficult to approach others.
E3	Know how to captivate people.
E4	Make friends easily.
E5	Take charge.
N1	Get angry easily.
N2	Get irritated easily.
N3	Have frequent mood swings.
N4	Often feel blue.
N5	Panic easily.
O1	Am full of ideas.
O2	Avoid difficult reading material.
O3	Carry the conversation to a higher level.
O4	Spend time reflecting on things.
O5	Will not probe deeply into a subject.
gender	Male = 1, Female =2
education	1 = HS, 2 = finished HS, 3 = some college, 4 = college graduate 5 = graduate degree
age	age in years

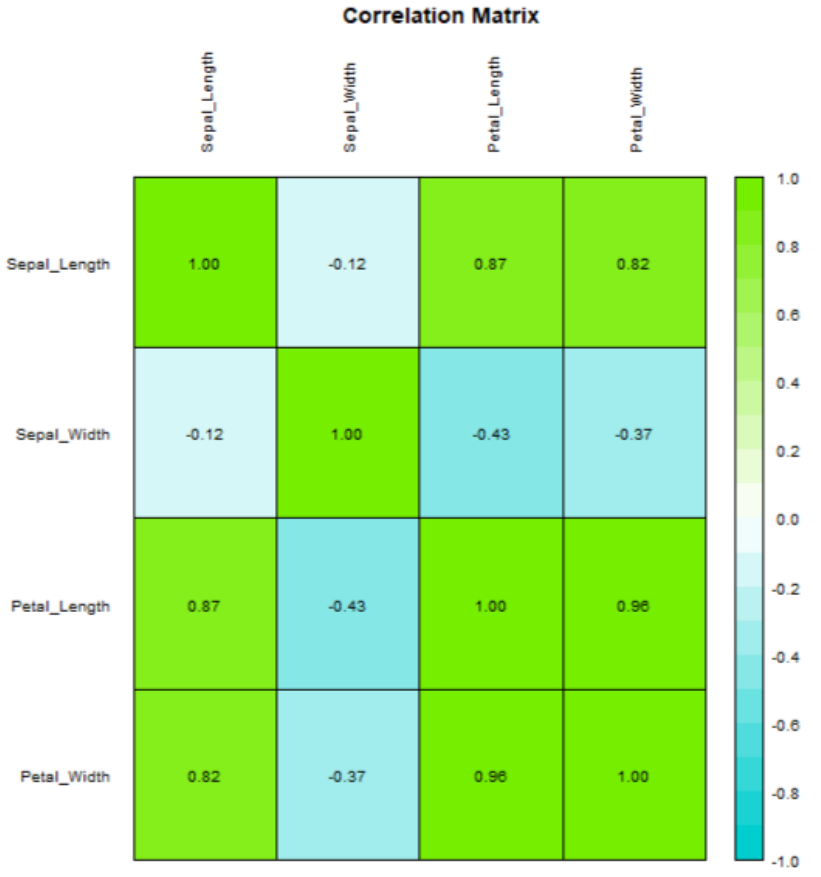


Fig. 6.10: Correlation heatmap. The saturation of the color of each cell indicates the strength of each correlation.

The scree plot shown in Fig. 6.13 helps you choose the number of factors in the data. Scree is the ruble at the bottom of a cliff, representing the left over factors that do not explain much of the original variance in the dataset. In our case, the ideal screeplot would be a nearly vertical start, with the cliff consisting of five factors, then a horizontal set of all the others, all with eigenvalues below a value of one (the dashed line). However, in our plot, it is not so clear where the cliff stops and the rubble begins. That is a common situation.

The eigenvalues on the y-axis represent the amount of variance that each factor contains. By default, the analysis will save as many factors as there are eigenvalues greater than one. Below a value of one means that those factors do not even explain as much variance as one of the original variables. In our plot, we see that six factors have eigenvalues greater than one, but just barely. From this, we could conclude that there are either five

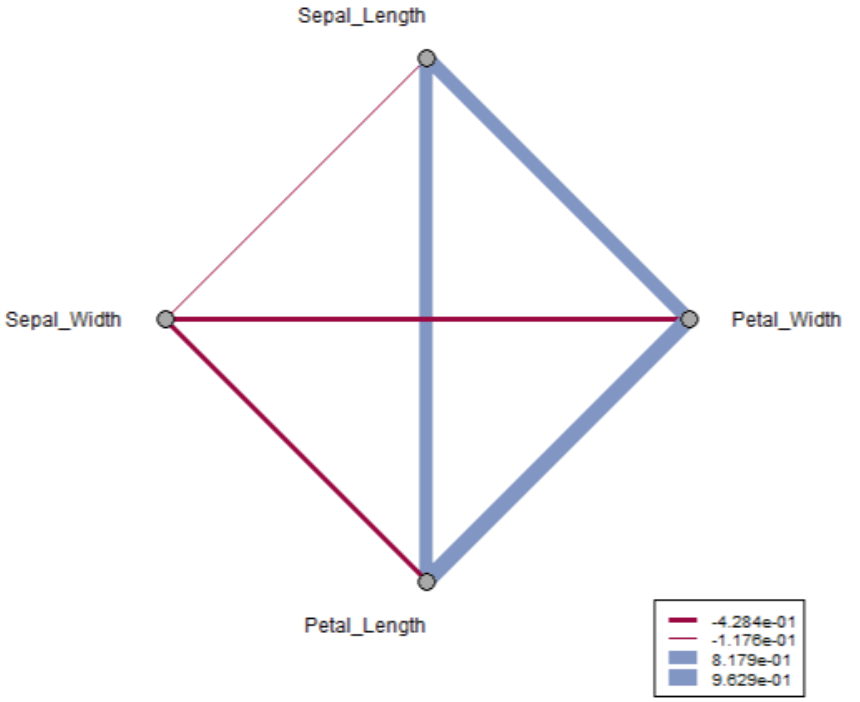


Fig. 6.11: Correlation web plot. The thickness of each line indicates the strength of each correlation

or six factors in the dataset. We could rerun this and force it to choose five factors, but it is often a good idea to leave in one extra factor rather than force a variable to become part of one that is not that good a fit for it.

We are rotating our factors. That is a process that transforms the factors to get each variable to load as highly as possible on a single factor. Some of these methods, such as Varimax, Quartimax, and GeominT assume the underlying constructs are uncorrelated, i.e., they are orthogonal rotations. Others, such as Promax, Oblimin, Simplimax, GeominQ, and BentlerQ allow the factors to correlate, i.e., they are oblique rotations. A commonly used approach is to start with an oblique rotation and if the correlations among the factors are all low, say, lower than 0.32, then switch to an orthogonal rotation. That level of correlation would mean that no factor explains more than 10% of the variance in another (the correlation squared). Since Promax is an efficient algorithm, it can handle large datasets well and is a popular choice for an oblique rotation.

Since we are rotating our factors, we will ignore the output before the rotation happens. The uniqueness table, shown in Tab. 6.29, is the percentage of variance in that variable, which is not explained by the factors. Since factors are collections of variables that have common variability, you might consider excluding a unique variable from the analysis. Subtracting

uniqueness from 1 yields "communality" or the variance that the variable *does* have in common with the factors.

The rotated factor loadings, shown in Tab.6.30, show the weights that are applied to each variable to create the factors. Factor analysis can only show us which variables varied together; it cannot tell us what the factors mean. Exploratory factor analysis does not usually begin with likely known set of factors as we have done. Usually, you would need to study the items that load highly on a given factor to decide what construct the factor represents. Looking at the loadings for Factor 1, we would have to read the questions and conclude from them that Extroversion was the underlying construct. However, we do now see that those questions all began with the letter E for Extroversion and no other variables loaded on this factor very strongly. Loadings of less than 0.10 are automatically left blank for ease of study. Note that items E1 and E2 have negative loadings. Looking back at Tab. 6.28 we see that those two items are worded oppositely from the others. In a correlation matrix, those two are indeed negatively correlated with the others (not shown). Variable A5 also loads fairly highly with Factor 1, with a loading of 0.377. Factor analyses often include such inconsistencies. Changing the number of factors kept or the method of rotation may yield a more consistent solution.

Factor 2 loads highly on all the neuroticism items. Factor 3 matches the conscientiousness variables. Factor 4 matches the agreeableness ones. Factor 5 matches the openness to new ideas variables.

While the default approach of keeping all the factors that have an eigenvalue of 1 has yielded six factors, here only the variable C4 loads highly on it.

Table 6.31 shows the proportion of variance that each factor explains. Extroversion, neuroticism, and conscientiousness each explain around 10% of the variance contained in the original variables. Agreeableness and openness each explain around 6%. Even including our lonely sixth factor, our factors explain only 47.1% of the variance of the original variables. The hypothesis test that six factors are sufficient to explain the variance in the original variables is rejected, with a p-value of 0.0000 (not shown), which makes sense given the small percentage of variance these factors explain. However, that test is often not very useful in choosing the number of factors.

Table 6.32 shows how the factors correlate. The largest correlation, -0.377, is between extroversion and neuroticism. That seems to make sense as it might be hard for someone to talk to other people if he or she is prone to mood swings and panic attacks. With a correlation that high, it is best to continue using an oblique rotation.

Looking at the Datagrid, you should see the new factor score variables named Factor_1 through Factor_6. Keep in mind that the Datagrid only shows a subset of variables, which is controlled by the arrows at the bottom right of the grid. If you don't see the new factor scores in your dataset, use them to scroll further toward the grid's right side. We can use the factor scores in any other analysis. Are males more extroverted than females? We can now efficiently address such questions.

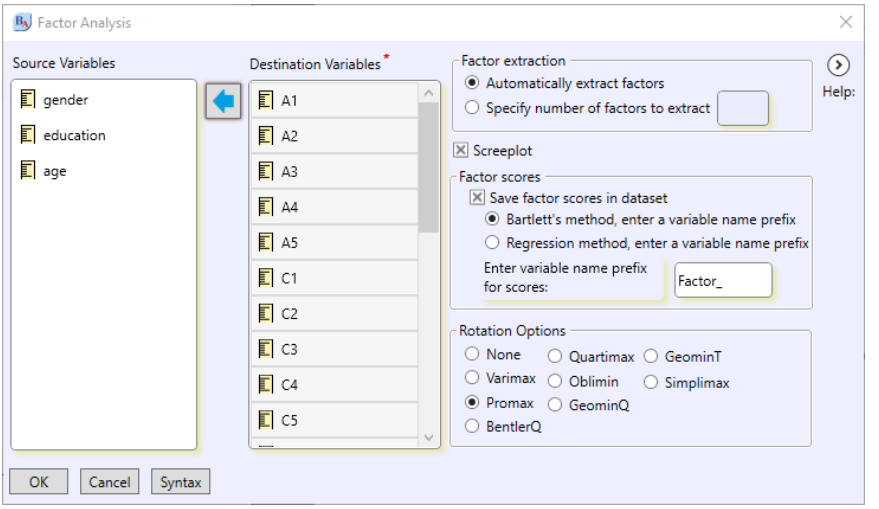


Fig. 6.12: Factor analysis dialog.

Table 6.29: Factor analysis variable uniqueness. This is truncated to get it to fit on the page.

Uniqueness (promax rotation)

A1	A2	A3	A4	A5	C1	C2	C3	C4	C5	E1	E2	E3
0.675	0.482	0.473	0.696	0.516	0.637	0.498	0.685	0.424	0.566	0.613	0.453	0.522

The `BlueSky` package's `BSkyFactorAnalysis` function performed the calculations in this section. That function calls the `stats` package's `factanal` function and several of the `GPArotation` package's functions for rotation.

6.6.2 Principal Components Analysis

PCA is a data reduction method that aims to capture the variability of many variables using a subset of new variables, called components, that are uncorrelated weighted averages of the original ones. The components are created in such a way that the first one captures as much variance as possible. The second one captures the next largest amount of variance, while maintaining a zero correlation with the first, and so on. In the end, you have just as many components as there were variables, but only the first few are of interest since they contain most of the variance. An example will help clarify this rather odd idea.

There are 206 bones in the human body. You might have the length and width of each stored as 412 variables. But comparing, say, males and females on all those would be quite complicated. Performing PCA could likely collapse all those measures into just two: length and thickness. Those two might explain 90% of the original variance contained in the 412 measures.

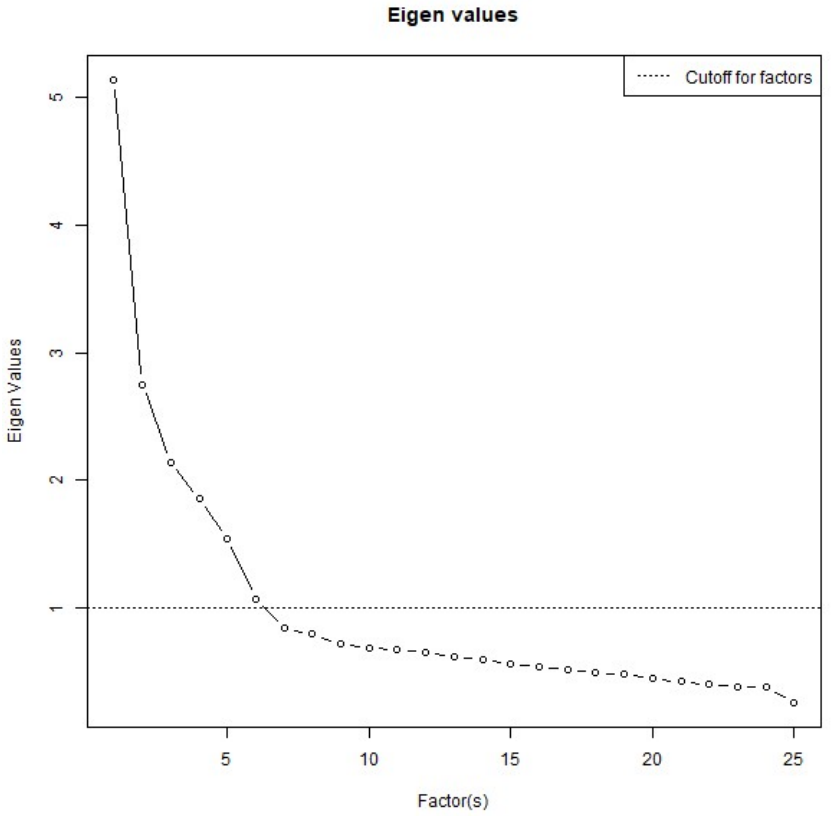


Fig. 6.13: Factor analysis scree plot.

With such a useful simplification, further analyses become much simpler. In addition, since the PCA variables are created in such a way that they correlate zero with each other, they provide an advantage for methods like linear regression or cluster analysis which have trouble dealing with highly correlated variables.

PCA's focus is on explaining variance. Measures that have larger means also have larger variances, so we need to prevent such variables from dominating the PCA analysis. This is accomplished by running the PCA on the correlation matrix of the variables instead of the covariance matrix (this is BlueSky's default, not R's). That is the equivalent to first subtracting the mean of each original variable and dividing by its standard deviation. It allows the PCA to focus on a comparable measured variance across all variables. Each of the resulting principal component variables is scaled with a mean of zero, and with around 95% of its values falling between plus and minus two (the standard deviation is 1).

Let us apply PCA to a famous dataset of iris flowers. Here are the steps to follow:

Table 6.30: Factor analysis rotated loadings.

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
A1	0.167			-0.605		0.188
A2	0.109			0.679		
A3	0.262			0.555		0.157
A4	0.158		0.195	0.349	-0.18	
A5	0.377	-0.189		0.359		0.285
C1			0.601		0.128	
C2			0.749			
C3			0.607			-0.185
C4			-0.7	-0.105		0.569
C5	-0.107	0.1	-0.571			0.33
E1	-0.651	-0.207	0.18			
E2	-0.732					
E3	0.56				0.233	0.358
E4	0.724				-0.133	0.242
E5	0.488	0.223	0.232		0.177	
N1	0.147	0.936				-0.122
N2		0.941				-0.196
N3		0.649				0.102
N4	-0.369	0.35				0.212
N5	-0.167	0.37		0.121	-0.171	0.161
O1	0.171				0.477	0.247
O2					-0.502	0.164
O3	0.272				0.568	0.293
O4	-0.273			0.145	0.348	0.198
O5					-0.581	0.165

Table 6.31: Factor analysis proportion of variance explained.

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
SS loadings	2.732	2.615	2.267	1.492	1.487	1.187
Proportion Var	0.109	0.105	0.091	0.06	0.059	0.047
Cumulative Var	0.109	0.214	0.305	0.364	0.424	0.471

1. Go to the Analysis window and open the iris dataset by choosing "File> Load dataset from a package."
2. Enter the name "iris" in the lower right of the dialog box.
3. Choose "Analysis> Factor Analysis> Principal Components Analysis."
4. Fill in the dialog, as shown in Fig. 6.14. Be sure to check the "Add components to the dataset" and fill in "PCA_" as the prefix for the new variable names. Also check the correlation matrix and screeplot items.

Table 6.32: Factor correlations.

	Factor1	Factor2	Factor3	Factor4	Factor5	Factor6
Factor1	1	-0.3773	-0.3327	-0.1848	-0.3263	0.1603
Factor2	-0.3773	1	0.0747	0.0754	0.1136	-0.4859
Factor3	-0.3327	0.0747	1	0.2033	0.3019	0.2775
Factor4	-0.1848	0.0754	0.2033	1	0.1016	-0.0214
Factor5	-0.3263	0.1136	0.3019	0.1016	1	0.2177
Factor6	0.1603	-0.4859	0.2775	-0.0214	0.2177	1

Table 6.33: Principal component variances.

Comp.1	Comp.2	Comp.3	Comp.4
2.9185	0.914	0.1468	0.0207

The first piece of output is the scree plot (Fig. 6.15). This plot shows the amount of variance that each new PCA variable contains. The term "scree" is the name for the rubble you often find at the bottom of a cliff. To interpret a screeplot, you look to see how many variables form a cliff-like high set of bars, and the rest end up as the tiny scree-like ones at the bottom. In this case, first two bars are tall relative to the others. That indicates that there are not really four unique measurements to these flowers. Just two components are likely to explain a large proportion of the variance.

Table 6.34 shows the component loadings. They indicate how the four components are constructed. To create the first principal component, we calculate:

$Comp.1 = 0.5211 \times Sepal\_Length - 0.2693 \times Sepal\_Width + 0.5804 \times Petal\_Length + 0.5649 \times Petal\_Width$

Component 2 consists of mostly the sepal width variable.

Table 6.33 shows the variances of each component. Table 6.35 converts those variances into the proportion of variance explained by each component. We see that 95.81% of the variance is explained by the first two components. The first three explain 99.48% of the variance. Since we saved those components to the dataset, we can use them for other analyses. See Sec. 6.3 for an example.

The calculations in this section were performed by the `BlueSky` package's `BSkyFactorAnalysis` function. That calls the `stats` package's `factanal` function and several of the `GPArotation` package's functions for rotation.

6.7 Market Basket Analysis

Market Basket Analysis (MBA) finds simple association rules which help you understand how items or sets of items (itemsets) coincide. In retail, it

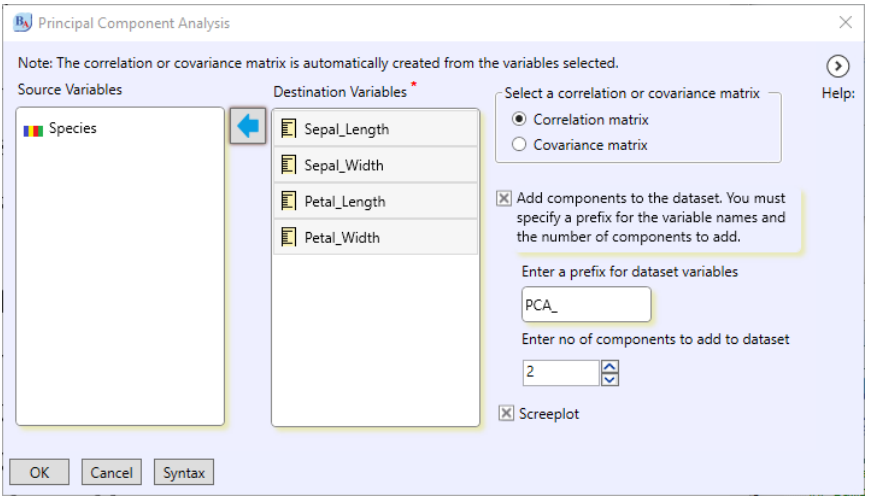


Fig. 6.14: Dialog box from "Analysis> Factor Analysis> Principal Components."

Table 6.34: Principal components analysis loadings.

	Comp.1	Comp.2	Comp.3	Comp.4
Sepal_Length	0.5211	0.3774	0.7196	0.2613
Sepal_Width	-0.2693	0.9233	-0.2444	-0.1235
Petal_Length	0.5804	0.0245	-0.1421	-0.8014
Petal_Width	0.5649	0.0669	-0.6343	0.5236

Table 6.35: PCA Importance of components.

	Comp.1	Comp.2	Comp.3	Comp.4
Standard deviation	1.7084	0.956	0.3831	0.1439
Proportion of Variance	0.7296	0.2285	0.0367	0.0052
Cumulative Proportion	0.7296	0.9581	0.9948	1

could address the question, "Does buying cookies increase the probability of buying milk?" In medicine, one might ask, "Does smoking increase the probability of emphysema, and if so, by how much?"

The terminology of MBA has many unusual aspects to it. The output is generally displayed in tables, with each row showing the predicting set of items on the left-hand side (LHS) and the single outcome on the right-hand side(RHS).

Three main measures control the algorithm:

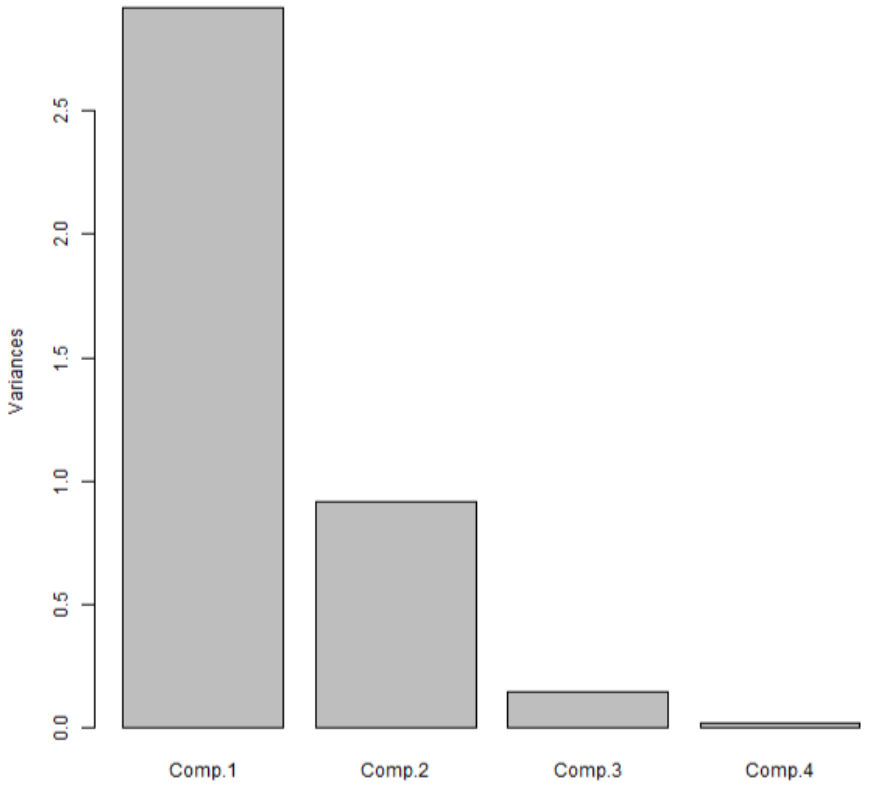


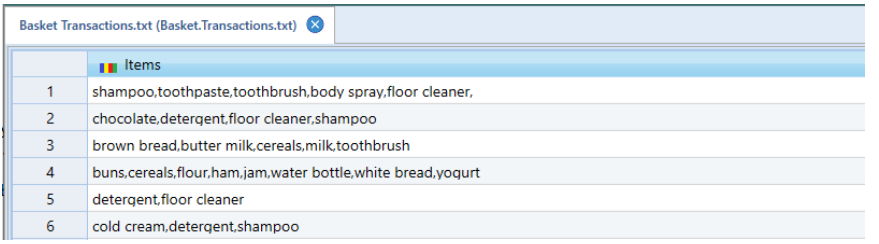
Fig. 6.15: Principal components analysis scree plot.

- Support – the proportion of all transactions that contain the predicting itemset (LHS). This is abbreviated "support(LHS)."
- Confidence – the conditional probability that customer buying items on the LHS will also buy item on RHS. Stated another way, the $support(LHS + RHS) \div support(LHS)$
- Lift – the change in outcome probability, given the itemset of predictor values. If greater than 1, the predictor itemset is increasing the probability of the given outcome. If less than 1, the predictor itemset is decreasing the probability of the given outcome.

Since MBA views any item interesting to predict, we will see some examples where those predictions make little sense. To avoid such cases, the analysis offers the ability to target specific items.

Since MBA's view is that everything is an "item," when there are continuous variables, it will automatically chop them into categories such as "Age 18-24." That process is called discretization or binning.

One of the most unusual features of MBA is its ability to analyze data stored in *very* different formats. The following sections cover each of these formats.



Basket Transactions.txt (Basket.Transactions.txt)

	Items
1	shampoo,toothpaste,toothbrush,body spray,floor cleaner,
2	chocolate,detergent,floor cleaner,shampoo
3	brown bread,butter milk,cereals,milk,toothbrush
4	buns,cereals,flour,ham,jam,water bottle,white bread,yogurt
5	detergent,floor cleaner
6	cold cream,detergent,shampoo

Fig. 6.16: The Datagrid showing basket data format. Note that there is only one variable but it contains many comma-separated values.

The `arules` package's `apriori` function performs the calculations for this topic. The `arulesViz` package's `plot.associations` and `plot.rules` provide the visualizations.

6.7.1 Basket Data Format

The item "Analysis> Market Basket> Basket Data Format" covers three main topics, all using what is called the "basket data format." This format is popular when the event combinations studied are a tiny subset of all possible combinations. For example, the typical grocery store has around 40,000 items, but shoppers purchase only a tiny proportion of them at a time. In cases like that, market basket data format saves a huge amount of space by entering only the event combinations that occurred. Here are the top four lines of the dataset that we will be using in this section:

```
shampoo,toothpaste,toothbrush,body spray,floor cleaner
chocolate,detergent,floor cleaner,shampoo
brown bread,butter milk,cereals,milk,toothbrush
buns,cereals,flour,ham,jam,water bottle,white bread,yogurt
```

Perhaps the most surprising thing about this format is that each of the above rows is stored as a single value! Normally, comma separated values end up spread across multiple variables, but those all end up in one variable. Let us see how this happens:

1. Go to the Analysis window and choose "File> Open."
2. Browse to the file, "`C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Market Basket\ Basket Transactions.txt`" and open it.
3. Check the box labeled, "Load all comma separated values in a single column/variable" then click OK.

Figure 6.7.1 shows how the data should appear. You can continue to any other section which uses this type of data to generate rules or plots.

Generate Rules, basket data

Once you have read data that is stored in basket data format, (described in Sec. 6.7.1), you may use the item, "Analysis> Market Basket> Basket Data Format> Generate Rules, basket data" to generate the rules that relate the items to one another. To do so, follow these steps:

1. Choose "Analysis> Market Basket Analysis> Basket Data Format> Generate Rules, basket data."
2. Move the variable named "Item" to the "Single variable..." box. If you're using your own data, use whatever single variable name you chose.
3. Check the box for "Prune rules."
4. Set "Minimum Support Value" to 0.05.
5. When the dialog looks like Fig 6.17 click OK.

The first table of output that we will examine is the list of pruned rules, shown in Tab. 6.36. It shows the predictor items on the Left Hand Side (LHS), and the item being predicted is on the Right Hand Side (RHS). Next comes support, where we see the top combination occurred 6.62% of the time. That is followed by confidence, where we see that 100% of the time that toothbrush and water bottle were purchased, white bread was also purchased. Finally, we see lift, which indicates that purchasing a toothbrush and water bottle increased the probability of purchasing white bread by 3.89 times.

A previous table contained redundant rules (not shown). Those are rules which are made redundant by the presence of simpler ones that are subsets of them and which are at least as high in confidence. For example, the rule that says people who bought flour, turkey, and white bread, also bought oil 100% of the time (i.e., confidence = 1). However, we already saw that the pruned set of rules contained a simpler rule that says people who bought only turkey and white bread also bought oil 100% of the time. Thus, the more complex rule is redundant and was pruned.

An odd aspect of the algorithm is that if you ask it to prune the rules when there are none to prune, it will ask you to un-check the prune box and run it again.

Table 6.37 shows some basic statistics about the pruned set of rules. They say that the analysis found 62 rules that met the criteria we set. The small table of lengths is a bit confusing at first:

```
rule length distribution (lhs +rhs) : sizes
2  3  4
3 51  9
```

The top row is the length of the total number of items, adding those on the RHS and LHS. The second row is the counts that match. So there were 3 sets that contained only 2 items, i.e., one predictor and one outcome, that met our criteria. There were 51 sets that had two predictors and one outcome, and so on.

The next set of stats shows that rules went from 2 to 4 items with a median of 3. Similar stats for support, confidence, lift, and count follow.

Table 6.36: Market basket pruned rules.

LHS	RHS	Support	Confidence	Lift
c("toothbrush", "water bottle")	white bread	0.0662	1	3.8865
c("toothbrush", "white bread")	water bottle	0.0662	1	4.2882
c("cereals", "curd")	milk	0.0701	1	2.6476
c("cereals", "coffee")	milk	0.0604	1	2.6476
c("turkey", "white bread")	oil	0.0676	1	5.9453
c("milk", "turkey")	salty snack	0.0556	1	4.0128
c("oil", "sweet spreads")	brown bread	0.057	1	2.8474
c("oil", "water bottle")	flour	0.0558	1	5.6148
c("flour", "salty snack")	oil	0.0669	1	5.9453
c("butter", "oil")	flour	0.0574	1	5.6148
c("ice cream", "oil")	brown bread	0.0572	1	2.8474
c("flour", "ice cream")	brown bread	0.068	1	2.8474
c("butter", "sugar")	brown bread	0.0564	1	2.8474
c("chocolate", "curd", "juice")	candy	0.0555	1	3.9262
c("butter", "candy", "milk")	chocolate	0.058	1	3.6643
c("butter", "cereals", "juice")	milk	0.058	1	2.6476
c("butter", "cereals", "milk")	juice	0.058	1	3.4965
c("brown bread", "chocolate", "milk")	butter	0.0555	1	4.0225
c("juice", "milk")	cereals	0.1057	0.9136	2.9281
c("cereals", "juice")	milk	0.1057	0.898	2.3777

Finally, there were 10,000 transactions when we set the minimum support criteria at 0.05 and confidence at 0.8. For a review of those definitions, see Sec. 6.7.

Figure 6.18 shows a scatter plot of our rules with support on the x-axis and confidence on the y-axis. The points are shaded by the amount of lift each provides, with a scale shown on the right side of the plot. At first glance you might miss the pile of points that have perfect confidence of 1 but lower levels of support. Those are rules that show perfectly consistent purchasing patterns though they didn't occur that often.

Figure 6.19 shows a plot of how each set of predictors relates to each outcome. The size of each point indicates the level of support for the rule, and the color of it indicates the amount of lift it offers.

Finally, Fig. 6.20 shows the relationship between rules. For example, at the top we see that toothbrushes and water bottles tend to be purchased with white bread. The size of each circle indicates the item's support (proportion of sales), while the color of it indicates the lift provided by the rule. Such complex plots are best viewed on large computer screens with the plot size set as large as possible. You can adjust the image size using the dialog described in Sec. 11.4.3.

Item Frequency Plot, basket data

An item frequency plot, as shown in Fig. 6.21, is a simple bar chart showing the counts of each item. It does not involve rules at all. However, it does

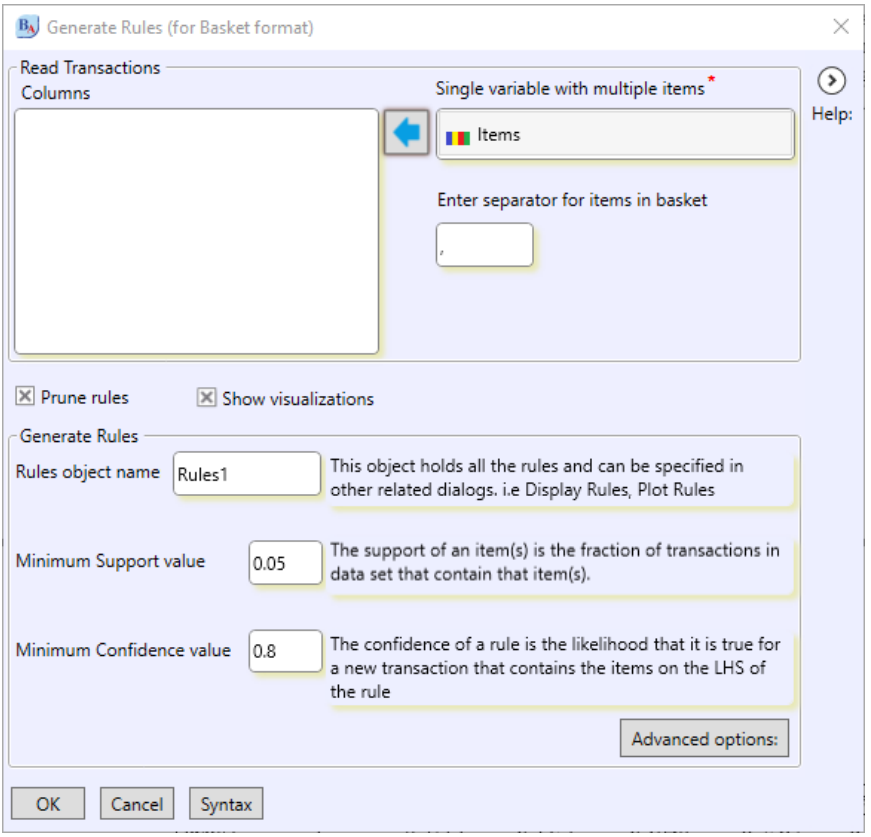


Fig. 6.17: "Analysis> Market Basket> Basket Data Format> Generate Rules" dialog.

require that the type of data structure matches the title of the menu, in this case "basket data format."

Here are the steps to follow to create an example plot:

1. Go to the Analysis window and choose "File> Open."
2. Browse to the file, "`C:\Program Files\BlueSky Statistics\Sample Datasets and Demos\Market Basket\Basket Transactions.txt`" and open it.
3. Check the box labeled, "Load all comma separated values in a single column/variable" then click OK.
4. Choose "Analysis> Market Basket Analysis> Basket Data Format> Item Frequency Plot, basket data."
   Move the "Item" variable to the "Single variable" box.
   Check the box, "Show bars horizontally" then click OK.

The horizontal layout of bars is especially helpful as it allows more clear association between the item and its bar. In a vertical orientation, the labels

Table 6.37: Market basket statistics.

```
 Summary of Rule Statistics
set of 63 rules

rule length distribution (lhs + rhs):sizes
 2  3  4
 3 51  9

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   3.000   3.000   3.095   3.000   4.000

summary of quality measures:
    support         confidence          lift            count
 Min.   :0.05250   Min.   :0.8040   Min.   :2.193   Min.   : 525.0
 1st Qu.:0.05650   1st Qu.:0.8373   1st Qu.:2.624   1st Qu.: 565.0
 Median :0.06410   Median :0.8668   Median :3.091   Median : 641.0
 Mean   :0.06688   Mean   :0.8939   Mean   :3.453   Mean   : 668.8
 3rd Qu.:0.06815   3rd Qu.:1.0000   3rd Qu.:3.970   3rd Qu.: 681.5
 Max.   :0.10570   Max.   :1.0000   Max.   :7.041   Max.   :1057.0

mining info:
 data ntransactions support confidence
   T1          10000    0.05        0.8
```

appear beneath the bars at a 45° angle, making them harder to read, and making it less clear which label matches each bar.

Targeting Items, basket data

Thus far in our study of market basket analysis, we have focused on letting the algorithm choose what to display. By choosing The item "Analysis> Market Basket> Basket Data> Targeting Items, basket data," you can focus the analysis on a particular item. This example uses the same dataset described in Sec. 6.7.1. After opening that dataset, we can study the itemsets that might impact the purchase of beer using these steps:

1. Choose "Analysis> Market Basket Analysis> Basket Data Format> Targeting Items, basket data."
2. Move the variable named "Item" to the "Single variable..." box. If you're using your own data, use whatever single variable name you chose.
3. Check the box for "Prune rules."
4. Set "Minimum Support Value" to 0.001.
5. Set "Minimum Confidence Value" to 0.20.
6. Under "Targeting Items" enter beer in the "Items appearing on the right-hand side." Note that spelling is critically important on this step! Be sure to match the upper and lower case of the item value, or it will tell you that no such item exists.
7. Under "Default Appearance" choose "lhs"
8. Under "Advanced Options," choose "Sort by Lift," then click OK.

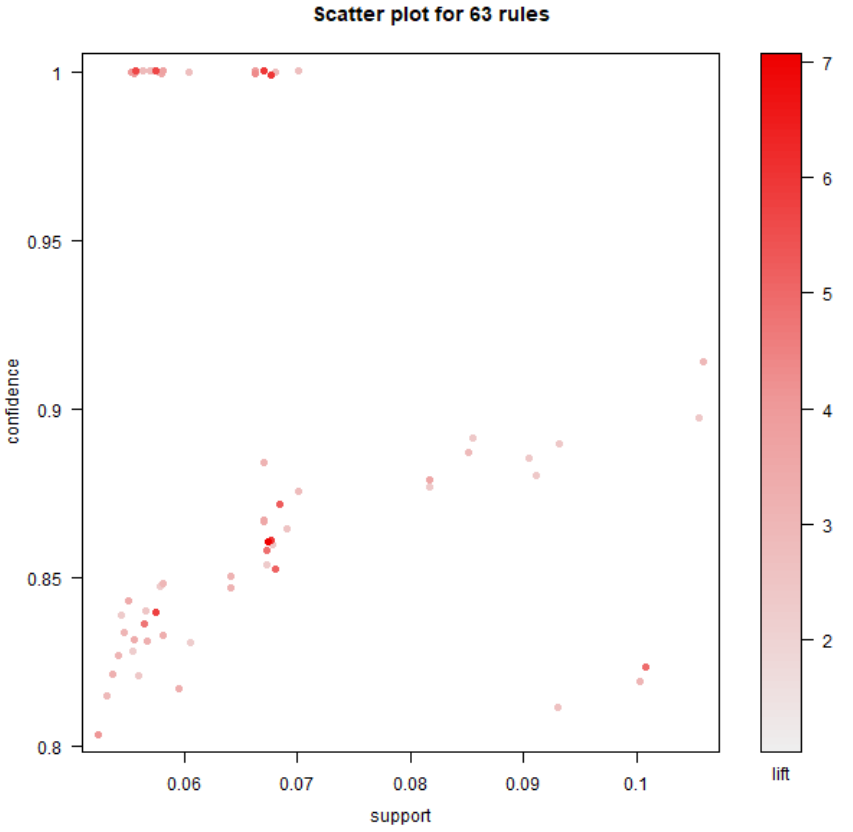


Fig. 6.18: Market basket scatter plot of rules.

Table 6.38 shows the results. The first rule shows that buying pastry and salty snacks will increase the probability of buying beer by nearly six times. That makes sense, but then we see that lots of combinations lead to the same amount of lift.

6.7.2 Display Rules

The item "Analysis> Market Basket> Display Rules" allows you to display a rule set that you already created on one of the "Generate Rules" analyses. Each of those dialogs created a rules object named Rules1 by default. With a large dataset, generating rules can take quite a long time. This option lets you start with the rules object you created previously, sorting them by different measures or changing the number of rules you display. The dialog is so similar to the "Generate Rules" dialog that its use should be immediately apparent.

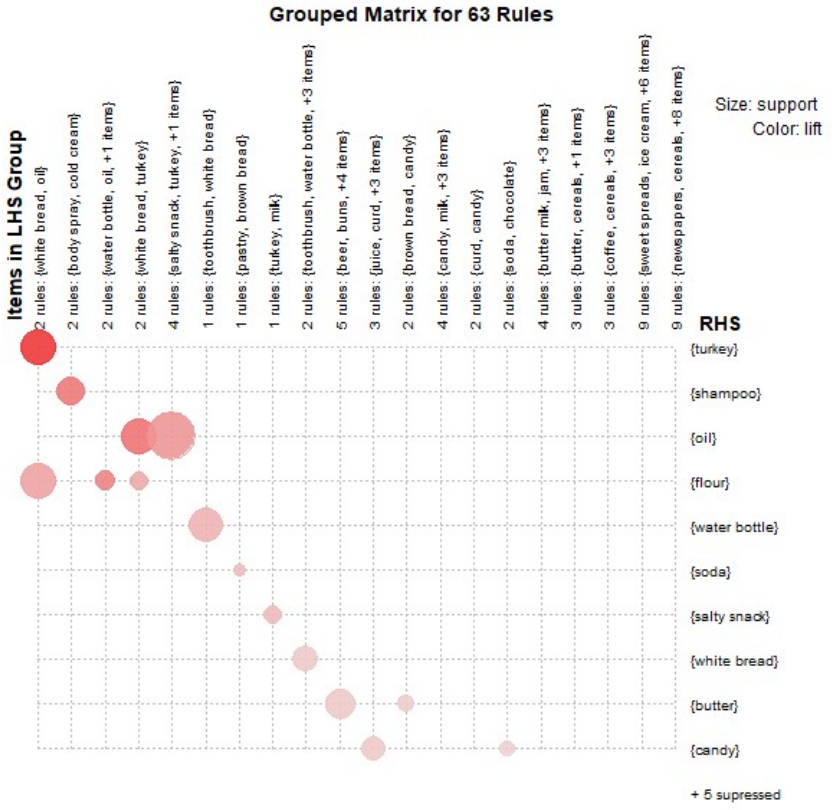


Fig. 6.19: Market basket matrix plot of rules.

6.7.3 Multi-line Transaction Format

Market basket's multi-line transaction format takes all the items in an itemset, stores them with one value on each row, and includes a set ID number. The first two sets of items from our practice dataset appear in Tab. 6.39.

To open this dataset, follow these steps:

1. Go to the Analysis window and choose "File> Open."
2. Browse to the file, "`C:\Program Files\BlueSky Statistics\Sample Datasets and Demos\Market Basket\Multi-line Transactions.csv`" and open it.
3. Check the box labeled, "First row of CSV/TXT file contains column headers."
4. This time, *do not* check the box labeled, "Load all comma separated values in a single column/variable"
5. Click OK.

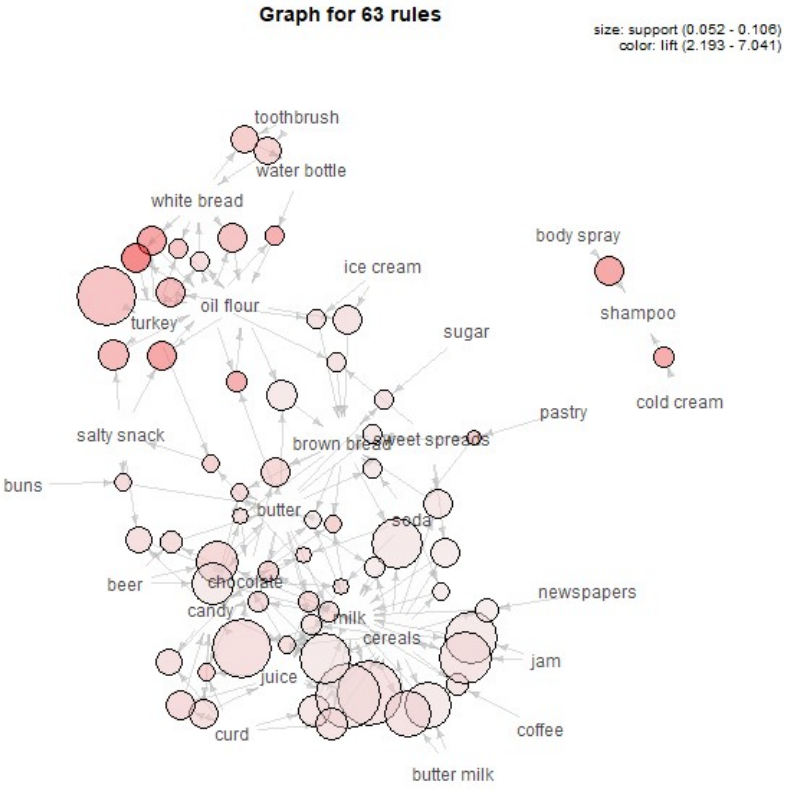


Fig. 6.20: Market basket network plot of items.

The data should open looking as in Tab. 6.39. You can now continue to any other section which uses this type of data to generate rules or plots.

Generate Rules, multi-line

The item "Analysis> Market Basket> Multi-line Transaction Format> Generate Rules" works in almost the exact same way as the section on basket data format, Sec. 6.7.1. The only difference is that instead of supplying a single variable that contains a complete itemset, in this case, you supply a "Transaction id column name" and a "Transaction items column name."

Using the data opened in Sec. 6.7.3, you can generate rules using these steps:

1. Choose "Analysis> Market Basket> Multi-line Transaction Format> Generate Rules."
2. Select "id" as the "Transaction id column name."
3. Select "name" as the "Transaction items column name."

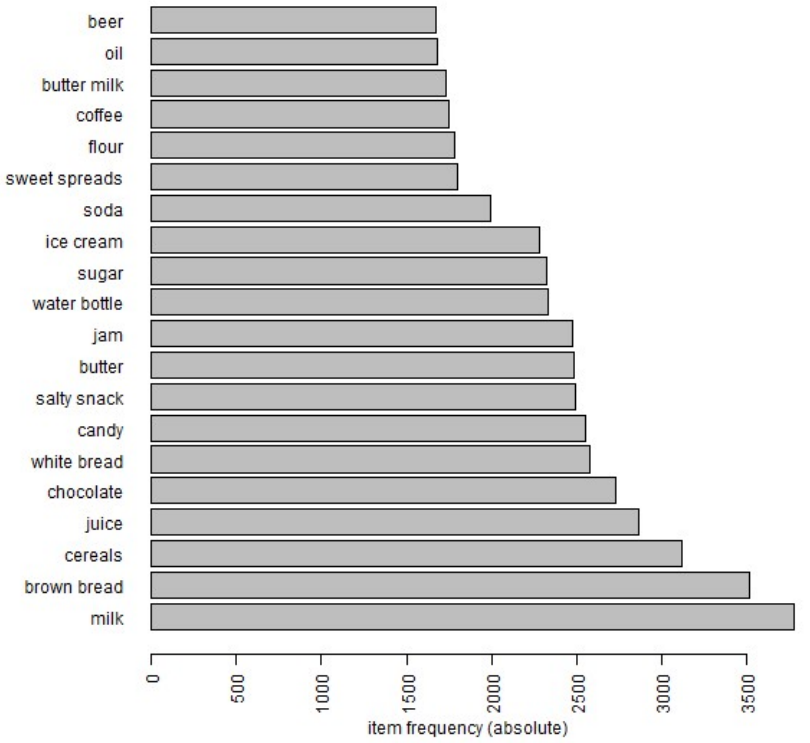


Fig. 6.21: Market basket frequency plot of items.

4. Set "Minimum Support Value" to 0.09 (you can play around with that number), then click, OK.

Your rules should appear as shown in Tab. 6.40. Rule interpretation and example rule plots are covered in Sec. 6.7.1.

Item Frequency Plot, basket data

The item "Analysis> Market Basket> Multi-line Transaction Format> Item Frequency Plot, multi-line" creates a simple bar plot of item counts.

Here are the steps to follow to create an example plot:

1. Choose "Analysis> Market Basket Analysis> Basket Data Format> Item Frequency Plot, basket data."
2. Select "id" as the "Transaction id column name."
3. Select "name" as the "Transaction items column name."
4. Check the box, "Show bars horizontally," then click OK.

The bar plot of counts will appear as shown in Fig. 6.22.

Table 6.38: Market basket analysis with beer target.

LHS	RHS	Support	Confidence	Lift
c("pastry", "salty snack")	beer	0.0128	1	5.9666
c("ham", "toothpaste")	beer	0.0115	1	5.9666
c("toothpaste", "turkey")	beer	0.0115	1	5.9666
c("buns", "toothpaste")	beer	0.0115	1	5.9666
c("oil", "toothpaste")	beer	0.0115	1	5.9666
c("buns", "toothbrush")	beer	0.0109	1	5.9666
c("ham", "sweet spreads")	beer	0.0115	1	5.9666
c("coffee", "turkey")	beer	0.0112	1	5.9666
c("buns", "coffee")	beer	0.0105	1	5.9666
c("coffee", "oil")	beer	0.0112	1	5.9666
c("butter", "coffee")	beer	0.0105	1	5.9666
c("buns", "turkey")	beer	0.0224	1	5.9666
c("buns", "sweet spreads")	beer	0.0115	1	5.9666
c("buns", "oil")	beer	0.0224	1	5.9666
c("soda", "white bread")	beer	0.0099	1	5.9666
c("coffee", "pastry", "sugar")	beer	0.0128	1	5.9666
c("coffee", "pastry", "salty snack")	beer	0.0128	1	5.9666
c("pastry", "soda", "white bread")	beer	0.0099	1	5.9666
c("cereals", "pastry", "soda")	beer	0.0099	1	5.9666
c("pastry", "salty snack", "sugar")	beer	0.0128	1	5.9666

Table 6.39: Market basket multi-line transaction format.

id	name
1	Apple
1	Mango
1	Banana
1	Grapes
2	Pineapple
2	Avocado
2	Strawberry
2	Orange...

Targeting Items, basket data

The item "Analysis> Market Basket Analysis> Basket Data Format> Item Frequency Plot, basket data" allows you to choose a target item and view the rules that affect it. This is described in more detail in Sec. 6.7.1. Here we examine a version applied to the multi-line dataset format we opened back in Sec. 6.7.3. To target an item on such a dataset, follow these steps:

1. Choose "Analysis> Market Basket Analysis> Basket Data Format> Target Item, basket data."
2. Select "id" as the "Transaction id column name."
3. Select "name" as the "Transaction items column name."

Table 6.40: Market basket rules for the multi-line fruit data.

LHS	RHS	Support	Confidence	Lift
c("pclass=1st", "sex=female")	survived=1	0.1224	0.9624	2.3576
c("pclass=2nd", "survived=0")	sex=male	0.1291	0.9247	1.4699
c("survived=0", "age=[34,80]")	sex=male	0.1855	0.894	1.4212
c("survived=0", "age=[23,34)")	sex=male	0.175	0.8592	1.3658
c("pclass=2nd", "sex=male")	survived=0	0.1291	0.8544	1.4438
survived=0	sex=male	0.5	0.8449	1.3431
c("survived=0", "age=[0.167,23)")	pclass=3rd	0.1511	0.836	1.7454
c("pclass=3rd", "sex=male")	survived=0	0.2772	0.8309	1.4042
c("sex=male", "age=[34,80]")	survived=0	0.1855	0.8291	1.401
c("sex=female", "age=[34,80]")	survived=1	0.1013	0.8217	2.0129
c("pclass=3rd", "sex=male", "age=[0.167,23)")	survived=0	0.1128	0.8194	1.3847
c("survived=0", "sex=male", "age=[0.167,23)")	pclass=3rd	0.1128	0.8082	1.6874

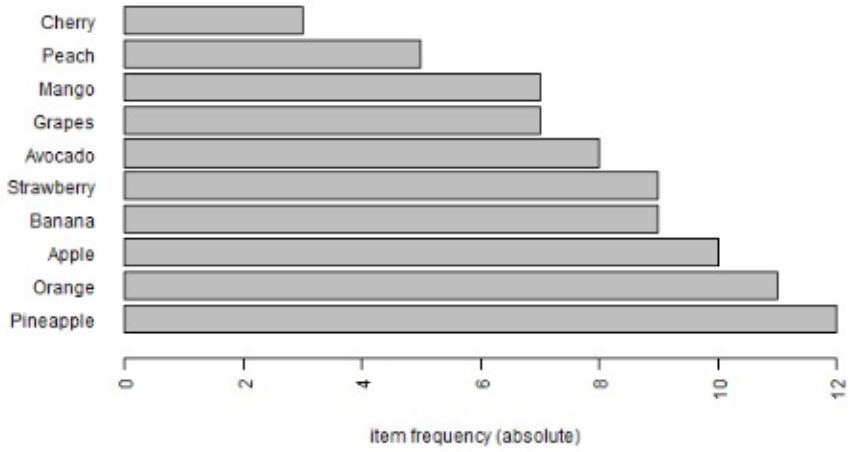


Fig. 6.22: Bar plot of item counts from a multi-line format dataset.

Table 6.41: Market basket rules for fruit target.

LHS	RHS	Support	Confidence	Lift
c("Avocado", "Cherry")	Banana	0.0645	1	3.4444
c("Avocado", "Cherry", "Pineapple")	Banana	0.0645	1	3.4444

4. Set “Minimum Support Value” to 0.05.
5. Set “Items appearing on right hand side” to Banana.
6. Set “Default Appearance” to lhs, then click OK.

Table 6.41 shows the main output. We see that the purchase of avocados and cherries increases the probability that the person will purchase bananas by 3.44 times.

etitanic (etitanic)

	pclass	survived	sex	age	sibsp	parch
1	1st	1	female	29	0	0
2	1st	1	male	0.916700006	1	2
3	1st	0	female	2	1	2
4	1st	0	male	30	1	2
5	1st	0	female	25	1	2
6	1st	1	male	48	0	0
7	1st	1	female	63	1	0
8	1st	0	male	39	0	0

Fig. 6.23: Market basket data in multiple variable format.

6.7.4 Multiple Variable Format

In market basket analysis, multiple variable format looks the most familiar. The various items are simply variables in columns, and an itemset is simply the row of values. Let us examine the Titanic data, which is stored in this format. It will allow us to see how people's attributes, such as gender or passenger class, affected their probability of surviving.

1. Go to the File menu and choose "Open."
2. Navigate to the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \Titanic.RData" select it, and click "Open."

The data will appear as shown in Fig. 6.23. In the following sections, we will use this data to do the same three tasks we did with the other two data formats: generate rules, plot item counts, and generate targeted rules.

Generate Rules, multiple variable

When using the multiple variable data format described in Sec. 6.7.4, generating rules often results in sets of rules that are not of interest. Let us look at an example.

1. Choose "Analysis> Market Basket Analysis> Multiple Variable Format> Generate Rules, multiple variable format."
2. Select "id" as the "Transaction id column name."
3. Select "name" as the "Transaction items column name."
4. Set "Minimum Support Value" to 0.10.
5. Set "Minimum Confidence Value" to 0.80, then click OK.

Table 6.42 will appear. Since the table is sorted in confidence order by default, we see that the combination of being a female in 1st class leads to survival 96.24% of the time. The lift value shows that being a female in 1st class increases the probability of survival by 2.36 times.

However, the second rule shows the rather nonsensical "prediction" that being a 2nd class passenger who did not survive led to being male 92.47%

Table 6.42: Market basket rules for Titanic dataset.

LHS	RHS	Support	Confidence	Lift
c("pclass=1st", "sex=female")	survived=1	0.1224	0.9624	2.3576
c("pclass=2nd", "survived=0")	sex=male	0.1291	0.9247	1.4699
c("survived=0", "age=[34,80]")	sex=male	0.1855	0.894	1.4212
c("survived=0", "age=[23,34)")	sex=male	0.175	0.8592	1.3658
c("pclass=2nd", "sex=male")	survived=0	0.1291	0.8544	1.4438
survived=0	sex=male	0.5	0.8449	1.3431
c("survived=0", "age=[0.167,23)")	pclass=3rd	0.1511	0.836	1.7454
c("pclass=3rd", "sex=male")	survived=0	0.2772	0.8309	1.4042
c("sex=male", "age=[34,80]")	survived=0	0.1855	0.8291	1.401
c("sex=female", "age=[34,80]")	survived=1	0.1013	0.8217	2.0129
c("pclass=3rd", "sex=male", "age=[0.167,23)")	survived=0	0.1128	0.8194	1.3847
c("survived=0", "sex=male", "age=[0.167,23)")	pclass=3rd	0.1128	0.8082	1.6874

of the time! That is because market basket analysis views every "item" as being potentially interesting to predict. Therefore, with this type of data, it is often of greater interest to target a particular item or event as we will do in the next section (Sec. 6.7.4.)

The third rule shows how the discretization process works. The age category is shown as [34,80], i.e., from 34 to 80 years old is now viewed as an "item" to combine into itemsets for predicting, or for being predicted.

Targeting Items, basket data

In this section, we will target specific items for datasets stored in multiple variable format. We will do this using the Titanic dataset described in Sec. 6.7.4. Once you have that data open, then follow these steps:

- Choose "Analysis> Market Basket Analysis> Basket Data Format> Target Item, basket data."
- Select the variables pclass, sex, age and survived as "Variables representing items in a basket."
- Set "Minimum Support Value" to 0.05.
- Set "Minimum Confidence Value" to 0.40.
- In "Items appearing on right hand side" enter "survived=1". Note that there are *no spaces* around the equal sign and that a single equal sign is used, not the double one, "==" that is generally used in R for logical equivalence. Be careful with the spelling too, or it will tell you there is no such item.
- In the "Default Appearance" box, choose "lhs" to display items on the Left Hand Side, then click OK.

The output shown in Tab. 6.43 will appear. Now the right hand side contains predictions only on survival, and the left hand side includes all the combinations that affect it. The line labeled "character(0)" is an empty line in the dataset.

Table 6.43: Market basket Titanic rules when targeting survived=1.

LHS	RHS	Support	Confidence	Lift
c("pclass=1st", "sex=female")	survived=1	0.1224	0.9624	2.3576
c("pclass=1st", "sex=female", "age=[34,80]")	survived=1	0.0688	0.96	2.3517
c("pclass=2nd", "sex=female")	survived=1	0.088	0.8932	2.188
c("sex=female", "age=[34,80]")	survived=1	0.1013	0.8217	2.0129
sex=female	survived=1	0.2792	0.7526	1.8436
c("sex=female", "age=[23,34)")	survived=1	0.0841	0.7458	1.8269
c("sex=female", "age=[0.167,23)")	survived=1	0.0937	0.695	1.7026
pclass=1st	survived=1	0.173	0.6373	1.5612
c("pclass=1st", "age=[34,80]")	survived=1	0.0994	0.5843	1.4313
c("pclass=3rd", "sex=female")	survived=1	0.0688	0.4737	1.1604
pclass=2nd	survived=1	0.1099	0.4406	1.0793
age=[0.167,23)	survived=1	0.1386	0.4341	1.0635
character(0)	survived=1	0.4082	0.4082	1
age=[34,80]	survived=1	0.1396	0.4022	0.9853

6.7.5 Plot Rules

The item "Analysis> Market Basket> Plot Rules," creates interactive network plots of rules. You can use it on rule sets created by any of the various methods described in previous sections. As seen in Fig. 6.22, rule network plots often get very messy and hard to read. An interactive plot will allow you to resize, rotate, and otherwise interact with these plots, making them much easier to read.

An essential prerequisite to making these plots is to install the Java Runtime Environment (JRE), available from https://java.com. Be sure to install the version that matches the type of BlueSky you are running. For example, if you are running 64-bit BlueSky (recommended), then you must also get the 64-bit JRE.

Here are the steps to make an interactive network plot:

1. Install the JRE that matches your version of BlueSky.
2. Choose "Analysis> Market Basket Analysis> Basket Data Format> Target Item, basket data."
3. Select "id" as the "Transaction id column name."
4. Select "name" as the "Transaction items column name."
5. Enter a "Rules object name" such "Rules1"
6. Set "Minimum Support Value" to 0.05.
7. Set "Items appearing on right hand side" to Banana.
8. Set "Default Appearance" to lhs, then click OK.
9. Choose "Analysis> Market Basket Analysis> Plot Rules" and fill in the name of your rules object.
10. Choose graph as your visualization method.
11. Make sure interactive is set to TRUE, then click OK.

The plot shown in Fig. 6.24 should appear. Note that resizing the window does not cause the plot to change size right away. You have first to choose "View> Fit to Screen" in that plot window. You can also have it redraw the

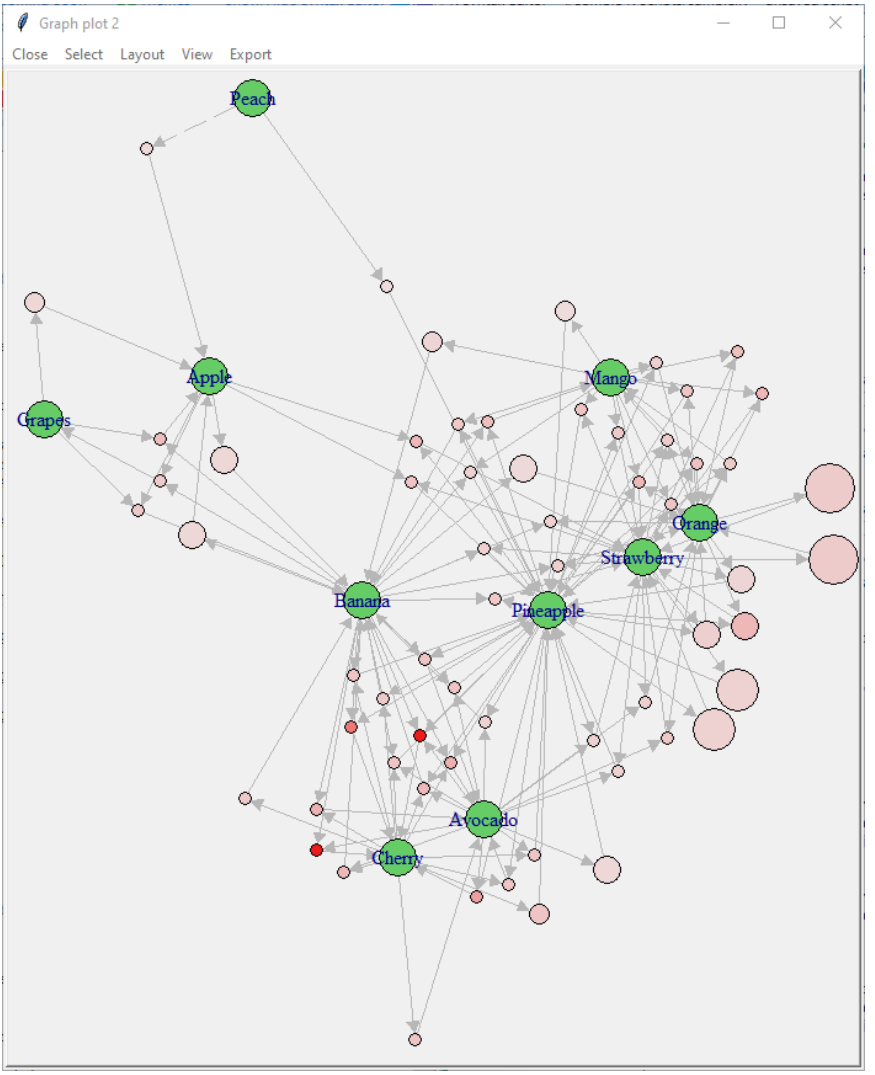


Fig. 6.24: Market basket interactive plot of rules.

plot to fill the window by choosing an item from the Layout menu, such as Random. There are several layouts possible. Those named after their developers are optimized in various ways to clarify the relationships among the items plotted.

The best way to learn the many options these plots have is to try them in various combinations. While there is a web page for the `iplots` package that BlueSky uses to create these plots at http://rosuda.org/software/iPlots/, it does not cover its network plots.

Table 6.44: T-test group statistics.

Group Statistics

		N	Mean	Std Deviation	Std Error Mean
extra	1	10	0.75	1.789	0.5657
extra	2	10	2.33	2.0022	0.6332

6.8 Means

The "Analysis> Means" menu contains many different methods for testing means, including t-tests, analysis of covariance, and one- and two-way analysis of variance.

More complex mean comparisons are possible in the Model Fitting menu. In particular, the Mixed Models, Sec. 8 item there handles analysis of variance models that include both fixed and random effects.

6.8.1 T-test, independent samples

The t-test is the most common analysis to test for mean differences between two independent or unrelated groups. This menu choice is designed for "long" format data, where the data for the two groups are stacked into a single numeric variable, and a factor variable identifies group membership. If you have "wide" data that has two numeric variables to compare, use the approach demonstrated in Sec. 6.8.2.

Let us try an example. A sleep study used drugs to attempt to get people extra sleep at night. That variable is approximately normally distributed, so let us use that to perform a t-test using the following steps:

1. In the Analysis window, use "File> Load Dataset from a Package" and enter "datasets" as the package name and "sleep" as the dataset name.
2. Choose "Analysis> Means> T-Test, independent samples."
3. Move the variable "extra" into the Selected Variables box.
4. Move the variable "group" into the "Single factor variable" box.
5. Leave the Alternative Hypothesis box at its default setting of "Population Mean != mu" setting.
6. Click, "OK."

The first piece of output contains the group sample sizes, means, standard deviations, and standard errors (Tab. 6.44). The second table is rather messy, so I display it in two different figures. The left-hand side of the table (Tab. 6.45) contains Levene's test for equality of variances. The F value is 0.2482, and the matching p-value is labeled "Sig" for significance, which is 0.6244. Since this is not less than the standard cutoff of 0.05, we cannot reject the hypothesis that the two groups have equal variances.

The right-hand side of the table contains two versions of the t-test (Tab. 6.46). The top row is labeled as the "Equal variances assumed" version,

Table 6.45: T-test output Levene test of equality of variances. Table is continued toward the right-hand side in Tab. 6.46.

		Levene's Test for Equality		
		F	Sig.	t
		F	Sig.	t
extra	Equal variances assumed	0.2482	0.6244	-1.8608
extra	Equal variances not assumed			-1.8608

Table 6.46: T-test statistics. The top row assumes equal variances, bottom row does not. This is the right-hand side of a very wide table that begins with Tab. 6.45

T-Test for equality of means						
t	df	Sig.(2-tailed)	Mean Difference	Std. Error Difference	Confidence interval of the Diff.	
t	df	Sig.(2-tailed)	Mean Difference	Std. Error Difference	lower	Upper
-1.8608	18	0.0792 .	-1.58	-0.0674	-3.3639	0.2039
-1.8608	17.7765	0.0794 .	-1.58		-3.3655	0.2055

while the lower row contains the "Equal variances non assumed" version. If you wish to choose a t-test based on Levene's test, select the top one, for a t value of -1.86 with 18 degrees of freedom, and a p-value of 0.0792. Since that is not less than 0.05, we cannot reject the hypothesis that the means are equal.

That describes the two-tailed tests. If your initial alternative hypothesis was that group two's mean would be greater, then this p-value would be $0.0792 \div 2 = 0.0396$. That is the value you would obtain if you re-ran this choosing "Population Mean $<$ mu". Keep in mind that if you choose a one-tailed test and the group you hypothesized would be smaller comes out larger, your p-value will become insignificant even if it would have been very significant if you had done a two-tailed test! Because of the temptation to change the directions of hypotheses after the fact, journals may be reluctant to accept results that only become significant when doing a one-tailed test.

Table 6.47: T-test output for "wide" format data. Note that the results are the same as the "Equal variance not assumed" version shown in the bottom row of Tab.6.46.

Welch Two Sample t-test

	Null Value Considered: 0					
			sample estimate	sample estimate	confidenc e: 0.95	confidenc e: 0.95
t	df	p-value	mean of x	mean of y	lower	upper
-1.8608	17.7765	0.0794 .	0.75	2.33	-3.3655	0.2055

Note. Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Of course, if prior results have shown that direction, they should be fine with a one-tailed test.

BlueSky performs the calculations in this section using the `stats` package's `t.test` function.

6.8.2 T-test, independent samples (two numeric variables)

The t-test is the most common analysis to test for mean differences between two independent or unrelated groups. This menu choice is designed for "wide" format data, where the numeric variables for the two groups appear side-by-side as two variables. This structure is very unusual for independent samples that I advise you to be doubly sure that these variables are indeed independent. If they are, what on earth were they doing in this data structure? What logic paired them in the dataset in the first place?

We are going to use the same dataset we used in the "long" version analysis that we used in Sec. 6.8.1, so we need to convert it to "wide" format first, then do the t-test using the following steps:

1. In the Analysis window, use "File> Load Dataset from a Package" and enter "datasets" as the package name and "sleep" as the dataset name.
2. Begin to reshape the dataset by choosing "Data>Reshape> Reshape long to wide"
3. Enter the new dataset name of, "sleep_wide" (top box).
4. Choose "group" as the repeated factor (middle box).
5. Choose "extra" as the repeated value (bottom box).
6. Begin the t-test by choosing "Analysis> Means> T-test, independent samples (two numeric variables)."
7. Choose X1 and X2 as first and second numeric variables.
8. Leave the Alternative Hypothesis box at its default setting of "Difference != 0" setting.
9. Click, "OK."

Table 6.47 shows the output. The title says, "Welch Two Sample t-test." The Welch version of the t-test assumes that the variances of the two group are not equal. Therefore the table shows only the bottom row from Tab.6.46.

They both describe two-tailed tests. If your initial alternative hypothesis was that group two's mean would be greater, then this p-value would be $0.0792 \div 2 = 0.0396$. That is the value you would obtain if you re-ran this choosing "Difference < 0". Keep in mind that if you choose a one-tailed test and the group you hypothesized would be smaller comes out larger, your p-value will become insignificant even if it would have been very significant if you had done a two-tailed test! Because of the temptation to change the direction of hypotheses after the fact, journals may be reluctant to accept results that only become significant when doing a one-tailed test. Of course, if prior published results have already established that direction, the journal editors should be fine with a one-tailed test.

BlueSky performs the calculations in this section using the `stats` package's `t.test` function.

6.8.3 T-test, one sample

A one sample t-test compares the mean of one variable to a given value. That value might be a known value of a population. It subtracts value from each case's measure and then compares the mean to zero. When the variable in question is a difference between pairs of subjects, the result is precisely the same as a paired-samples t-test. The only advantage the latter provides is to do the subtraction for you.

Let us assume that it is known that a particularly arduous workout makes people need an extra 2.5 hours of sleep the next day. A drug is used to reduce the effects of fatigue and we wish to know if, on average, the amount of extra sleep needed is significantly reduced. We can answer that question using the following steps:

1. In the Analysis window, use "File> Load Dataset from a Package" and enter "datasets" as the package name and "sleep" as the dataset name.
2. Begin the single-sample t-test by choosing "Analysis> Means> T-test, one sample."
3. Select the variable "extra."
4. Leave the Alternative Hypothesis box at its default setting of "Population Mean != mu" setting.
5. Fill in 2.5 as the test value.
6. Click, "OK."

Table 6.48 shows the output. The t statistic is -2.1276, with a p-value of 0.0467. Since that value is below the usual cutoff of 0.05, we can reject the hypothesis that our sample mean is not different from the population mean of 2.5.

This one sample t-test is a two-tailed test. If your initial alternative hypothesis was that the measure would be greater the second time, then this p-value would be $0.0467 \div 2 = 0.0233$. That is the value you would obtain if you re-ran this choosing "Difference < 0". Keep in mind that if you choose a one-tailed test and the group you hypothesized would be smaller comes out larger, your p-value will become insignificant even if it would have been very

Table 6.48: Single sample t-test output.

One Sample Statistics

	Mean	N	Std Deviation	Std Error Mean
extra	1.54	20	2.0179	0.4512

One Sample Test

	Test Value = 2.5					
	t	df	Sig.(2-tailed)	Mean Difference	Confidence : 0.95	
	t	df	Sig.(2-tailed)	Mean Difference	lower	Upper
extra	-2.1276	19	0.0467 *	-0.96	0.5956	2.4844

Note. Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

significant if you had done a two-tailed test! Because of the temptation to change the direction of hypotheses after the fact, journals may be reluctant to accept results that only become significant when doing a one-tailed test. Of course, if prior published results have already established that direction, the journal editors should be fine with a one-tailed test.

BlueSky performs the calculations in this section using the `stats` package's `t.test` function.

6.8.4 T-test, paired samples

The paired t-test is the most common analysis to test for mean differences between two related measures. Often the first variable is a baseline measure of subject's physiology or knowledge. The experimental condition, such as a drug, or training, is then applied, and the second variable measures the resulting impact. The paired t-test essentially subtracts the second variable from the first and then compares the mean of that difference to zero.

The paired t-test is designed for "wide" format data, where the numeric variables for the two groups appear side-by-side as two variables. If you have "long" data that has a single numeric variable containing the values from both groups along with a factor indicating group membership (that's the most common data structure), then use the approach shown in Sec. 6.8.1. In that example, we assumed that the people were two independent groups of subjects. Here, we assume they are the same subjects measured two times.

We are going to use the same dataset we used in the "tall" version analysis that we used in Sec. 6.8.1, so we need to convert it to "wide" format first, then do the t-test using the following steps:

1. In the Analysis window, use "File> Load Dataset from a Package."
   In the "Select an R Package..." field, enter "datarium".
   In the "Select a Dataset..." field, enter "anxiety."
2. Begin the paired t-test by choosing "Analysis> Means> T-test, paired samples."

Table 6.49: Paired t-test summary statistics. A few values have been truncated from the right side to increase legibility.

	vars	n	mean	sd	median	trimmed	mad	min	max
t1	1	45	16.9156	1.489	17	16.9378	1.3343	13.7	19.8
t3	2	45	15.1978	1.9679	15.3	15.2243	2.0756	11	19.4

Table 6.50: Paired t-test.

			Null Value Considered: 0		
			sample estimate	confidence: 0.95	confidence: 0.95
t	df	p-value	mean of the differences	lower	upper
8.5756	44	0.0000	1.7178	1.3141	2.1215

3. Choose t1 and t3 as the first and second numeric variables.
4. Leave the Alternative Hypothesis box at its default setting of "Difference != 0" setting.
5. Click, "OK."

The first table of output contains summary statistics such as the mean, group size, standard deviation. and so on, shown in Tab. 6.49. The table, shown in Tab. 6.50 contains the t statistic, 8.58, with a p-value of 0.0000, or $p < .001$. In Sec. 6.8.2 we did a very similar t-test, but we did not assume that the cases were paired. The p-value that resulted was a non-significant 0.079. How could such similar-sounding tests yield such different results? Imagine giving a set of patients a drug that miraculously lowered every subject's blood pressure by exactly 3 points. That's not a huge amount, but since everyone got the same improvement, the standard deviation is zero! So it is easy to see that high correlations between the two measures lead to very low variability and a greater chance of getting a significant result. The correlation between our two variables is 0.79, which is plenty of additional information to help reduce unknown sources of variation.

A common mistake is to analyze a paired t-test as an independent samples one. That can lead to costly errors. It is much less likely to analyze independent-samples data as a paired t-test, since there is no obvious way to pair them.

This paired t-test is a two-tailed test. If your initial alternative hypothesis was that the measure would be greater the second time, then this p-value would be $0.0028 \div 2 = 0.0014$. That is the value you would obtain if you re-ran this choosing "Difference < 0". Keep in mind that if you choose a one-tailed test and the group you hypothesized would be smaller comes out larger, your p-value will become insignificant even if it would have been very significant if you had done a two-tailed test! Because of the temptation to

change the direction of hypotheses after the fact, journals may be reluctant to accept results that only become significant when doing a one-tailed test. Of course, if prior published results have already established that direction, the journal editors should be fine with a one-tailed test.

BlueSky performs the calculations in this section using the `stats` package's `t.test` function.

6.8.5 ANCOVA

The goal of Analysis of Covariance (ANCOVA) is to test for group differences after controlling for the effects of a covariate. In this section's example, we will see if three types of treatments can lower anxiety. Comparing three groups could be done with a one-way analysis of variance. However, the groups might not have had comparable levels of anxiety before the treatment. ANCOVA can correct for any pre-test differences by fitting a regression line between the pretest and the posttest, then "predicting" what the posttest mean would have been if each group had had the same pretest value. The catch with this approach is that the regression lines for the groups must be parallel, or the correction they apply would change depending on what level of pretest you wanted to compare the corrected means at. Let us study an example:

1. In the Analysis window, use "File> Load Dataset from a Package."
   In the "Select an R Package..." field, enter "datarium".
   In the "Select a Dataset..." field, enter "anxiety."
2. Begin the ANCOVA by choosing "Analysis> Means> ANCOVA."
   In the "Dependent Variable" field, enter t3.
   In the "Fixed Factor" field, enter group.
   In the "Covariate" field, enter t1.
   Check all the option boxes, then click OK.

Although it is not the first piece of output, let us examine Fig. 6.25. ANCOVA uses linear regression, so the lines for each group should be close to straight. With this type of smoothed fit, the lines will never be perfectly straight, but these are pretty close. We also get the idea that the slopes of the lines are parallel. Let us see if the statistical results verify that impression.

In Tab. 6.51, the term "group:t1" is what tests to see if each group needs its own slope. As usual, the null hypothesis is that there are no differences among the slopes of the groups. The p-value of 0.42 is well above the standard 0.05 cutoff, so we cannot reject the null hypothesis. In this case, that is a good outcome since that is an assumption of ANCOVA. Once we make that determination, we can drop that interaction term, which forces the groups' slopes to be the same. In that case, the differences in the y-intercepts will show us what the overall mean differences are.

Table 6.52 shows the ANOVA table with the slope interaction term removed. The test for the group variable here is group corrected for the impact of our covariate, t1. So we know there are indeed group differences on our posttest variable, t3, but we do not know which groups differ.

The linear model summary table (not shown) provides information regarding the overall model, which is not particularly useful in this case. It shows the model has an R-squared value of .94, and the overall model is significant, though the F-test it provides doesn't get more specific than that.

A residuals summary table follows (not shown). Residuals are the model's prediction errors, so we would like them to be small, hopefully within plus or minus 3 (their standard deviation). They range from -1.49 to 0.81 so those look fine. If your model has residuals that are far beyond plus or minus 3, the model might not be fitting very well, and you might want to think about other covariates that you could add.

Table 6.53 shows the model coefficients. Linear regression equations are not designed to handle factor variables. If we put our group variable in as a numeric variable, with values 1, 2, and 3, it would be implying that group 2 is twice as important as group 1! That makes no sense, so what the underlying R code does is convert factors into "dummy" variables. In this case, a new variable that it named "groupgrp2" has the values 1 for group 2 people and zero otherwise. Similarly, "groupgrp3" has the values 1 for group 3 people and zero otherwise. There is no variable needed for people in group 1 because it would be redundant. If a person is not in groups 2 and 3, then they must be in group 1.

Dummy variables provide a change in y-intercept, so the coefficient estimate for "groupgrp2" of -.55 means that group 2 has a mean posttest score, t3, that is about a half-point lower than group 1. The coefficient estimate for "groupgrp3" is -2.87, which is that far below the covariate corrected mean of group 1.

A plot of fitted values (i.e., predictions) against the model residuals is shown in Fig. 6.26. Ideally, the points would show no sign of bias; that is, there should be around the same number of values above and below the zero value on the y-axis across predictions' range. The vertical spread of residuals should also be consistent across the range of the fitted values, indicating homogeneity of variance. If that is not the case, transforming the numeric variables might help (e.g. using a logarithm or square root).

Figure 6.27 shows a histogram of the model residuals. The residuals should be normally distributed in order for the p-values to be accurate. If they pile up on one side or the other, transformations can often fix the problem.

BlueSky performs the calculations in this section using the `stats` package's `lm` function, followed by the `car` package's `Anova` function.

6.8.6 ANOVA, one-way and two-way

Analysis of Variance (ANOVA) is a method that allows you to compare groups on the mean of some numeric measure. We will examine the mean of a job satisfaction score to see if males differ from females and if people with various levels of education differed. ANOVA assumes that the model's errors (residuals) are normally distributed and have roughly equal variability in all of the groups. To begin our example, follow these steps:

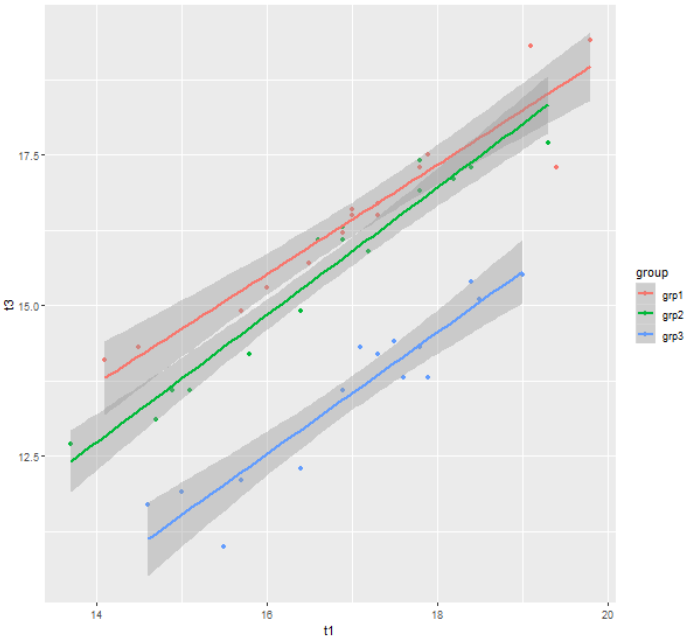


Fig. 6.25: ANCOVA slopes plot.

Table 6.51: ANOVA table with interaction term.

	Sum Sq	Df	F value	Pr(>F)
group	69.8652	2	150.5178	<.001***
t1	93.3662	1	402.2971	<.001***
group:t1	0.4173	2	0.8989	0.4153
Residuals	9.0512	39	NA	NA

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 6.52: ANOVA table controlling for covariate.

	Sum Sq	Df	F value	Pr(>F)
group	69.8652	2	151.2636	<.001***
t1	93.3662	1	404.2903	<.001***
Residuals	9.4685	41	NA	NA

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1. In the Analysis window, use "File> Load Dataset from a Package."
   In the "Select an R Package..." field, enter "datarium".
   In the "Select a Dataset..." field, enter "jobsatisfaction." Note that there are other datasets named, "jobsatisfaction," so you must enter the package name of "datarium" to get the correct one.

Table 6.53: ANCOVA model coefficients.

Coefficients

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.3535	0.8477	-0.417	0.6789
groupgrp2	-0.5458	0.1768	-3.0873	0.0036 **
groupgrp3	-2.8743	0.1755	-16.3766	<.001***
t1	0.9867	0.0491	20.107	<.001***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

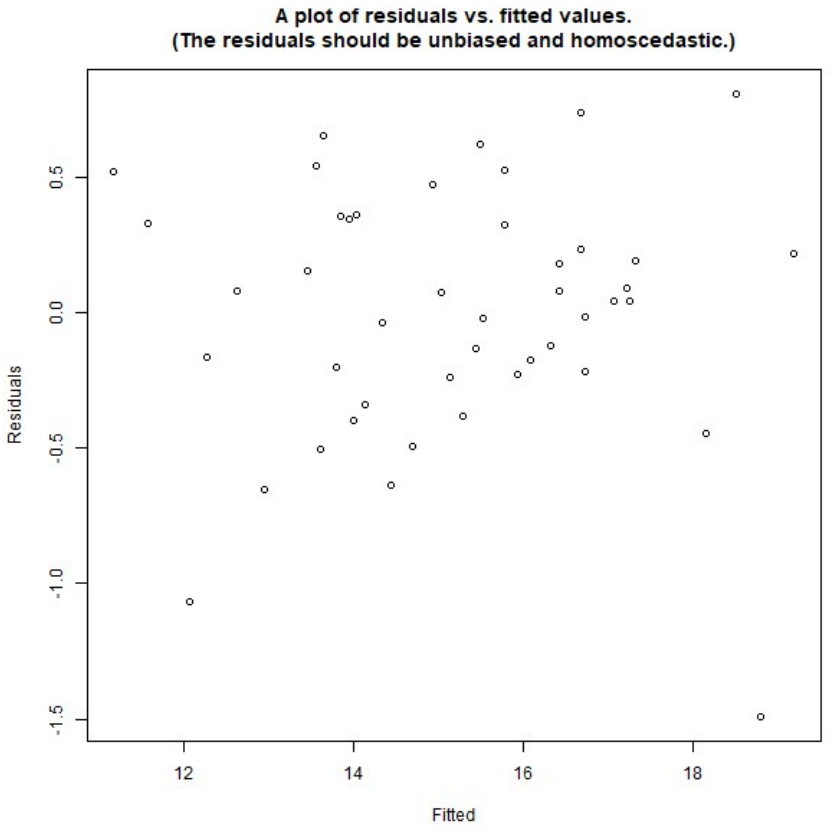


Fig. 6.26: Scatterplot of ANCOVA residuals to examine for homogeneity of variance.

2. Begin the ANCOVA by choosing "Analysis> Means> ANOVA, one-way and two-way."

   In the "Target Variable" field, enter Score.

   In the "Specify a maximum of 2 factor variables" field, enter gender and education_level.

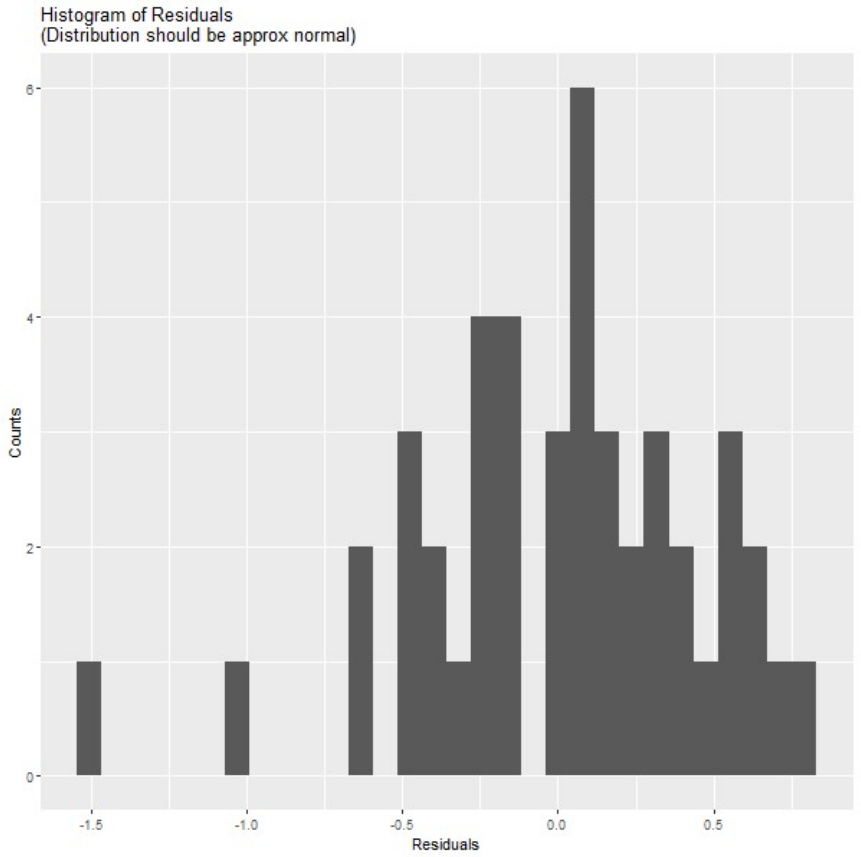


Fig. 6.27: Histogram of ANCOVA residuals, to examine for normality.

In the "Covariate" field, enter t1.

Under Options check all boxes, *except for* "Ignore interaction terms in model."

Under Options choose Holm from the "Adjust p-values using" menu, then click OK (twice).

The first several tables of output (e.g. Tab. 6.54) provide basic statistics such as means and standard deviations for each group or combination of groups (in a two-way analysis). However, these statistics are of only minor interest. Consider a situation in which in the university group, there were twice as many women as men. To calculate the mean for the university group, you might take two approaches. First, you could calculate the arithmetic mean of all individual university student scores. In that case, the women would have much more impact on the mean. That might be fine if you only wanted to get the mean of graduates from a single university that always tended to have that many more women. However, if you instead wanted to generalize your mean to any university, and you knew that typically there were equal numbers of male and female students, you would want to do a

different calculation. In that case, you would get the males' mean, and the female's mean, then average the two. That would give the genders equal weight in the final result. That is what you usually want to report when doing ANOVA. They are called "estimated marginal means" (EMM) or, in other software, "least-squares means." Those will appear after the model runs.

Table 6.54 contains regular (i.e. not EMM) means and various other measures. The minimum (min) and maximum (max) values shown for each group bear careful examination. You should always study them carefully before going further! They help screen out data entry errors before you waste considerable time in trying to interpret the rest of the output. In my 35-year career as a statistician, I have diagnosed more trouble with these simple measures than any other.

Another useful thing to study in Tab. 6.54 is the variance. ANOVA assumes the group variances are equal, and we will examine some ways to visualize this assumption later in this section. But glancing down the column we see the variances range from .1154 to .8797, a ratio of 7.6. ANOVA is known to be robust with regard to violations of this assumption.

The next set of outputs BlueSky creates is a series of four diagnostic plots. Figure 6.28. On the y-axis are the values predicted by the model. Since we have six categories, two genders by three educational levels, there are only six values on that axis. For example, every female with a university education is predicted to score the same. The residuals are the actual values minus the means of each group. They represent how far off each person was from their group mean. Cases 52 and 57 stick out the top and bottom of the group with a fitted value of around 8.5, but they do not appear too extreme.

Another thing to look for is that the means of these points form a horizontal line, like the one shown in red. This indicates that the model is not biased in a particular direction. If the red line were instead curved, or obviously not horizontal, then the model would not be fitting well. In that case, transforming the dependent variable may be helpful.

Figure 6.29 plots the theoretical quantiles (percentiles divided by 100) against the standardized residuals. The result is known as a Z-score, that should have around 99% of its values between plus and minus 3 points, regardless of the original scale of the dependent measure. Before we saw that observation 57 was a bit of an outlier, at around 2 points below the group mean. But without knowing more about the dependent variable, it is hard to know how bad that might be. Now we see that 57 is more than 3 standard deviations away from the mean, which is more useful information. However, it is not that unusual a case. We will see more about influential cases soon.

If the data meet the assumption of normally distributed residuals, then this plot would form a perfect 45° diagonal line, and this is close. Again, only case 57 seems very far from that line.

Figure 6.30 plots the fitted values against the square root of the standardized residuals. ANOVA assumes that the groups all have the same variance, and this plot provides a way to visualize that assumption. The

Table 6.54: Arithmetic means and other statistics.

Summaries for score by factor variables gender*education_level

gender	education_level	n	mean	median	min	max	sd	variance
male	school	9	5.4267	5.51	4.78	5.94	0.3639	0.1324
male	college	9	6.2233	6.3	5.58	6.74	0.3396	0.1154
male	university	10	9.292	9.205	8.7	10	0.4445	0.1976
female	school	10	5.741	5.725	4.93	6.38	0.4744	0.2251
female	college	10	6.463	6.45	5.65	7.1	0.4747	0.2253
female	university	10	8.406	8.48	6.52	9.57	0.9379	0.8797

vertical spread of all groups of points should be roughly the same. These meet that assumption, with the possible exception of the group of points at the predicted value of around 8.5 on the x-axis. Later in the output are two small tables of Levene's test of homogeneity of variance (not shown). Neither the test of gender nor educational level is significant at the usual 0.05 cutoff, so we do not reject the null hypothesis that the group variances are equal.

The last diagnostic plot is shown in Fig. 6.31. It plots leverage on the y-axis, which is a measure of influence each case has on the overall model. If we saw one or two points plotted in the upper right, or lower left of this plot, it would indicate that the cases were particularly influential. In the bottom right we see case 57 again, but its measure of leverage is no different than most of the others. If we had an extreme case or two, some dashed red lines would appear showing another measure of influence known as Cook's distance. For either leverage or Cook's distance, the key thing to look for is a small number of points that plot far away from all the rest. That is usually a sign of either data entry errors or data in need of transformation.

The core of the analysis appears in the ANOVA table itself, Tab. 6.55. It shows the F test for each main effect and the interaction. Gender is not significant, $F(1,52)=1.78$, $p=0.19$ (both rounded off). Education level is significant, $F(1,2)=56.84$, $p<.001$. However, since there is a gender by education level interaction, $F(1,2)=4.44$, $p=0.0016$, interpreting the effect of education level by itself may not make sense (more on that soon). The table even warns at the bottom, "NOTE: Results may be misleading due to the involvement in interactions."

The ANOVA table is followed by several tables of estimated marginal means (emmeans). The means for the genders are nearly identical, which is no surprise given the ANOVA results (Tab. 6.56). The means of job satisfaction at each educational level, shown in Tab. 6.57, appear to increase as education increases. Looking over the 95% confidence intervals, they appear to be significantly different. The pairwise comparisons in Tab. 6.58 indicate that all three educational levels differ. However, we still need to be cautious until we understand its interaction with gender.

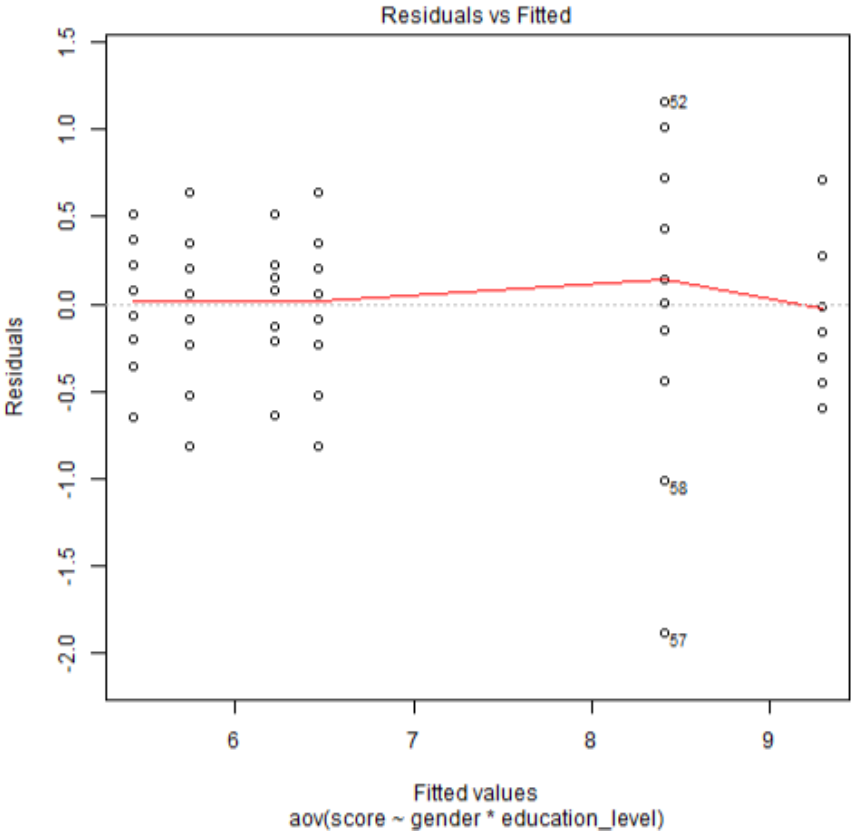


Fig. 6.28: Plot of residuals vs. fitted values.

Since educational level has order to it, we could have made these pairwise comparisons in a more powerful way. On the options dialog, we could have chosen, "Compare Means Using trt.vs.ctrl1". That would have used high school as a control-style group and would have compared the other two levels only to that. Alternatively, we could have chosen "conseq" which would have compared only high school to community college, then community college to university (i.e. sequentially). Whenever you can answer your research question by performing fewer tests, your chances of getting a significant result increase, often substantially.

The table of means for all gender and education levels follows (not shown), but it is complex enough to be hard to interpret. Luckily, the interaction plot shown in Fig. 6.32 clarifies the nature of the interaction. There we see that educational level does seem to have a consistent effect of increasing job satisfaction for both males and females. However, how the genders compare at each level of education is not as simple. For those who stopped at high school education or community college, females have slightly higher means, but not significantly so. For those with university degrees, males are significantly more satisfied than females.

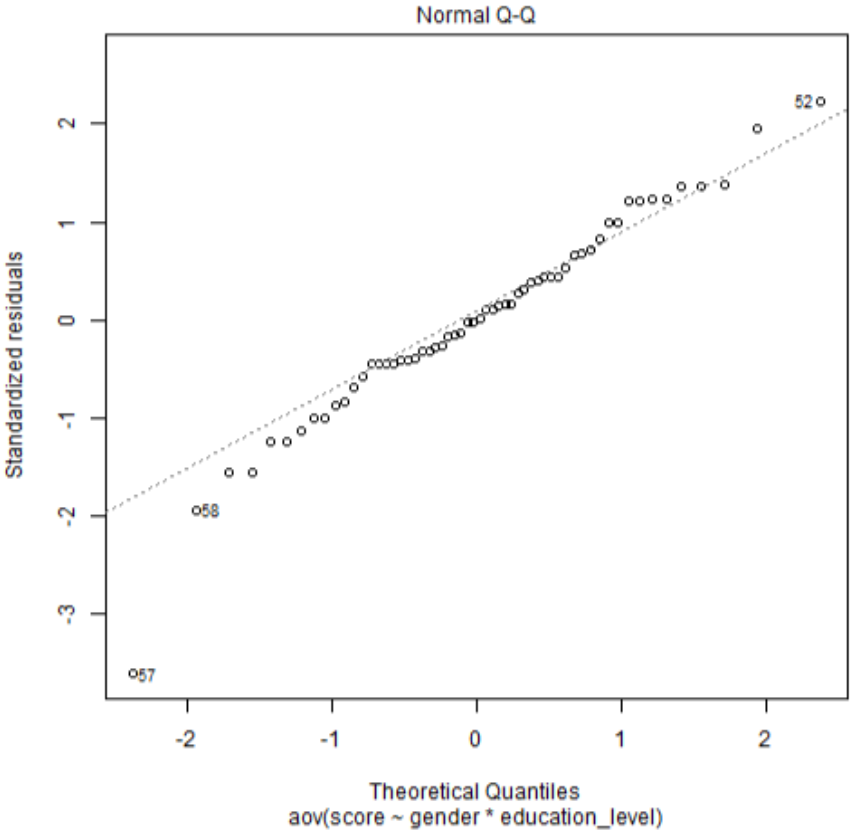


Fig. 6.29: Q-Q plot of residuals. The straight line indicates a normal distribution.

BlueSky performs the calculations used in one- and two-way ANOVA using the packages and functions shown in Tab. 6.59.

6.8.7 ANOVA, one-way with blocks

The Analysis of Variance (ANOVA) test compares groups for mean differences. It does so by comparing the within-group variance to the between-group variance, which is why it is called the analysis of variance rather than the analysis of means. You can imagine that if the means of two groups differed by two points on a school exam, and everyone got between 74 and 76 in one group, and 77 to 79 in another, there is no overlap, and the groups clearly differ. But with the same mean difference, if the groups both ranged from 20-100 on the test, a mean difference of two points is not very much given the broad range of scores within each group.

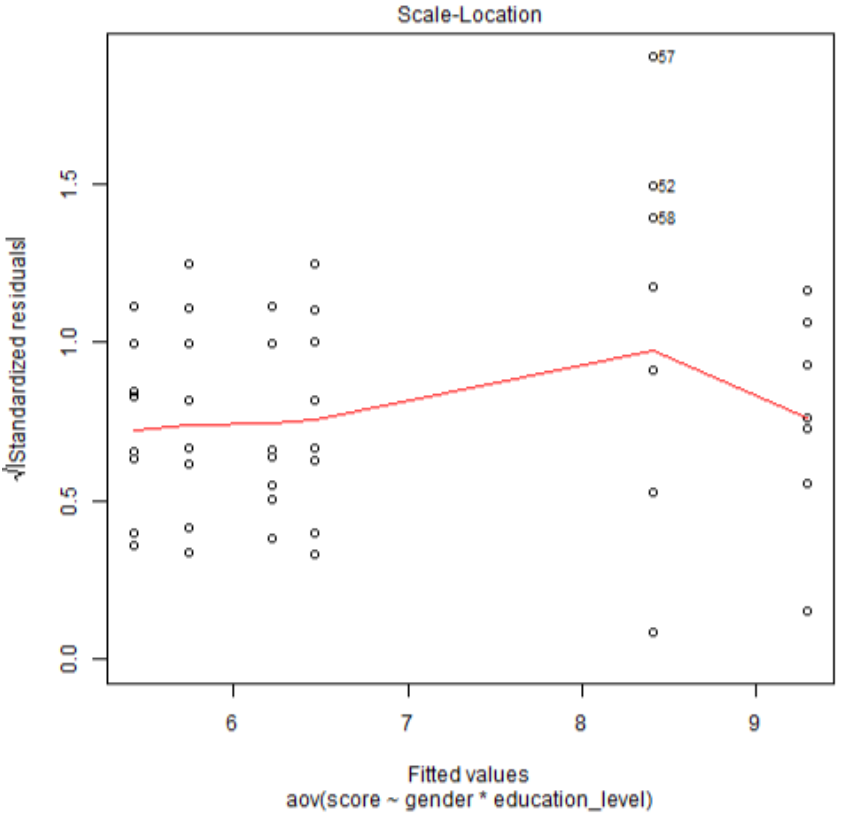


Fig. 6.30: Scale-location plot. The variability for each group should be roughly the same.

Adding a blocking factor to a study decreases unexplained variance. Usually, we do not care about differences between blocks; we just want to reduce the amount of unexplained variance. Reducing unexplained variance makes it easier to get significant differences on other, more interesting factors.

Let us consider an example in which instructors teach a nutritional class using different methods. In this case, the variable "instructor" is essentially tracking "teaching method."

They teach in different towns, and we expect these towns differ, perhaps due to the educational levels of their inhabitants, socioeconomic status etc. but we do not have measures on those, we only know what town the students are taught in. In addition, these are the only towns to which we wish to generalize our findings. That is to say, we do not want to generalize across all towns, for which these are a random sample. These towns are then called "fixed blocks." Here are the steps to analyze this problem.

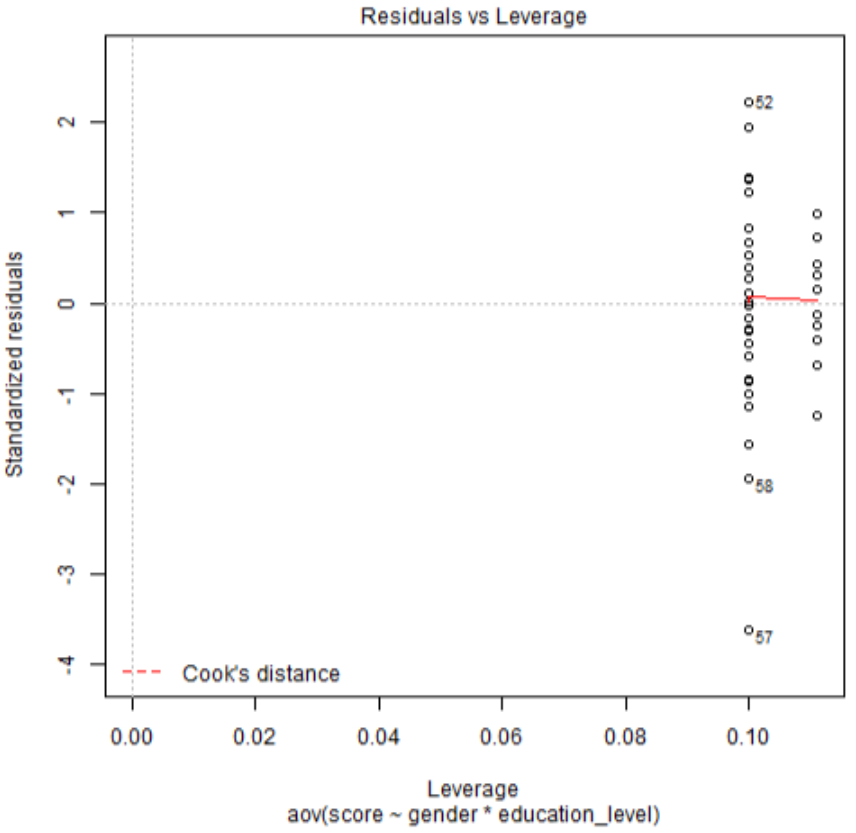


Fig. 6.31: Residuals vs. leverage plot.

Table 6.55: ANOVA table. The various types of sums of squares are explained in the text.

Anova table with type III sum of squares for score by gender*education_level

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
gender	1	0.5407	0.5407	1.7872	0.1871
education_level	2	113.6841	56.8421	187.8921	<.001***
gender:education_level	2	4.4398	2.2199	7.3379	0.0016 **
Residuals	52	15.7313	0.3025	NA	NA

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

NOTE: Results may be misleading due to involvement in interactions
```

Table 6.56: Estimated marginal means for gender.

gender	emmean	SE	df	lower.CL	upper.CL
male	6.9807	0.1041	52	6.7718	7.1895
female	6.87	0.1004	52	6.6685	7.0715

Table 6.57: Estimated marginal means for educational level.

Estimated Marginal Means for score by education_level

education_level	emmean	SE	df	lower.CL	upper.CL
school	5.5838	0.1264	52	5.3303	5.8374
college	6.3432	0.1264	52	6.0896	6.5967
university	8.849	0.123	52	8.6022	9.0958

Table 6.58: Pairwise comparisons for educational level.

Post-hoc tests for score by education_level (using method = pairwise)

contrast	estimate	SE	df	t.ratio	p.value
school - college	-0.7593	0.1787	52	-4.2492	<.001***
school - university	-3.2652	0.1763	52	-18.5172	<.001***
college - university	-2.5058	0.1763	52	-14.2109	<.001***

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 6.59: Packages and functions used for one- and two-way ANOVA.

Package	Function
`stats`	`aov`
`car`	`Anova`
`car`	`leveneTest`
`emmeans`	`emmeans`
`emmeans`	`contrast`
`multcomp`	`cld`

1. Use “File> Open” and browse to the folder: “`C:\Program Files \BlueSky Statistics\Sample Datasets and Demos\ANOVA \OneWayAnova.RData`”.

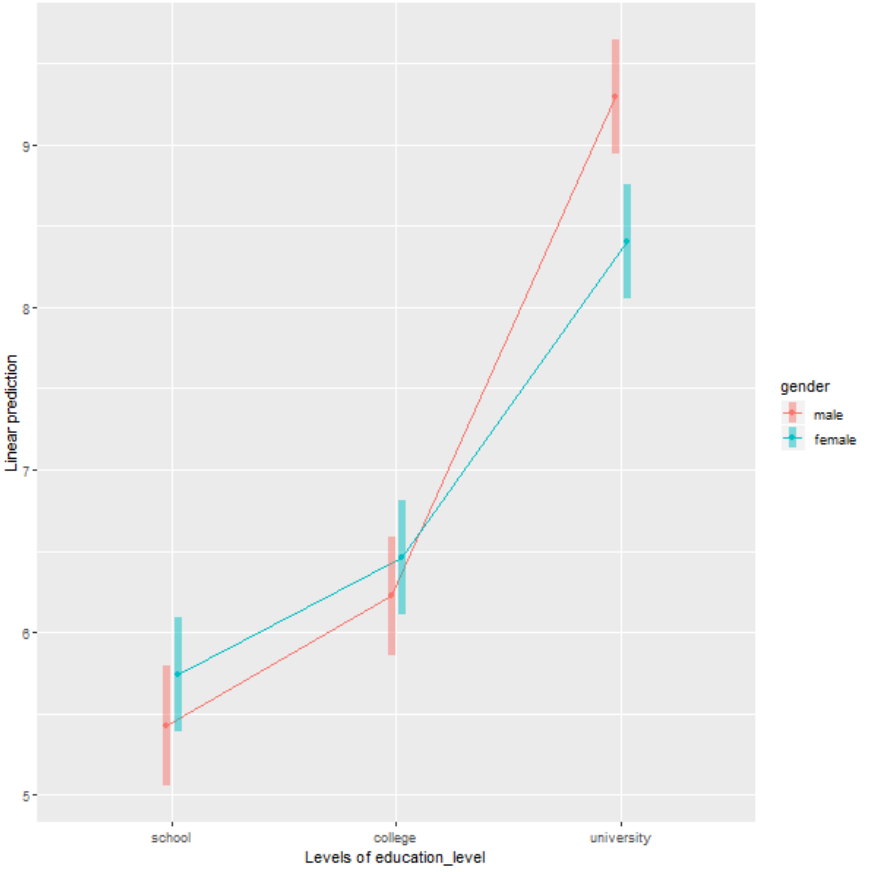


Fig. 6.32: Interaction plot of estimated marginal means with confidence intervals.

2. Choose "Analysis> Means> Oneway ANOVA with Blocks."
3. For the response variable, choose Sodium. We wish to see if one of the teaching methods reduce the use of salt in the participant's diets.
4. For the fixed effect, choose Instructor (synonymous with the teaching method used.)
5. For the blocking variable, choose Town.
6. Make sure to check the histogram and post-hoc analysis boxes, then click OK to run it.

The first table of output shows summary statistics for the fixed effect, Tab. 6.60. A similar table follows that breaks the statistics down by both the fixed effect and the blocking variable(s), Tab. 6.61.

Next comes the overall model summary, Tab. 6.62. We see that the R-squared is 0.3485, indicating that our model explains approximately 34.85% of the variance in sodium score.

The residuals summary comes next, Tab. 6.63. The median residual is 16.27, indicating that our model estimates the means that much higher than their actual values. We would prefer that to be close to zero, but no model is perfect.

The model coefficients are shown next, in Tab. 6.64. These indicate whether there is a y-intercept difference between the factor levels. Each factor has been dummy-coded. We see that Coach McGuirk has a dummy variable that is 1 for him, and zero for anyone else. The same for Melissa Robins. Where is Brendan Small? The default factor level order is alphabetical, so he is the "reference level." That means the analysis is comparing the two to Brendon Small. Since we are hoping for an outcome of low sodium usage, both instructors doing better than Brendan Small, though only Melissa Robins' estimate of -124.47 has a significant p-value. We also see that Metalocalypse has a significantly lower outcome than the other town by 151.12 points, though we do not care about that.

Next, we see the overall ANOVA table, Tab. 6.65. This shows is that Instructor is significant, at a p-value of 0.0388. We have already seen that this is mostly due to one instructor's method. We again see that town had a significant effect. In fact, that is almost exactly the same test. The F value of 15.9332 is the square of the coefficient's t value of -3.9916 and the p-value is identical. That is because there are only two towns to compare, and so only one coefficient to test.

Now we arrive at the table of pairwise comparisons, Tab. 6.66. Here we see all possible comparisons, and only Melissa Robins' teaching method comes out significantly lower than Bredon Small's approach. Note that in the table of coefficients, that comparison had a p-value of 0.0098. Here it is only 0.0261. That is because the Tukey test corrects the p-values for the number of comparisons made. Another way to do that is to multiply the p-value by the number of tests, called the Bonferroni correction. In this case that is $0.0098 \times 3 = 0.0294$, a very similar result.

Next, we see two plots of residuals. Figure 6.33 shows a histogram of residuals. They are supposed to be normally distributed to meet a key assumption of ANOVA. They have a slight positive skew but are in pretty good shape.

Figure 6.34 shows the fitted values plotted against the model residuals. We hope to see them evenly distributed around the zero point on the y-axis without a trend toward having greater spread toward one end or the other. There is a slight increase in the spread from one group to the next, but these look good.

6.8.8 ANOVA, one-way with random blocks

In this section we will continue our discussion of ANOVA with Blocks, Sec. refANOVAwithBlocks. If you have not already done so, read that section before trying this one.

Previously we viewed town as a fixed effect. More often, though, blocks are viewed as random effects. That is because we are hoping to generalize what we learn about the fixed effect, the instructor's teaching method,

Table 6.60: Summaries for the fixed effect.

Instructor	n	mean	sd	min	Q1	median	Q3	max
Brendon Small	20	1287.5	193.734	950	1150	1300	1400	1700
Coach McGuirk	20	1246.25	142.412	1000	1143.75	1212.5	1350	1525
Melissa Robins	20	1123.75	143.149	900	1006.25	1112.5	1231.25	1400

Table 6.61: Summaries for the fixed effect and blocking variables.

Instructor	Town	n	mean	sd	min	Q1	median	Q3	max
Brendon Small	Squiggleville	11	1336.364	162.928	1150	1225	1300	1400	1700
Coach McGuirk	Squiggleville	14	1292.857	134.96	1100	1200	1287.5	1387.5	1525
Melissa Robins	Squiggleville	6	1275	74.162	1200	1231.25	1250	1306.25	1400
Brendon Small	Metalocalypse	9	1227.778	220.597	950	1050	1300	1325	1600
Coach McGuirk	Metalocalypse	6	1137.5	97.147	1000	1068.75	1162.5	1200	1250
Melissa Robins	Metalocalypse	14	1058.929	112.919	900	950	1100	1118.75	1300

Table 6.62: Summary of the entire model

LM Summary							
Residual Std. Error	df	R-squared	Adjusted R-squared	F-statistic	numdf	dendf	p-value
143.8167	56	0.3485	0.3136	9.9871	3	56	0.0000

Table 6.63: ANOVA with blocks residuals.

Min	1Q	Median	3Q	Max
-251.0861	-112.8055	16.2687	77.8558	398.9139

more broadly. Is there a difference that is likely to be significant across all towns on average? To answer that question, we need to view blocks as random effects, i.e., random samples from all possible towns. The three in this example is a tiny sample to make such a sweeping generalization, but it makes a simple example to learn with. Let us perform the very same analysis, this time with town as a random effect.

1. Use "File> Open" and browse to the folder: "`C:\Program Files \BlueSky Statistics\Sample Datasets and Demos\ANOVA \OneWayAnova.RData`".
2. Choose "Analysis> Means> Oneway ANOVA with Random Blocks."
3. For the response variable, choose Sodium. We wish to see if one of the teaching methods reduce the use of salt in the participant's diets.

Table 6.64: ANOVA with blocks model coefficients.

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1358.202	36.7137	36.9944	<.001***
InstructorCoach McGuirk	-64.8174	45.8605	-1.4134	0.1631
InstructorMelissa Robins	-124.471	46.5312	-2.675	0.0098 **
TownMetalocalypse	-157.1161	39.3612	-3.9916	<.001***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Table 6.65: ANOVA with blocks ANOVA table.

	Sum Sq	Df	F value	Pr(>F)
Instructor	148943.8	2	3.6006	0.0338 *
Town	329551	1	15.9332	<.001***
Residuals	1158261	56	NA	NA

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Table 6.66: ANOVA with blocks pairwise comparisons using the Tukey test.

contrast	estimate	SE	df	t.ratio	p.value
Brendon Small - Coach McGuirk	64.8174	45.8605	56	1.4134	0.341
Brendon Small - Melissa Robins	124.471	46.5312	56	2.675	0.0261 *
Coach McGuirk - Melissa Robins	59.6536	48.127	56	1.2395	0.4352

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

4. For the fixed effect, choose Instructor (synonymous with the teaching method used.)
5. For the blocking variable, choose Town.
6. Make sure that the post-hoc analysis box is checked, then click OK to run the analysis.

The results begin with the same two tables of statistical summaries, that I do not repeat here.

Next, we see a summary of the fixed effects, Tab. 6.67. It does not show a p-value, but we will get that later.

In Tab. 6.68 we do get a p-value of 0.0012, which we usually do not care about. Random blocks are almost always different, but we do not care why.

Next, is the deviance table, Tab. 6.69. That gives us Akaike's Information Criterion (AIC), the Bayesian Information Criterion (BIC), and the log likelihood, all of which you can use to compare compare models. A low value

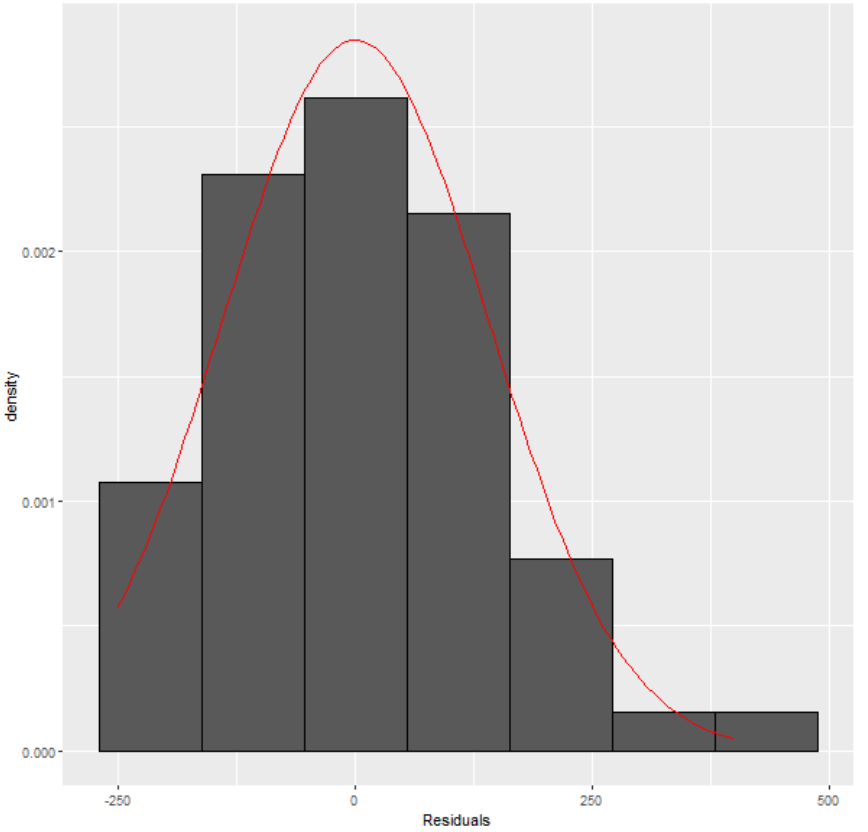


Fig. 6.33: ANOVA with blocks histogram of residuals.

of AIC/BIC indicates a better model in general. We see that overall, this model is significant, with a p-value of 0.0232.

After that, it displays the number of observations in the model, 60.

Next, comes the likelihood ratio test, Tab. 6.70 with a Chi-squared test and p-value of 0.023.

Several measures of pseudo R-squared measures follows in Tab. 6.71. Higher values are better here as usual, though they do not provide a direct "proportion of variance explained" interpretation.

Now we arrive at the table of pairwise comparisons, Tab. 6.72. Here we see all possible comparisons, and, as before, only Melissa Robins' teaching method comes out significantly lower than Brendon Small's approach. We now have evidence that suggests that the instructor's difference in teaching method may generalize across random towns, not just the three in this sample.

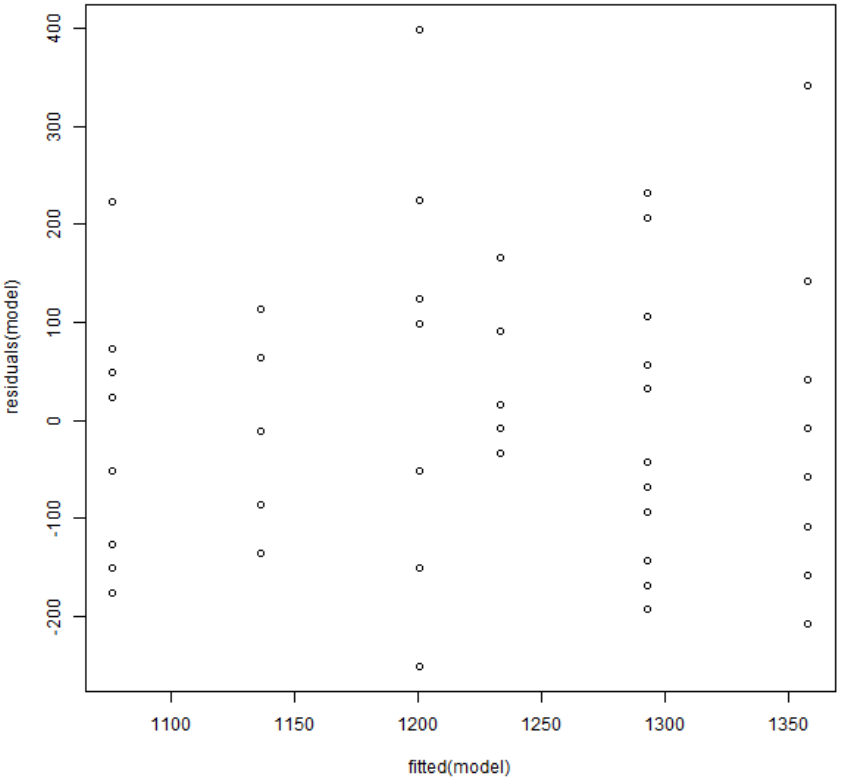


Fig. 6.34:]
ANOVA with blocks plot of residuals vs. fitted values.]

Table 6.67: ANOVA with random effects, fixed effects summary.

	Df	Sum Sq	Mean Sq	F value
Instructor	2	154766.5	77383.23	3.7413

Table 6.68: ANOVA with random effects, random effects summary.

	npar	logLik	AIC	LRT	Df	Pr(>Chisq)
<none>	5	-369.9641	749.9281	NA	NA	NA
(1 | Town)	4	-375.2114	758.4229	10.4948	1	0.0012

Table 6.69: ANOVA with random blocks deviance table.

	Df	AIC	BIC	logLik	deviance	Chisq	Chi Df	Pr(>Chisq)
model.null	3	782.3432	788.6263	-388.1716	776.3432	NA	NA	NA
model	5	778.8177	789.2895	-384.4089	768.8177	7.5255	2	0.0232

Table 6.70: ANOVA with random blocks likelihood ratio test.

Df.diff	LogLik.diff	Chisq	p.value
-2	-3.7731	7.5461	0.023 *

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Table 6.71: ANOVA with random blocks R-squared measures.

	Pseudo.R.squared
McFadden	0.009715
Cox and Snell (ML)	0.118181
Nagelkerke (Cragg and Uhler)	0.118181

Table 6.72: ANOVA with random blocks pairwise comparisons.

contrast	estimate	SE	df	t.ratio	p.value
Brendon Small - Coach McGuirk	63.3383	45.9337	56.1134	1.3789	0.3587
Brendon Small - Melissa Robins	126.9362	46.7314	56.2915	2.7163	0.0234 *
Coach McGuirk - Melissa Robins	63.5979	48.621	56.6182	1.308	0.3967

```
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6.9 Missing Values

Missing values are a fact of life in most fields of research. When only a tiny percent are missing, they do not cause much of a problem. However, it is important to know how big a problem they present in any given dataset. If you ever come across a dataset with *no* missing values at all, I recommend maintaining a skeptical attitude. The people collecting the data may not have passed along the fact that they used some code, such as "-999" or even zero, to represent missing values. In that case, you can use the recoding methods discussed in Sec. 4.12 to convert those values to be the "NA" value that BlueSky and R use to represent Not Available or missing values.

The items under "Analysis> Missing Values" provide summaries of missing values. Those missing values can be imputed (estimated) by using

Table 6.73: Counts of missing (top) and non-missing (bottom) values per variable, in column format.

Missing Value (NAs) Count Per Variable

BodyWgt	BrainWgt	NonD	Dream	Sleep	Span	Gest	Pred	Exp	Danger
0	0	14	12	4	4	4	0	0	0

Non Missing Value Count Per Variable

BodyWgt	BrainWgt	NonD	Dream	Sleep	Span	Gest	Pred	Exp	Danger
62	62	48	50	58	58	58	62	62	62

"Data> Missing Values" in Sec. 4.10. That approach is usually preferable to the "listwise deletion" that remove cases that have *any* missing values.

The examples in this section will use the mammal sleep dataset from the VIM package. You can load it using the following steps:

1. In the Analysis window, choose "File> Load Dataset from a Package."
2. In the "Select a Package..." box, enter "VIM."
3. In the "Select a Dataset to Load..." box, enter "sleep" then click OK. WARNING: this is not the sleep dataset from the datasets package used in many other examples! Therefore, you must enter the package name of VIM to get the correct one.

6.9.1 Analysis of Missing Values, column output

The item "Analysis> Analysis of Missing Values, column output," calculates counts of missing and non-missing values for each variable. We see in Tab. 6.73 (top) that some variables, such as BodyWgt (body weight) and BrainWgt (brain weight), have no missing values. Others, such as Dream and NonD (non-Dream), have the most missing at 12 and 14 observations, respectively.

The bottom table in Tab. 6.73 shows the number of non-missing values for each variable. In cases like this where the number of variables you have can fit across one page, the column-style output is nice as it lines these two tables up, making it easy to look up and down comparing the numbers reported for each variable. However, when the variables do not fit on a page, you can switch to the row-style output covered in the next section.

The last table, Tab 6.74, focuses on observation numbers that have missing values for each variable. In the bottom row, we see that the variable Gest is missing for observations 13, 19, 20, and 56.

6.9.2 Analysis of Missing Values, row output

The item "Analysis> Analysis of Missing Values, row output," generates the same count of missing values per observation shown in Tab. 6.74. However,

Table 6.74: Missing value counts per observation, in column format.

Row Numbers for Variables that have NAs

	Row numbers with NAs													
NonD	1	3	4	14	21	24	26	30	31	41	47	53	55	62
Dream	1	3	4	14	24	26	30	31	47	53	55	62		
Sleep	21	31	41	62										
Span	4	13	35	36										
Gest	13	19	20	56										

Table 6.75: Counts and percents of missing values per variable, in row format.

Missing Variable Summary

variable	n_miss	pct_miss
NonD	14	22.5806
Dream	12	19.3548
Sleep	4	6.4516
Span	4	6.4516
Gest	4	6.4516
BodyWgt	0	0
BrainWgt	0	0
Pred	0	0
Exp	0	0
Danger	0	0

it displays the information in rows, and adds a scroll-bar to make it easier to study. An example is shown in Tab. 6.75.

6.10 Non-Parametric Tests

Non-parametric tests require fewer assumptions than their parametric equivalents. They don't assume any particular distribution, but they do assume that groups share the same distribution, other than a shift in location (i.e., the medians can differ). Nonparametric tests can also yield accurate significance tests when using ordinal data, measures for which the intervals between numbers may not hold consistent meaning. For example, a change

Table 6.76: Counts and percents of missing values per observation, in row format. Note that on your screen, the scroll bar would allow you to see the values for all observations.

Missing Case Summary

case	n_miss	pct_miss
4	3	30
31	3	30
62	3	30
1	2	20
3	2	20
13	2	20
14	2	20

of 1 to 2 on a measure of pain may be perceived as a small change, while an increase from 9 to 10 may be perceived as much larger.

The nature of a numeric variable's distribution can be assessed visually using density plots, histograms, Q-Q plots, or P-P plots. If you prefer a hypothesis test of normality, see the Shapiro-Wilk test in Sec. 6.13.7.

6.10.1 Chi-squared Test

The item "Analysis> Non-Parametric Tests> Chi-squared Test" performs a test on the proportions in one-way tables. The far more common test applied to two-way tables is done using, "Analysis> Contingency Tables> Crosstabs, Multi-way," which is covered in Sec. 6.4.2. A related test is the Binomial test discussed in Sec. 6.11.1. That provides exact probabilities, while this Chi-squared test provides only approximate ones. However, the Binomial test can handle measures with only two values while the Chi-squared test in this section handles any number of values. The proportion test in Sec. 6.11.3 provides an almost identical test to this one.

An example of a one-way table comes from the 2010 U.S. Census, which found that the population was 49.2% males and 50.8% females. Let us examine the people on the Titanic to see if the ratio of males to females differs from the U.S. 2010 Census values:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C: \Program Files \BlueSky Statistics \Sample Datasets and Demos \Sample R Datasets(.rdata) \Titanic.RData" select it, and click "Open."
3. Choose "Analysis> Non-Parametric Tests> Chi-squared Test"
4. Move the variable sex to the "Selected Variables" box.

Table 6.77: Observed and expected values for one-way chi-squared test.

Frequencies for variable sex

	Observed	Expected	Residuals
female	388	531.368	-6.2195
male	658	514.632	6.3198

5. Enter the proportions to test against as ".508,.492". Note that these are proportions that add to 1, not percentages that add to 100. Then click, OK.

The results are shown in Tab. 6.77. The observed counts are those on the ship, while the expected ones are based on the proportions we provided. It is important to be sure to check these to ensure that you entered them in the proper order. Factor levels are stored in alphabetical order by default, but that can be overridden when they are created. Females appear first in the table, and we provided the female proportion first in the dialog box to match that. We also see a larger than expected number for females, another sign that we have the correct order of the proportions. If the proportion of females matched the Census, then there would have been 531 females on the ship, while instead, there were only 388.

The Pearson residuals are in the last column. They are the observed values, minus the expected, divided by the square root of the expected. The size of the residual indicates how far off each count is from the expected value and its sign shows whether there are more than expected (positive) or less (negative). The next table (not shown) contains the Chi-squared test value of 78.62, with 1 degree of freedom and a p-value of 0.0000, or $p<.001$. Since that is much smaller than the standard cutoff of 0.05, we can reject the hypothesis that there is no difference between the Titanic's gender proportions and those of the U.S. 2010 Census.

The R function used to compute the values in this section is `chi.square` from the `stats` package. We will examine this same question later using the Binomial test, in Sec. 6.11.1.

6.10.2 Friedman Test

The Friedman test is a one-way repeated measures test for data that are not normally distributed. In the case of a measure taken only twice, it yields the same p-value as the Wilcoxon paired-samples test covered in Sec. 6.10.6. As with that test, the measurements are assumed to be of at least ordinal scale, sharing the same distribution except for location.

For an example of the Friedman test, let us use the anxiety data which, measures that variable three times, labeled t1, t2, and t3. The experiment has other factors, but we will pretend those do not exist for the sake of demonstration.

1. In the Analysis window, use "File> Load Dataset from a Package."
   In the "Select an R Package..." field, enter "datarium".
   In the "Select a Dataset..." field, enter "anxiety."
2. Select the menu, "Analysis> Non-Parametric Tests> Friedman Test."
   In the "Response Variable (two or more)" field, enter t1, t2, t3. Then click, OK.

The first table of output (not shown) lists the medians of the three times in order: 17, 16.1, 15.3. We see anxiety is decreasing but is the decrease significant? The Friedman test shows a Chi-squared value of 69.37, with 2 degrees of freedom and a p-value of 0.0000, i.e., p<.001. Since the p-value is less than the standard cutoff of 0.05, we can reject the null hypothesis that the anxiety levels do not change. That still leaves us with a question of which ones differ. For that, you can repeat the analysis, including only the comparisons of interest. Since these are in order, you might compare t1 to t2, then t2 to t3. Alternatively, you might compare t1 to t2 and t3. In both cases, you can correct the p-values using the Bonferroni adjustment and multiplying them by 2. You could make all three possible comparisons, but then you would have to multiply by 3, making it harder to get significance with minimal gain in information.

6.10.3 Kruskal-Wallis Test

The Kruskal-Wallis test is the most common analysis to test for median or mean rank differences among more than two independent or unrelated groups. If done using only two groups, it is exactly the same as the Wilcoxon-Mann-Whitney test. If the numeric variable is normally distributed, or it can be transformed to become normally distributed, then the one-way analysis of variance test is slightly more powerful and would be preferred.

The Kruskal-Wallis test assumes that the two groups you are comparing follow the same distribution, other than shifting the median's location. If that is true, then this is a test comparing the medians. However, if the distributions are *not* the same, then it is a test on the groups' mean ranks. This difference causes confusion when one group's distribution is more skewed to one side. The medians are unaffected by the skewness, and the test can yield a significant result when the medians are exactly the same! In that case, the means ranks will differ.

Let us examine the impact of educational level on job satisfaction. These are the steps to run the test:

1. In the Analysis window, use "File> Load Dataset from a Package."
   In the "Select an R Package..." field, enter "datarium".
   In the "Select a Dataset..." field, enter "jobsatisfaction".
2. Select the menu, "Analysis> Non-Parametric Tests> Kruskal-Wallis Test."
   Choose score, as the "Response Variable."
   Choose education_level, as the "Factors Variable." Then click, OK.

The output of this test is quite simple. The medians for high school, community college, and university are 5.51, 6.38, and 9.06, respectively. There seems to be an upward trend in job satisfaction as educational level increases, but are these differences significant? The Kruskal-Wallis Chi-squared test follows, at 45.44 with 2 degrees of freedom and a p-value of 0.0000. Since that is well below the usual cutoff of 0.05, we can reject the null hypothesis that the medians are equal.

To discover which ones differ requires additional steps. The item "Data> Subset Dataset" can be used to select, for example, "`education_level=="school" | education_level=="college"`." Then repeating the analysis above would compare only those two educational levels.

6.10.4 Wilcoxon Test, independent samples

The Wilcoxon test (a.k.a. Mann-Whitney U) is the most common analysis to test for median or mean rank differences between two independent or unrelated groups. If the numeric variable is normally distributed, or it can be transformed to become normally distributed, then the independent samples t-test is slightly more powerful and would be preferred.

The Wilcoxon test assumes that the two groups you are comparing follow the same distribution, other than shifting the location of the median. If that is true, then this is a test comparing the medians. However, if the distributions are *not* the same, then it is a test on the mean ranks of the two groups. This difference causes confusion when one group's distribution is more skewed to one side. The medians are unaffected by the skewness, and the test can yield a significant result when the medians are exactly the same! The mean ranks in that case will be different.

Let us examine the "extra" variable from a sleep study that used drugs to attempt to get people additional sleep at night. While that variable is fairly normally distributed, we will use it with the Wilcoxon test for demonstration purposes. These are the steps to run the test:

1. In the Analysis window, use "File> Load Dataset from a Package" and enter the "datasets" as the package name and "sleep" as the dataset name.
2. Choose "Analysis> Non Parametric Tests> Wilcoxon Test, independent samples."
3. Move the variable "extra" into the Response Variable box.
4. Move the variable "group" into the "Factor" variable box.
5. Leave the Alternative Hypothesis box at its default setting of "Population Mean != mu" setting.
6. Click, "OK."

The first output table shows the medians are 0.35 and 1.75 for groups 1 and 2, respectively (table not shown). The second piece of output, (Tab. 6.78), contains the test itself with a p-value of 0.0693 indicating that we cannot reject the hypothesis that the medians of the two groups are equal. Since this is not less than the standard cutoff of 0.05, we cannot reject the hypothesis

Table 6.78: Wilcoxon test for independent samples.

Wilcoxon rank sum test with continuity correction

Null Value Considered: 0				
W	p-value	sample estimate difference in location	confidence: 0.95 lower	confidence: 0.95 upper
25.5	0.0693	-1.3464	-3.5999	0.1

that the groups' medians are equal. The table also includes the median difference of -.13464 with a 95% confidence interval that includes zero, another indication that the median difference is not significant.

That describes the two-tailed tests. If your initial alternative hypothesis was that group two's median would be greater, then this p-value would be $0.0693 \div 2 = 0.0347$. That is the value you would obtain if you re-ran this, choosing "Difference < 0".

6.10.5 Wilcoxon Signed-Rank Test, one sample

A one sample Wilcoxon signed-rank test compares the median of one variable to a given value. That value might be a known value of a population. The value is subtracted from the measure on each case, and the median is then compared to zero. When the variable in question is a difference between pairs of subjects, the result is exactly the same as a paired-samples Wilcoxon test. The only advantage the latter provides is to do the subtraction for you.

This test assumes that the variable has a symmetric distribution. If the variable is normally distributed, a more powerful test would be "Analysis> Means> T-test, one sample" covered in Sec. 6.10.5.

Let us repeat the example we covered in that section, this time using the non-parametric test. Assume that it is known that a particularly arduous workout makes people need an extra 2.5 hours of sleep the next day. A drug is used to reduce the effects of fatigue, and we wish to know if the median amount of extra sleep needed is significantly reduced. We can answer that question using the following steps:

1. In the Analysis window, use "File> Load Dataset from a Package" and enter "datasets" as the package name and "sleep" as the dataset name.
2. Choose "Analysis> Non-parametric Tests> Wilcoxon Signed-Rank Test, one sample."
3. Select the outcome variable, "extra."
4. Leave the Alternative Hypothesis box at its default setting of "Population Median != mu" setting.
5. Fill in 2.5 as the test value, and click OK to run it.

Table 6.79: Wilcoxon signed-rank test for one sample.

		Null Value Considered: 2.5		
		sample estimate	confidence: 0.95	confidence: 0.95
V	p-value	(pseudo)median	lower	upper
55	0.0645	1.5	0.4	2.6

Table 6.80: Wilcoxon signed-rank test additional details.

	Additional Comments
Test Method Performed	Wilcoxon signed rank test with continuity correction
Alternative	two.sided
Null Value	2.5
Alternative Hypothesis	True location is not equal to 2.5
Data Variables Used	sleep$extra

Table 6.79 appears first in the output, showing the test value, V, and its p-value of 0.0645. That is not quite significant. When we tested this hypothesis using a t-test, the p-value was 0.0467, demonstrating the increased power of that test with this variable that is approximately normally distributed. After the 2.5 comparison value is subtracted out, the pseudo median is 1.5, with 95% confidence intervals from 0.4 to 2.6. That interval includes the value of 2.5, which is why the p-value is not lower than the usual 0.05 cutoff.

Table 6.80 appears next, showing details of how the test was done, including the fact that it was a two-sided test. If we were confident enough that the drug could only have a positive effect, then we could have run the test as one-sided, which would yield a p-value of $0.0645 \div 2 = 0.03225$. Because of the temptation to change the direction of hypotheses after the fact, journals may be reluctant to accept results that only become significant when doing a one-tailed test. Of course, if prior published results have already established that direction, the journal editors should be fine with a one-tailed test.

BlueSky performs the calculations in this section using the `stats` package's `wilcox.test` function.

6.10.6 Wilcoxon Test, paired samples

The Wilcoxon paired samples test (a.k.a. the signed rank test) is the most common analysis to test for median differences between two related measures. Often the first variable is a baseline measure of the subjects' physiology or knowledge. The experimental condition, such as a drug, or training, is then applied, and the second variable measures the resulting impact. The test

Table 6.81: Paired Wilcoxon test output

Null Value Considered: 0				
		sample estimate	confidence: 0.95	confidence: 0.95
V	p-value	(pseudo)median	lower	upper
988.5	0.0000	1.8	1.1	2.15

essentially subtracts the ranks of the second variable from those of the first, and then compares the median of that difference to zero.

This paired test is designed for "wide" format data, where the numeric variables for the two groups appear side-by-side as two variables. If you have "long" data which has a single numeric variable containing the values from both groups along with a factor indicating group membership, then reshape the data as shown in the example below:

1. In the Analysis window, use "File> Load Dataset from a Package."
2. In the "Select an R Package..." field, enter "datarium".
3. In the "Select a Dataset..." field, enter "anxiety."
4. Choose, "Analysis> Non-Parametric Tests> Wilcoxon Test, paired samples."
5. Choose t1 and t3 as first and second numeric variables, then click, "OK."

The first piece of output is the median difference of 1.3. Next, comes a table of test results, shown in Tab. 6.81. The V statistic is 988.5, with a p-value of 0.0000, far less than the 0.05 standard cutoff. Therefore, we reject the null hypothesis that the medians are equal.

This is a two-tailed test, so if your initial hypothesis was that group two's median would be greater than group one's, you would correct that by dividing by two, which would still be very significant. That is the result you would obtain by rerunning the analysis and choosing "Difference < 0". It always subtracts group two from group one.

The `stats` package's `wilcox.test` function performs the calculations for this topic.

6.11 Proportions

Proportion tests examine several different hypotheses regarding proportions involving only two levels of a factor, such as infected/not, lived/died, male/female, and so on. While such proportions are often compared to 0.50, they can be compared to any population value. You can also test to see if that ratio is the same across the levels of another factor. These tests are very similar to those covered elsewhere as Chi-squared tests applied to various types of contingency tables, as referenced in each following section.

Table 6.82: Binomial test, single sample.

		Null Value Considered: 0.508			
			sample estimate	confidence: 0.95	confidence: 0.95
number of successes	number of trials	p-value	probability of success	lower	upper
388	1046	0.0000	0.3709	0.3416	0.401

6.11.1 Binomial Test, single sample

The item "Analysis> Proportions> Binomial Test, single sample," is used to test the hypothesis that a variable with two (and only two) outcomes has a particular proportion. Let's examine the same question we looked at in the section on the one-way Chi-squared test in Sec. 6.10.1. There we compared the proportion of females on the Titanic to the proportion in the 2010 U.S. Census, which found the population then was 49.2% males and 50.8% females. However, in this case, we use the exact probability provided by the Binomial test rather than the approximate one provide by the Chi-squared test. Being exact, this test is preferred, especially when sample sizes are small.

Here are the steps to follow:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample CSV Datasets\Titanic.csv" select it, and click "Open."
3. Choose "Analysis> Proportions> Binomial Test, single sample."
4. Move the variable sex to the "Factor" box.
5. Enter the proportions to test against as ".508,.492". Note that these are proportions that add to 1, not percentages that add to 100. Then click, OK.

The results are shown in Tab. 6.82. The first table (not shown) simply lists the number of females, 388, and the number of males, 658. It is easy to see that the proportion of females is nowhere near 0.50! The Binomial test bases its calculations on the factor level that it considers the "success." By default, that is the first level of the factor. Note that there are 388 "successes," which matches the number of females. Based on the Binomial distribution, the chance of 388 successes from a process whose mean is 0.508 is 0.0000. We would have to use "Tools> Settings> Output" to turn on scientific notation to tell the difference between this p-value and the one we got from the Chi-squared test in Sec. 6.10.1. Both are highly significant, allowing us to reject the null hypothesis that the proportion of females matches that of the U.S. Census.

The calculations for this test are performed by the `stats` package's `binom.test` function.

Table 6.83: Crosstabulation of sex and survival.

survived	sex			
	female	male	Total	Count
0	15.5	84.5	100	619
1	68.4	31.6	100	427

Table 6.84: Two-sample test for equality of proportions (without continuity correction)

Test Result						
			sample estimate	sample estimate	confidence: 0.95	confidence: 0.95
X-squared	df	p-value	prop 1	prop 2	lower	upper
302.7586	1	0.0000	0.1551	0.6838	-0.5813	-0.4762

6.11.2 Proportion Test, independent samples

The item "Analysis> Proportions> Proportion Test, independent samples," compares the proportions of two (and only two) factor levels across the levels of a second variable. This is essentially the same problem that we solved using the contingency table approach covered in Sec. 6.4.

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample CSV Datasets\Titanic.csv" select it, and click "Open."
3. Choose "Data> Convert Variables to Factor(s)" choose the variable survived, and click OK.
4. Choose "Analysis> Proportions> Proportion Test, independent samples".
5. Choose survived as the "GroupBy" factor.
6. Choose sex as the response variable, then click OK.

The first table shows the number and percent of males and females in each survival condition, Tab. 6.83. In the group that survived, 68.4% were female, while in the non-survival group, only 15.5% were female.

The test results are shown in Tab. 6.84. The Chi-squared value is 302.76 with 1 degree of freedom, and a p-value of 0.0000, or $p < .001$. Using the usual 0.05 cutoff value, we can reject the null hypothesis that the proportion of females is the same in each survival outcome.

The calculations for this test are performed by the `stats` package's `prop.test` function.

Table 6.85: One-sample proportion test (without continuity correction).

Null Value Considered: 0.508					
			sample estimate	confidence: 0.95	confidence: 0.95
X-squared	df	p-value	p	lower	upper
78.622	1	0.0000	0.3709	0.3422	0.4006

6.11.3 Proportion Test, single sample

The item "Analysis> Proportions> Proportion Test, single sample," compares the proportions of the values within a single variable. This is the same type of problem solved by the Binomial test covered in Sec. 6.11.1, but there we used an exact solution that was useful for small samples. Here, we use an approximate solution that was easy to calculate in the pre-computer days and is still popular. Let us duplicate the example from that section to see how similar it is:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample CSV Datasets\Titanic.csv" select it, and click "Open."
3. Choose "Analysis> Proportions> Proportion Test, single sample."
4. Move the variable sex to the "Factor" box.
5. Enter the proportions to test against as ".508". Note that in the Chi-squared version, there was the option to enter multiple values. This one only handles two. Then click, OK.

The results are shown in Tab. 6.85. The Chi-squared value is 78.62, with 1 degree of freedom. Its p-value is 0.0000, or $p < 0.001$, so we can reject the null hypothesis that the proportion of females is 0.508. The proportion, labeled as P, is 0.37, with 95% confidence intervals ranging from 0.34 to 0.40. Those are nearly identical to the confidence intervals obtained from the Binomial test. While this is an approximate test and the Binomial is an exact one, we can see that the approximation of this test is quite good.

This test can also be performed using, "Analysis> Contingency Tables> Crosstab, multi-way." While that test can handle factors that have more than two levels, when limited to two levels, it yields the same Chi-squared values as this test (See Sec. 6.10.1 for details).

The `stats` package's `prop.test` function performs the calculations in this section.

6.12 Reliability Analysis

The term "reliability" is applied to two very different types of analysis. In the fields of medicine, engineering, and industry, reliability analyzes the

Table 6.86: Variable descriptions for reliability analysis example. Note that E1 and E2 are asked in the opposite direction of the other items.

Variable	Question
E1	Don't talk a lot.
E2	Find it difficult to approach others.
E3	Know how to captivate people.
E4	Make friends easily.
E5	Take charge.

time until an organism or device fails. That type of reliability is also known as survival analysis, or time-to-event analysis. We cover that in Sec. 6.14.

This section is about the consistency of measurement devices or scales. Let us return to the example of measuring the personality trait of extroversion discussed above in Factor Analysis, Sec. 6.6. You can ask, "On a scale of 1 to 100, how extroverted are you?" However, you are unlikely to get a reliable answer. Each person may define extroversion in a slightly different way. Imagine someone just attended a party and did not have a good time. With only that on their mind, they could rate themselves low on extroversion. The next time you ask the person, they may have just given a speech in front of a large audience that went well. With that on their mind, they may rate themselves high on extroversion. This is why reliable measures of people's traits usually consist of "scales," which are sets of questions. The averages of such scales yield more reliable estimates.

The simplest way to measure the reliability of a measure is to record the measurement once, wait a while, and retake. How long to wait depends on what you are trying to measure. Let us say you have measured a group of people on an extroversion scale and now have a single score for each. After a month, you measure them again and calculate the Pearson correlation between the two. That is called "test-retest" reliability, and it is one of the best ways to do it. However, it turns out that you can divide a scale into two parts, like the first and second half, average them, then correlate them. This "split-half reliability" provides a good estimate of the test-retest figure (after correcting for the loss of half the number of items), and it takes less time. Alternatively, you might score the odd items and the even ones. The reliability measures covered in the next two sections are related to split-half reliability but are both better.

6.12.1 Reliability Analysis, Cronbach's Alpha

When considering how to measure psychometric reliability, the split-half approach is simple to understand but time-consuming (see Sec. 6.12 for details.) You might use many different splits, but the measure known as Cronbach's Alpha is the mean of all possible split-half reliability measures, corrected for the number of items. This type of reliability is also known as the internal consistency of a test. If the scale items do not form a single scale, i.e., if a factor analysis on the scale items would find more than one factor,

Table 6.87: Crohbach's Alpha and associated statistics

Cronbach's Alpha	Cronbach's Standardized Alpha	Guttman's Lambda 6	average _r	S/N	ase	mean	sd	median_r
0.7617	0.7618	0.7266	0.3901	3.198	0.007	4.145	1.061	0.382

Table 6.88: Cronbach's Alpha and associated statistics, if each item were removed

	Cronbach's Alpha	Cronbach's Standardized Alpha	Guttman's Lambda 6	average _r	S/N	alpha se	var.r	med.r
E1-	0.7257	0.7254	0.6731	0.3978	2.6423	0.0084	0.0044	0.3818
E2-	0.6902	0.6931	0.6342	0.3608	2.2582	0.0095	0.0028	0.3546
E3	0.7279	0.7262	0.6737	0.3988	2.6529	0.0082	0.0071	0.396
E4	0.7019	0.7032	0.6464	0.372	2.3697	0.0091	0.0033	0.3767
E5	0.7436	0.7442	0.6914	0.4211	2.9093	0.0078	0.0043	0.419

Table 6.89: Nunnally and Bernstein's recommendations on Cronbach's Alpha.

Cronbach's Alpha	Sufficient for:
0.90 to 0.95	Making important decisions
0.80 to 0.89	Applied research
0.70 to 0.79	Early stage of research

then Alpha is not an appropriate measure. Alpha also assumes that the items would have identical loadings in a factor analysis, and the error terms are uncorrelated. If such assumptions are not met, then McDonald's Omega Hierarchical is a more appropriate measure (see Sec. 6.12.2 for details).

Let us examine an example.

1. Go to the Analysis window and open the bfi dataset by choosing "File> Load dataset from a package."
2. Enter the name "bfi," in the lower right of the dialog box. That will load the dataset from the "psych" package, but you probably do not need to fill in the package name.
3. Choose "Analysis> Reliability> Reliability Analysis, Cronbach's Alpha."
4. Choose the variables E1 through E5 and move them to the "All Items" box.
5. Choose the variables E1 and E2 and move them to the "Enter items above that are reverse scored." box. As you can see in Table 6.86, those two items are asked in an opposite manner than the other items.
6. Click OK.

The first table of output is shown in Tab. 6.87. Alpha itself is 0.76, which according to Nunnally, is adequate only for an early stage of research (Tab. 6.89) [19].

The Standardized Alpha is calculated on the correlation matrix rather than the covariance matrix. That means it is the equivalent of taking each item, subtracting its mean, then dividing by its standard deviation. That puts all items on the same scale. In this case, the items are already on the same scale, which is why these two figures are identical in the first three decimal places.

Guttman's Lambda 6 is a measure that is similar to Alpha, but which is more appropriate if the scale is "lumpy", in that there are subsets of items that correlate somewhat more highly with one another than they do with the other sets of items in the scale.

The Average r is simply the mean inter-item correlation. Median.r (Med.r in 2nd table) is the median of those correlations.

The Signal to Noise ratio (S/N) is another index of test quality and is $s/n = n \times \bar{r}/(1 - \bar{r})$, where $\bar{r}$ is the average inter-item correlation. Alpha.se is the standard error of Alpha, and var.r is the variance of the inter-item correlations.

The next table of output, Tab. 6.88, shows the impact of removing each item, one at a time. If you saw Alpha increase sharply for a given item, it would indicate that the item is not being answered in as consistent a manner as the others. Removing such items will then increase the overall Alpha. Note that items E1 and E2 are followed with a minus sign indicating that they measure the reverse of the others.

The final two tables of output (not shown) include basic summary statistics of the items.

The `psych` package's `alpha` function performs the calculations for this section.

6.12.2 Reliability Analysis, McDonald's Omega

The menu item "Analysis> Reliability> Reliability Analysis, McDonald's Omega," calculates McDonald's Omega, an internal-consistency type of reliability measure similar to Cronbach's Alpha. While Alpha is best when all the items in a scale correlate among themselves equally, Omega is best when the scale might have sub-scales nested within it. If sub-scales were nested within the scale's items, they would show up as separate factors in a factor analysis. To calculate Omega, it performs a factor analysis, extracting three factors by default. Since we are usually hoping that there are no factors, three is a reasonable number. It then rotates the factors using an oblique method and saves the factor scores. Then it does a secondary (hierarchical) factor analysis on those factor scores. Omega is based on how well one general factor, called "g," can explain the variability in the saved factor scores. See the help file for details.

Let us calculate Omega for the extroversion scale we used for Alpha.

1. Go to the Analysis window choose "File> Load dataset from a package."

2. Enter the name "bfi" in lower right of the dialog box. That will load the dataset from the "psych" package, but you probably do not need to fill in the package name.
3. Choose "Analysis> Reliability> Reliability Analysis, McDonald's Omega."
4. Choose the variables E1 through E5 and move them to the "All Items" box.
5. Click OK.

The output is, unfortunately, not in nicely formatted tables but is raw mono-spaced output directly from R. The first result begins with the R function call that uses the `psych` package's `omega` function. It returns Cronbach's Alpha of 0.76 and Guttman's Lambda 6 of 0.73, both of which we saw in the section on Cronbach's Alpha, Sect 6.12.1.

Next, is Omega Hierarchical at 0.66. It is a measure of reliability that does not assume that the scale is uni-dimensional. That is the one of greatest interest. Omega Total is based on the sum of squared loadings on all resulting factors, rather than just the general factor upon which Omega Hierarchical is based. Omega H Asymptotic is the highest value Omega could ever be if the scale could be made infinitely long (longer scales are inherently more reliable so long as the added items are of similar value to the scale).

```
Loading required namespace: GPArotation
Omega
Call: psych::omega(m = bfi[, c("E1","E2","E3","E4","E5")])
Alpha:                 0.76
G.6:                   0.73
Omega Hierarchical:    0.66
Omega H asymptotic:    0.83
Omega Total            0.79
```

Next, the output shows the relationships between the scale items and the factors. They all load fairly highly on the general factor, g, which indicates that this scale does tend to be uni-dimensional.

```
Schmid Leiman Factor loadings greater than  0.2
       g   F1*  F2*   F3*   h2   u2   p2
E1- 0.49 -0.36            0.37 0.63 0.65
E2- 0.59 -0.50            0.60 0.40 0.57
E3  0.66                  0.47 0.53 0.92
E4  0.58 -0.38       0.22 0.53 0.47 0.64
E5  0.59                  0.40 0.60 0.87

With eigenvalues of:
   g  F1*  F2*  F3*
1.70 0.53 0.04 0.11
```

The first is a chi-squared goodness of fit test for the factor analysis. The second one is the chi-squared for model fit for a model with just a general factor.

```
general/max  3.22   max/min =   13.29
mean percent general =  0.73
   with sd =  0.15 and cv of  0.21
Explained Common Variance of the general factor =  0.72

The degrees of freedom are -2  and the fit is  0
The number of observations was  2800
   with Chi Square =  0  with prob <  NA
The root mean square of the residuals is  0
The df corrected root mean square of the residuals is  NA
```

This next block of output provides a goodness of fit test to see if the factors it has chosen fit adequately. It is followed by the more useful goodness of fit test with a null hypothesis that a single general factor is sufficient. We hope this will be non-significant when we expect that this scale is measuring a single trait. In this case, it is significant, so we can reject the hypothesis that a single factor is adequate.

```
Compare this with the adequacy of just a general factor and
   no group factors
The degrees of freedom for just the general factor are 5
   and the fit is  0.11
The number of observations was  2800  with
   Chi Square = 312.5  with prob <  2.1e-65
The root mean square of the residuals is  0.09
The df corrected root mean square of the residuals is  0.13

RMSEA index =  0.148  and the 10% confidence intervals
   are  0.135 0.162
BIC =  272.81

Measures of factor score adequacy
                                       g   F1*   F2*   F3*
Correlation of scores with factors  0.83  0.61  0.20  0.41
Multiple R square of scores with factors
                                    0.68  0.38  0.04  0.17
Minimum correlation of factor score estimates
                                    0.37 -0.25 -0.92 -0.66

Total, General and Subset omega for each subset
                                       g  F1*  F2*  F3*
Omega total for total scores and subscales
                                    0.79 0.74 0.45 0.38
Omega general for total scores and subscales
                                    0.66 0.47 0.43 0.35
Omega group for total scores and subscales
                                    0.13 0.27 0.02 0.03
```

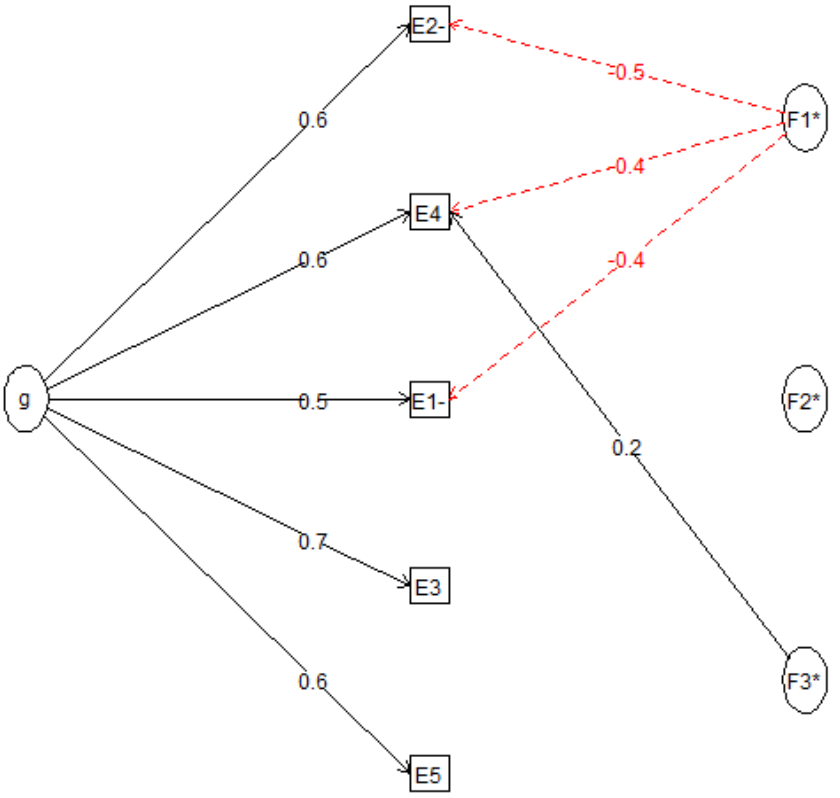


Fig. 6.35: McDonald's Omega plot.

The factor loadings among g, the three factors, and the original scale items is shown in Fig. 6.35. It shows that g does load fairly highly on each of the scale items. The first factor, F1, also loads on E2 and E1 (negatively) and E4. The third factor, F3, also loads on E4.

The `psych` package's `omega` function perform the calculations for this topic.

6.13 Summary Analysis

For most of the topics in this section we will obtain summary statistics for the passengers on the Titanic, described in Sec. 1.4. Here is how to load that dataset:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample CSV Datasets\Titanic.csv" select it, and click "Open."

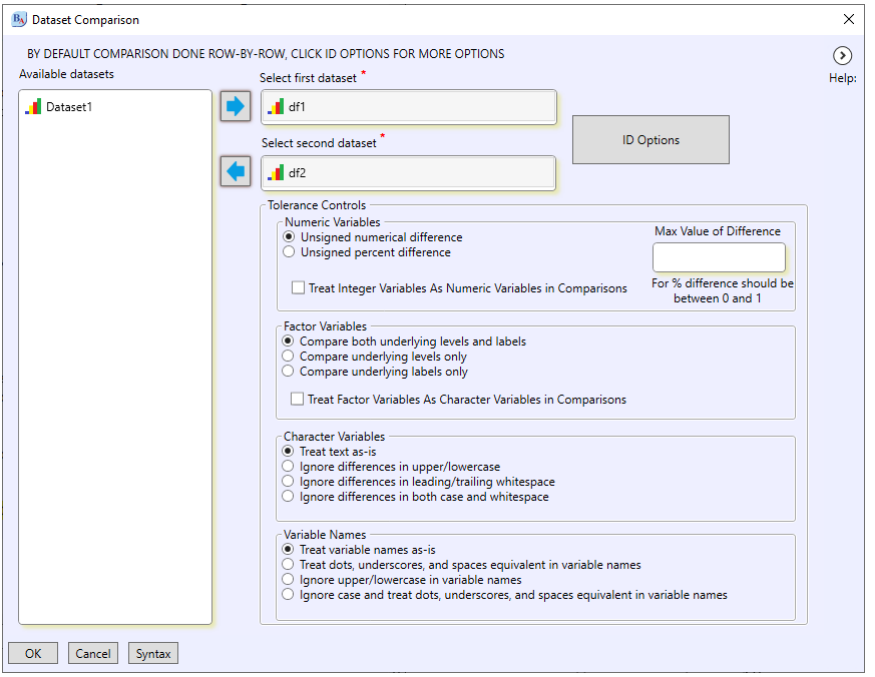


Fig. 6.36: Dialog for comparing datasets.

6.13.1 Compare Dataset

The item "Analysis> Summary Analysis> Compare Dataset" can compare two datasets. Let us dive right into an example that will demonstrate the types of comparisons it can make:

1. Use "File> Open" and browse to the folder: "`C:\Program Files\BlueSky Statistics\Sample Datasets and Demos\Comparing Datasets \df1.RData`".
2. Use "File> Open" again, browse to the same folder and open the file `df2.RData`.
3. Choose, "Analysis> Compare Dataset" and fill in df1 and df2 as the first and second datasets, then click OK.

The dialog should appear, as shown in Fig. 6.36. It offers great control over how different variable types can be compared and even allows numeric values to be slightly different using its "Max Value of Difference" field. That can be particularly helpful for decimal values which may differ in a very tiny percent.

The first table (not shown) specifies the number of rows and columns in each dataset, and assigns a label of x or y to each.

The second table is an overall summary of the comparison, shown in Tab. 6.90. The comparison is quite extensive, but each measure is clearly labeled.

Table 6.90: Dataset comparison summary

statistic	value
Number of by-variables	0
Number of non-by variables in common	3
Number of variables compared	3
Number of variables in x but not y	1
Number of variables in y but not x	1
Number of variables compared with some values unequal	2
Number of variables compared with all values equal	1
Number of observations in common	3
Number of observations in x but not y	0
Number of observations in y but not x	0
Number of observations with some compared variables unequal	2
Number of observations with all compared variables equal	1
Number of values unequal	4

The next table (not shown) says that dataset "x" has variable "c" as its fourth variable, while dataset "y" has variable "d" in that position.

The "differences detected by variable" table simply counts the number of differences found for each variable, while the "differences detected" table lists the values of each difference found (neither shown.)

The calculations for this test are performed by the `arsenal` package's `comparedf` function.

6.13.2 Explore Dataset

The item "Analysis> Summary Analysis> Explore Dataset," lets you obtain summary statistics, bar charts, and histograms on an entire dataset at once. Its dialog is shown in Fig. 6.37. It offers a check-box to "Display Output in a Webpage" which creates the nice look shown in Fig. 6.38. When *not* checked, the output appears as classic mono-spaced R output, in which the graphs are shown with crude text characters (not shown). Another check-box asks if graphs should be included.

The "Maximum Distinct Values" box asks for a number at or below which frequencies and percents will be included for numeric variables (they're always included for factors). By default, if a numeric variable has 10 or fewer values, frequencies and percents will be included along with summaries such as mean and standard deviation.

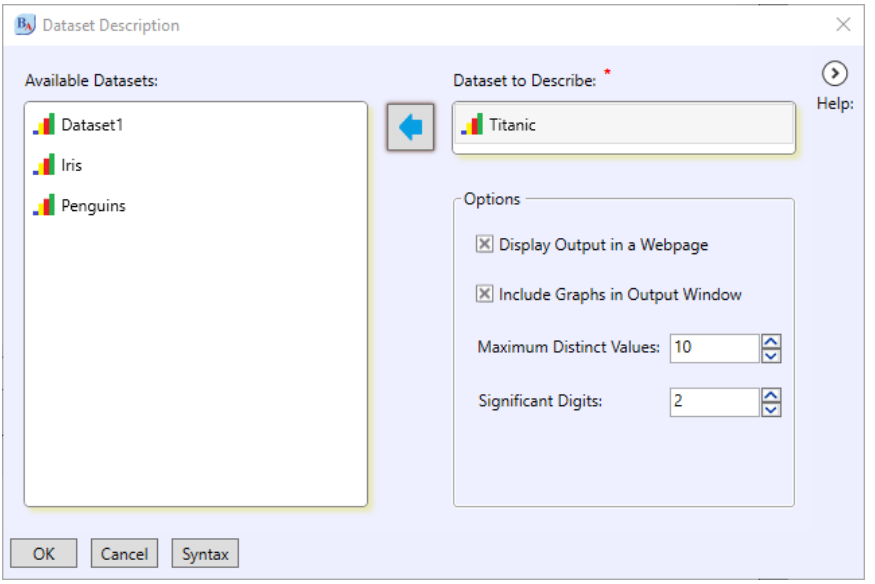


Fig. 6.37: Explore Dataset dialog. Note that the box is checked for "Display Output in a Webpage" which creates the output in Fig. 6.38.

A second entry box allows you to set the number of significant digits, though the default value of two is usually sufficient.

The calculations for this analysis are performed by the `summarytools` package's `dfSummary` function.

6.13.3 Frequency Table

The item "Analysis> Summary Analysis> Frequency Table," provides counts, cumulative counts, percents, and cumulative percents. It does this for both factors and numeric variables. Let us run it using all the variables in the Titanic dataset we opened in Sec. 6.13. There are not any options in this dialog box, so simply choose all the variables and click OK.

The first table of output shows a summary of all variables. For factors, all it shows is counts. For numeric variables, the mean, median, and quantiles are displayed.

After that single table, each variable gets its own frequency table like the one shown in Tab. 6.92. For numeric variables with many values, like age, the tables become so long that a secondary scroll-bar will appear to help you navigate it. At that point, you may wish to re-run the analysis by choosing it from the History menu, and delete such variables.

The `BlueSky` package's `BskyFrequency` function perform the calculations for this item.

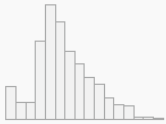
No	Variable	Stats / Values	Freqs (% of Valid)	Graph	Valid	Missing
1	pclass [factor]	1. 1st 2. 2nd 3. 3rd	284 (27.2%) 261 (24.9%) 501 (47.9%)		1046 (100%)	0 (0%)
2	survived [factor]	1. 0 2. 1	619 (59.2%) 427 (40.8%)		1046 (100%)	0 (0%)
3	sex [factor]	1. female 2. male	388 (37.1%) 658 (62.9%)		1046 (100%)	0 (0%)
4	age [numeric]	Mean (sd) : 29.9 (14.4) min < med < max: 0.2 < 28 < 80 IQR (CV) : 18 (0.5)	98 distinct values		1046 (100%)	0 (0%)
5	sibsp [integer]	Mean (sd) : 0.5 (0.9) min < med < max: 0 < 0 < 8 IQR (CV) : 1 (1.8)	0: 685 (65.5%) 1: 280 (26.8%) 2: 36 (3.4%) 3: 16 (1.5%) 4: 22 (2.1%) 5: 6 (0.6%) 8: 1 (0.1%)		1046 (100%)	0 (0%)
6	parch [integer]	Mean (sd) : 0.4 (0.8) min < med < max: 0 < 0 < 6 IQR (CV) : 1 (2)	0: 768 (73.4%) 1: 160 (15.3%) 2: 97 (9.3%) 3: 8 (0.8%) 4: 5 (0.5%) 5: 6 (0.6%) 6: 2 (0.2%)		1046 (100%)	0 (0%)

Fig. 6.38: Explore Dataset output when routed to a web page.

Table 6.91: Frequency tables summary of all variables.

pclass	survived	sex	age	sibsp	parch
1st:284	0:619	female:388	Min. : 0.1667	Min. :0.0000	Min. :0.0000
2nd:261	1:427	male :658	1st Qu.:21.0000	1st Qu.:0.0000	1st Qu.:0.0000
3rd:501	NA	NA	Median :28.0000	Median :0.0000	Median :0.0000
NA	NA	NA	Mean :29.8811	Mean :0.5029	Mean :0.4207
NA	NA	NA	3rd Qu.:39.0000	3rd Qu.:1.0000	3rd Qu.:1.0000
NA	NA	NA	Max. :80.0000	Max. :8.0000	Max. :6.0000

6.13.4 Frequency Table, top N

The item "Analysis> Summary Analysis> Frequency Table, top N" yields the same type of frequency tables offered in the previous section (Sec. 6.13.3)

Table 6.92: Frequency table of passenger class on the Titanic.

pclass	Frequency	Percent	CumPercent	Valid Percent	Valid CumPercent
1st	284	27.1511	27.1511	27.1511	27.1511
2nd	261	24.9522	52.1033	24.9522	52.1033
3rd	501	47.8967	100	47.8967	100
NA	0	0	100		

but with two changes. First, it accepts only factors. This can be helpful when you have both factors and numeric variables intermingled. Second, it offers a setting to "Show the top N factor levels by count." By default, this is set to 30, so it will show the 30 most frequent values and only those values. That is extremely helpful in cases where factors have far more values than are of interest at the moment. An example would be patient diagnosis. There are thousands of potential diagnoses, but just a few dozen probably accounts for 90% of the patients in a hospital at a given moment in time.

The `BlueSky` package's `BskyFactorVariableAnalysis` function performs the calculations for this analysis.

6.13.5 Numerical Statistical Analysis

The item "Analysis> Summary Analysis> Numerical Statistical Analysis," calculates various descriptive measures on numeric variables, as shown in Tab. 6.93. While it does not analyze factor variables, it does accept them in its "Group By" box. If included there, the table will repeat for each level of the grouping variable(s). Doing so is quicker than using the Split File feature described in Sec. 4.28.1.

The `BlueSky` package's `BSkySummaryStats` function performs the calculations for this analysis. That function combines the calculations of many functions that are built into R's `base` and `stats` packages.

6.13.6 Numerical Statistical Analysis, using describe

The menu selection, "Analysis> Summary Analysis> Numerical Statistical Analysis, using describe" calculates a broader range of statistical measures than some of the other summary options. In addition, it provides them using a single line per variable, making it a particularly useful choice when you have many variables. The dialog accepts only a list of variables to analyze (not shown). The results using the Titanic dataset are shown in Tab. 6.94.

The calculations for this analysis are performed by the `psych` package's `describe` function.

Table 6.93: Output from numerical statistical analysis.

stat function	age	sibsp	parch
min	1	0	0
1st Qu	32	0	0
mean	45.2906	0.5029	0.4207
median	43	0	0
3rd Qu	60	1	1
max	98	8	6
sd	19.7505	0.9122	0.8398
std. error	0.6107	0.0282	0.026
IQR	28	1	1
sum	47374	526	440
n	1046	1046	1046
NAs	0	0	0

Table 6.94: Summary analysis using describe function. Kurtosis and standard error of the mean were removed from the right-hand side to keep the font size legible.

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew
age	1	1046	45.29	19.75	43	44.84	19.27	1	98	97	0.26
sibsp	2	1046	0.5	0.91	0	0.31	0	0	8	8	2.8
parch	3	1046	0.42	0.84	0	0.22	0	0	6	6	2.66

6.13.7 Shapiro-Wilk Normality Test

The menu selection, "Analysis> Summary Analysis> Shapiro-Wilk Normality Test" performs a test to ascertain whether a numeric variable has a normal distribution. The dialog has only a single box to select its variables. When run on the Titanic dataset's age variable, it yields a Shapiro-Wilk statistic of 0.99 and a p-value of .0000, or $p < .001$. The null hypothesis is that the data are normally distributed, and using the standard 0.05 cutoff, we could reject that hypothesis.

However, it is important to consider that every hypothesis test becomes significant if the sample size is large enough. In the case of normality, no distribution is perfectly normal. Figure 6.39 shows a histogram of age, and its distribution is as normal as you are ever likely to see with real data (i.e., not simulated from a mathematical function). This is why statistical tests

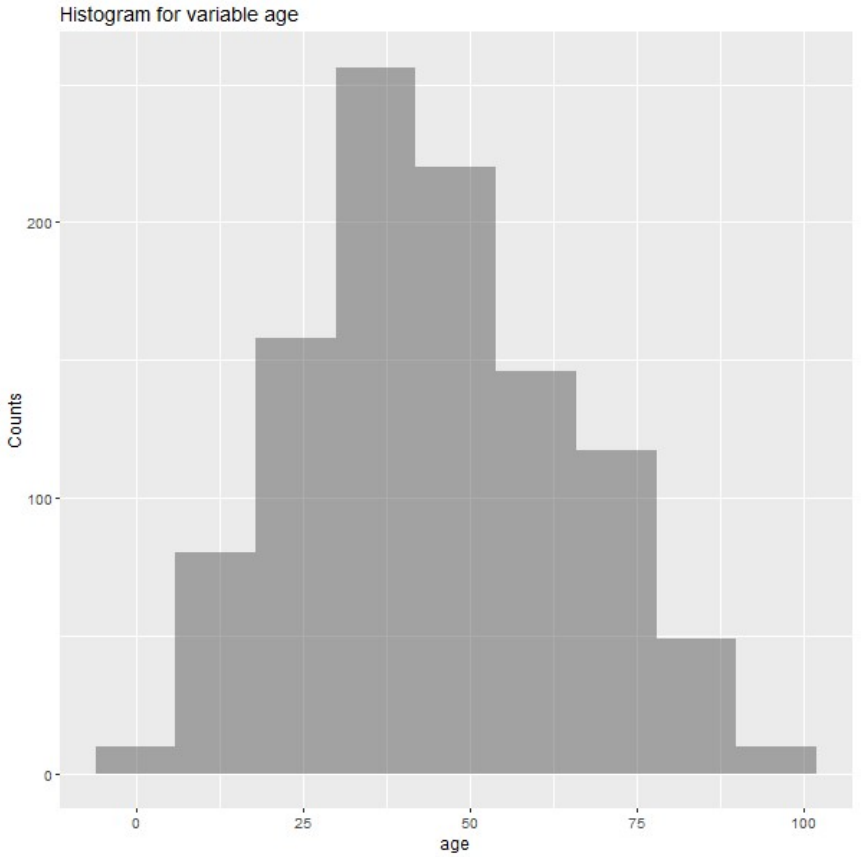


Fig. 6.39: Histogram of age from the Titanic dataset.

of normality are not as popular as using histograms, Q-Q plots, and density plots, described in Chap. 5.

Despite the limits of hypothesis testing in this situation, the Shapiro-Wilk statistic itself is a useful measure. A common rule-of-thumb uses a cutoff value of 0.90; when greater, the variable is considered normally distributed. With its value here of 0.99, that certainly holds in this case.

The calculations for this analysis are performed by the `stats` package's `shapiro.test` function.

6.13.8 Summary Statistics, by group

The menu item, "Analysis> Summary Analysis> Summary Statistics, by Group," provides basic descriptive statistics of numeric variables, including the mean, median, min, max, and quartiles. While it accepts factors in its main variable selection box, it provides only simple counts of each value for them.

Table 6.95: Summary of Titanic variables by group.

sex	base..summary				
	pclass	survived	age	sibsp	parch
female	1st:133	0: 96	Min. : 1.00	Min. :0.000	Min. :0.0000
	2nd:103	1:292	1st Qu.:29.00	1st Qu.:0.000	1st Qu.:0.0000
	3rd:152	NA	Median :42.00	Median :0.000	Median :0.0000
	NA	NA	Mean :43.65	Mean :0.616	Mean :0.6624
	NA	NA	3rd Qu.:58.00	3rd Qu.:1.000	3rd Qu.:1.0000
	NA	NA	Max. :97.00	Max. :5.000	Max. :6.0000
male	1st:151	0:523	Min. : 2.00	Min. :0.0000	Min. :0.0000
	2nd:158	1:135	1st Qu.:32.00	1st Qu.:0.0000	1st Qu.:0.0000
	3rd:349	NA	Median :43.00	Median :0.0000	Median :0.0000
	NA	NA	Mean :46.26	Mean :0.4362	Mean :0.2781
	NA	NA	3rd Qu.:60.00	3rd Qu.:1.0000	3rd Qu.:0.0000
	NA	NA	Max. :98.00	Max. :8.0000	Max. :6.0000

The dialog's "Group By" box allows for one or more factors, and it will repeat the summary analysis for every level of factor provided. Table 6.95 shows the Titanic data repeated by the variable sex.

The `stats` package's `summary` function performs the calculations for this analysis.

6.13.9 Summary Statistics, all variables

The menu item, "Analysis> Summary Analysis> Summary Statistics, all variables," provides basic descriptive statistics of numeric variables, including the mean, median, min, max, and quartiles. It is a unique feature in BlueSky in that it does not have a dialog box. Since its goal is to analyze all variables, it does not need one. However, that would create a problem if you wanted to paste its syntax. If you were building an R program and you want the syntax from this step, you would instead use the menu item described in Sec. 6.13.10, and simply select all the variables manually.

Table 6.96 shows its output. You can also obtain the same type of output broken down by the levels of one or more factors by using the item described in Sec. 6.13.8.

The `stats` package's `summary` function performs the calculations for this analysis.

6.13.10 Summary Statistics, selected variables

The menu item, "Analysis> Summary Analysis> Summary Statistics, all variables" provides basic descriptive statistics of numeric variables, including

Table 6.96: Summary of all Titanic variables.

pclass	survived	sex	age	sibsp	parch
1st:284	0:619	female:388	Min. : 1.00	Min. :0.0000	Min. :0.0000
2nd:261	1:427	male :658	1st Qu.:32.00	1st Qu.:0.0000	1st Qu.:0.0000
3rd:501			Median :43.00	Median :0.0000	Median :0.0000
			Mean :45.29	Mean :0.5029	Mean :0.4207
			3rd Qu.:60.00	3rd Qu.:1.0000	3rd Qu.:1.0000
			Max. :98.00	Max. :8.0000	Max. :6.0000

Table 6.97: Advanced summary of all Titanic variables, with control levels.

pclass	survived	sex	age	sibsp	parch
1st:284	0:619	female:388	Min. : 1.00	Min. :0.0000	Min. :0.0000
2nd:261	1:427	male :658	1st Qu.:32.00	1st Qu.:0.0000	1st Qu.:0.0000
3rd:501			Median :43.00	Median :0.0000	Median :0.0000
			Mean :45.29	Mean :0.5029	Mean :0.4207
			3rd Qu.:60.00	3rd Qu.:1.0000	3rd Qu.:1.0000
			Max. :98.00	Max. :8.0000	Max. :6.0000

the mean, median, min, max, and quartiles. The only selections possible in its dialog are the variables to analyze. The resulting output is the same as for the approach that automatically uses all variables, shown in Tab. 6.96.

If you have many variables to analyze, you might use the approach described in Sec. 6.13.9, which automatically selects all variables.

The `stats` package's `summary` function performs the calculations for this analysis.

6.13.11 Advanced Summary Statistics, all variables, control levels

The menu item, "Analysis> Summary Analysis> Summary Statistics, all variables" provides basic descriptive statistics of numeric variables, including the mean, median, min, max, and quartiles. That is the same information provided by "Summary Statistics, all variables" described in Sec. 6.13.9 with one addition. This one asks you to "specify the number of levels to display counts for factor variables. Note: Less frequent levels of factor variables are summarized in the (Other) category. This is very useful in the case of factors that have many levels, but the counts of many levels are tiny. Except for combining rare levels into "Other," the output will look the same as in Tab. 6.96.

Table 6.98: Survival packages and functions used by Kaplan-Meier.

Package	Function
`survival`	`Surv`
`survminer`	`surv_summary`
`survminer`	`ggsurvplot`
`arsenal`	`tableby`

The `stats` package's `summary` function performs the calculations for this analysis.

6.13.12 Advanced Summary Statistics, selected variables, control levels

The menu item, "Analysis> Summary Analysis> Summary Statistics, all variables" provides basic descriptive statistics of numeric variables, including the mean, median, min, max, and quartiles. That is the same information provided by "Summary Statistics, selected variables" described in Sec. 6.13.10 with one addition. This one asks you to, "specify the number of levels to display counts for factor variables. Note: Less frequent levels of factor variables are summarized in the (Other) category. This is very useful in the case of factors that have many levels but the counts of many levels are tiny. Other than combining rare levels into "Other" the output will look the same as in Tab. 6.96.

The `stats` package's `summary` function performs the calculations for this analysis.

6.14 Survival Analysis

Survival analysis models the time until some event occurs, such as time until the infection, recovery, death, device breakage, customer attrition, and so on. If every observation had the timed outcome, you could use linear regression. However, this type of data collection usually has to conclude before that can happen. Imagine modeling time until a heart attack, but some people never have one. Data lacking the event in question is called "censored" data. It is this censoring that makes linear regression an inappropriate type of analysis.

There are two main types of survival analysis. This section covers models with time the only predictor. Those that have other predictors, e.g., blood pressure, maximum heart rate, and so on, are modeled using Cox Proportional Hazards methods, which I cover in Sec. 8.

Table 6.98 lists the packages and functions BlueSky uses for Kaplan-Meier models.

6.14.1 Kaplan-Meier Estimation, compare groups

The item "Analysis> Survival> Kaplan-Meier Estimation, compare groups" plots survival probabilities across time for two or more groups. Although

you can add confidence lines or shading to the lines, it can get messy to view, especially if you choose to use the shaded ribbon as was used in the one-group example, Fig. 6.42. For two or more groups, the "step" confidence interval is much easier to read. Let us examine a survival model for leukemia.

1. In the Analysis window, use "File> Load Dataset from a Package."
2. In the "Select an R Package..." field, enter "survival."
3. In the "Select a Dataset..." field, enter "aml."
4. Choosing "Analysis> Survival> Kaplan-Meier Estimation, compare groups."
5. In the "Time Variable" field, enter time.
6. In the "Event" field, enter status.
7. In the "Group" field, enter group.
8. Leave the "Plot Type" on the default setting of survival.
9. Under Style Options, choose a plot theme of "theme_grey."
10. Check the "Include Number at Risk" box in the "out" position.
11. Check the "include 95% CI" box, and set its style to "step."
12. If you like, check "Include Censored Times."
13. Check the "include p-value" box.
14. The Style Options dialog should now appear as in Fig. 6.40, so click OK.

The resulting plot is shown in Fig. 6.41. Note the subtle "+" signs that indicate censored points. The p-value of 0.065 indicates that we cannot reject the null hypothesis that the two groups are equal.

Note that the Number at Risk tallies at the bottom shows how many are alive at each point in time. That part of the plot is an integral part of the plot immediately above it, i.e., they form a single image when saved.

6.14.2 Kaplan-Meier Estimation, one group

The Kaplan-Meier plot for one group shows the survival time for one group as it passes through time. Optionally, you can add confidence lines or shading. Let us examine a survival model for leukemia.

1. In the Analysis window, use "File> Load Dataset from a Package."
2. In the "Select an R Package..." field, enter "survival."
3. In the "Select a Dataset..." field, enter "aml."
4. Choosing "Analysis> Survival> Kaplan-Meier Estimation, one group."
5. In the "Time Variable" field, enter time.
6. In the "Event" field, enter status.
7. Leave the "Plot Type" on the default setting of survival.
8. Under Style Options, choose a plot theme of "theme_grey," check the "include 95% CI" box, and set its style to "ribbon."
9. If you like, check "Include Censored Times," and click OK.

The resulting plot is shown in Fig. 6.42. Note the subtle "+" signs that indicate censored points.

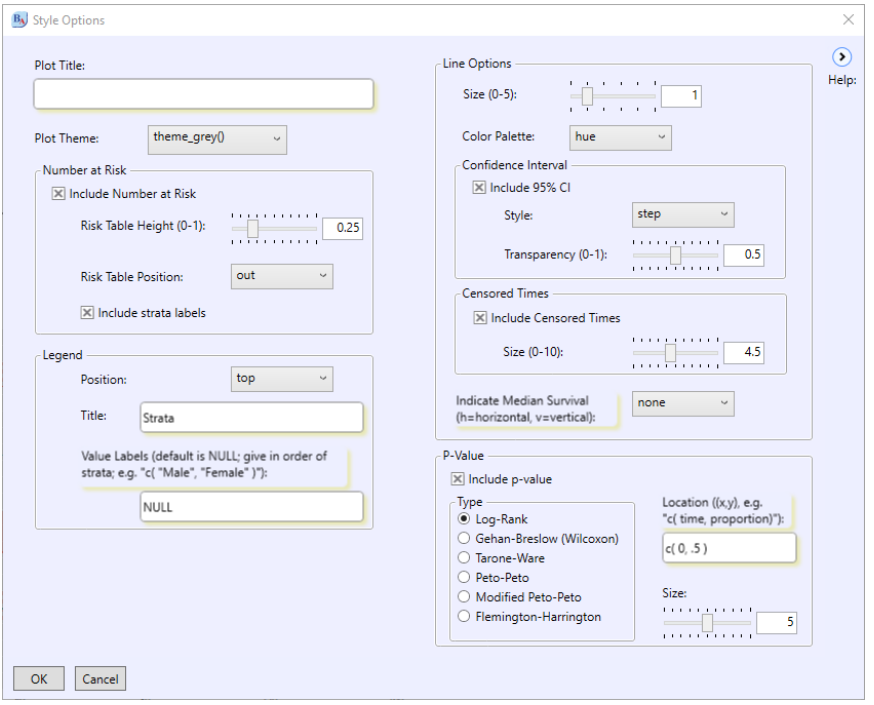


Fig. 6.40: "Analysis> Survival Kaplan-Meier> Style Options" dialog.

6.15 Tables

The "Analysis> Tables" menu offers ways to combine statistical summaries, measures of variability, and hypothesis tests all in a single table. An example is shown in Tab. 6.99. Such tables are frequently the first ones shown in research papers that focus on group comparisons. While showing where the groups *do* differ is, of course, important, such tables are also useful for showing where they *do not* differ. That strengthens the claim that group differences are caused by the study's main intervention, not by other factors.

The main dialog box is so simple that I'm not showing it. It contains only three fields: "Variables to Summarize," "Groups to Compare," and "Strata". If you fill in only variables to summarize, you'll get a single column of statistics for each variable. Adding a group variable will cause that column to split into multiple columns, one for each group. At that point, you might want to add statistical tests, which is a check-box under "Options." Finally, adding a strata variable will get the entire row and column structure to repeat downward as additional rows.

When you include a group variable, both the Basic and Advanced table menus perform ANOVA or Kruskal-Wallis tests. When your group variable has only two levels, then the ANOVA will provide the same p-value as a t-test, and the Kruskal-Wallis will provide the same p-value as a Wilcoxon-Mann-Whitney test. Also, when comparing time-to-event variables, log-rank tests are used.

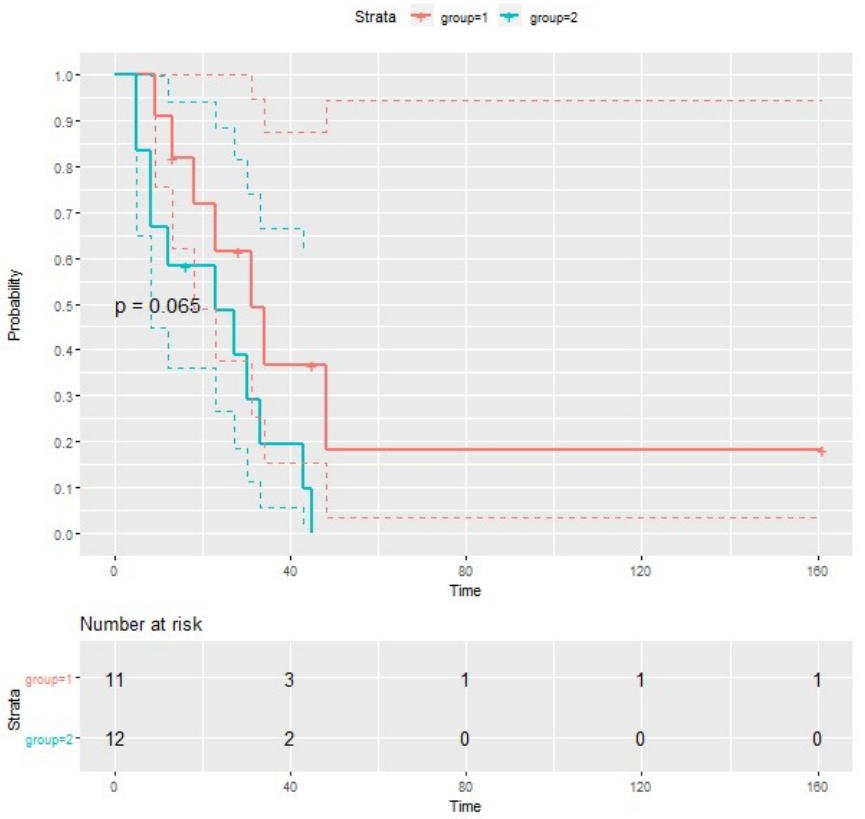


Fig. 6.41: Survival Kaplan-Meier estimates, two groups. Censored times are indicated with "+" signs.

The examples in this section use the mockstudy dataset that is included in the arsenal package. This is a very useful dataset that allows you to practice many different types of tables. To load that dataset into BlueSky, use "File> Load Dataset from a Package" and enter "arsenal" as the package name and "mockstudy" into the dataset name box on the lower right. The dataset consists of 1,499 observations on the following 15 variables:

1. case – a numeric identifier-patient ID.
2. age – age in years.
3. arm – treatment arm (i.e., group) divided into 3 groups, a character string, *not* a factor!
4. sex – a factor with levels Male Female.
5. race – self-reported race/ethnicity, a character string, *not* a factor!
6. fu.time – survival or censoring time in years.
7. fu.stat – censoring status; 1=censor, 2=death.
8. ps – integer, ECOG performance score.
9. hgb – numeric, hemoglobin count.

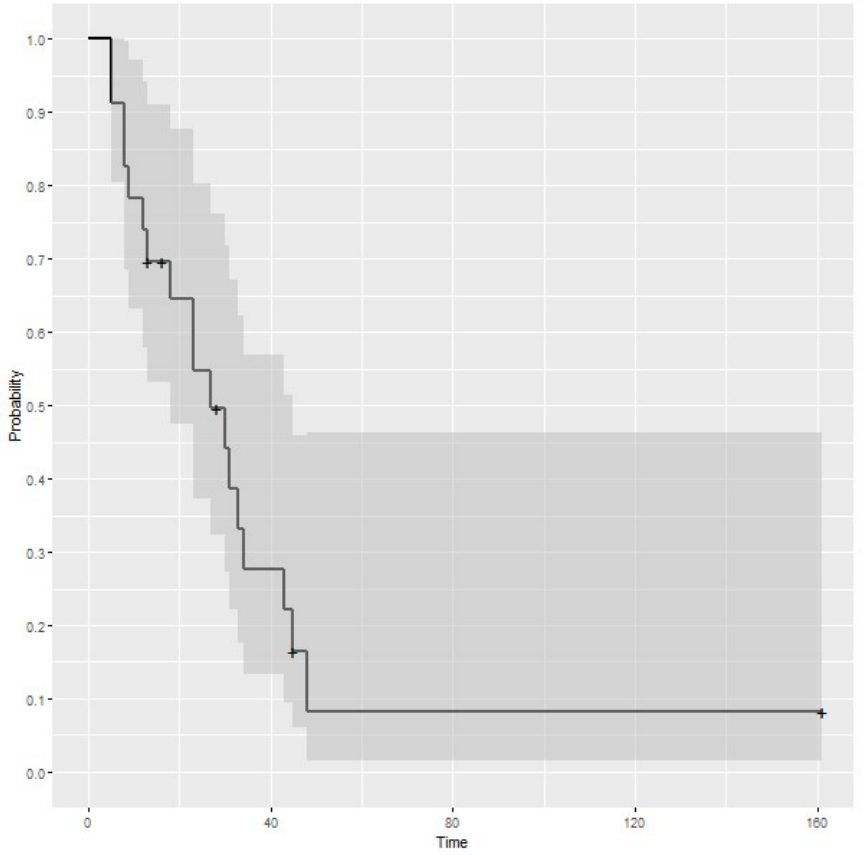


Fig. 6.42: Survival Kaplan-Meier estimates, one group. The 95% confidence intervals are shown using the grey "ribbon" setting, which works well for a single group. Censored times are indicated with "+" signs.

10. bmi – numeric, body mass index, $kg \div m^2$.
11. alk.phos – numeric, alkaline phosphatase.
12. ast – numeric, aspartate transaminase.
13. mdquality.s – integer, LASA QOL 0=Clinically Deficient, 1=Not Clinically Deficient.
14. age.ord – an ordered factor split of age, with levels 10-19 < 20-29 < 30-39 < 40-49 < 50-59 < 60-69 < 70-79 < 80-89.

It is important to notice that arm and race are character variables. While such variables can be used as "Groups to Compare," they cannot be used as "Variables to Summarize" unless you first convert them to factors. See Sec. 4.6 for factor conversion examples.

6.15.1 Basic

Despite its name, the "Analysis> Tables> Basic" menu can create quite complex tables. The "Basic" part of the name comes from the fact that the default values are often sufficient for making the table you need. If not, the "Advanced" menu item allows you to choose the statistical test for each variable in the table. Using the dataset described above in Sec. 6.15, the steps to create Tab. 6.99 are quite simple:

1. Begin creating the table by choosing "Analysis> Tables> Basic."
2. Choose the variables, sex, age, and fu_time, and move them to the "Variables to Summarize" box. If you clicked OK at this point, you would have a single column of statistics.
3. Choose arm and move it to the "Groups to Compare" box. If you clicked OK at this point, you would have three columns of statistics, one for each treatment arm, but no statistical tests.
4. Click the "Options" button and the relatively massive dialog shown in Fig. 6.43 will appear. Check the "Statistical Tests."
5. To completely duplicate our example table, check the Median box, and un-check the box labeled, "Median, (25th %-ile, 75th %-ile)." That causes the median to appear, but not the first and third quartiles, which would provide a measure of variability around the median.

Let us examine the resulting table shown in Tab. 6.99. It compares arms (i.e., treatment groups) of the study in a variety of ways. The sample size of each arm is shown at the top. Next, comes the number and percent of males and females in each arm. The p-value of 0.19 is for a Chi-squared test of independence. Since this value is not below the usual 0.05 cutoff, we cannot reject the null hypothesis that sex is independent of arm. That is probably a good thing as it provides some assurance that differences between treatments would not be due to a large mismatch of genders in each arm.

The next rows contain summaries of two continuous measures, age and fu_time, the survival (or censoring) time. They include the means, followed by the standard deviations (SD) in parentheses, then the medians. By default, the medians would have been followed by the first (Q1) and third (Q3) quartiles. I un-checked that option to save space. Off to the right are p-values from analyses of variance. The test for Age is not significant, but the test for survival time is. That's quite a lot to pack into a single table!

The dialog also contains an optional Strata box. That allows you to repeat the table for levels of another factor.

The "Options" window shown in Fig. 6.43 offers quite a lot of control over the contents of each table. The "Numerical Statistics" options let you choose whether to display the overall sample size (no by default), the number missing if any (yes, by default), and whether to always show the number missing, even if it is zero (no, by default). It also lets you set the way means are displayed, with the option of just the mean, the mean followed by the standard deviation in parentheses (the default), or the mean followed by a 95% confidence interval in parentheses. Lastly, it offers a variety of ways

Table 6.99: Table combining statistical summaries, measures of variability, and statistical tests.

arm	A: IFL (N=428)	F: FOLFOX (N=691)	G: IROX (N=380)	p value
sex				0.19
- Male	277 (64.7%)	411 (59.5%)	228 (60.0%)	
- Female	151 (35.3%)	280 (40.5%)	152 (40.0%)	
Age in Years				0.639
- Mean (SD)	59.673 (11.365)	60.301 (11.632)	59.763 (11.499)	
- Median	61	61	61	
fu_time				< 0.001
- Mean (SD)	553.584 (419.607)	731.246 (487.744)	607.242 (435.509)	
- Median	446.5	601	515.5	

to display quantiles, such as the median, and a variety of quantile-related measures of variability.

The "Categorical Statistics" section offers the same "Sample Size" options, but this time the Mean and Quantiles boxes are replaced with Frequency and Proportion options that are fairly self-explanatory.

The "Date Statistics" section is identical to the "Numerical Statistics" options described above.

The "Global Options" section lets you fine-tune various aspects of the table. The "Include total column" item lets you add totals to the group columns (no, by default). The "Include group variable name" is checked by default, but you can often turn this off as the group's values often sufficient for identifying the group variable. For example, if we were comparing the sexes in the columns, the labels of Male and Female make that obvious, so we might choose to not display the variable name. However, in Tab. 6.99, leaving out the variable name of "arm" might well leave the readers perplexed as to the meaning of the column headings.

The "Statistical Tests" section of the "Global Options" lets you turn on or off the statistical tests. It is turned off by default. When you turn it on, then the four boxes below it are activated, and they allow you to choose the tests for numerical, categorical, and categorical ordinal tests. Those are set the same for all the variables in your table. If you instead wanted to do an ANOVA test for one numeric variable and a non-parametric Kruskal-Wallis test for another variable, then you would have to use the advanced menu described in the next section, Sec. 6.15.2.

The "Pearson and Fisher Simulations" box lets you simulate the p-values using Monte-Carlo methods. This is particularly useful when Fisher's Exact Test would take an inordinate amount of time to calculate.

The `arsenal` package's `tableby` function creates the tables in this section.

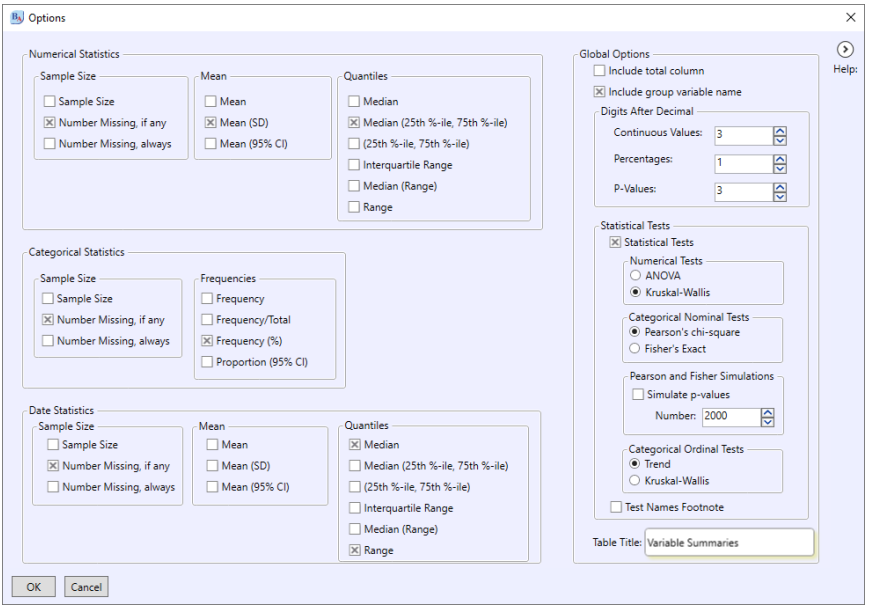


Fig. 6.43: Options dialog from “Analysis> Tables> Basic.”

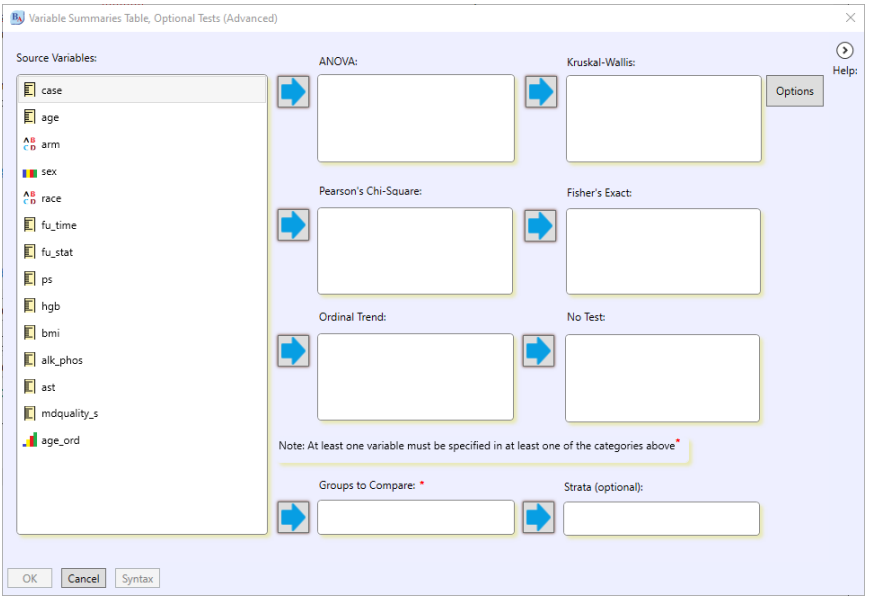


Fig. 6.44: Dialog from “Analysis> Tables> Advanced”.

6.15.2 Advanced

The “Analysis> Tables> Advanced” dialog extends the capabilities of the Basic menu. You should be familiar with that section before reading this one.

The main limitation of the Basic menu is that you can set which statistical test is applied to all variables of a given type. The Advanced dialog, shown in Fig. 6.44 offers statistical test boxes into which you can place variables. Placing a variable in the box named for a particular test tells it to apply that test to the variable chosen. Otherwise, the Advanced tables are essentially identical to the Basic ones.

The calculations for advanced tables are performed by the `arsenal` package's `tableby` function.

6.16 Time Series Analysis

Time series analysis is a way to forecast future values by analyzing the patterns of the past behavior of the variable itself. For example, sales of warm coats peak every twelve months as the winter season arrives. Such data are said to have "seasonality," a change in pattern that is roughly the same each season. The definition of season is quite loose though. If paychecks tend to arrive at the end of each month, causing grocery sales to increase, then month is the "season."

In the case of winter coat sales, the overall growth of the population through the years will slowly cause coat sales to increase, even as the seasonal cycles continue. That type of long term change is called a "trend."

Time series methods make their forecasts by analyzing seasonality and trend and assume that they will continue following that pattern into the future. Since the predictions are into the future, they are adding *rows* to the dataset. That is quite different from most other modeling predictions that BlueSky makes. With linear regression, for example, the prediction would add a new column onto the right side of the dataset. Since time series predictions add rows to the dataset, if you tell it to add the predictions to the existing dataset, it will ask if you wish to over-write it! Therefore, it is best to write predictions to a new dataset. If you make predictions again afterward, then it will ask if you wish to replace that new dataset, which would be fine.

Note that time series predictions depend only on patterns found within the variable itself. There are other, potentially causative variables that might be involved. With our coat sales example, economic health measures such as the unemployment level might also impact sales. Families with an unemployed member might well make do with older coats. The methods of analysis in this section do not address such factors. That is the realm of specialized regression models that handle autoregressive data (data in which observations are correlated through time). BlueSky does not yet handle such models.

In this section, we will predict accidental deaths in the U.S. using the USAccDeaths dataset:

1. In the Analysis window, use "File> Load Dataset from a Package."
2. In the "Select an R Package..." field, enter "datasets".
3. In the "Select a Dataset..." field, enter "USAccDeaths."

For time series analysis, BlueSky uses the popular `forecast` package which is documented in Forecasting: Principles and Practice, a highly recommended book by Hyndman and Athanasopoulos.

6.16.1 Automated ARIMA

ARIMA modeling is a very popular approach to forecasting. ARIMA stands for autoregressive integrated moving average. Autoregressive means that it predicts from past values of the same variable. Moving average is a way of smoothing changes through time by replacing each point with an average of two or more points around it. Weighting is applied so that the closer other values are in time, the greater the weight they receive.

Let us use ARIMA to forecast the USAccDeaths dataset described in Sec. 6.16:

1. Choosing "Analysis> Time Series> Automated ARIMA."
2. In the "Variables to plot" field, enter x.
3. In the "Time of first observation" field, enter "1973,1" which means the first month of 1973.
4. In the "Number of observations per unit of time" field, enter "12".
5. Leave the model criteria at "bic." That stands for Bayesian Information Criterion. Also available are Akaiki's Information Criterion (aic) and that criteria corrected for model complexity (aicc).
6. Click the Options button and check the box for "Plot residuals."
7. Check the "Make predictions using model," set "intervals to predict" to 12, and "confidence interval to "95."
8. Check the "Save predicted values" and fill in the name "Pred_USAcc" as the dataset to save them in. Do *not* use your original dataset name here, as it will be overwritten!
9. Check the "Plot predicted values."
10. Check "Generate correlogram" and enter 36 a the "Max lag."
11. Check the "Ljung-Box test" box.
12. When finished, your dialogs should look like those shown in Fig. 6.45. Click OK to execute the analysis.

We did not ask it to plot the series because the plot, which includes 12 months of predictions, displays the original series, as shown in Fig. 6.46. There, the original values are shown in black, and the predictions are shown with a blue line with gray shading around it for the 95% confidence interval.

The plot of residuals actually preceded the plot of predictions since the model had to be done to get them. The residuals are shown in Fig. 6.47. We would hope to see very small residuals which show no clear pattern. However, no model is perfect, and given this type of model, these are likely the best that can be found.

The next plot is a correlogram (Fig. 6.48), which shows how much autocorrelation exists in the model error at each lag in time. The highest value appears at around 11 months into the series, with a correlation of just over 0.40. We would prefer there to be no such un-accounted for correlations

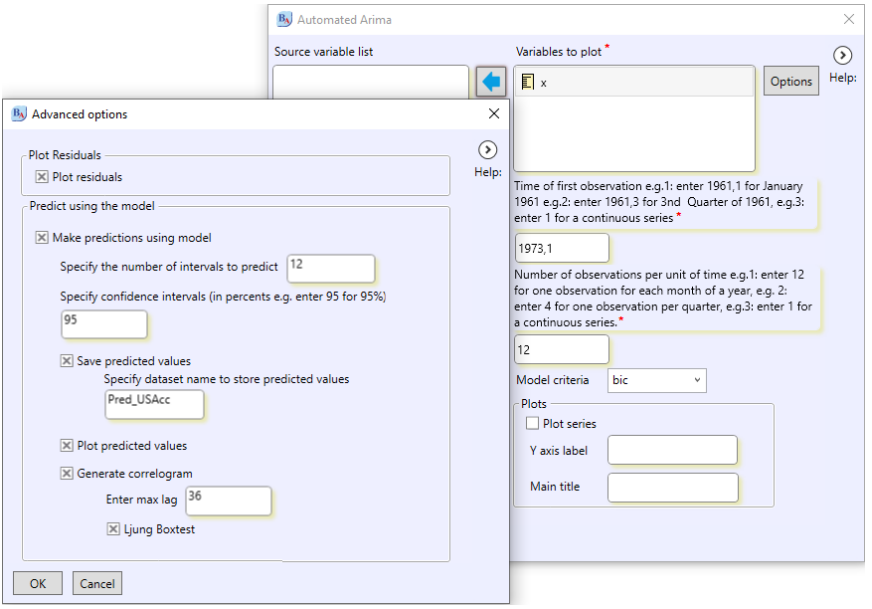


Fig. 6.45: Dialog from "Analysis> Time Series> Automated ARIMA".

left in the data, but some always remain. We will try other models to see if we can do better.

Toward the end of the output are some model measures that are beyond our scope.

The `BlueSky` package's `BSkyAutoArima` function performs the calculations for this analysis. That function uses the `forecast` package's `auto.arima` function. `forecast` package's `auto.arima` function.

6.16.2 Exponential Smoothing

Exponential smoothing is another way to make future forecasts based on patterns found exclusively within the variable itself. Some exponential models have equivalent ones in ARIMA, and vice versa.

Let us use exponential smoothing to forecast the USAccDeaths dataset described in Sec. 6.16:

1. Choosing "Analysis> Time Series> Exponential Smoothing."
2. In the "Variables to plot" field, enter x.
3. In the "Time of first observation" field, enter "1973,1" which means the first month of 1973.
4. In the "Number of observations per unit of time" field, enter "12."
5. Click the Options button and check the boxes for "Save fitted values," and fill in the dataset name that is something new, like "Fitted." Do *not* use your original dataset name here, as it will be overwritten!
6. Check the box for "Plot original vs. fitted."

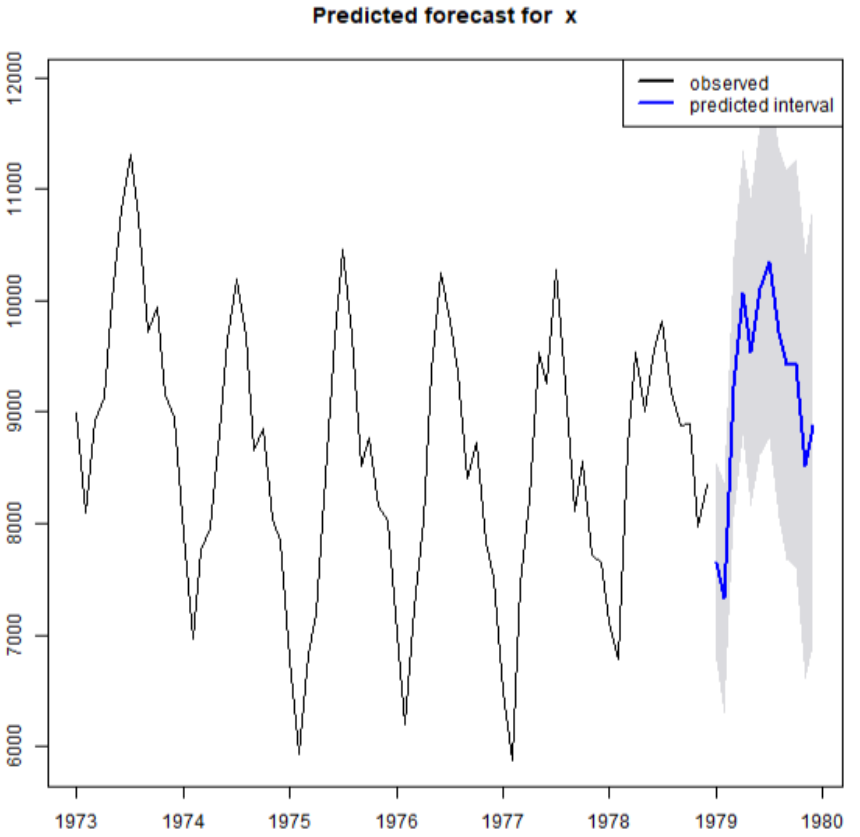


Fig. 6.46: Time series of U.S. accidental deaths with 12 months of predictions.

7. Check the "Make predictions using model," set "intervals to predict" to 12.
8. Check the "Save predicted values" and fill in a new name like "Predicted" as the dataset to save them in. Do *not* use your original dataset name here as it will be over-written!
9. Check the "Plot predicted values."
10. Check "Generate correlogram" and enter 36 a the "Max lag."
11. Check the "Ljung-Box test" box, and click OK.

When finished, your dialogs should look like "Fig. 6.49.

The first plot displays the actual data in black and the fitted values overlaid in red, as shown in Fig. 6.50. Here we are hoping to see the two lines overlap as much as possible.

The next plot shows the time series itself with the twelve predicted values added on to the end in blue. The 95% confidence intervals shade the area around the predictions.

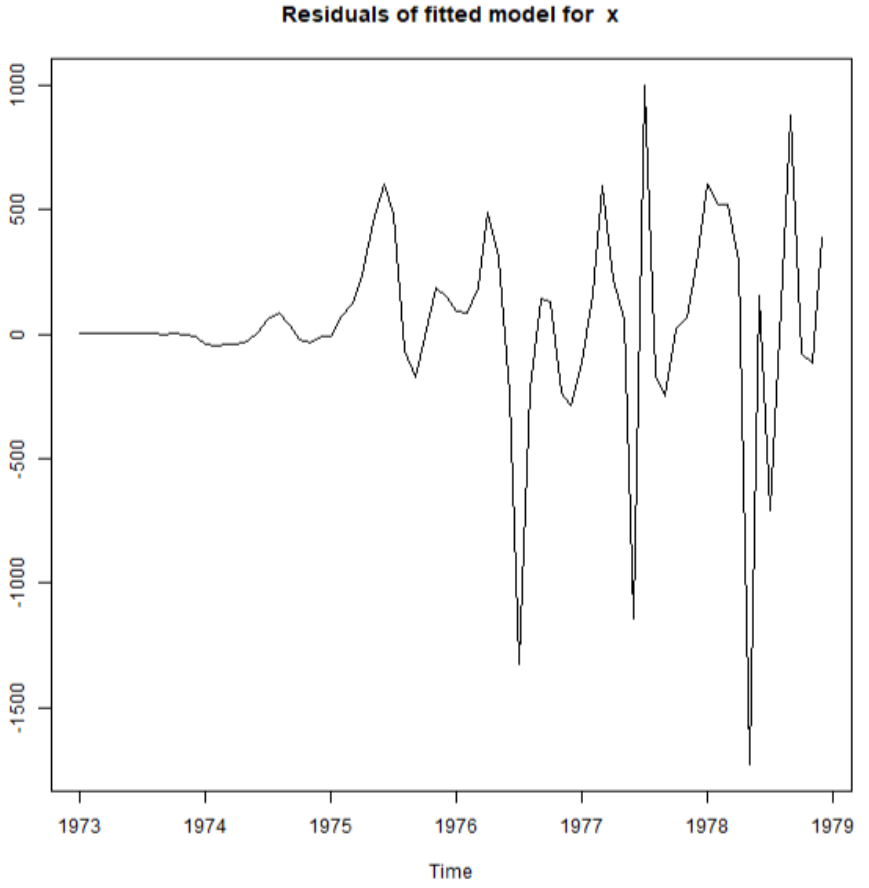


Fig. 6.47: ARIMA model residuals.

A correlogram is shown next, which displays how the model errors are correlated at various time lags (not shown). Various smoothing measures follow that.

6.16.3 Holt-Winters, seasonal

The item "Analysis> Time Series> Holt-Winters, seasonal" calculates a type of exponential smoothing and BlueSky controls it in exactly the same way as covered in Sec. 6.16.2.

6.16.4 Holt-Winters, non-seasonal

The item "Analysis> Time Series> Holt-Winters, non-seasonal" calculates a type of exponential smoothing and BlueSky controls it in exactly the same way as covered in Sec. 6.16.2. The main difference is that this one focuses on data that has an overall trend but no cyclical seasonality.

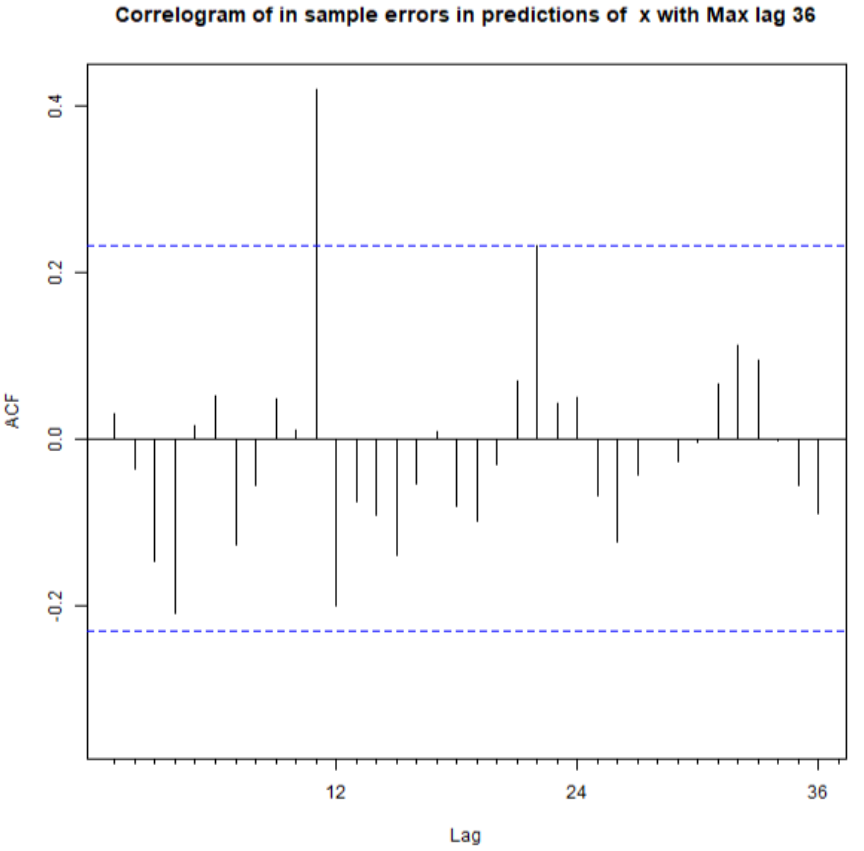


Fig. 6.48: ARIMA model correlogram.

6.16.5 Plot Time Series, separate or combined

The item "Analysis> Time Series> Plot Time Series, separate or combined" does not do any forecasting. It merely allows you to plot one or more variables through time. These can be overlaid on the same plot, or each plot can be done on its own set of axes.

To plot the USAccDeaths dataset described in Sec. 6.16:

1. Choosing "Analysis> Time Series> Plot Time Series, separate or combined."
2. In the "Variables to plot" field, enter x.
3. In the "Time of first observation" field, enter "1973,1" which means the first month of 1973.
4. In the "Number of observations per unit of time" field, enter "12".
5. Choose to plot each series on separate graphs or to combine the series into a single plot.
6. Choose to take the natural log of the data or not, add labels if you like, then click OK.

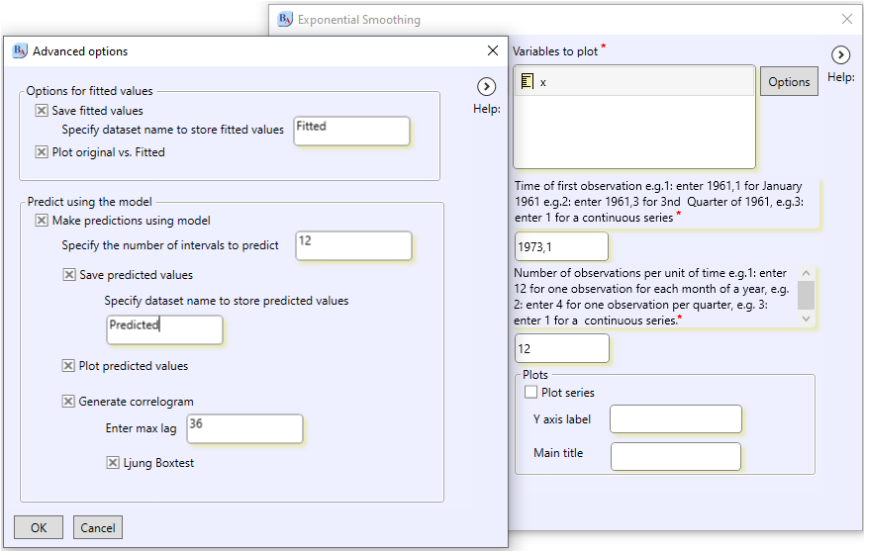


Fig. 6.49: Dialog from "Analysis> Time Series> Exponential Smoothing".

The plots will then appear. The individual plots can also be included in the other analyses, so their output is not repeated here.

6.16.6 Plot Time Series, with correlations

The item "Analysis> Time Series> Plot Time Series, separate or combined" doesn't do any forecasting. It lets you plot autocorrelations, autocovariances and partial correlations. To plot these using the USAccDeaths dataset described in Sec. 6.16:

1. Choosing "Analysis> Time Series> Plot Time Series, with correlations."
2. In the "Variables to plot" field, enter x.
3. In the "Time of first observation" field, enter "1973,1" which means the first month of 1973.
4. In the "Number of observations per unit of time" field, enter "12".
5. Choose to plot autocorrelations, autocovariances, and/or partial autocorrelations.
6. Add labels if you like, then click OK.

The plots will then appear. The autocorrelation plots can also be included in the other analyses, so their output is not repeated here.

6.17 Variance

Many statistical tests that compare groups on the mean of a numeric variable assume that the variance of that variable is the same in each group. Variance tests help you decide of that is true or not. If that is not true, then you can

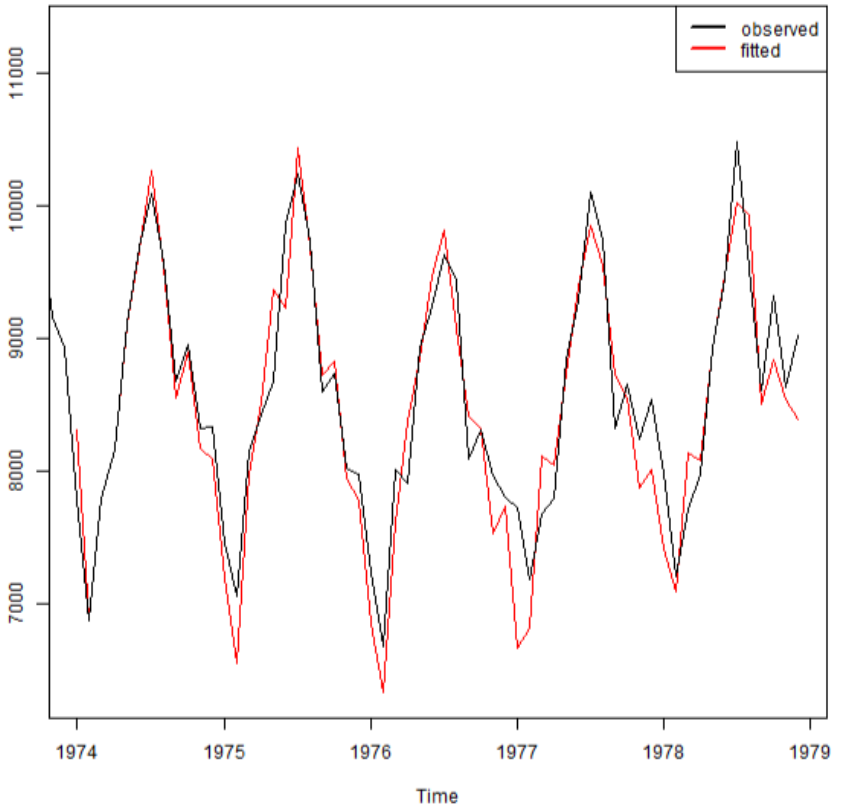


Fig. 6.50: Plot of observed data with exponential smooth fitted values overlaid in red.

consider transforming the data, perhaps taking logarithms or square roots and testing again. Levene's test is generally considered more accurate than Bartlett's test.

In this section we will test to see if the variance of age differs between the Titanic survivors and non-survivors. The variable, "survived" is a numeric one which we will convert to a factor. Here is how to load and transform that dataset:

1. Go to the File menu and choose "Open."
2. Navigate to the file "C:\Program Files\BlueSky Statistics \Sample Datasets and Demos\Sample CSV Datasets\Titanic.csv" select it, and click "Open."
3. Choose "Data> Convert Variables to Factors" and move the variable survived to the "Variables to be converted" box and click OK.

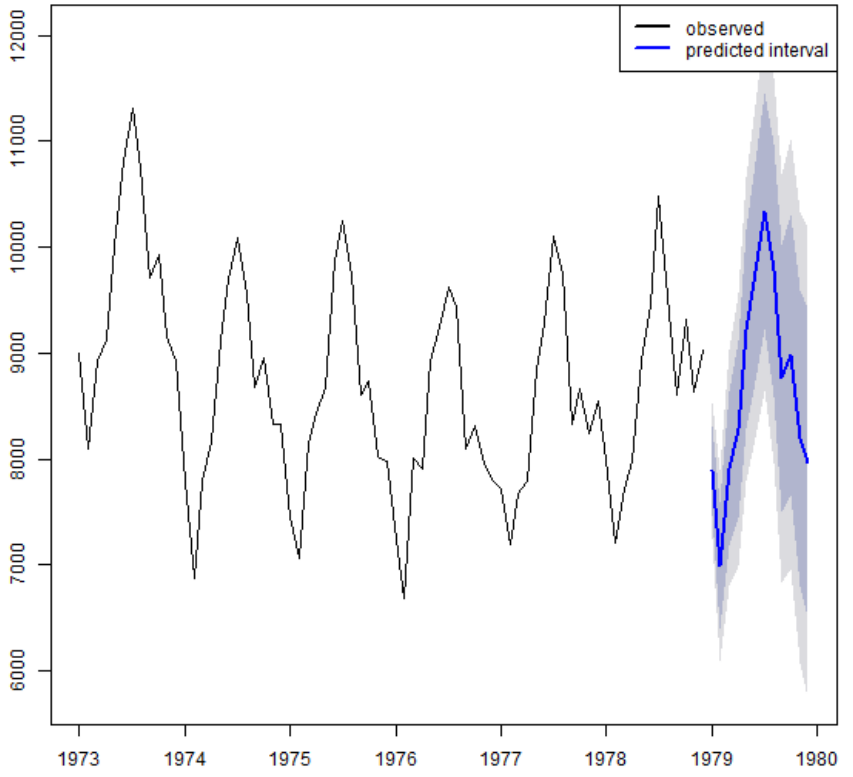


Fig. 6.51: Plot of observed data with exponential smooth predicted values added on in blue. Confidence intervals are shown in gray.

6.17.1 Bartlett Test

The item "Analysis> Variance> Bartlett's Test" offers only two selection boxes in its dialog: the response variable (age, in our case), and Factor Variables (survived).

The first output table shows the variance of each group, 423.18 for survivors and 366.01 for non-survivors. Next, it shows Bartlett's K-squared statistic of 2.68 with 1 degree of freedom and a p-value of 0.10. At the usual 0.05 significance level, we would not reject the hypothesis that there is no difference in variance between these two groups.

The calculations for this test are performed by the `bartlett.test` function from the `stats` package.

6.17.2 Levene Test

The item "Analysis> Variance> Levene's Test," prompts for two variables in its dialog: the response variable (age, in our case), and Factor Variable (survived). It also asks for the measure to use as the center of the distribution: the mean or median.

Unfortunately, at the time of writing, this shows summary statistics, but not the variances themselves. You can obtained those by running Bartlett's test, as described in Sec. 6.17.1.

Next, it shows the F test of 3.05 with 1 degree of freedom, and a p-value of 0.08. At the usual 0.05 significance level, we would not reject the hypothesis that there is no difference in variance between these two groups.

The `car` package's `leveneTest` function performs the calculations for this test.

6.17.3 Variance Test, two samples

The item "Analysis> Variance> Variance Test, two samples," provides a variance test on the ratio of the group variances. It also provides a confidence interval for that ratio. It is the only test in this section which provides the option for a directional alternative hypothesis.

As with the other methods, its dialog asks for two variables: the response variable (age, in our case) and Factor Variables (survived).

In addition, it asks for your alternative hypothesis, which allows you to choose if the difference in the variances is greater or less than one. The default setting is not equal to one, the same as the other variance tests discussed in this part of the menu structure. Note that at the time of writing, the dialog box is labeled as comparing to zero, not one. Hopefully, this will be fixed by the time you read this.

The first output table shows the variance of each group, 423.18 for survivors and 366.01 for non-survivors. Next, is the F test of 0.86, exactly the same as with Levene's test. This time degrees of freedom are listed as 618 in the numerator and 426 in the denominator. So we would write this as $F(618, 426) = 0.86$, $p = 0.10$. As before, we cannot reject the null hypothesis.

The results then provide a ratio of the two variances, and 95% confidence interval around that ratio.

The calculations for this test are performed by the `leveneTest` function from the `car` package.

7

Distribution Menu

BlueSky's Distribution menu offers ways to obtain information from probability distributions, such as the bell-shaped normal curve. For each distribution, it offers four choices:

7.1 Probabilities

With this set of menu items, you provide the parameters of interest, and it generates the probability that matches it. Let us examine the item, "Distribution> Continuous> Normal Distribution> Normal Probabilities." Using its dialog shown in Fig. 7.1, I filled in a value of 1.96. It offered a normal distribution with a mean of zero and a standard deviation of 1, which I accepted. I left the choice of tail on "Lower tail" and clicked OK. It returned the probability of 0.975. That is the value that has 2.5% of the probability in each tail of the distribution, which yields a 95% confidence interval.

7.2 Quantiles

The quantiles menu items are the reverse of the probability ones: you enter a quantile, and it returns the distribution value that matches it. I used the item "Distribution> Continuous> Normal> Normal Quantiles" to open the dialog shown in Fig. 7.2. I entered the value of 0.975, and it returned a probability of 1.96.

7.3 Plot Distribution

BlueSky can generate plots of most distributions to your specifications. The item, "Distribution> Continuous> Normal Distribution> Plot Normal distribution" displays the dialog shown in Fig. 7.3. You can choose the density function or the distribution function. Regions under the density

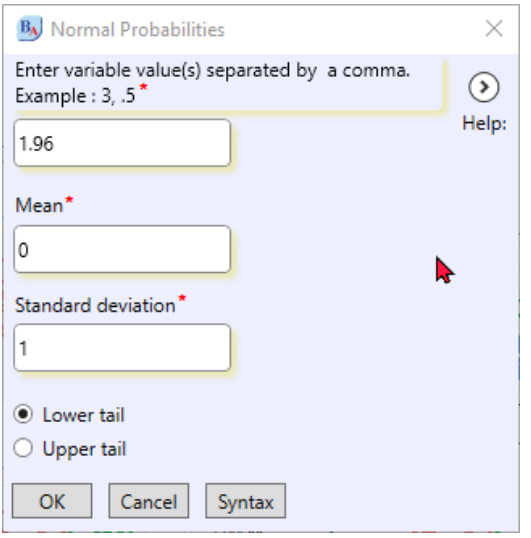


Fig. 7.1: Normal probabilities dialog settings which generate a cumulative probility of 0.975.

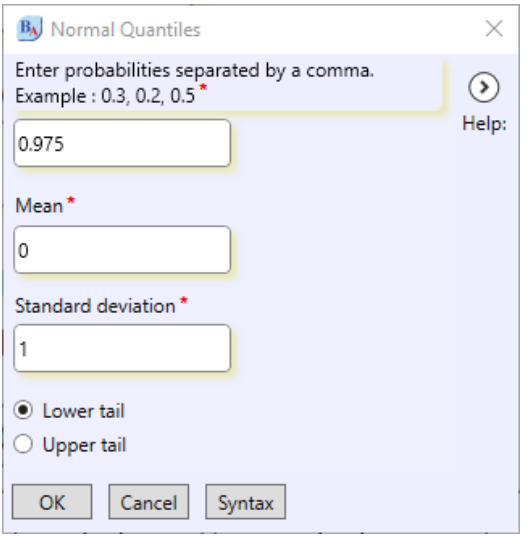


Fig. 7.2: Normal quantiles dialog settings that generate a value of 1.96.

function can be specified by either x-values, or quantiles. I have asked it to use a mean of zero and a standard deviation of one. In addition, I requested shading the area under the curve from -1.96 to +1.96 standard deviations, a 95% confidence interval. The plot is shown in Fig. 7.4.

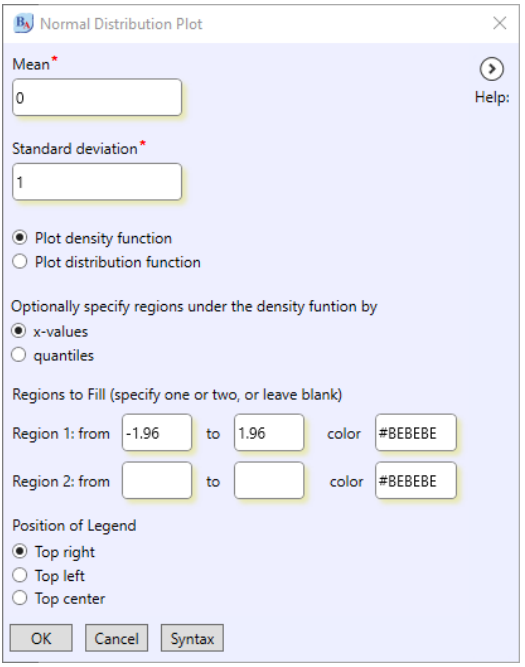


Fig. 7.3: Normal plot dialog settings ranging from +/- 1.96.

7.4 Sample From Distribution

BlueSky will generate variables or entire datasets from many distributions using the item, "Distribution> Continuous> Normal Distribution> Sample From Normal Distribution." Its dialog is shown in Fig. 7.5. I have asked it to generate 1,000 rows and 1 column from a standard normal distribution.

Next, I used, "Graphics> Histogram" to plot the data, which is shown in Fig. 7.6. Note that to get the very same result, you need to make sure your random generator seed is set to "1234."

7.5 Discrete Tail Probabilities

For discrete distributions, BlueSky includes one additional menu item: tail probabilities. For example, if we wanted to know the probability of getting at least 1 success (we will call that "heads") out of 2 coin tosses, the outcomes could be:

```
HH
HT
TH
TT
```

so the probability is 3/4 or 0.75 in the lower tail and 1/4 or 0.25 in the upper. Use the item "Distribution> Discrete> Binomial Distribution> Binomial

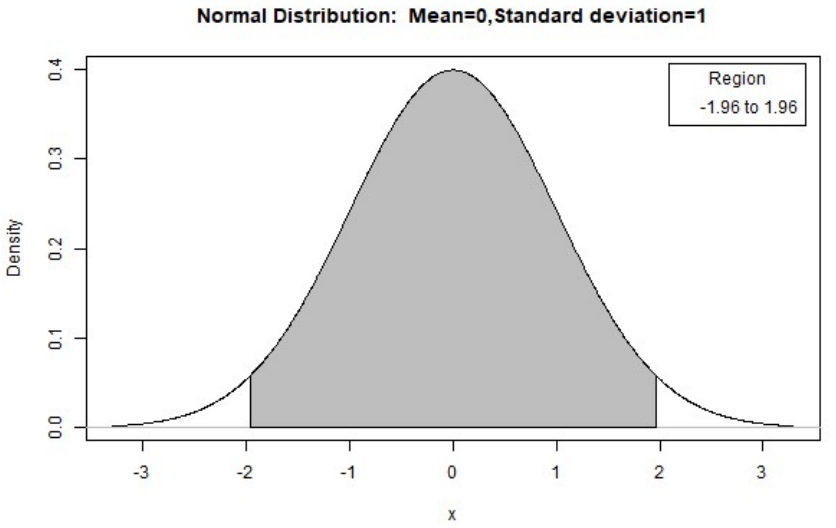


Fig. 7.4: Normal distribution plot with shading from +/-1.95 standard deviations.

Tail Probabilities" and enter those values into the dialog as shown in Fig. 7.7. After clicking OK, BlueSky responded that the probability of obtaining at least one head in two trials is 0.75.

7.6 Continuous Distributions

BlueSky can perform all of the tasks demonstrated in the preceding section (Sec. 7) for the following continuous distributions:

- Beta
- Cauchy
- Chi-squared
- Exponential
- F
- Gamma
- Gumbel
- Logistic
- Lognormal
- Normal
- t
- Uniform
- Weibull

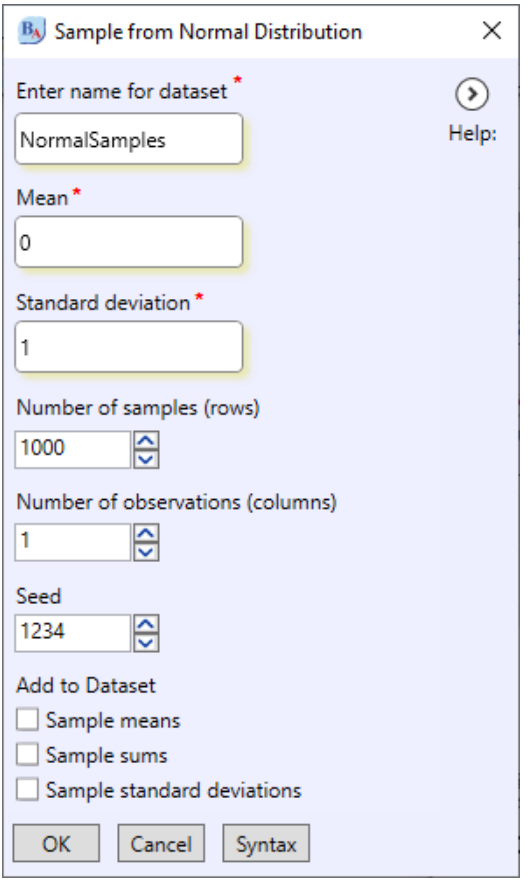


Fig. 7.5: Normal sample dialog.

7.7 Discrete Distributions

BlueSky can perform all of the tasks demonstrated in the preceding section (Sec. 7) for the following discrete distributions:

- Binomial
- Geometric
- Hypergeometric
- Negative Binomial
- Poisson

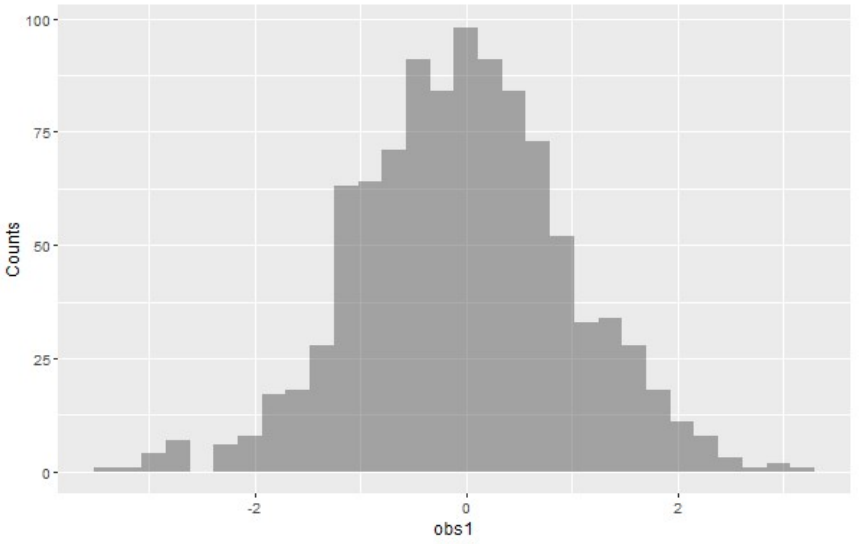


Fig. 7.6: Histogram of the generated normal sample.

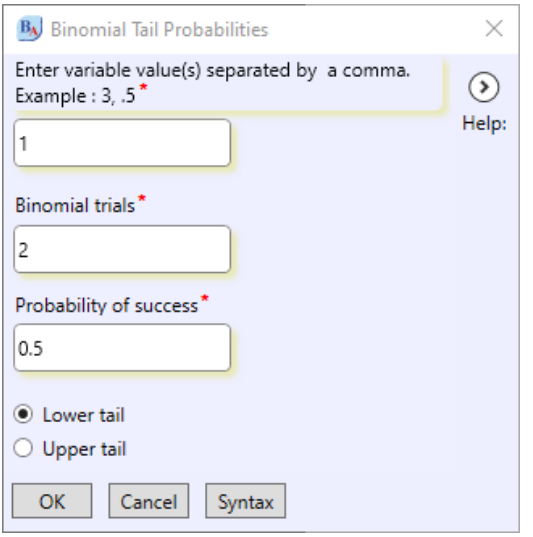


Fig. 7.7: Binomial dialog for tail probabilities.

8

Model Fitting, Brief Overview

The Model Fitting menu items offer precise manual control over the model building process. Validating models created with these menu items is also a manual process. To perform it, you would split the data into training, and testing sets using "Data> Split Dataset for Partitioning" covered in Sec. 4.28.2. Afterward, you would use the "Score Current Dataset" box to generate predictions and evaluate their accuracy, as described in Sec. 10.

In contrast, the Model Tuning menu (Sec. 9) offers more automated model building, tuning, and validation all at once using the `caret` package, but with less fine control. Each of these menus is independent; you do not fit the model here and tune it there. After using either the Fitting or Tuning menu, you can then use the models you created as described in Chapter 10.

The additional topics listed below are included only in the *BlueSky Statistics 7.1 User Guide* available at http://Lulu.com and other bookstores.

- Modeling Basics
- Contrasts Display
- Contrasts Set
- Cox Proportional Hazards Model
- Decision Trees
- Extreme Gradient Boosting
- GLZM - Generalized Linear Models
- IRT, Item Response Theory
- KNN, K-Nearest Neighbors
- Linear Modeling
- Linear Regression
- Logistic Regression
- Mixed Models, basic
- Multinomial Logit
- Naive Bayes
- Neural Networks
- Ordinal Regression
- Quantile Regression
- Random Forest

- Summarizing Models for Each Group

9

Model Tuning, Brief Overview

Model tuning offers automation via the popular `caret` package, but with less fine control than the Model Fitting menu items have. Each of these menus is independent; you *do not* fit the model there and then tune it here. The Model Tuning menu offers access to more model types and more R packages than does the Model Training menu. That is due to the fact that the `caret` package makes developing dialogs particularly easy. After using either the Tuning or Fitting menus, you can then use your models to do the tasks described in the Model Using chapter.

The model tuning dialogs all use the dependent variable's class to determine the type of model to provide. If that variable is numeric, it fits a regression model. If the variable is a factor, it fits a classification model instead.

The additional topics listed below are included only in the *BlueSky Statistics 7.1 User Guide* available at http://Lulu.com and other bookstores.

- Validation Methods
- Preparing Data for Model Tuning
- AdaBoost Classification Trees
- Bayesian Ridge Regression
- Boosted Trees
- Decision Trees
- K Nearest Neighbors
- Linear Regression
- Multi-variate Adaptive Regression Spline
- Naive Bayes
- Neural Network
- Random Forest
- Robust Linear Regression
- Support Vector Machines

10

Model Using, Brief Overview

Once you have created a model using either the Model Fitting or Model Tuning menus, you can then use them by selecting items from the Model Statistics menu. You can also use your models with the "Score Current Dataset" panel located in the upper-right corner of the Analysis window. That will make predictions by applying the model you choose to the active dataset.

The additional topics listed below are included only in the printed *BlueSky Statistics User Manual* available at http://Lulu.com and other bookstores.

- Score Current Dataset
- Model Type
- Load Model
- Pick a Model
- Save Model
- Score
- Add Model Statistics to Observations
- AIC
- ANOVA & Likelihood Ratio Test
- BIC
- Bonferroni Outlier Test
- Compare Models
- Confidence Interval
- Hosmer-Lemeshow Test
- IRT
- Model-Level Statistics
- Parameter-Level Statistics
- Plot a Model
- Pseudo R Squared
- Stepwise
- Summarize a Model
- Variance Inflation Factors

11

Tools Menu

The Tools menu contains dialogs that help you manage BlueSky's setup and the underlying R language that it uses to accomplish each task.

11.1 Dialog Inspector

More advanced BlueSky users use the Dialog Inspector menu. Choosing it brings up a file browser dialog, which prompts you to find a BlueSky dialog for it to inspect. Those dialogs are all located in the Dialog Files folder, located here:

```
C:\Program Files\BlueSky Statistics\Dialog Files
```

That folder contains one folder for each of the main menu headings, such as Analysis, Data, Distribution, Graphics, and so on. You can open any one of those folders and choose a dialog to inspect. Let us examine the dialog used for creating scatterplots.

1. Choose "Tools> Dialog Inspector."
2. Navigate to "C:\Program Files\BlueSky Statistics\Dialog Files\Graphics."
3. Choose the file "Scatterplot.bsky."
4. Click on "Open."

The chosen dialog should appear, as seen in Figure 11.1. The top of the dialog contains some text describing the message that appears whenever someone hovers their pointer over the dialog buttons. In this case, they are not very informative, saying things like, "Clicking on the Syntax button will bring up the syntax in the Command Editor." At the bottom, a tiny property grid displays the properties of the dialog. There you can see useful information such as the R packages required, ggplot2 and ggthemes, in this case. It also shows a field named RHelpText, which is where it will find the R help file related to the dialog. In this case, it says, "help(geom_point, package = ggplot2)." That tells you the name of the R function, geom_point, and the package it comes from, ggplot2. You can learn more about that function by choosing "Help> Help on function" and entering its name.

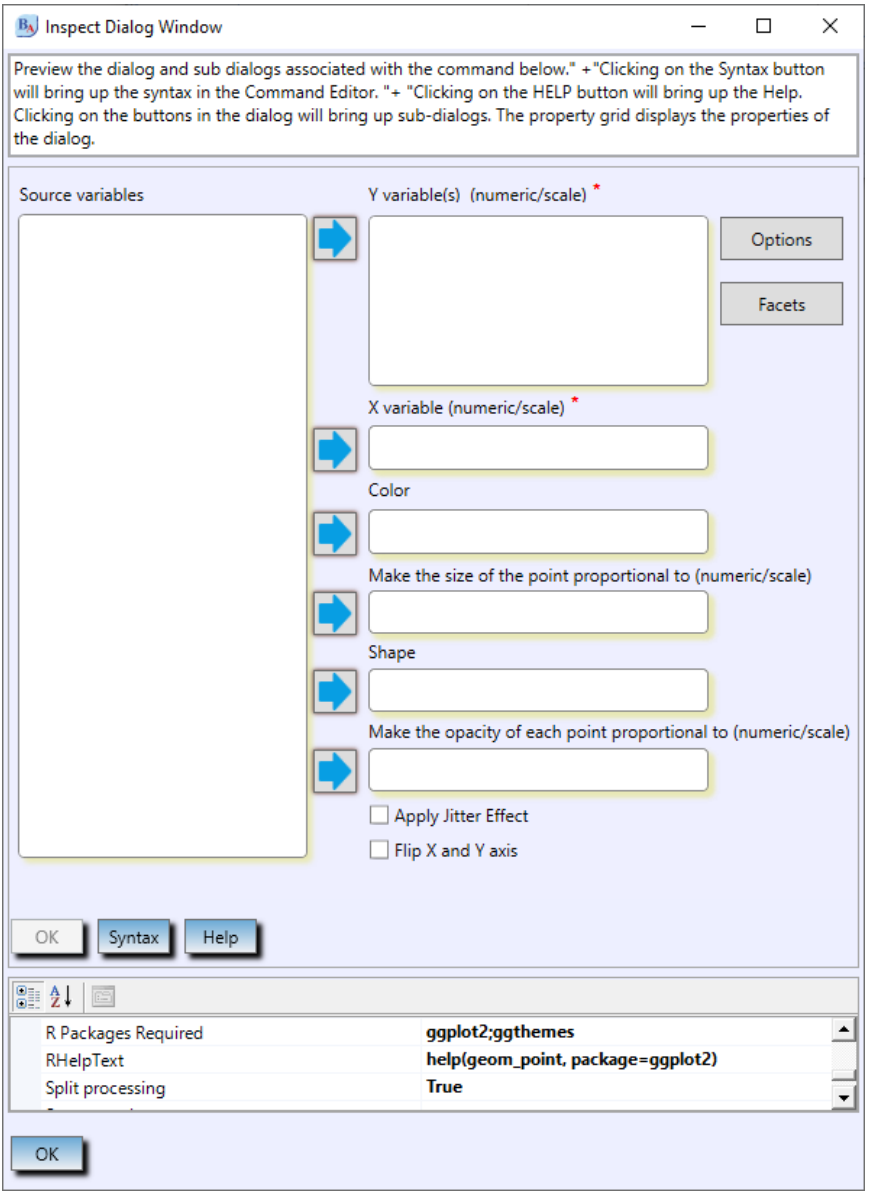


Fig. 11.1: File Inspector's view of the scatterplot dialog.

11.2 Dialog Installer

The main job of the Dialog Installer is to, well, install dialogs! When you download a dialog from http://BlueSkyStatistics.com, or you build your own, install it using "Tools> Dialog Installer." Choosing it will bring up a window that lays out the entire menu structure, offering you a place to put the new one. In Fig. 11.2, I wanted to install a new menu under the Analysis

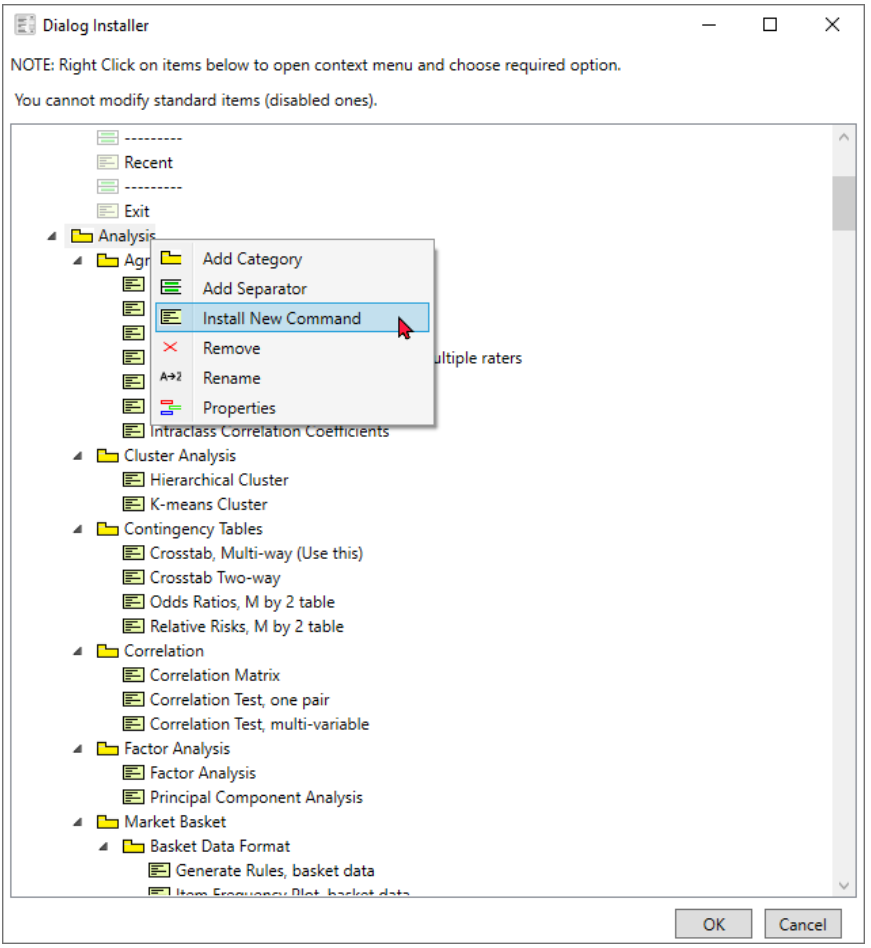


Fig. 11.2: The Dialog Installer's layout of the menu structure.

menu. I right-clicked on it, and you can see that it offers me several choices. From there I can "Add Category," under which I could install several menu items. If I choose "Install New Command," it will bring up a file browser to choose the command to install.

The Dialog Installer also lets you remove or rename menu items, install a gray separator line to help differentiate sections of a menu, or examine a menu item's properties.

11.3 Package

BlueSky does its work through the R language, which organizes its functions (commands) into collections called packages. When you install BlueSky, you also install a copy of R and all the packages it needs. However, if you write R programs, you will need to be able to manage the R packages manually.

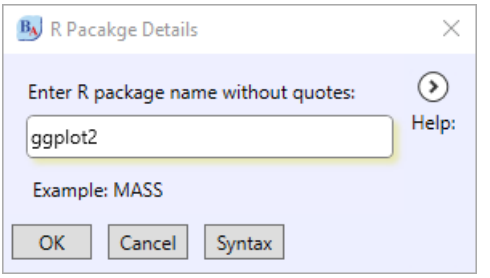


Fig. 11.3: R package details dialog.

The "Tools> Package" menu contains the sub-menus to accomplish this task.

11.3.1 R Version Details

To see the version of R BlueSky is using, choose "Tools> Package> R Version Details." When I did so, it showed me details about both R and my operating system:

```
   R Version Currently In Use

platform       x86_64-w64-mingw32
arch           x86_64
os             mingw32
system         x86_64, mingw32
status
major          3
minor          6.1
year           2019
month          07
day            05
svn rev        76782
language       R
version.string R version 3.6.1 (2019-07-05)
nickname       Action of the Toes
```

Whenever you ask a tech support question, this is important information to add to the end of your problem description.

If BlueSky ever gets confused about what version of R to use, it stores that information in the folder "C:/Users/YourID/AppData/Roaming/BlueSky." Upon startup, BlueSky looks in that folder for the location of your copy of R. If that folder does not exist, it will recreate it, which should clear up any confusion it may have over what version of R you are using.

11.3.2 R Package Details

To obtain detailed information about an R package, choose "Tools> Package> R Package Details," and enter its name. In Fig. 11.3, I filled in the name of the popular ggplot2 package. Here's a subset of all the data it provided:

```
Package: ggplot2
Version: 3.2.1

Title: Create Elegant Data Visualisations Using
   the Grammar of Graphics
Description: A system for 'declaratively' creating
  graphics, based on "The Grammar of Graphics"...

Authors\@R: c( person("Hadley", "Wickham", ,
   "hadley@rstudio.com",
   c("aut", "cre")), person("Winston", "Chang", ,
   role = "aut"), ...

Depends: R (>= 3.2)

Imports: digest, grDevices, grid, gtable (\>= 0.1.1),
   lazyeval, MASS, mgcv, reshape2, rlang (>= 0.3.0),
   scales (\>= 0.5.0), stats,...

Enhances: sp

License: GPL-2 | file LICENSE

URL: http://ggplot2.tidyverse.org,
     https://github.com/tidyverse/ggplot2
BugReports: https://github.com/tidyverse/ggplot2/issues
...
Date/Publication: 2019-08-10 22:30:13 UTC
Built: R 3.6.1; ; 2019-11-07 18:33:34 UTC; windows

-- File: C:/Program Files/BlueSky Statistics/...
   R-3.6.1/library/ggplot2/Meta/package.rds
```

11.3.3 Show Installed Packages

To obtain a list of all R packages installed in BlueSky's copy of R, choose "Tools> Package> Show Installed Packages." It returns quite an impressive list. At the time of writing, it was 976 total packages! By the time you read this, it will likely be far more. Here is a tiny subset of the output it provided:

```
"abind"  "acepack"  "acs"  "aplpack"
```

```
"arsenal"  "arules"  "arulesViz"  "askpass"
...
"xtable"  "xts"  "yaImpute"  "yaml"
"yardstick"  "zeallot"  "zip"  "zoo"
```

11.3.4 Update BlueSky Package from Zip

BlueSky Statistics comes with everything it needs to run in a single installation. Part of that installation is a set of R functions called the "BlueSky Package." It is rarely necessary to update this, but if you run into a problem that the BlueSky Technical Support team tells you requires a software update, they will send you a new package. To install it, choose, "Tools> Package> Update BlueSky Package from Zip." It will open a file browser that will let you locate the file they gave you. The process will complete automatically from there, but you will need to restart BlueSky for it to take effect.

11.3.5 Install Package(s) from Zipped File(s)

Packages containing additional R language features are available on the Comprehensive R Archive Network, CRAN https://cran.r-project.org/. The easiest way to obtain and install these is directly from CRAN, as described in Sec. 11.3.6. However, it is occasionally useful to install packages from zipped files that are also downloadable from CRAN. An example is when teaching a workshop for many people when they will not have Internet access, or if everyone downloading a large set of packages would overload the WiFi connection.

With each analysis you run in BlueSky, various packages are loaded into memory. Having some packages loaded may interfere with the installation of other packages. Therefore, it is best to install new packages immediately after starting BlueSky.

When you have a zipped R package to install, choose, "Tools> Package> Install Package(s) from Zipped File(s)." That will open a file browser, allowing you to locate and open the needed package. It will install directly from the zipped package, with no need to unzip it beforehand. When finished, restart BlueSky to make the new package available.

11.3.6 Install/Update Package(s) from CRAN

Packages containing additional R language features are available on the Comprehensive R Archive Network, CRAN https://cran.r-project.org/. However, with each analysis you run in BlueSky, various packages are loaded into memory. Having some packages loaded may interfere with the installation of other packages. Therefore, it is best to install new packages immediately after starting BlueSky.

The easiest way to obtain and install new packages is to choose "Tools> Package> Install/Update Package(s) from CRAN." BlueSky will prompt

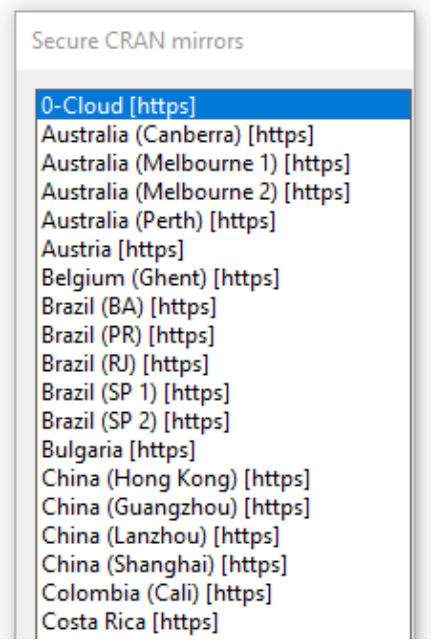


Fig. 11.4: The list of CRAN "mirror" sites from which you can download R packages.

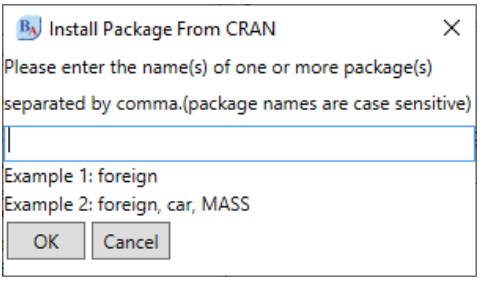


Fig. 11.5: The prompt asking which package you wish to install.

you to choose a download site, called a "mirror," shown in Fig. 11.4. It is best to choose the default one named, "0-Cloud [https]." That one has copies on every continent, and it allows for the easy tracking of downloads, which helps determine package usage.

Next, BlueSky will prompt you for the name of the package you need, as shown in Fig. 11.3.6. You can give it one name, or multiple names separated by commas. Note that case matters, so never capitalize the first letter of a package's name unless you see that is precisely how the name is written.

If you ever need to install an R package in a situation with limited or no Internet access, see the instructions on installing a package from zip files, Sec. 11.3.5.

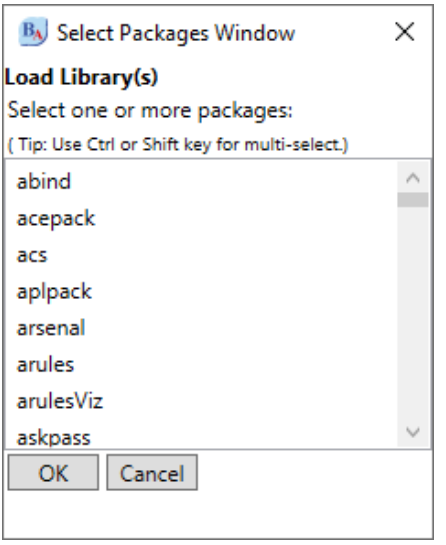


Fig. 11.6: Load packages dialog.

11.3.7 Load Package(s)

R packages are stored on your hard drive. The software must load them from there into your computer's main memory. BlueSky does that automatically for any task that you choose from its menus. However, if you are entering your own R code, then you must load the package yourself. To do so, choose "Tools> Package> Load Package(s)," and it will offer a list of packages that are currently installed, as shown in Fig. 11.6.

You then choose the package(s) you need to load by clicking on its name. You can also use Shift-click to make contiguous selections and Ctrl-click (i.e. hold own the Ctrl button, then click) to select multiple packages that are not next to each other in the list. Clicking, "OK" will then cause BlueSky to enter the appropriate library function call to load the chosen packages. Afterward, you can enter any R code that uses those packages.

Since this step is only needed when programming in R, it is often easier to manually enter your own library call. But for entering long lists of packages, using this menu can be quicker and more error-free. Of course, you can always save the library command that BlueSky writes to your R program for future use. As I write this, there is no Syntax button on this dialog, but you can always copy it from the Output window and paste it into the Syntax panel with the rest of your R code.

11.3.8 Load User Session Package(s)

As pointed out previously, R packages are stored on your computer's hard drive. They are then loaded into your computer's main memory when needed. Each task in BlueSky will load the packages it needs at the time of execution. However, you can instead choose to load a custom set of packages in advance.

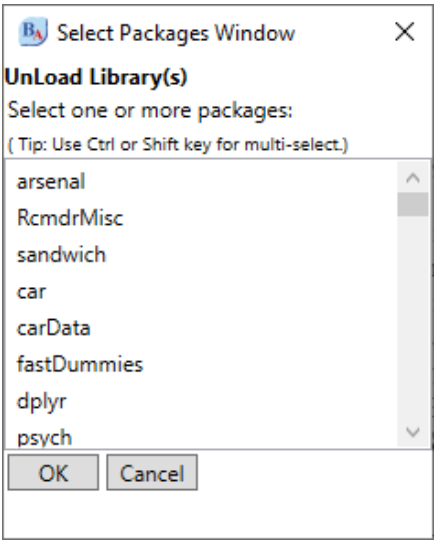


Fig. 11.7: Unload packages dialog.

That trades off memory for a slight increase in speed later. Once loaded, the packages do not unload automatically, so you will give up that amount of memory for the duration of your BlueSky session.

To create your own list of packages to load, follow the instructions below in Sec. 11.4.5. Then you load that set into main memory by choosing "Tools> Package> Load user Session Packages." This is one of the few BlueSky menus that does not display a dialog box. It will simply load them all at once as soon as the menu name is selected.

11.3.9 Unload Package(s)

It may be occasionally necessary to remove an R package from memory. This most often happens when you install and load your own packages. The more recently loaded one may contain a function with the same name as one loaded earlier, interfering with its use. If that happens, use the item "Tools> Package> Unload Package(s)." The dialog in Fig. 11.7 will appear. It is labeled "Unload Library(s)" but that is a typo. In R, the term "package" is what many other languages call a "library." To unload a package, choose its name, and click OK.

11.3.10 Uninstall Package(s)

When you no longer need an R package that you have manually added to BlueSky, you can uninstall it to save some disk space. To do this, choose "Tools> Package> Uninstall Packages." The dialog shown in Fig. 11.8 will appear. Choose the package(s) you wish to uninstall and click "OK." BlueSky then removes them. The use of Shift-click and Ctrl-click work as ways to select multiple packages in the dialog box.

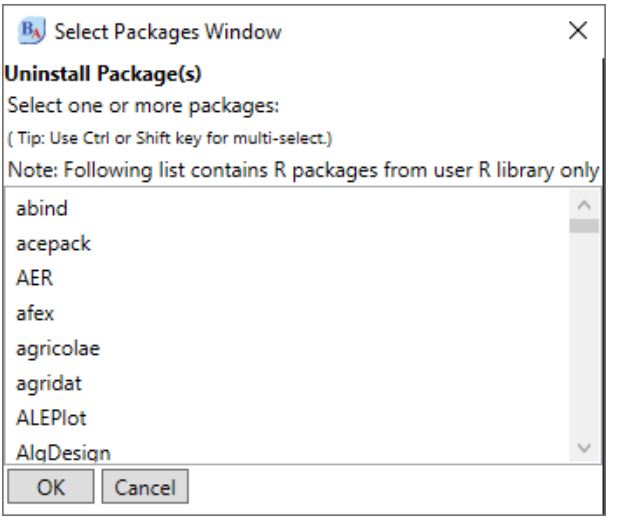


Fig. 11.8: Uninstall package dialog.

Note that there are several approaches to use to install a package, but when uninstalling, that does not matter. Regardless of the method of installation, the uninstall process is the same.

11.4 Configuration Settings

The default settings for many of BlueSky's actions are controlled by choosing "Tools> Configuration Settings." A dialog will appear, offering several tabs across the top. Choosing one of those tabs will display all the settings for it. For example, in Fig. 11.9, I chose the Output tab, one of the most widely used.

The following sections describe the various settings on each tab. Note that the "Reset Defaults" button will restore the default settings of *all tabs* simultaneously, regardless of which tab is selected!

These setting are stored in the folder: "C:\Users\YourID\AppData\Roaming\BlueSky\Config\BlueSky.exe.config". The list of recent files you have used is also stored in that file. You can edit it directly with any text editor, but it is safer to make changes through the "Tools> Configuration Settings menu choice."

11.4.1 Paths

The Paths tab shows you the R Home Directory for BlueSky's own copy of R. You should never change this except under extreme circumstances, such as when instructed by the BlueSky Technical Support team.

This tab also shows your "User R Library Path" which is where any R packages you install are stored, regardless of whether you used BlueSky to

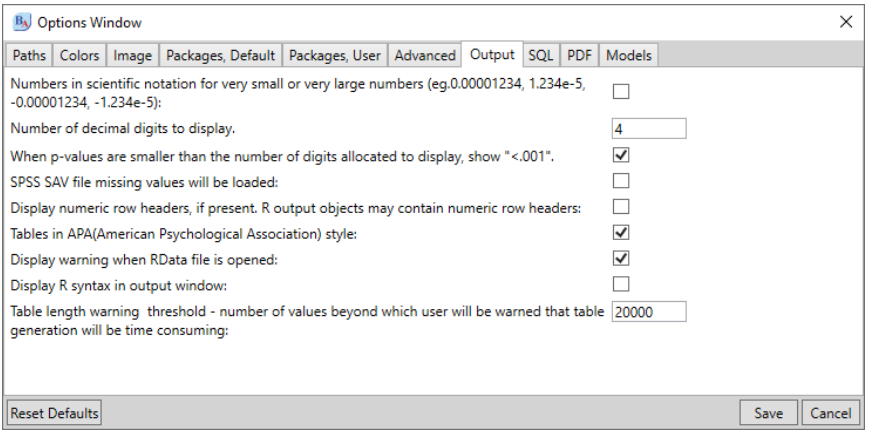


Fig. 11.9: The settings window with the output tab chosen.

install them or another copy of R, perhaps used by RStudio. When trying to load a package, BlueSky will first look in its own area. If it doesn't find the package that it is seeking, then it will look in your library. This allows you to have your own copy of R without there being any conflict between the two. While having two copies of R may seem like a waste of disk drive space, it does not take very much space on today's large drives, and it offers a significant benefit: stability! R packages change frequently, and by providing its own set, BlueSky can guarantee that your output will be stable across time.

11.4.2 Colors

The Colors tab lets you choose the colors of command titles, error messages, the mouse hover box in the Output window, Navigation tree, and R syntax. My personal preference is to see error messages printed in bright red rather than the default dark red.

11.4.3 Image

The Image tab lets you set the height and width of all graphics images that BlueSky generates.

11.4.4 Packages, default

The Packages Default tab shows you the packages that are loaded automatically every time you start BlueSky. Each one it loads slows the start-up process a bit, but it makes using the software a bit quicker after that. Each package also takes up some of your computer's main memory, leaving less room for data and calculations. However, to date, I have never found a need to change the default settings here.

At the time of writing, the default set of packages is: BlueSky, foreign, data.table, readxl, readr, haven, openxlsx, stringr, and gdata.

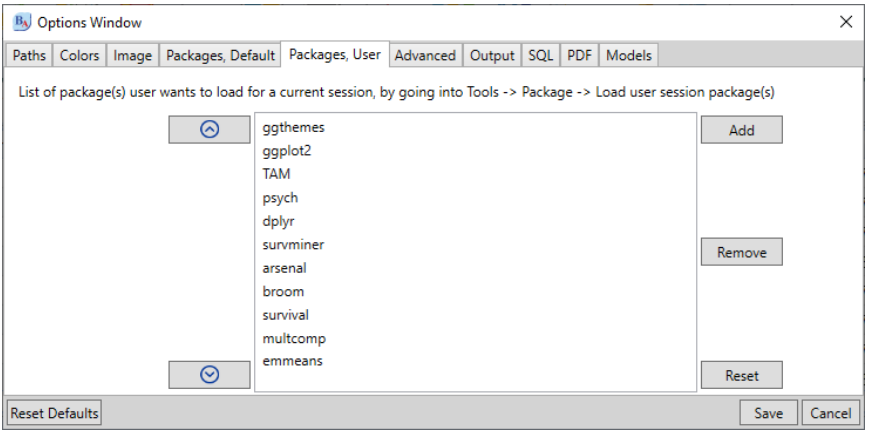


Fig. 11.10: The user package configuration dialog.

11.4.5 Packages, user

The "Packages, User" tab contains the list of packages that you use so often that you would like to load them all at once, but not necessarily at startup. For example, when doing machine learning modeling, you might want to load a dozen of your favorite packages. Loading them all at once would be convenient, but only when you perform that type of task. Otherwise, it would waste memory.

The menu "Tools> Package> Load User Session Package(s)" described in Sec. 11.3.8 lets you load many packages at once, and this is the place you use to maintain that list. Choosing the item "Tools> Configuration Settings> Packages, user" opens the dialog shown in Fig. 11.10. You can select package names and click Add or Remove to cause them to be loaded every time BlueSky starts up, or not. You can also change their loading order by using the arrow keys to move them up or down the list. Those loaded toward the bottom of the list are loaded later. As a result, if two or more packages share a function name, the one loaded later will be dominant. It will overwrite the one higher on the list, replacing its function of the same name.

11.4.6 Advanced

The Advanced tab includes a miscellaneous list of settings that do not classify neatly under the other tabs.

The "Expiration reminder" setting is for commercial licenses only. It lets you set the number of days you wish to be reminded of the next license expiration. By default, BlueSky will warn you 30 days in advance that your license is about to expire. It will then warn 15 days in advance, then 7, then 3. Finally, it will stop working. At that point, you need to renew the license. If your organization is slow at processing invoices, you may wish to set a warning 60 days in advance.

Under "Enter maximum number of factor levels allowed..." you can set a value that will block a numeric-to-factor conversion. By default, if a numeric or integer variable has more than 20 values, BlueSky will warn you and will not convert it to a factor.

The "Log Advanced" and "Log Level" settings are only there in the rare event that the BlueSky technical support staff ask you to turn logging on to help debug a problem.

The two "Convert all character columns" settings determine what happens to character (a.k.a. string) columns as they are read in. By default, CSV or Excel files *do not* convert to factors. However, when entering data manually into the Datagrid, character values *do* convert to factors. The reason for the difference is that during manual entry, it is easy to make corrections. When reading data from external files, there are often many more lines read, and fixing errors may require the use of string manipulation functions. Afterward, you can convert them all to factors as described in Sec. 4.6.

11.4.7 Output

The Output tab controls all the settings for tables that appear in the Output window. The tab itself is shown in Fig. 11.9. Most of these settings are described in the chapter on Output, especially in Sec. 3.2. Here we will discuss the remaining ones.

The checkbox for "SPSS SAV file missing values will be loaded" decides whether or not SPSS' special missing values will be loaded when reading a file of type ".SAV".

The "Display numeric row headers" option determines if columns are numbered 1, 2, 3,... for R objects that may contain such headers. It is turned off by default.

The "Display warning when RData file is opened" is there because opening such a file could possibly replace datasets and other objects that you might have in memory. If all your files were created with BlueSky, this is unlikely to ever happen. When opening an RData file, there is a checkbox to ask it to not bother you with that warning for the duration of your session. There is currently no way to turn that warning off permanently.

The "Table length warning" setting determines the point at which BlueSky will warn you that an extremely large table may take a very long time to compute. Since its default value is 20,000 getting such a warning should definitely get you to think about what you're asking BlueSky to do!

11.4.8 SQL

The SQL tab allows you to specify the default information needed when querying data stored in a relational database.

11.4.9 PDF

The PDF tab lets you configure how you wish your PDF files to appear when you use "File> Save as PDF." It includes such things as font size, page size, margin settings, and the like.

11.4.10 Models

The models tab allows you to control the type of models that are displayed in the "Score Current Dataset" box on the top right-hand side of BlueSky's main Analysis window. By default, all model types are turned on, so you should rarely need to reduce this set.

12

Working with the R Language

12.1 The R Language

BlueSky does all its work using the open source R language. Norman Nie, one of the founders of SPSS, called R [24] "The most powerful statistical computing language on the planet." [1] R was written by Ross Ihaka, Robert Gentleman, the R Core Development Team, and countless volunteers. R is a language and a massive collection of pre-written procedures for graphics, statistics, machine learning, and more. R is also free and open source software, which has enabled the BlueSky development team to offer its software in the same way.

R is a variant of the S language, which was developed by John Chambers, with help from Douglas Bates, Rick Becker, Bill Cleveland, Trevor Hastie, Daryl Pregibon, and Allan Wilks.[2] The Association of Computing Machinery presented John Chambers with a Software System Award and said that the S language, "*. . . will forever alter the way people analyze, visualize, and manipulate data. . .*" and went on to say that it is, "*. . . an elegant, widely accepted, and enduring software system, with conceptual integrity. . . .*"

12.2 The R Installation

Since BlueSky Statistics is free and open source software, anyone can download it at `http://BlueSkyStatistics.com`. The open source version installs without a license code, of course. If you or your organization licenses the commercial version, follow the installation directions provided by BlueSky LLC.

BlueSky includes its own copy of the R language in its installation. While you could install a separate copy of R, it is never necessary to do so. To manage the installation of R packages, see Sec. 11.3.

[1] He said this after moving to Revolution Analytics, a company that was selling a version of R.

[2] Details of S and R history are in Appendix A of *Software for Data Analysis: Programming with R* [4].

Not only can you blend the use of BlueSky's menus and R code as you work, but you can also use the BlueSky Dialog Designer application to add your own dialogs to BlueSky. The documentation on how to write such extensions and the list of currently available ones is available at the company's website, http://BlueSkyStatistics.com. Many of these are already installed when you download BlueSky. However, it is worth checking to see if any of the additional ones would be useful in your work.

12.3 R Code Typographic Conventions

Any R programming code and the names of all R functions and packages are written in `this Courier font`. BlueSky occasionally packs its R code tightly together, making it harder to read when you are learning it. Therefore, when showing BlueSky's R code, I add spacing in some places to improve legibility.

12.4 Getting Started with R Code

When using BlueSky dialogs, clicking "OK," tells it that you have finished specifying how you want to run the task. Then, behind the scenes, it writes some R programming code, which BlueSky refers to using the generic term "syntax." However, there is another way to execute the task. Rather than click "OK," you can click the "Syntax" button. That will cause the "Syntax Editor" panel to pop out of the Analysis window's right side. That allows you to learn some R code and, if you like, modify it before running it.

Let us consider an example:

1. Load the "iris" dataset by choosing "File> Load Dataset from a Package" and enter "datasets" as the package name and "iris" as the dataset name.
2. To create an aggregate (summary) report containing some variable means for each group, choose "Data> Aggregate to Output".
3. Fill in the box, as shown in In Fig.12.1.
4. Instead of clicking "OK", click on "Syntax." The Syntax Editor will pop out as shown in Fig.12.2. It will contain the R code that BlueSky just wrote.
5. Try out the Show/Hide button to see how the Syntax Editor can easily be popped out and then put away again (but don't touch the code!)
6. Note that the syntax is highlighted, meaning it is all selected and ready to run. Click on the "Run" icon at the top of the Syntax Editor. The output will appear as if you had clicked "OK," in the first place. If you accidentally click on the code, the highlighting will vanish, and you'll have to select it manually before running it.
   R is an interpreted computer language, not a compiled one, so no compilation step is needed; the code runs immediately.

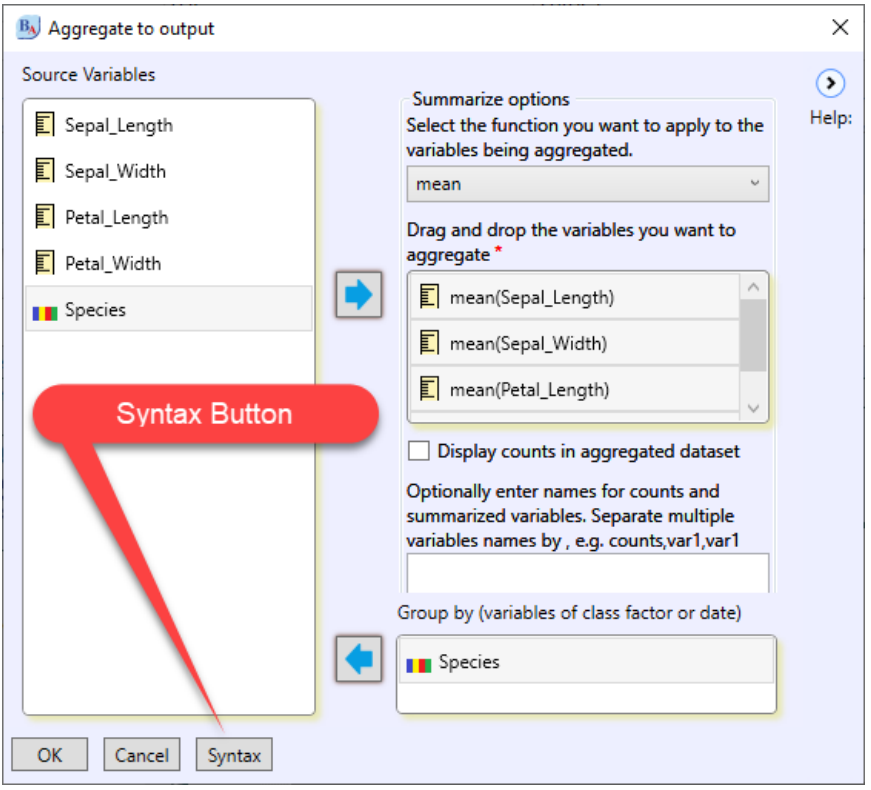


Fig. 12.1: A dialog showing the location of the Syntax button.

Here I repeat the R syntax to make it easier to read. If you're an R user, you probably recognize that this is the type of code written by fans of the "tidyverse" set of packages:

```
## [Aggregate to output]
require(dplyr);

#BSkyFormat is a function that displays the aggregated
   dataset in the output

BSkyFormat(as.data.frame(

iris %>%
  dplyr::group_by(Species) %>%
  dplyr::summarize(
    mean_Sepal_Length = mean(Sepal_Length,na.rm = TRUE),
    mean_Sepal_Width  = mean(Sepal_Width,na.rm  = TRUE),
    mean_Petal_Length = mean(Petal_Length,na.rm = TRUE),
    mean_Petal_Width  = mean(Petal_Width,na.rm  = TRUE))),
```

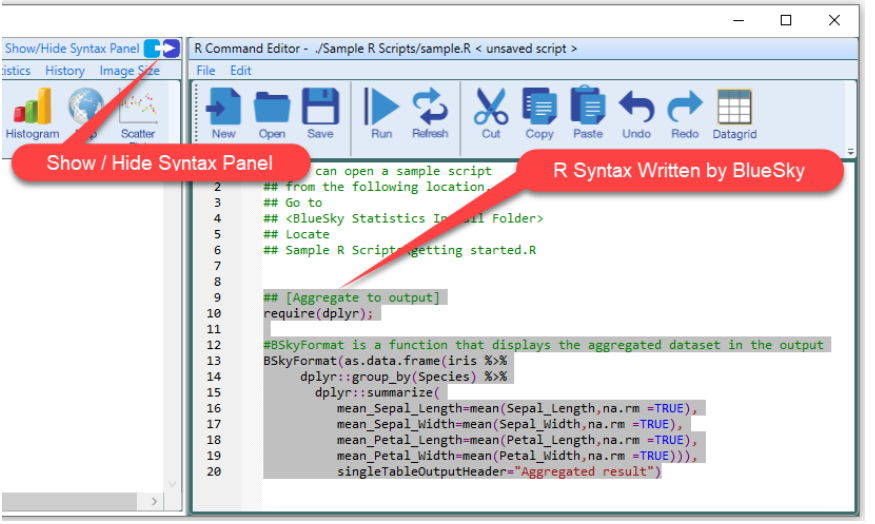


Fig. 12.2: The right-hand side of the Analysis window, showing the Syntax Editor.

```
singleTableOutputHeader="Aggregated result")
```

As long as you don't change the code that BlueSky wrote, the output will be the same. However, its label will be different. In this example, clicking "OK," will result in output labeled precisely as, "Aggregate to Output." But when you run it from the Syntax Editor, BlueSky label it vaguely as, "R-Syntax" since it does not know if you changed the code before running it. Oddly enough, the Navigation pane labels it as "R Session."[3].

You can build long error-free R programs by using this approach. Any time you see some output you want to save the code for, simply use the History menu to bring back the last dialog you used and click "Syntax" instead of "OK" this time. There is no need to execute the code a second time if all you want is to get the R code.

If you look closely at the R code in Fig.12.2, you will see that it is color-coded. That is a big help for avoiding problems. For example, if you use an open quote and do not close it, an R program will generate a lot of error messages. However, the Syntax Editor shows all quoted strings in red, helping you avoid that type of error.

The Syntax Editor also contains a toolbar with the usual commands that are common to any word processor or editor. The most frequently used icon is the Run button. As we have seen, it will run any code that you have selected. In addition, if you place your cursor on any line of R code and click Run, that line will run, and then the cursor will move automatically to the next line. That is a convenient way to step through a program one line at a time when debugging. However, occasionally BlueSky will write a line

[3] I've reported this to the developers, so that may be consistent by the time you read this

of code that must be submitted with one or more that follow. In that case, you must select that set before running. That can be frustrating at times, but the developers are aware of the issue and are working to rectify it.

12.5 Converting R Objects to Data Frames

The only R objects that BlueSky can open are data frames, tibbles (an enhanced type of data frame), and R models[4]. An R matrix is similar to a data frame, though all its values must be the same type, often all numeric. To import a matrix, you must first use R code to convert it to a data frame using code like this:

```
my_data_frame <- data.frame(my_matrix)
save(my_data_frame, file = "my_data_frame.RData")
```

The matrix will become a data frame whose values will appear in the exact same column and row positions.

An R array is a multiple-dimensional matrix. Its dimensions can be selected using values in square brackets: [row, column, layer]. Leaving a value out means you want all of that level. For example, if you have an array with three dimensions, you could extract all rows and all columns in layer 1, and covert that layer to a data frame with:

```
my_data_frame <- data.frame(my_array[ , , 1])
save(my_data_frame, file = "my_data_frame.RData")
```

12.6 Generating True Word Processing Tables

Output tables from R typically appear as simple text, with no formatting. Not even tabs appear between columns, so using a mono-spaced font is needed to line them up. The development of R Markdown[5] has raised the awareness of nice quality output and R package developers are adding the ability to write nicely formatted tables. BlueSky writes its nicely formatted tables using its `BSkyFormat` function. It is easy to use, as you can see in the following simple example. The resulting output is shown in Fig. 12.3.

```
data(iris)

subset <- iris[1:5, ]

BSkyFormat(subset)
```

BSkyFormat has nicely formatted output styles for a wide range of R objects, including the following classes:

[4] Future versions may save and open graph objects.

[5] https://rmarkdown.rstudio.com/authoring_quick_tour.html

Results

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
	5.1	3.5	1.4	0.2	setosa
	4.9	3	1.4	0.2	setosa
	4.7	3.2	1.3	0.2	setosa
	4.6	3.1	1.5	0.2	setosa
	5	3.6	1.4	0.2	setosa

Fig. 12.3: Example output from the BskyFormat function. Note that APA style was turned off at the time.

- Anova
- by
- character vector
- data.frame
- glm
- htest
- list
- lm
- matrix
- numeric vector
- numSummary
- polr
- summary.glm
- summary.lm
- summary.multinorm
- summary.polr
- table
- tapply
- xtabs

12.7 Loading and Refreshing the Datagrid

When you load a dataset into memory using BlueSky's menus, it will automatically appear in the Datagrid. However, when you use R code to load a dataset, BlueSky will not load it into the Datagrid automatically. That is the job of the `BSkyLoadRefreshDataframe` function. It can load a new dataset into the Datagrid, and if you have modified a previously loaded dataset, it can refresh the view that Datagrid shows you.

Let us return to the example from the previous Sec. 12.6. There we created a dataset called "subset" and printed it nicely. However, that dataset never appears in the Datagrid! It resided in memory, and any R code could use it. But without loading it into the Datagrid, you cannot use it with BlueSky's dialogs. This R code recreates the subset data and then loads it into the Datagrid:

```
data(iris)

subset <- iris[1:5, ]

BSkyLoadRefreshDataframe (subset)
```

If the subset data frame had already been in the Datagrid, this would have over-written the entire dataset with the contents of subset! Note that

this is a one-way process. Any changes you make in the Datagrid happen the instant that you press the Enter key or *leave the last cell you edited!* Any R code that you run after those changes will be reflected in your R output.

12.8 Getting Help on R Packages and Functions

To get help on R functions, use the menu item "Help> Help on Function" or enter the following command in the Syntax Editor, select it, then click "Run." For example, here is how to get help on the `BlueSky` package itself.

```
help(package="BlueSky")
```

To get help on a specific function, use the item "Help> Help on Function" or simply enter its quoted name in the help function and run it. The quotes are optional for most function names, but some do require it, e.g., the `if` function. Here is an example of getting help on one of the `BlueSky` package's functions.

```
help("BSkyLoadRefreshDataframe")
```

13

Glossary

Appendix A: Glossary of BlueSky Terminology

Below are definitions of key BlueSky and R terms used in this book.

Active Dataset
: The dataset (a.k.a. data frame, or tibble) that is foremost in the Datagrid. It is the one that will be used for any analyses done using the menus. Any open dataset can become the active one by clicking the tab with its name on it, bringing it to the front.

Apply
: The process of having a function work on variables or observations. "Data> Computing New Variables> Compute" applies functions down a dataset's columns, while "Data> Computing New Variables> Compute, apply a function across all rows" applies it across rows.

Arguments
: The options or parameters that BlueSky uses to control an analysis or plot are called arguments. You set BlueSky arguments using dialog box entries. You can also override them manually by clicking the "Syntax" button instead of "OK," then editing the R code BlueSky sends to the Syntax Editor.

Array
: A matrix with more than two dimensions whose values must all be of only one type (e.g., all numeric or all character). To convert an array to a data frame to read into BlueSky, see Sec. 12.5.

Attribute
: An attribute is the set of values of a single measure for all the analysis units. With people data, an example might be the height of every person in the dataset. Usually stored in columns. Also called a variable, feature, field, or column.

Assignment function
: The two-key sequence "`<-`" that places data or results of procedures or transformations into a variable or data set (like the equals sign in most languages). BlueSky uses this behind the scenes as it writes R code to perform the work.

Attributes
: Traits of an object. Dataset attributes include variable names and value labels. Model attributes include the function call that created it, its predictions, residuals, and so on. You can view dataset attributes and set their values on the Datagrid's Variables tab.

BlueSky Utility
: The BlueSky Utility is a software program that generates a log file to help the technical support team figure out what might be going wrong with an installation. The tool and its documentation are downloadable from the company website.

Case
: A case is the set of measures on single unit of analysis. With people data, that would be all the measurements for one person. Also called an observation, an instance, subject, or row.

Class
: An attribute of a variable that determines what roles it can play in an analysis or graph. In BlueSky, the most important classes are numeric, factor, character (string), and POSIXct (dates & times).

Classification Problem
: The type of problem that predicts or estimates each observation's class. For example, who will and who will not recover from cancer.

CRAN
: The Comprehensive R Archive Network at http://cran.r-project.org/. Consists of a set of sites worldwide called mirrors that distribute R and its add-on packages. BlueSky's installation process automatically installs the ones it needs. If you need additional packages, you can install them manually from CRAN.

Data frame
: A data set. Data frames also contain *attributes* such as each variable's name and class. A special type of data frame is called a "tibble." BlueSky works with standard data frames or tibbles equally well, so it is seldom necessary to differentiate between them

Dataset
: What BlueSky calls a data frame (or a tibble).

Data View
: The Data View is the part of the Datagrid which shows the data itself in a spreadsheet format. It is the default view, but it also appears whenever you click the Data tab at the bottom left of the Datagrid. Data View's alternative is Variable View.

Datagrid
: BlueSky's Datagrid is its dataset editor. It has a spreadsheet-like grid that allows you to enter variables in its columns and observations in its rows.

Dependent Variable
: The measure which we are trying to predict or analyze with a model. Also called the target or outcome variable. This is often assumed to be caused by the predictor variables, but only a carefully designed experiment can prove causation.

Factor
: A categorical variable and its value labels. It is essential to tell BlueSky which of your variables are factors, otherwise, it will not allow you to choose variables in the appropriate fields in various dialog boxes. Value labels may be nothing more than "1," "2,"..., if not assigned explicitly.

Feature
: Another term for an independent variable or predictor in a model.

Feature Engineering
: Another term for variable transformation or preparation for modeling.

Function
: A procedure that makes BlueSky do something. It could be as simple as taking the mean of a variable or as complex as an automated grid search for the optimum machine learning model. BlueSky executes all of its functions though the R language.

Generic function
: A function that changes its output based on the kind of data you give it. For example, a summary analysis might take the mean of a numeric variable but instead, get frequency counts on the same variable if stored as a factor.

`ggplot2`
: The graphics package BlueSky uses to create nearly all of its graphics.

Hyper-parameter
: See parameters.

Independent Variable
: The measures used to predict or model the dependent variable, or outcome. Also called features or predictors.

Installation
: The process of placing BlueSky and all the components it needs on your computer's hard drive. This includes BlueSky's own copy of the R language and all packages that it needs. You can also manually install your own R add-on modules. If you install R manually, it will be an independent copy which will not interfere with BlueSky's copy.

Instance
: An instance is the set of measures on single unit of analysis. With people data, that would be all the measurements for a person. Also called an observation, a case, a record, or a row.

Levels
: The values that a categorical variable can have. They are stored as a part of the variable itself in what appears to be a very short character variable (even when the values themselves are numbers).

Library
: The location where a given version of BlueSky (or R) stores its packages and the add-on modules you have installed.

Load
: Brings a dataset or an R package from disk to memory so that BlueSky can use it.

Matrix
: A data set that must contain only one type of variable, e.g., all numeric or character.

Model Builder
: A collection of buttons that, when clicked, enter combinations of variable names and math symbols to build a model. You can always manually type the model yourself if you prefer. The model builder appears in each dialog that has, "...with formula" as part of its title.

Modeling Fitting
: The BlueSky menu that allows you to control the finer points of models by specifying a formula. The items on this menu drive R modeling functions.

Modeling function
: In BlueSky, modeling functions are usually those that the Model Fitting menu lists. They use functions like linear regression to make predictions or study relationships. They often have options to specify a formula using the *Formula Builder*. In R, modeling functions accept a formula (e.g., `y~x`) and a "`data =`" argument.

Model objects
: An equation or set of equations created to describe relationships or make predictions. BlueSky's modeling functions create them. You can create them using R code, too.

NA
: A missing value. Stands for *N*ot *A*variable. See also NaN.

Names
: Variable names. They are stored in a character variable that is a part of a data set. In BlueSky, names can be displayed and changed using the Variables tab of the Datagrid. More formally in R, names are attributes of many objects that label the elements or components of the object.

NaN
: A missing value. Stands for *N*ot a *N*umber. Something undefined mathematically such as zero divided by zero.

Numeric
: A variable that contains only numbers. Its storage modes include integer, single, and double, i.e. double precision.

Object
: In BlueSky, objects are datasets (data frames) and models (and perhaps graphics object, by the time you read this). R has many other types of objects, but BlueSky's menus do not use them.

Object-oriented programming
: A style of software in which the output of a procedure depends on the type of data you provide. R has an object-orientation.

Observation
: The set of measures on single unit of analysis. With people data, that would be all the measurements for a person. Also called a case, an instance, a record, a subject, or a row.

Option

BlueSky dialogs often include an Options button you use to set the arguments that control R functions. Settings which are viewed by R as options are, in BlueSky, Configuration Settings. They are controlled by the item "Tools> Configuration Settings," the Themes dialog on the main Application window, and the "Image Size" button on the Output window. In R, `option` is a function that sets general parameters, such as the width of each line of output. Calling this function manually in the R Syntax Editor will also impact how BlueSky operates.

Output Window

The window that contains BlueSky's analysis' or graphics' results. On the left, it also includes the Output Navigator. On the right, it includes the R Syntax Editor. Both of these pop-out panes are displayed and hidden by the Show/Hide icon on the Output Window control bar.

Output Navigator

The Output Navigator is a pane that provides an interactive outline view of your output. The Navigator pops out of the Output Window's left side when you click the Show/Hide icon at the top left of each Output Window. Clicking on an entry in the Navigator will take you directly to that piece of output.

Package

An add-on module and related files such as help or practice data sets bundled together. May come with R or be written by its users. More formally, a collection of functions, help files, and, optionally, data objects. The BlueSky installation includes R and many of its packages.

Parameters

This term has a variety of meanings. When used to describe the control over how an analysis runs, the official BlueSky terminology is "options." In the R language that actually does BlueSky's calculations, the official term is "arguments." The optimal settings that a modeling method can choose for itself are also called parameters. For example, in a regression model $y = a+bX$, a and b are model parameters found by the modeling process itself. Hyper-parameters are parameters that the modeling process cannot find by itself; the best settings can be found only by the trial-and-error process called "tuning."

R

A free and open source language and environment for statistical computing and graphics. BlueSky does all of its work by converting settings in dialog boxes into R code that does the actual analysis or graphics work. The BlueSky installation includes R and many of its packages. You can install a separate version of R that will not interfere with BlueSky's copy.

Regression Problem

In general, this is a class of problems in which a continuous numeric variable is estimated or predicted. Many modeling methods can solve

regression problems. However, the linear regression model, solved via the ordinary least-squares (OLS) method, is so well known that many people use just the name "regression" to mean that type of linear regression.

Row Names

A special type of ID variable. The row names are stored in a character variable that is a part of a data set. By default, row names are the character strings "1," "2," . . . , etc. BlueSky does not currently support the display of row names. However, future plans include offering to move row names to the first column in the dataset via a checkbox control.

S

The language from which R evolved. R can run many S programs, but S cannot use R packages.

Score Current Dataset

A panel at the top right of the Analysis window. It allows you to choose a model, then "score" it, or use it to make predictions on the *active dataset*.

Scoring a Model

The process of taking a trained model and using it to make predictions on a dataset. That dataset could be the one that trained it, a hold-out sample, or a completely new dataset.

Script

An R program you can control from BlueSky's Syntax Editor.

Search list

The ordered list of objects that BlueSky will examine when looking for objects or functions.

Show/Hide Icons

The BlueSky output window contains icons that cause panes to pop out the sides of the window. On the left is the Output Navigator pane and on the right is the R Syntax Editor pane.

Supervised Learning

A type of modeling in which a known outcome is used to train a model.

Syntax Editor

BlueSky's Syntax Editor allows you to edit and execute R code. BlueSky dialogs can write that code by clicking the Syntax button. You can also write and execute R code that you write yourself in the Syntax Editor.

Theme

Themes are styles that determine the overall look of graphs. They can set the color of the background, the nature of tick-marks or grid lines, the location of legends, and so on. However, themes do not determine parts of the graph such as which *variables* control colors, sizes, shapes, or line-types.

Tidyverse

The tidyverse is a set of R packages that make R easier to use by providing a consistent style of working. Tidyverse functions always read

tibbles (datasets or data frames) and, after processing, return their results as tibbles too. BlueSky does nearly all of its data wrangling using tidyverse functions.

Tibble

A data frame with some additional classes which give it useful abilities. In BlueSky, there is not much of a difference between analyzing a data frame or a tibble; they both display data the same way in the Datagrid.

Training Dataset

A training dataset is the one used to create a model. Evaluating the effectiveness of a model on its training dataset will always overestimate its effectiveness. However, you can only obtain certain important diagnostic measures from the training dataset. For example, the influence of each case on the model.

Tuning

Tuning is the trial-and-error process of searching a set of modeling parameters to find the best set to use. The set of candidates is called a tuning or search grid.

Unit of Analysis

A unit of analysis is one of the things you are studying. With medical data, each unit of analysis is a person. For educational data, it could be a teacher, or a student. For a geologist, it could be a rock from which measurements are taken.

Unsupervised Learning

A type of modeling in which no outcome is known. Therefore, you can only search for clusters of similar observations. Examples of unsupervised learning methods are cluster analysis and self-organizing maps.

Variable

The set of all the values of one measure for all the units of analysis. With people data, an example might be the height of every person in the dataset. Usually stored in columns. Also called a feature, attribute, field, or column.

Variable View

The Variables tab is the part of the Datagrid that shows data about the variables (names, labels, etc.). It appears whenever you click the Variables tab at the bottom left of the Datagrid. It allows you to change this "metadata" by making changes to things like variable names.

Vector

A variable outside of a dataset. While vectors can exist in BlueSky's Syntax Editor, its Datagrid and menus can operate only on data frames. Therefore, a vector must be stored as a data frame as a single column before you can use it in BlueSky.

Workspace

It is the area of your computer's main memory where BlueSky (and R) do all the work. When you exit BlueSky, it will delete any objects you created, unless you save them to your hard drive first.

References

[1] F. J. Anscombe. Graphs in statistical analysis. *The American Statistician*, 27:17–21, 1973.

[2] American Psychological Association. *Purdue Online Writing Lab APA Style Guide*. Purdue University, 2019.

[3] J. M. Bland and D. G. Altman. Statistical methods for assessing agreement between two methods of clinical measurement. *Lancet*, 327:307–310, 1986.

[4] J. M. Chambers. *Software for Data Analysis: Programming with R*. Springer, Berlin Heidelberg New York, 2008.

[5] K. Elliot. 39 studies about human perception in 30 minutes. *Medium.com*, 2016.

[6] John Fox. *RcmdrMisc: R Commander Miscellaneous Functions*, 2020. R package version 2.7-0.

[7] S. Milborrow; Derived from mda:mars by Trevor Hastie and Rob Tibshirani; Uses Alan Miller's Fortran utilities with Thomas Lumley's leaps wrapper. *earth: Multivariate Adaptive Regression Splines*, 2019. R package version 5.1.2.

[8] K. B. Gorman, T. D. Williams, and W. R. Fraser. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PLoS ONE*, 9(3)(e90081):–13, 2014.

[9] R. Hoyt and R. A. Muenchen. *Introduction to Biomedical Data Science*. Lulu.com, 2019.

[10] G. G. Koch J. R. Landis. The measurement of observer agreement for categorical data. *Biometrics*, 22(1):159–174, 1977.

[11] J. Kaplan. *fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variables*, 2020. R package version 1.6.1.

[12] M. Kuhn. *caret: Classification and Regression Training*, 2020. R package version 6.0-86.

[13] Palmer Station Antarctica LTER and K. Gorman. Structural size measurements and isotopic signatures of foraging among adult male and female adélie penguins (pygoscelis adeliae) nesting along the palmer

archipelago near palmer station, 2007-2009. *Environmental Data Initiative*, 2020.

[14] Palmer Station Antarctica LTER and K. Gorman. Structural size measurements and isotopic signatures of foraging among adult male and female gentoo penguin (pygoscelis papua) nesting along the palmer archipelago near palmer station, 2007-2009 ver 5. environmental data initiative. *Environmental Data Initiative*, 2020.

[15] Palmer Station Antarctica LTER and K. Gorman. Structural size measurements and isotopic signatures of foraging among adult male and female chinstrap penguin (pygoscelis antarcticus) nesting along the palmer archipelago near palmer station, 2007-2009 ver 6. *Environmental Data Initiative*, 2020.

[16] R. A. Muenchen. *R for SAS and SPSS Users*. Springer, Berlin Heidelberg New York, 2nd edition, 2011.

[17] R. A. Muenchen. *BlueSky Statistics 7.1 User Guide*. Lulu.com, 2020.

[18] R. A. Muenchen and J. M. Hilbe. *R for Stata Users*. Springer, Berlin Heidelberg New York, 2010.

[19] J. C. Nunnally and I. H. Bernstein. *Psychometric Theory*. McGraw-Hill, New York, NY, 3rd edition, 1994.

[20] J. M. Chambers R. A. Becker. *S: An Interactive Environment for Data Analysis and Graphics*. Wadsworth & Brooks/Cole., Pacific Grove, CA, USA, 1984.

[21] R Development Core Team. *R: A language and environment for statistical computing*. Vienna, 2003.

[22] W. Revelle. *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois, 2019. R package version 1.9.12.

[23] D. Robinson, Alex Hayes, and Simon Couch. *broom: Convert Statistical Objects into Tidy Tibbles*, 2020. R package version 0.7.0.

[24] R Development Core Team. *R: A Language and Environment for Statistical Computing*. http://www.R-project.org, Vienna, 2008.

[25] M. van der Loo. *simputation: Simple Imputation*, 2020. R package version 0.2.4.

[26] H. Wickham, R. François, L. Henry, and K. Müller. *dplyr: A Grammar of Data Manipulation*, 2020. R package version 1.0.0.

[27] Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

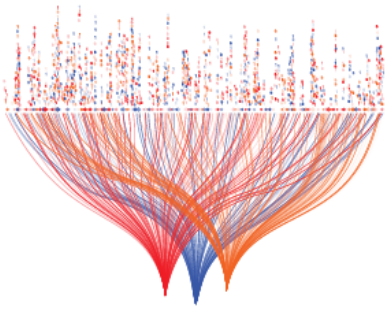
[28] Hadley Wickham. *forcats: Tools for Working with Categorical Variables (Factors)*, 2020. R package version 0.5.0.

[29] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019.

Index

BLUESKY STATISTICS

7.1 INTRO GUIDE



BlueSky Statistics is an easy-to-use and powerful menu-based system for data science. Its menus and dialog boxes make quick work of graphs and analyses without having to learn to program. Behind the scenes, it writes code using the powerful R language. It can show you the code it writes, allowing you to learn R and modify what it is doing. R programmers can easily add menus and dialog boxes to BlueSky.

This guide is a subset of the ***BlueSky Statistics 7.1 User Guide***. This one keeps the cost low by skipping graphics examples and advanced modeling, such as the Model Fitting and Model Tuning menus. The straightforward writing style used in both guides assumes a minimal background in statistics or machine learning. It includes:

- ✓ **An introduction that gets you going quickly.**
- ✓ **Instructions for setting graphics themes and table styles.**
- ✓ **Simple step-by-step instructions for each item on the Analysis menu.**
- ✓ **How to interpret each output table on the Analysis menu.**
- ✓ **Details of the Analysis Window, Datagrid, Output Window, and Model Scorer.**
- ✓ **Extensive cross-references helping you relate one topic to another.**
- ✓ **A detailed index allowing you to search by topic or by R function or package name.**
- ✓ **A glossary of BlueSky terminology.**

Robert A. Muenchen is a member of the BlueSky Statistics development team, and he has contributed many features to the software. He wrote ***R for SAS and SPSS Users*** and co-authored ***R for Stata Users*** and ***Introduction to Biomedical Data Science***. An ASA Accredited Professional Statistician, Bob wrote or co-authored over 70 articles published in scientific journals and conference proceedings. At The University of Tennessee, he guided more than 1,000 graduate theses and dissertations.

9 781716 443527