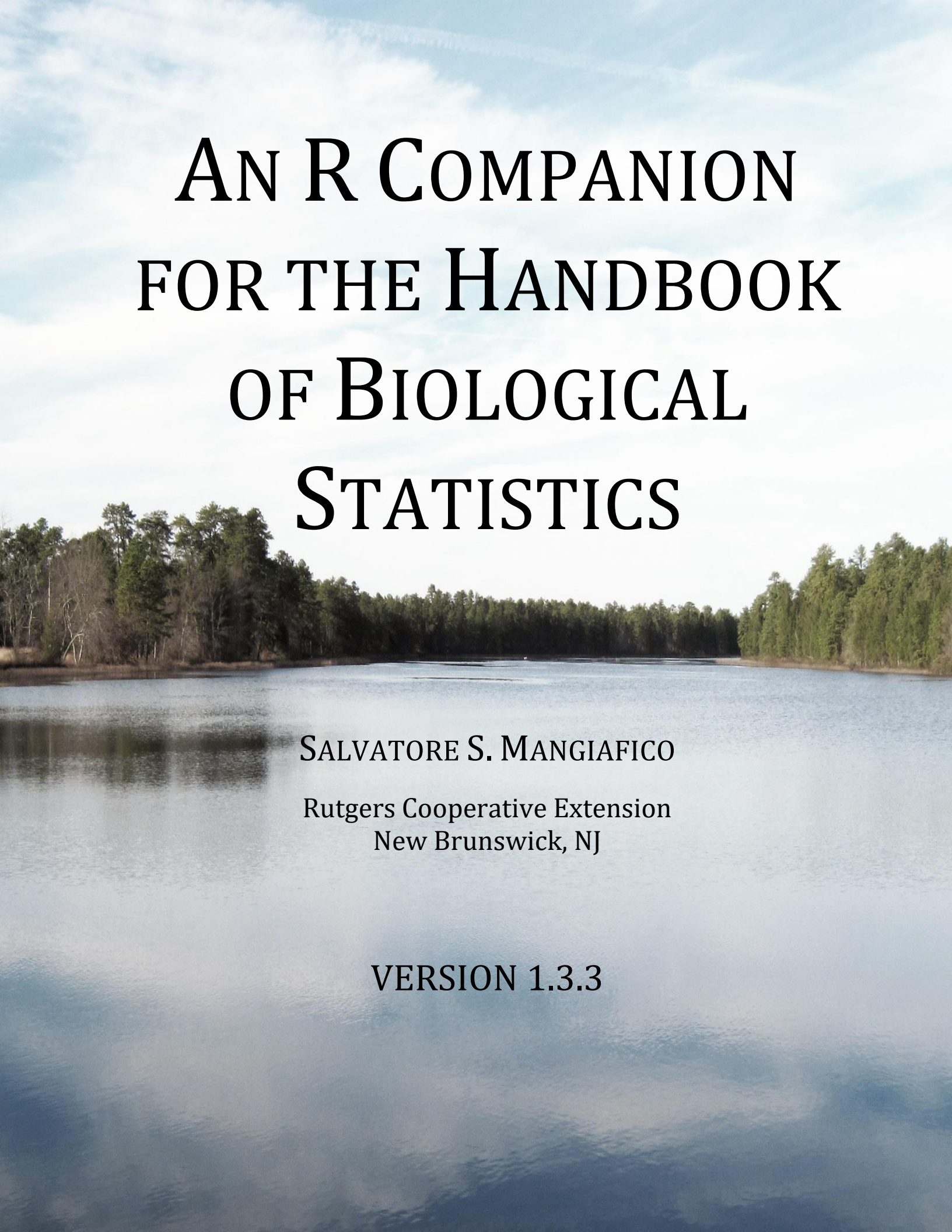




AN R COMPANION FOR THE HANDBOOK OF BIOLOGICAL STATISTICS

SALVATORE S. MANGIAFICO

Rutgers Cooperative Extension
New Brunswick, NJ

VERSION 1.3.3

©2015 by Salvatore S. Mangiafico, except for organization of statistical tests and selection of examples for these tests ©2014 by John H. McDonald. Used with permission.

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

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

Mangiafico, S.S. 2015. An R Companion for the Handbook of Biological Statistics, version 1.3.3.
rcompanion.org/documents/RCompanionBioStatistics.pdf . (Web version:
rcompanion.org/rcompanion/).

Table of Chapter

<i>Introduction</i>	<i>1</i>
Purpose of This Book.....	1
The Handbook for Biological Statistics	1
About the Author of this Companion	1
About R	2
Obtaining R.....	2
A Few Notes to Get Started with R	3
Avoiding Pitfalls in R	10
Help with R.....	11
R Tutorials	12
Formal Statistics Books	13
<i>Tests for Nominal Variables</i>	<i>14</i>
Exact Test of Goodness-of-Fit	14
Power Analysis	23
Chi-square Test of Goodness-of-Fit	24
G–test of Goodness-of-Fit	32
Chi-square Test of Independence.....	35
G–test of Independence	47
Fisher’s Exact Test of Independence	53
Small Numbers in Chi-square and G–tests.....	61
Repeated G–tests of Goodness-of-Fit.....	61
Cochran–Mantel–Haenszel Test for Repeated Tests of Independence.....	66
<i>Descriptive Statistics.....</i>	<i>78</i>
Statistics of Central Tendency.....	78
Statistics of Dispersion	84
Standard Error of the Mean.....	87
Confidence Limits.....	88
<i>Tests for One Measurement Variable.....</i>	<i>94</i>
Student’s <i>t</i> –test for One Sample.....	94
Student’s <i>t</i> –test for Two Samples	97
Mann–Whitney and Two-sample Permutation Test	101

Chapters Not Covered in This Book.....	103
Type I, II, and III Sums of Squares	104
One-way Anova	106
Kruskal–Wallis Test	118
One-way Analysis with Permutation Test.....	129
Nested Anova	133
Two-way Anova	143
Two-way Anova with Robust Estimation.....	161
Paired t–test	169
Wilcoxon Signed-rank Test	178
<i>Regressions</i>	<i>182</i>
Correlation and Linear Regression	182
Spearman Rank Correlation.....	190
Curvilinear Regression.....	193
Analysis of Covariance	206
Multiple Regression	216
Simple Logistic Regression.....	228
Multiple Logistic Regression	242
<i>Multiple tests</i>	<i>256</i>
Multiple Comparisons	256
<i>Miscellany.....</i>	<i>263</i>
Chapters Not Covered in this Book	263
<i>Other Analyses</i>	<i>264</i>
Contrasts in Linear Models	264
Cate–Nelson Analysis	275
<i>Additional Helpful Tips</i>	<i>282</i>
Reading SAS Datalines in R	282

Table of Contents

Introduction	1
Purpose of This Book	1
The Handbook for Biological Statistics	1
About the Author of this Companion	1
About R	2
Obtaining R	2
Standard installation	2
R Studio	3
Portable application	3
R Online: R Fiddle	3
A Few Notes to Get Started with R	3
Packages used in this chapter	3
A cookbook approach	3
Color coding in this book	3
Copying and pasting code	3
From the website	4
From the pdf	4
A sample program	4
Assignment operators	4
Comments	5
Installing and loading packages	5
Data types	5
Creating data frames from a text string of data	5
Reading data from a file	6
Variables within data frames	7
Using <i>dplyr</i> to create new variables in data frames	8
Extracting elements from the output of a function	8
Exporting graphics	9
Avoiding Pitfalls in R	10
Grammar, spelling, and capitalization count	10
Data types in functions	10
Style	11
Help with R	11
Help in R	11
CRAN documentation	12
Summary and Analysis of Extension Education Program Evaluation in R	12
Other online resources	12
R Tutorials	12
Formal Statistics Books	13
Tests for Nominal Variables	14
Exact Test of Goodness-of-Fit	14
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	14

Packages used in this chapter	14
How the test works	14
Binomial test examples	14
Sign test	16
Post-hoc example with manual pairwise tests	17
Post-hoc test alternate method with custom function	18
Examples	19
Binomial test examples	19
Multinomial test example	20
How to do the test	21
Binomial test example where individual responses are counted	21
Power analysis	22
Power analysis for binomial test	22
Power Analysis	23
Packages used in this chapter	23
Examples	23
Power analysis for binomial test	23
Power analysis for unpaired t-test	23
Chi-square Test of Goodness-of-Fit	24
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	24
Packages used in this chapter	24
How the test works	24
Chi-square goodness-of-fit example	24
Examples: extrinsic hypothesis	25
Example: intrinsic hypothesis	26
Graphing the results	26
Simple bar plot with barplot	26
Bar plot with confidence intervals with ggplot2	28
How to do the test	31
Chi-square goodness-of-fit example	31
Power analysis	31
Power analysis for chi-square goodness-of-fit	31
G-test of Goodness-of-Fit	32
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	32
Packages used in this chapter	32
Examples: extrinsic hypothesis	32
G-test goodness-of-fit test with DescTools and RVAideMemoire	32
G-test goodness-of-fit test by manual calculation	33
Examples of G-test goodness-of-fit test with DescTools and RVAideMemoire	33
Example: intrinsic hypothesis	34
Chi-square Test of Independence	35
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	35
Packages used in this chapter	35
When to use it	36
Example of chi-square test with matrix created with read.table	36
Example of chi-square test with matrix created by combining vectors	36
Post-hoc tests	37
Post-hoc pairwise chi-square tests with rcompanion	38
Post-hoc pairwise chi-square tests with pairwise.table	38
Examples	39

Chi-square test of independence with continuity correction and without correction	39
Chi-square test of independence	40
Graphing the results	40
Simple bar plot with error bars showing confidence intervals	41
Bar plot with categories and no error bars	42
How to do the test	45
Chi-square test of independence with data as a data frame	45
Power analysis	46
Power analysis for chi-square test of independence	46
G-test of Independence	47
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	47
Packages used in this chapter	47
When to use it	48
G-test example with functions in DescTools and RVAideMemoire	48
Post-hoc tests	48
Post-hoc pairwise G-tests with RVAideMemoire	49
Post-hoc pairwise G-tests with pairwise.table	49
Examples	50
G-tests with DescTools and RVAideMemoire	50
How to do the test	52
G-test of independence with data as a data frame	52
Fisher's Exact Test of Independence	53
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	53
Packages used in this chapter	53
Post-hoc tests	54
Post-hoc pairwise Fisher's exact tests with RVAideMemoire	54
Examples	55
Examples of Fisher's exact test with data in a matrix	55
Similar tests – McNemar's test	58
McNemar's test with data in a matrix	58
McNemar's test with data in a data frame	58
How to do the test	59
Fisher's exact test with data as a data frame	59
Power analysis	60
Small Numbers in Chi-square and G-tests	61
Yates' and William's corrections in R	61
Repeated G-tests of Goodness-of-Fit	61
Packages used in this chapter	61
How to do the test	62
Repeated G-tests of goodness-of-fit example	62
Example	64
Repeated G-tests of goodness-of-fit example	64
Cochran–Mantel–Haenszel Test for Repeated Tests of Independence	66
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	67
Packages used in this chapter	67
Examples	67
Cochran–Mantel–Haenszel Test with data read by read.ftable	67
Cochran–Mantel–Haenszel Test with data entered as a data frame	69
Cochran–Mantel–Haenszel Test with data read by read.ftable	71
Graphing the results	73

Simple bar plot with categories and no error bars	73
Bar plot with categories and error bars	74
Descriptive Statistics	78
Statistics of Central Tendency	78
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	78
Packages used in this chapter	78
Example	78
Arithmetic mean	79
Geometric mean	79
Harmonic mean	79
Median	79
Mode	79
Summary and describe functions for means, medians, and other statistics	80
Histogram	80
DescTools to produce summary statistics and plots	81
DescTools with grouped data	83
Statistics of Dispersion	84
Example	85
Statistics of dispersion example	85
Range	85
Sample variance	85
Standard deviation	86
Coefficient of variation, as percent	86
Custom function of desired measures of central tendency and dispersion	86
Standard Error of the Mean	87
Example	87
Standard error example	87
Confidence Limits	88
How to calculate confidence limits	89
Confidence intervals for mean with t.test, Rmisc, and DescTools	89
Confidence intervals for means for grouped data	90
Confidence intervals for mean by bootstrap	90
Confidence interval for proportions	92
Confidence interval for proportions using DescTools	93
Tests for One Measurement Variable	94
Student's t-test for One Sample	94
Example	94
One sample t-test with observations as vector	94
How to do the test	95
One sample t-test with observations in data frame	95
Histogram	95
Power analysis	96
Power analysis for one-sample t-test	96
Student's t-test for Two Samples	97
Example	97
Two-sample t-test, independent (unpaired) observations	97
Plot of histograms	98
Box plots	99

Similar tests	100
Welch's t-test	100
Power analysis	100
Power analysis for t-test	100
Mann–Whitney and Two-sample Permutation Test	101
Mann–Whitney U-test	101
Box plots	102
Permutation test for independent samples	102
Chapters Not Covered in This Book	103
Homoscedasticity and heteroscedasticity	104
Type I, II, and III Sums of Squares	104
One-way Anova	106
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	106
Packages used in this chapter	106
How to do the test	107
One-way anova example	107
Checking assumptions of the model	109
Tukey and Least Significant Difference mean separation tests (pairwise comparisons)	110
Graphing the results	113
Welch's anova	116
Power analysis	117
Power analysis for one-way anova	117
Kruskal–Wallis Test	118
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	118
Packages used in this chapter	118
Kruskal–Wallis test example	118
Example	121
Kruskal–Wallis test example	122
Dunn test for multiple comparisons	124
Nemenyi test for multiple comparisons	125
Pairwise Mann–Whitney U-tests	126
Kruskal–Wallis test example	127
How to do the test	128
Kruskal–Wallis test example	128
References	128
One-way Analysis with Permutation Test	129
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	129
Packages used in this chapter	129
Permutation test for one-way analysis	129
Pairwise permutation tests	131
Nested Anova	133
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	133
Packages used in this chapter	133
How to do the test	133
Nested anova example with mixed effects model (nlme)	133
Mixed effects model with lmer	138
Nested anova example with the aov function	140
Two-way Anova	143

Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	143
Packages used in this chapter	144
How to do the test	144
Two-way anova example	144
Post-hoc comparison of least-square means	150
Graphing the results	151
Rattlesnake example – two-way anova without replication, repeated measures	154
Using two-way fixed effects model	154
Using mixed effects model with nlme	158
Using mixed effects model with lmer	158
Two-way Anova with Robust Estimation	161
Packages used in this chapter	161
Example	162
Produce Huber M-estimators and confidence intervals by group	162
Interaction plot using summary statistics	163
Two-way analysis of variance for M-estimators	163
Produce post-hoc tests for main effects with mcp2a	164
Produce post-hoc tests for main effects with pairwiseRobustTest or pairwiseRobustMatrix	164
Produce post-hoc tests for interaction effect	166
Paired t-test	169
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	169
Packages used in this chapter	169
How to do the test	169
Paired t-test, data in wide format, flicker feather example	169
Paired t-test, data in wide format, horseshoe crab example	173
Paired t-test, data in long format	175
Permutation test for dependent samples	177
Power analysis	178
Power analysis for paired t-test	178
Wilcoxon Signed-rank Test	178
Examples in <i>Summary and Analysis of Extension Program Evaluation</i>	178
Packages used in this chapter	178
How to do the test	179
Wilcoxon signed-rank test example	179
Sign test example	180
Regressions	182
Correlation and Linear Regression	182
How to do the test	182
Correlation and linear regression example	182
Correlation	183
Pearson correlation	183
Kendall correlation	184
Spearman correlation	184
Linear regression	184
Robust regression	187
Linear regression example	188
Power analysis	189
Power analysis for correlation	189
Spearman Rank Correlation	190

Example	190
Example of Spearman rank correlation	190
How to do the test	191
Example of Spearman rank correlation	191
Curvilinear Regression	193
How to do the test	193
Polynomial regression	193
B-spline regression with polynomial splines	199
Nonlinear regression	201
Analysis of Covariance	206
How to do the test	206
Analysis of covariance example with two categories and type II sum of squares	206
Analysis of covariance example with three categories and type II sum of squares	211
Multiple Regression	216
How to do multiple regression	217
Multiple correlation	217
Multiple regression	221
Simple Logistic Regression	228
How to do the test	228
Logistic regression example	230
Logistic regression example	233
Logistic regression example with significant model and abbreviated code	238
Multiple Logistic Regression	242
How to do multiple logistic regression	242
Multiple correlation	243
Multiple logistic regression example	246
Multiple tests	256
Multiple Comparisons	256
How to do the tests	256
Multiple comparisons example with 25 p-values	257
Multiple comparisons example with five p-values	260
Miscellany	263
Chapters Not Covered in this Book	263
Other Analyses	264
Contrasts in Linear Models	264
Contrasts within linear models	Error! Bookmark not defined.
Example for single degree-of-freedom contrasts	264
Example with lsmeans	265
Example with multcomp	266
Example for global F-test within a group of treatments	268
Tests of contrasts with lsmeans	269
Tests of contrasts with multcomp	271
Tests of contrasts within aov	273
Cate–Nelson Analysis	275
Custom function to develop Cate–Nelson models	275

Example of Cate–Nelson analysis	276
Example of Cate–Nelson analysis with negative trend data	279
References	280
<i>Additional Helpful Tips</i>	282
Reading SAS Datalines in R	282

Purpose of This Book

This book is intended to be a supplement for *The Handbook of Biological Statistics* by John H. McDonald. It provides code for the R statistical language for some of the examples given in the *Handbook*. It does not describe the uses of, explanations for, or cautions pertaining to the analyses. For that information, you should consult the *Handbook* before using the analyses presented here.

The Handbook for Biological Statistics

This *Companion* follows the .pdf version of the third edition of the *Handbook of Biological Statistics*.

The *Handbook* provides clear explanations and examples of some the most common statistical tests used in the analysis of experiments. While the examples are taken from biology, the analyses are applicable to a variety of fields.

The *Handbook* provides examples primarily with the SAS statistical package, and with online calculators or spreadsheets for some analyses. Since SAS is a commercial package that students or researchers may not have access to, this *Companion* aims to extend the applicability of the *Handbook* by providing the examples in R, which is a free statistical package.

The .pdf version of the third edition is available at www.biostathandbook.com/HandbookBioStatThird.pdf.

Also, the *Handbook* can be accessed without cost at www.biostathandbook.com/. However, the reader should be aware that the online version may be updated since the third edition of the book.

Or, a printed copy can be purchased from <http://www.lulu.com/shop/john-mcdonald/handbook-of-biological-statistics/paperback/product-22063985.html>.

About the Author of this Companion

I have tried in this book to give the reader examples that are both as simple as possible, and that show some of the options available for the analysis. My goal for most examples is to make things comprehensible for the user without extensive R experience. The reader should realize that these goals may be partially frustrated either by the peculiarities in the R language or by the complexity required for the example.

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

About R

R is a free, open source, and cross-platform programming language that is well suited for statistical analyses. This means you can download R to your Windows, Mac OS, or Linux computer for free. It also means that, in theory, you can look at the code behind any of the analyses it performs to better understand the process, or to modify the code for your own purposes.

R is being used more and more in educational, academic, and commercial settings. A few advantages of working with R as a student, teacher, or researcher include:

- R functions return limited output. This helps prevent students from sorting through a lot of output they may not understand, and in essence requires the user to know what output they're asking R to produce.
- Since all functions are open source, the user has access to see how pre-defined functions are written.
- There are powerful packages written for specific type of analyses.
- There are lots of free resources available online.
- It can also be used online without installing software.

For a brief summary of some the advantages of R from the perspective of a graduate student, see thetarzan.wordpress.com/2011/07/15/why-use-r-a-grad-students-2-cents/.

It is also worth mentioning a few drawbacks with using R. New users are likely to find the code difficult to understand. Also, I think that while there are a plethora of examples for various analyses available online, it may be difficult as a beginner to adapt these examples to her own data. One goal of this book is to help alleviate these difficulties for beginners. I have some further thoughts below on avoiding pitfalls in R.

Obtaining R

This topic in SAEPER

For a more complete discussion of installing and running R, see [SAEPER: Using R](#).

Standard installation

To download and install R, visit cran.r-project.org/. There you will find links for installation on Linux, Mac OS, and Windows operating systems.

R Studio

I also recommend using R Studio. This software is an environment for R that makes it easier to see code, output, datasets, plots, and help files together on one screen.

www.rstudio.com/products/rstudio/. It is also possible to install R Studio as a portable application.

Portable application

R can be installed as a portable application. This is useful in cases where you don't want to install R on a computer, but wish to run it from a portable drive. My portable installation of R with a handful of added packages is about 250 MB. The version on R Studio I have is about 400 MB. So, 1 GB of space on a usb drive is probably sufficient for the software along with additional installed packages and projects.

R Online

It is also possible to access R online, without needing to install software. One example of this is [rdrr.io Snippets](http://rdrr.io). This site includes common add-on packages.

A Few Notes to Get Started with R

Packages used in this chapter

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

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

A cookbook approach

The examples in this book follow a “cookbook” approach as much as possible. The reader should be able to modify the examples with her own data, and change the options and variable names as needed. This is more obvious with some examples than others, depending on the complexity of the code.

Color coding in this book

The text in blue in this book is R code that can be copied, pasted, and run in R. The text in red is the expected result, and should not be run. In most cases I have truncated the results and included only the most relevant parts. Comments are in green. It is fine to run comments, but they have no effect on the results.

Copying and pasting code

From the website

Copying the R code pieces from the [website](#) version of this book should work flawlessly. Code can be copied from the webpages and pasted into the R console, the R Studio console, the R Studio editor, or a plain text file. All line breaks and formatting spaces should be preserved.

The only issue you may encounter is that if you paste code into the R Studio editor, leading spaces may be added to some lines. This is not usually a problem, but a way to avoid this is to paste the code into a plain text editor, save that file as a .R file, and open it from R Studio.

From the pdf

Copying the R code from the pdf version of this book may work less perfectly. Formatting spaces and even line breaks may be lost. Different pdf readers may behave differently.

It may help to paste the copied code in to a plain text editor to clean it up before pasting into R or saving it as a .R file. Also, if your pdf reader has a select tool that allows you to select text in a rectangle, that works better in some readers.

A sample program

The following is an example of code for R that creates a vector called *x* and a vector called *y*, performs a correlation test between *x* and *y*, and then plots *y* vs. *x*.

This code can be copied and pasted into the console area of R or R Studio, or into the editor area of R Studio, and run. You should get the output from the correlation test and the graphical output of the plot.

```
x = c(1,2,3,4,5,6,7,8,9) # create a vector of values and call it x
y = c(9,7,8,6,7,5,4,3,1)

cor.test(x,y)           # perform correlation test

plot(x,y)               # plot y vs. x
```

You can run fairly large chunks of code with R, though it is probably better to run smaller pieces, examining the output before proceeding to the next piece.

This kind of code can be saved as a file in the editor section of R Studio, or can be stored separately as a plain text file. By convention files for R code are saved as .R files. These files can be opened and edited with either a plain text editor or with the R Studio editor.

Assignment operators

In my examples I will use an equal sign, =, to assign a value to a variable.

```
height = 127.5
```

In examples you find elsewhere, you will more likely see a left arrow, <-, used as the assignment operator.

```
height <- 127.5
```


These are essentially equivalent, but I think the equal sign is more readable for a beginner.

Comments

Comments are indicated with a number sign, #. Comments are for human readers, and are not processed by R.

Installing and loading packages

Some of the packages used in this book do not come with R automatically, but need to be installed as add-on packages. For example, if you wanted to use a function in the *psych* package to calculate the geometric mean of *x* in the sample program above:

```
x = c(1,2,3,4,5,6,7,8,9)
```

First you would need to install the package *psych*:

```
install.packages("psych")
```

Then load the package:

```
library(psych)
```

You may then use the functions included in the package:

```
geometric.mean(x)
```

```
[1] 4.147166
```

In future sessions, you will need only to load the package; it should still be in the library from the initial installation.

If you see an error like the following, you may have misspelled the name of the package, or the package has not been installed.

```
library(psych)
```

```
Error in library(psych) : there is no package called 'psych'
```

Data types

There are several data types in R. Most commonly, the functions we are using will ask for input data to be a vector, a matrix, or a data frame. Data types won't be discussed extensively here, but the examples in this book will read the data as the appropriate data type for the selected analysis.

Creating data frames from a text string of data

For certain analyses you will want to select a variable from within a data frame. In most examples using data frames, I'll create the data frame from a text string that allows us to arrange the data in columns and rows, as we normally visualize data.

Here, *Input* is just a text string that will be converted to a data frame with the *read.table* function. Note that the text for the table is enclosed in simple double quotes and parentheses.

read.table is pretty tolerant of extra spaces or blank lines. But if we convert a data frame to a matrix—which we will later—with *as.matrix*—I’ve had errors from trailing spaces at the ends of lines.

Values in the table that will have spaces or special characters can be enclosed in simple single quotes (e.g. 'Spongebob & Patrick').

```
Input =("
Sex      Height
male     175
male     176
female   162
female   165
")
```

```
D1 = read.table(textConnection(Input),header=TRUE)
```

```
D1
```

```
      Sex Height
1  male    175
2  male    176
3 female    162
4 female    165
```

Reading data from a file

R can also read data from a separate file. For longer data sets or complex analyses, it is helpful to keep data files and r code files separate. For example,

```
D2 = read.table("male-female.dat", header=TRUE)
```

would read in data from a file called *male-female.dat* found in the working directory. In this case the file could be a space-delimited text file:

```
Sex      Height
male     175
male     176
female   162
female   165
```

Or

```
D2 = read.table("male-female.csv", header=TRUE, sep=",")
```

for a comma-separated file.

```
Sex,Height
```

```
male,175
male,176
female,162
female,165
```

```
D2
```

```
      Sex Height
1  male    175
2  male    176
3 female    162
4 female    165
```

R Studio also has an easy interface in the *Tools* menu to import data from a file.

The *getwd* function will show the location of the working directory, and *setwd* can be used to set the working directory.

```
getwd()
```

```
[1] "C:/Users/Salvatore/Documents"
```

```
setwd("C:/Users/Salvatore/Desktop")
```

Alternatively, file paths or URLs can be designated directly in the *read.table* function.

Variables within data frames

For the data frame *D1* created above, to look at just the variable *Sex* in this data frame:

```
D1$ Sex           # Note: the space is optional
```

```
[1] male  male  female female
Levels: female male
```

Note that *D1\$Height* is a vector of numbers.

```
D1$ Height
```

```
[1] 175 176 162 165
```

So if you wanted the mean for this variable:

```
mean(D1$ Height)
```

```
[1] 169.5
```

Using *dplyr* to create new variables in data frames

The standard method to define new variables in data frames is to use the *data.frame\$variable* syntax. So if we wanted to add a variable to the D1 data frame above which would double *Height*:

```
D1$ Double = D1$ Height * 2      # Spaces are optional
D1
```

	Sex	Height	Double
1	male	175	350
2	male	176	352
3	female	162	324
4	female	165	330

Another method is to use the *mutate* function in the *dplyr* package:

```
library(dplyr)

D1 =
mutate(D1,
       Triple = Height*3,
       Quadruple = Height*4)
```

```
D1
```

	Sex	Height	Double	Triple	Quadruple
1	male	175	350	525	700
2	male	176	352	528	704
3	female	162	324	486	648
4	female	165	330	495	660

The *dplyr* package also has functions to select only certain columns in a data frame (*select* function) or to filter a data frame by the value of some variable (*filter* function). It can be helpful for manipulating data frames.

In the examples in this book, I will use either the *\$* syntax or the *mutate* function in *dplyr*, depending on which I think makes the example more comprehensible.

Extracting elements from the output of a function

Sometimes it is useful to extract certain elements from the output of an analysis. For example, we can assign the output from a binomial test to a variable we'll call *Test*.

```
Test = binom.test(7, 12, 3/4,
                  alternative="less",
                  conf.level=0.95)
```

To see the value of *Test*:

```
Test
```

```
Exact binomial test
```

```
number of successes = 7, number of trials = 12, p-value = 0.1576
```

```
95 percent confidence interval:
0.0000000 0.8189752
```

To see what elements are included in *Test*:

```
names(Test)

[1] "statistic" "parameter" "p.value"    "conf.int"   "estimate"
"null.value" "alternative"
[8] "method"    "data.name"
```

Or with more details:

```
str(Test)
```

To view the p-value from *Test*:

```
Test$ p.value

[1] 0.1576437
```

To view the confidence interval from *Test*:

```
Test$ conf.int

[1] 0.0000000 0.8189752

[1] 0.95
```

To view the upper confidence limit from *Test*:

```
Test$ conf.int[2]

[1] 0.8189752
```

Exporting graphics

R has the ability to produce a variety of plots. Simple plots can be produced with just a few lines of code. These are useful to get a quick visualization of your data or to check on the distribution of residuals from an analysis. More in-depth coding can produce publication-quality plots.

In the Rstudio *Plots* window, there is an *Export* icon which can be used to save the plot as image or pdf file. A method I use is to export the plot as pdf and then open this pdf with either Adobe Photoshop or the free alternative, GIMP (www.gimp.org/). These programs allow you to import the pdf at whatever resolution you need, and then crop out extra white space.

The appearance of exported plots will change depending on the size and scale of exported file. If there are elements missing from a plot, it may be because the size is not ideal. Changing the export size is also an easy way to adjust the size of the text of a plot relative to the other elements.

An additional trick in Rstudio is to change the size of the plot window after the plot is produced, but before it is exported. Sometimes this can get rid of problems where, for example, words in a plot legend are cut off.

Finally, if you export a plot as a pdf, but still need to edit it further, you can open it in Inkscape, ungroup the plot elements, adjust some plot elements, and then export as a high-resolution bitmap image. Just be sure you don't change anything important, like how the data line up with the axes.

Avoiding Pitfalls in R

Grammar, spelling, and capitalization count

Probably the most common problems in programming in any language are syntax errors, for example, forgetting a comma or misspelling the name of a variable or function.

Be sure to include quotes around names requiring them; also be sure to use straight quotes (") and not the smart quotes that some word processors use automatically. It is helpful to write your R code in a plain text editor or in the editor window in R Studio.

Data types in functions

Probably the biggest cause of problems I had when I first started working with R was trying to feed functions the wrong data type. For example, if a function asks for the data as a matrix, and you give it a data frame, it won't work.

A more subtle error I've encountered is when a function is expecting a variable to be a factor vector, and it's really a character ("chr") vector.

For instance if we create a variable in the global environment with the same values as *Sex* and call it *Gender*, it will be a character vector.

```
Gender = c("male", "male", "female", "female")

str(Gender)      # what is the structure of this variable?

chr [1:4] "male" "male" "female" "female"
```

While in the data frame, *Sex* was read in as a factor vector by default:

```
str(D1$ Sex)

Factor w/ 2 levels "female","male": 2 2 1 1
```

One of the nice things about using R Studio is that it allows you to look at the structure of data frames and other objects in the *Environment* window.

Data types can be converted from one data type to another, but it may not be obvious how to do some conversions. Functions to convert data types include *as.factor*, *as.numeric*, and *as.character*.

Style

There isn't an established style for programming in R in many respects, such as if variable names should be capitalized. But there is a Google R Users Style Guide, for those who are interested. I don't necessarily agree with all the recommendations there. And in practice, people use different style conventions. google.github.io/styleguide/Rguide.xml.

Help with R

It's always a good idea to check the help information for a function before using it. Don't necessarily assume a function will perform a test as you think it will. The help information will give the options available for that function, and often those options make a difference with how the test is carried out.

Help in R

In order to see the help file for the *chisq.test* function:

```
?chisq.test
```

In order to specify the *chisq.test* function in the *stats* package, you would use:

```
?stats::chisq.test
```

or

```
help(chisq.test, package=stats)
```

In order to search all installed packages for a term:

```
??"chi-square"
```

In order to view the help for a package

```
help(package=psych)
```

CRAN documentation

Documentation for packages are also available in a .pdf format, which may be more convenient than using the help within R. Also very helpful, some packages include vignettes, which describe how a package might be used.

For a list of available packages, visit cran.r-project.org/web/packages/available_packages_by_name.html.

And clicking on the link for the *psych* package, will bring up a page with a link for the .pdf documentation, two .pdf vignettes, and other information.

Summary and Analysis of Extension Education Program Evaluation in R

Most of the analyses in this book are also presented in [Summary and Analysis of Extension Education Program Evaluation in R](#) (SAEPPER). It may be useful for the reader to consult that book for additional examples and discussion.

Other online resources

Since there are many good resources for R online, an internet search for your question or analysis including the term “r” will often lead to a solution. The reader is cautioned, however, to always check the original R documentation on functions to be sure it will perform an analysis as the user desires.

A convenient tool is the *RSiteSearch* function, which will open a browser window and search for a term in functions and vignettes across a variety of sources:

```
RSiteSearch("chi-square")
```

This tool can also be accessed from: search.r-project.org/nmz.html.

R Tutorials

The descriptions of importing and manipulating data and results in this section of this book don't even scratch the surface of what is possible with R. Going beyond this very brief introduction, however, is beyond the scope of this book. I have tried to provide only enough information so that the reader unfamiliar with R will find the examples in the rest of the book comprehensible.

Luckily, there are many resources available for users wishing to better understand how to program in R, manipulate data, and perform more varied statistical analyses.

One free online resource I've found helpful is *Quick-R* (www.statmethods.net/).

CRAN hosts a collection of R manuals (cran.r-project.org/manuals.html). One that might be helpful is *An Introduction to R* by Venables.

CRAN also hosts a collection of contributed documentation (cran.r-project.org/other-docs.html), in several languages, which may prove helpful.

If readers wish to purchase a more-comprehensive and well-written textbook, *The R Book* by Michael Crawley is one option.

Formal Statistics Books

When describing a particular statistical analysis—especially one that your readers may not be familiar with—it's a good idea to cite an authoritative statistical source. A few that may be useful for this purpose:

- *Biostatistical Analysis* by Jerrold Zar
- *Introduction to Biostatistics* by Sokal and Rohlf
- *Categorical Data Analysis* by Alan Agresti
- *Mixed-Effects Models in S and S-Plus* by José Pinheiro and Douglas Bates

Exact Test of Goodness-of-Fit

The exact test goodness-of-fit can be performed with the *binom.test* function in the native *stats* package. The arguments passed to the function are: the number of successes, the number of trials, and the hypothesized probability of success. The probability can be entered as a decimal or a fraction. Other options include the confidence level for the confidence interval about the proportion, and whether the function performs a one-sided or two-sided (two-tailed) test. In most circumstances, the two-sided test is used.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Goodness-of-Fit Tests for Nominal Variables](#)

Packages used in this chapter

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

```
if(!require(XNomial)){install.packages("XNomial")}
if(!require(BSDA)){install.packages("BSDA")}
if(!require(pwr)){install.packages("pwr")}
```

Introduction

When to use it

Null hypothesis

See the [Handbook](#) for information on these topics.

How the test works

Binomial test examples

```
### -----
### Cat paw example, exact binomial test, pp. 30-31
### -----
### In this example:
###   2 is the number of successes
###   10 is the number of trials
###   0.5 is the hypothesized probability of success

dbinom(2, 10, 0.5)           # Probability of single event only!
                             #   Not binomial test!

[1] 0.04394531

binom.test(2, 10, 0.5,
            alternative="less",      # One-sided test
            conf.level=0.95)

p-value = 0.05469
```

```
binom.test(2, 10, 0.5,
           alternative="two.sided", # Two-sided test
           conf.level=0.95)
```

```
p-value = 0.1094
```

```
# # #
```

Probability density plot

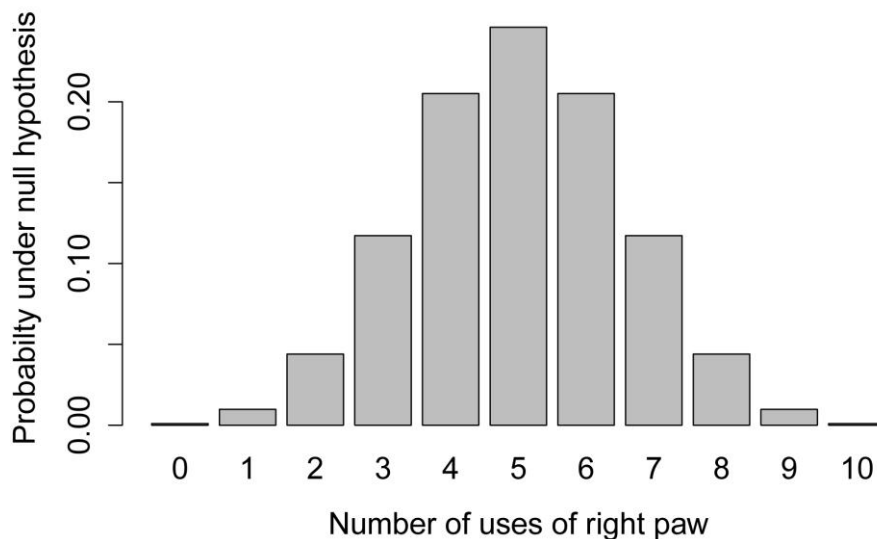
```
### -----
### Probability density plot, binomial distribution, p. 31
### -----
# In this example:
#   You can change the values for trials and prob
#   You can change the values for xlab and ylab

trials = 10
prob = 0.5

x = seq(0, trials) # x is a sequence, 1 to trials
y = dbinom(x, size=trials, p=prob) # y is the vector of heights

barplot (height=y,
         names.arg=x,
         xlab="Number of uses of right paw",
         ylab="Probability under null hypothesis")

# # #
```



Comparing doubling a one-sided test and using a two-sided test

```

#### -----
### Cat hair example, exact binomial test, p. 31-32
### Compares performing a one-sided test and doubling the
### probability, and performing a two-sided test
### -----

binom.test(7, 12, 3/4,
           alternative="less",
           conf.level=0.95)

p-value = 0.1576

Test = binom.test(7, 12, 3/4,           # Create an object called
           alternative="less",         # Test with the test
           conf.level=0.95)           # results.

2 * Test$ p.value                      # This extracts the p-value from the
                                       # test result, we called Test
                                       # and multiplies it by 2

[1] 0.3152874

binom.test(7, 12, 3/4, alternative="two.sided", conf.level=0.95)

p-value = 0.1893      # Equal to the "small p values" method in the Handbook
                       #
                       #

```

Sign test

The following is an example of the two-sample dependent-samples sign test. The data are arranged as a data frame in which each row contains the values for both measurements being compared for each experimental unit. This is sometimes called "wide format" data. The *SIGN.test* function in the *BSDA* package is used. The option *md=0* indicates that the expected difference in the medians is 0 (null hypothesis). This function can also perform a one-sample sign test.

```

#### -----
### Tree beetle example, two-sample sign test, p. 34-35
### -----

Input =("
Row  Angiosperm.feeding  A.count  Gymnosperm.feeding  G.count
1    Corthyliina         458      Pityophthorus       200
2    Scolytinae          5200     Hylastini_Tomacini  180
3    Acanthotomicus_P    123     Orhotomicus         11
4    Xyleborini_D        1500     Ipin                 195
5    Apion                1500     Antliarhininae      12
6    Belinae             150      Allocoryninae_Oxycorinae 30
7    H_Curculionidae     44002   Nemonychidae        85

```

8	H_Cerambycidae	25000	Aseminae_Spondylinae	78
9	Megalopodinae	400	Palophaginae	3
10	H_Chrysomelidae	33400	Aulocoscelinae_Orsod	26

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

```
library(BSDA)
```

```
SIGN.test(x = Data$ A.count,
          y = Data$ B.count,
          md = 0,
          alternative = "two.sided",
          conf.level = 0.95)
```

```
p-value = 0.001953
```

```
# # #
```

Exact multinomial test

See example below in the “Examples” section.

Post-hoc test

Post-hoc example with manual pairwise tests

A multinomial test can be conducted with the *xmulti* function in the package *YNomial*. This can be followed with the individual binomial tests for each proportion, as post-hoc tests.

```
### -----
### Post-hoc example, multinomial and binomial test, p. 33
### -----
```

```
observed = c(72, 38, 20, 18)
expected = c(9, 3, 3, 1)
```

```
library(XNomial)
```

```
xmulti(observed,
        expected,
        detail = 2) # 2: Reports three types of p-value
```

```
P value (LLR) = 0.003404 # log-likelihood ratio
P value (Prob) = 0.002255 # exact probability
P value (Chisq) = 0.001608 # Chi-square probability
```

```
### Note last p-value below agrees with Handbook
```

```
successes = 72
total      = 148
numerator  = 9
denominator = 16
```

```
binom.test(successes, total, numerator/denominator,
           alternative="two.sided", conf.level=0.95)
```

```
p-value = 0.06822
```

```
successes = 38
total     = 148
numerator = 3
denominator = 16
```

```
binom.test(successes, total, numerator/denominator,
           alternative="two.sided", conf.level=0.95)
```

```
p-value = 0.03504
```

```
successes = 20
total     = 148
numerator = 3
denominator = 16
```

```
binom.test(successes, total, numerator/denominator,
           alternative="two.sided", conf.level=0.95)
```

```
p-value = 0.1139
```

```
successes = 18
total     = 148
numerator = 1
denominator = 16
```

```
binom.test(successes, total, numerator/denominator,
           alternative="two.sided", conf.level=0.95)
```

```
p-value = 0.006057
```

```
#      #      #
```

Post-hoc test alternate method with custom function

When you need to do multiple similar tests, however, it is often possible to use the programming capabilities in R to do the tests more efficiently. The following example may be somewhat difficult to follow for a beginner. It creates a data frame and then adds a column called *p.Value* that contains the p-value from the *binom.test* performed on each row of the data frame.

```
### -----
### Post-hoc example, multinomial and binomial test, p. 33
### Alternate method for multiple tests
### -----
```

```
Input =("
```

```

Successes Total Numerator Denominator
72         148     9         16
38         148     3         16
20         148     3         16
18         148     1         16
")

```

```
D1 = read.table(textConnection(Input),header=TRUE)
```

```

Fun = function (x){
  binom.test(x["Successes"],x["Total"],
    x["Numerator"]/x["Denominator"])$ p.value
}

```

```
D1$ p.value = apply(D1, 1, Fun)
```

```
D1
```

```

      Successes Total Numerator Denominator      p.Value
1           72   148         9           16 0.068224131
2           38   148         3           16 0.035040215
3           20   148         3           16 0.113911643
4           18   148         1           16 0.006057012

```

```
#      #      #
```

Intrinsic hypothesis

Assumptions

See the *Handbook* for information on these topics.

Examples

Binomial test examples

```

### -----
### Parasitoid examples, exact binomial test, p. 34
### -----

```

```

binom.test(10, (17+10), 0.5,
  alternative="two.sided",
  conf.level=0.95)

```

```
p-value = 0.2478
```

```

binom.test(36, (7+36), 0.5,
  alternative="two.sided",
  conf.level=0.95)

```

```
p-value = 8.963e-06
```

```
#      #      #
```

```

#### -----
#### Drosophila example, exact binomial test, p. 34
#### -----

binom.test(140, (106+140), 0.5,
           alternative="two.sided",
           conf.level=0.95)

p-value = 0.03516

# # #

#### -----
#### First Mendel example, exact binomial test, p. 35
#### -----

binom.test(428, (428+152), 0.75, alternative="two.sided",
           conf.level=0.95)

p-value = 0.5022 # value is different than in the Handbook
                  # See next example

# # #

#### -----
#### First Mendel example, exact binomial test, p. 35
#### Alternate method with XNomial package
#### -----

observed = c(428, 152)
expected = c(3, 1)

library(XNomial)

xmulti(observed,
        expected,
        detail = 2) # 2: reports three types of p-value

P value (LLR) = 0.5331 # log-likelihood ratio
P value (Prob) = 0.5022 # exact probability
P value (Chisq) = 0.5331 # Chi-square probability

#### Note last p-value below agrees with Handbook

# # #

```

Multinomial test example

```

#### -----
#### Second Mendel example, multinomial exact test, p. 35-36

```



```

###      and SAS example, p. 38
###      -----

observed = c(315, 108, 101, 32)
expected = c(9, 3, 3, 1)

library(XNomial)

xmulti(observed,
        expected,
        detail = 2)          # reports three types of p-value

P value (LLR)  = 0.9261      # log-likelihood ratio
P value (Prob) = 0.9382      # exact probability
P value (Chisq) = 0.9272     # Chi-square probability

### Note last p-value below agrees with Handbook,
###      and agrees with SAS Exact Pr>=ChiSq

#      #      #

```

Graphing the results

Graphing is shown in the “Chi-square Goodness-of-Fit” section.

Similar tests

The *G*-test goodness-of-fit and chi-square goodness-of-fit are presented elsewhere in this book.

How to do the test

Binomial test example where individual responses are counted

```

###      -----
###      Cat paw example from SAS, exact binomial test, pp. 36-37
###      When responses need to be counted
###      -----

Input =("
Paw
right
left
right
right
right
right
left
right
right
right
")

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

Successes = sum(Gus$ Paw == "left")      # Note the == operator

```

```

Failures = sum(Gus$ Paw == "right")

Total = Successes + Failures

Expected = 0.5

binom.test(Successes, Total, Expected,
            alternative="less",      # One-sided test!
            conf.level=0.95)

p-value = 0.05469

binom.test(Successes, Total, Expected,
            alternative="two.sided", # Two-sided test
            conf.level=0.95)

p-value = 0.1094

# # #

```

Other SAS examples

R code for the other SAS example is shown in the examples in previous sections.

Power analysis

Power analysis for binomial test

```

#### -----
#### Power analysis, binomial test, cat paw, p. 38
#### -----

P0 = 0.50
P1 = 0.40
H = ES.h(P0,P1)          # This calculates effect size

library(pwr)

pwr.p.test(
  h=H,
  n=NULL,                # NULL tells the function to
  sig.level=0.05,        # calculate this value
  power=0.80,            # 1 minus Type II probability
  alternative="two.sided")

n = 193.5839             # Slightly different than in Handbook

# # #

```

Power Analysis

Packages used in this chapter

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

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

Introduction

Parameters

How it works

See the [Handbook](#) for information on these topics.

Examples

Power analysis for binomial test

```
### -----
### Power analysis, binomial test, pea color, p. 43
### -----

P0 = 0.75
P1 = 0.78
H = ES.h(P0,P1)           # This calculates effect size

library(pwr)

pwr.p.test(
  h=H,
  n=NULL,                 # NULL tells the function to
  sig.level=0.05,         # calculate this
  power=0.90,             # 1 minus Type II probability
  alternative="two.sided")

n = 2096.953              # Somewhat different than in Handbook

# # #
```

Power analysis for unpaired t-test

```
### -----
### Power analysis, t-test, student height, pp. 43-44
### -----

M1 = 66.6                 # Mean for sample 1
M2 = 64.6                 # Mean for sample 2
S1 = 4.8                  # Std dev for sample 1
S2 = 3.6                  # Std dev for sample 2

Cohen.d = (M1 - M2)/sqrt(((S1^2) + (S2^2))/2)
```

```
library(pwr)

pwr.t.test(
  n = NULL,          # Observations in _each_ group
  d = Cohen.d,
  sig.level = 0.05,  # Type I probability
  power = 0.80,      # 1 minus Type II probability
  type = "two.sample", # Change for one- or two-sample
  alternative = "two.sided")
```

Two-sample t test power calculation

n = 71.61288

NOTE: n is number in **each** group 71.61288

#

How to do power analyses

Methods are shown in the previous examples.

Chi-square Test of Goodness-of-Fit

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Goodness-of-Fit Tests for Nominal Variables](#)

Packages used in this chapter

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

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

When to use it

Null hypothesis

See the [Handbook](#) for information on these topics.

How the test works

Chi-square goodness-of-fit example

```
### -----
### Drosophila example, Chi-square goodness-of-fit, p. 46
### -----

observed = c(770, 230)      # observed frequencies
expected = c(0.75, 0.25)    # expected proportions
```

```
chisq.test(x = observed,
           p = expected)

X-squared = 2.1333, df = 1, p-value = 0.1441

#      #      #
```

Post-hoc test

Assumptions

See the *Handbook* for information on these topics.

Examples: extrinsic hypothesis

```
#### -----
#### Crossbill example, Chi-square goodness-of-fit, p. 47
#### -----

observed = c(1752, 1895)      # observed frequencies
expected = c(0.5, 0.5)       # expected proportions

chisq.test(x = observed,
           p = expected)

X-squared = 5.6071, df = 1, p-value = 0.01789

#      #      #

#### -----
#### Rice example, Chi-square goodness-of-fit, p. 47
#### -----

observed = c(772, 1611, 737)
expected = c(0.25, 0.50, 0.25)

chisq.test(x = observed,
           p = expected)

X-squared = 4.1199, df = 2, p-value = 0.1275

#      #      #

#### -----
#### Bird foraging example, Chi-square goodness-of-fit, pp. 47-48
#### -----

observed = c(70, 79, 3, 4)
expected = c(0.54, 0.40, 0.05, 0.01)

chisq.test(x = observed,
           p = expected)
```

```
x-squared = 13.5934, df = 3, p-value = 0.0035
```

```
# # #
```

Example: intrinsic hypothesis

```
### -----
### Intrinsic example, Chi-square goodness-of-fit, p. 48
### -----

observed      = c(1203, 2919, 1678)
expected.prop  = c(0.211, 0.497, 0.293)

expected.count = sum(observed)*expected.prop

chi2 = sum((observed- expected.count)^2/ expected.count)

chi2

[1] 1.082646

pchisq(chi2,
        df=1,
        lower.tail=FALSE)

[1] 0.2981064

# # #
```

Graphing the results

The first example below will use the *barplot* function in the native *graphics* package to produce a simple plot. First we will calculate the observed proportions and then copy those results into a matrix format for plotting. We'll call this matrix *Matriz*. See the "Chi-square Test of Independence" section for a few notes on creating matrices.

The second example uses the package *ggplot2*, and uses a data frame instead of a matrix. The data frame is named *Forage*. For this example, the code calculates confidence intervals and adds them to the data frame. This code could be skipped if those values were determined manually and put into a data frame from which the plot could be generated.

Sometimes factors will need to have the order of their levels specified for *ggplot2* to put them in the correct order on the plot, as in the second example. Otherwise R will alphabetize levels.

Simple bar plot with barplot

```
### -----
### Simple bar plot of proportions, p. 49
### Uses data in a matrix format
```

```

### -----

observed = c(70, 79, 3, 4)

expected = c(0.54, 0.40, 0.05, 0.01)

total = sum(observed)

observed.prop = observed / total

observed.prop

[1] 0.44871795 0.50641026 0.01923077 0.02564103

### Re-enter data as a matrix

Input =("
Value      Douglas.fir  Ponderosa.pine  Grand.fir  Western.larch
Observed   0.4487179      0.5064103      0.01923077 0.02564103
Expected   0.5400000      0.4000000      0.05000000 0.01000000
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

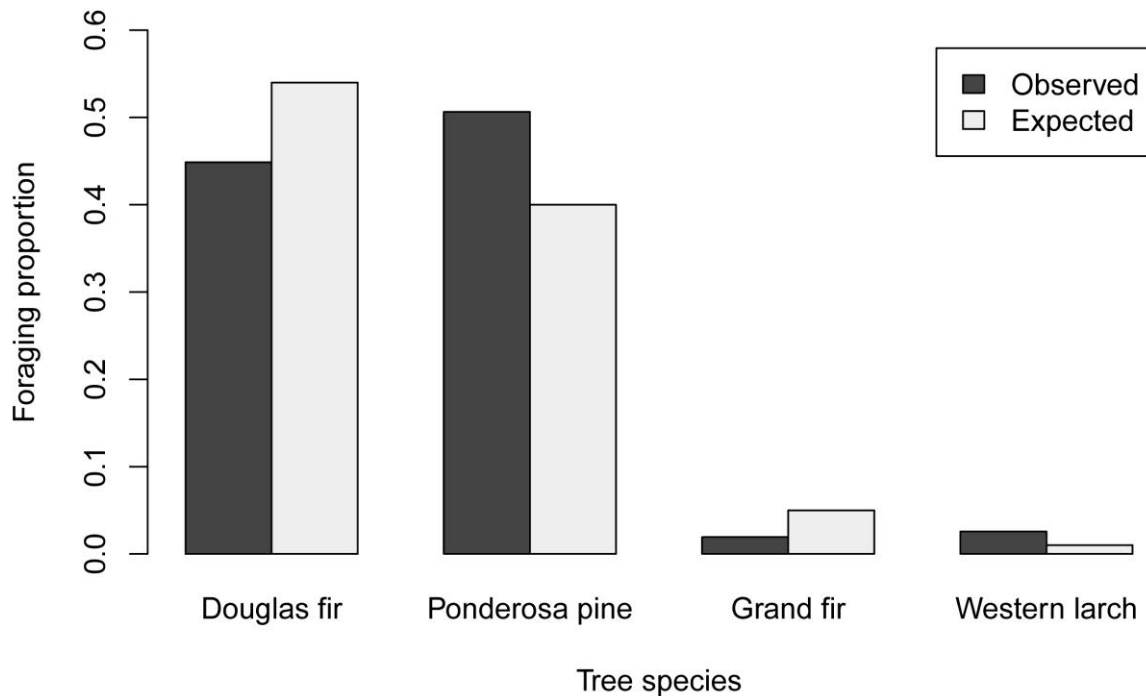
Matriz

      Douglas fir Ponderosa pine  Grand fir western larch
Observed   0.4487179      0.5064103 0.01923077    0.02564103
Expected   0.5400000      0.4000000 0.05000000    0.01000000

barplot(Matriz,
        beside=TRUE,
        legend=TRUE,
        ylim=c(0, 0.6),
        xlab="Tree species",
        ylab="Foraging proportion")

#      #      #

```



Bar plot with confidence intervals with ggplot2

The plot below is a bar chart with confidence intervals. The code calculates confidence intervals. This code could be skipped if those values were determined manually and put in to a data frame from which the plot could be generated.

Sometimes factors will need to have the order of their levels specified for *ggplot2* to put them in the correct order on the plot. Otherwise R will alphabetize levels.

```
### -----
### Graph example, Chi-square goodness-of-fit, p. 49
### Using ggplot2
### Plot adapted from:
### shinyapps.stat.ubc.ca/r-graph-catalog/
### -----
```

```
Input =("
Tree      Value      Count  Total Proportion Expected
'Douglas fir'  Observed   70     156   0.4487     0.54
'Douglas fir'  Expected   54     100   0.54       0.54
'Ponderosa pine' Observed   79     156   0.5064     0.40
'Ponderosa pine' Expected   40     100   0.40       0.40
'Grand fir'    Observed    3     156   0.0192     0.05
'Grand fir'    Expected    5     100   0.05       0.05
'Western larch' Observed    4     156   0.0256     0.01
'Western larch' Expected    1     100   0.01       0.01
")
```

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


Specify the order of factor levels. Otherwise R will alphabetize them.

```
library(dplyr)
```

```
Forage =
mutate(Forage,
  Tree = factor(Tree, levels=unique(Tree)),
  Value = factor(Value, levels=unique(Value)))
```

Add confidence intervals

```
Forage =
mutate(Forage,
  low.ci = apply(Forage[c("Count", "Total", "Expected")],
    1,
    function(x)
      binom.test(x["Count"], x["Total"], x["Expected"])
        )$ conf.int[1]),
  upper.ci = apply(Forage[c("Count", "Total", "Expected")],
    1,
    function(x)
      binom.test(x["Count"], x["Total"], x["Expected"])
        )$ conf.int[2]))
```

```
Forage$ low.ci [Forage$ Value == "Expected"] = 0
Forage$ upper.ci [Forage$ Value == "Expected"] = 0
```

```
Forage
```

	Tree	Value	Count	Total	Proportion	Expected	low.ci	upper.ci
1	Douglas fir	Observed	70	156	0.4487	0.54	0.369115906	0.53030534
2	Douglas fir	Expected	54	100	0.5400	0.54	0.000000000	0.000000000
3	Ponderosa pine	Observed	79	156	0.5064	0.40	0.425290653	0.58728175
4	Ponderosa pine	Expected	40	100	0.4000	0.40	0.000000000	0.000000000
5	Grand fir	Observed	3	156	0.0192	0.05	0.003983542	0.05516994
6	Grand fir	Expected	5	100	0.0500	0.05	0.000000000	0.000000000
7	western larch	Observed	4	156	0.0256	0.01	0.007029546	0.06434776
8	western larch	Expected	1	100	0.0100	0.01	0.000000000	0.000000000

Plot adapted from:

shinyapps.stat.ubc.ca/r-graph-catalog/

```
library(ggplot2)
```

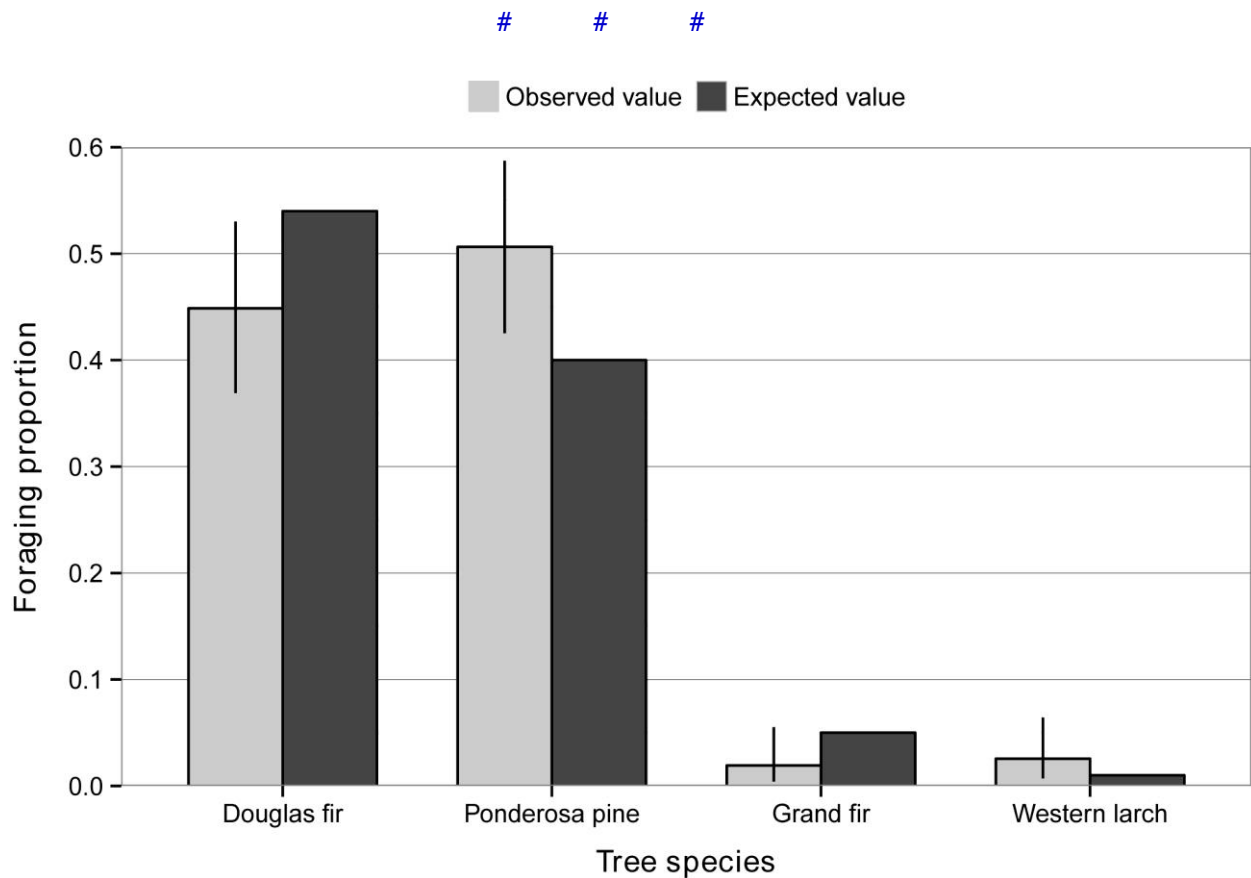
```
library(grid)
```

```
ggplot(Forage,
  aes(x = Tree, y = Proportion, fill = Value,
    ymax=upper.ci, ymin=low.ci)) +
  geom_bar(stat="identity", position = "dodge", width = 0.7) +
  geom_bar(stat="identity", position = "dodge",
    colour = "black", width = 0.7,
    show_guide = FALSE) +
  scale_y_continuous(breaks = seq(0, 0.60, 0.1),
```

```

limits = c(0, 0.60),
expand = c(0, 0)) +
scale_fill_manual(name = "Count type",
  values = c('grey80', 'grey30'),
  labels = c("Observed value",
    "Expected value")) +
geom_errorbar(position=position_dodge(width=0.7),
  width=0.0, size=0.5, color="black") +
labs(x = "Tree species",
  y = "Foraging proportion") +
## ggtitle("Main title") +
theme_bw() +
theme(panel.grid.major.x = element_blank(),
  panel.grid.major.y = element_line(colour = "grey50"),
  plot.title = element_text(size = rel(1.5),
    face = "bold", vjust = 1.5),
  axis.title = element_text(face = "bold"),
  legend.position = "top",
  legend.title = element_blank(),
  legend.key.size = unit(0.4, "cm"),
  legend.key = element_rect(fill = "black"),
  axis.title.y = element_text(vjust= 1.8),
  axis.title.x = element_text(vjust= -0.5))

```



Bar plot of proportions vs. categories. Error bars indicate 95% confidence intervals for each observed proportion.

Similar tests

Chi-square vs. G-test

See the *Handbook* for information on these topics. The *exact test of goodness-of-fit*, the *G-test of goodness-of-fit*, and the *exact test of goodness-of-fit* tests are described elsewhere in this book.

How to do the test

Chi-square goodness-of-fit example

```
### -----
### Pea color example, Chi-square goodness-of-fit, pp. 50-51
### -----

observed = c(423, 133)
expected = c(0.75, 0.25)

chisq.test(x = observed,
           p = expected)

x-squared = 0.3453, df = 1, p-value = 0.5568

# # #
```

Power analysis

Power analysis for chi-square goodness-of-fit

```
### -----
### Power analysis, Chi-square goodness-of-fit, snapdragons, p. 51
### -----

library(pwr)

P0      = c(0.25, 0.50, 0.25)
P1      = c(0.225, 0.55, 0.225)

effect.size = ES.w1(P0, P1)

degrees = length(P0) - 1

pwr.chisq.test(
  w=effect.size,
  N=NULL,
  df=degrees,
  power=0.80,
  sig.level=0.05)
# Total number of observations
# 1 minus Type II probability
# Type I probability

N = 963.4689

# # #
```

G-test of Goodness-of-Fit

The G-test goodness-of-fit test can be performed with the *G.test* function in the package *RVAideMemoire*, the *GTest* function in *DescTools*. As another alternative, you can use R to calculate the statistic and p-value manually.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Goodness-of-Fit Tests for Nominal Variables](#)

Packages used in this chapter

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

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

When to use it

Null hypothesis

How the test works

Post-hoc test

Assumptions

See the [Handbook](#) for information on these topics.

Examples: extrinsic hypothesis

G-test goodness-of-fit test with DescTools and RVAideMemoire

```
#### -----
#### Crossbill example, G-test goodness-of-fit, p. 55
#### -----

observed = c(1752, 1895)      # observed frequencies
expected = c(0.5, 0.5)       # expected proportions

library(DescTools)

GTest(x=observed,
      p=expected,
      correct="none")        # "none" "williams" "yates"

Log likelihood ratio (G-test) goodness of fit test

G = 5.6085, x-squared df = 1, p-value = 0.01787

library(RVAideMemoire)

G.test(x=observed,
       p=expected)

G-test for given probabilities
```

$G = 5.6085$, $df = 1$, $p\text{-value} = 0.01787$

#

G-test goodness-of-fit test by manual calculation

```
#### -----
#### Crossbill example, G-test goodness-of-fit, p. 55
#### Manual calculation
#### -----

observed      = c(1752, 1895)      # observed frequencies
expected.prop = c(0.5, 0.5)        # expected proportions

degrees = 1                        # degrees of freedom

expected.count = sum(observed)*expected.prop

G = 2 * sum(observed * log(observed / expected.count))

G
[1] 5.608512

pchisq(G,
      df=degrees,
      lower.tail=FALSE)

[1] 0.01787343

# # #
```

Examples of G-test goodness-of-fit test with DescTools and RVAideMemoire

```
#### -----
#### Rice example, G-test goodness-of-fit, p. 55
#### -----

observed = c(772, 1611, 737)
expected = c(0.25, 0.50, 0.25)

library(DescTools)

GTest(x=observed,
      p=expected,
      correct="none")           # "none" "williams" "yates"

Log likelihood ratio (G-test) goodness of fit test

G = 4.1471, x-squared df = 2, p-value = 0.1257

library(RVAideMemoire)
```

```

G.test(x=observed,
       p=expected)

G-test for given probabilities
G = 4.1471, df = 2, p-value = 0.1257

# # #

### -----
### Foraging example, G-test goodness-of-fit, pp. 55-56
### -----

observed = c(70, 79, 3, 4)
expected = c(0.54, 0.40, 0.05, 0.01)

library(DescTools)

GTest(x=observed,
      p=expected,
      correct="none")          # "none" "williams" "yates"

Log likelihood ratio (G-test) goodness of fit test

G = 13.145, x-squared df = 3, p-value = 0.004334

library(RVAideMemoire)

G.test(x=observed,
       p=expected)

G-test for given probabilities
G = 13.1448, df = 3, p-value = 0.004334

# # #

```

Example: intrinsic hypothesis

```

### -----
### Intrinsic example, G-test goodness-of-fit, amphipod, p. 56
### -----

observed      = c(1203, 2919, 1678)
expected.prop  = c(.21073, 0.49665, 0.29262)

### Note: These are recalculated for more precision
###       In this case, low precision probabilities
###       change the results

expected.count = sum(observed)*expected.prop

G = 2 * sum(observed * log(observed / expected.count))

```

```
G
[1] 1.032653

pchisq(G,
      df=1,
      lower.tail=FALSE)

[1] 0.3095363

# # #
```

Graphing the results

Graphing would be the same as in the “Chi-square Test of Goodness-of-Fit” section.

Similar tests

Chi-square vs. G-test

See the *Handbook* for information on these topics. The *exact test of goodness-of-fit* and the *chi-square test of goodness-of-fit* tests are described elsewhere in this book.

How to do the test

These examples are shown above.

Power analysis

Power analysis would be the same as in the “Chi-square Test of Goodness-of-Fit” section.

Chi-square Test of Independence

The Chi-square test of independence can be performed with the *chisq.test* function in the native *stats* package in R. For this test, the function requires the contingency table to be in the form of matrix. Depending on the form of the data to begin with, this can require an extra step, either combining vectors into a matrix, or cross-tabulating the counts among factors in a data frame. None of this is too difficult, but it requires following the correct example depending on the initial form of the data.

When using *read.table* and *as.matrix* to read a table directly as a matrix, be careful of extra spaces at the end of lines or extraneous characters in the table, as these can cause errors.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Association Tests for Nominal Variables](#)

Packages used in this chapter

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

```

if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(dplyr)){install.packages("dplyr")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(grid)){install.packages("grid")}
if(!require(pwr)){install.packages("pwr")}

```

When to use it

Example of chi-square test with matrix created with read.table

```

#### -----
### Vaccination example, Chi-square independence, pp. 59-60
### Example directly reading a table as a matrix
### -----

```

```

Input =("
Injection.area No.severe Severe
Thigh          4788      30
Arm            8916      76
")

```

```

Matriz = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

```

```
Matriz
```

```

      No.severe Severe
Thigh      4788      30
Arm        8916      76

```

```

chisq.test(Matriz,
            correct=TRUE)      # Continuity correction for 2 x 2
                                # table

```

Pearson's Chi-squared test with Yates' continuity correction

X-squared = 1.7579, df = 1, p-value = 0.1849

```

chisq.test(Matriz,
            correct=FALSE)     # No continuity correction for 2 x 2
                                # table

```

Pearson's Chi-squared test

X-squared = 2.0396, df = 1, p-value = 0.1533

```
# # #
```

Example of chi-square test with matrix created by combining vectors


```

### -----
### Vaccination example, Chi-square independence, pp. 59–60
### Example creating a matrix from vectors
### -----

R1 = c(4788, 30)
R2 = c(8916, 76)

rows = 2

Matriz = matrix(c(R1, R2),
                nrow=rows,
                byrow=TRUE)

rownames(Matriz) = c("Thigh", "Arm")      # Naming the rows and
colnames(Matriz) = c("No.severe", "Severe") # columns is optional.

Matriz

      No.severe Severe
Thigh      4788      30
Arm       8916      76

chisq.test(Matriz,
            correct=TRUE)      # Continuity correction for 2 x 2
                              # table

Pearson's Chi-squared test with Yates' continuity correction
X-squared = 1.7579, df = 1, p-value = 0.1849

chisq.test(Matriz,
            correct=FALSE)     # No continuity correction for 2 x 2
                              # table

Pearson's Chi-squared test
X-squared = 2.0396, df = 1, p-value = 0.1533

# # #

```

Null hypothesis

How the test works

See the [Handbook](#) for information on these topics.

Post-hoc tests

For the following example of post-hoc pairwise testing, we'll use the *pairwiseNominalIndependence* function from the package *rcompanion* to make the task easier. Then we'll use *pairwise.table* in the native *stats* package as an alternative.

Post-hoc pairwise chi-square tests with rcompanion

```

#### -----
#### Post-hoc example, Chi-square independence, pp. 60-61
#### -----

Input =("
Supplement      No.cancer  Cancer
'Selenium'      8177      575
'Vitamin E'     8117      620
'Selenium+E'    8147      555
'Placebo'       8167      529
")

Matriz = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

Matriz

chisq.test(Matriz)

      X-squared = 7.7832, df = 3, p-value = 0.05071

library(rcompanion)

pairwiseNominalIndependence(Matriz,
                             fisher = FALSE,
                             gtest  = FALSE,
                             chisq  = TRUE,
                             method = "fdr")

      Comparison p.Chisq p.adj.Chisq
1  Selenium : Vitamin E 0.17700      0.2960
2  Selenium : Selenium+E 0.62800      0.6280
3    Selenium : Placebo 0.19700      0.2960
4 Vitamin E : Selenium+E 0.06260      0.1880
5   Vitamin E : Placebo 0.00771      0.0463
6 Selenium+E : Placebo 0.44000      0.5280

```

Post-hoc pairwise chi-square tests with pairwise.table

```

#### -----
#### Post-hoc example, Chi-square independence, pp. 60-61
#### As is, this code works on a matrix with two columns,
#### and compares rows
#### -----

Input =("
Supplement      No.cancer  Cancer
'Selenium'      8177      575
'Vitamin E'     8117      620
'Selenium+E'    8147      555

```

```
'Placebo'      8167      529
")

Matriz = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

Matriz

chisq.test(Matriz)

X-squared = 7.7832, df = 3, p-value = 0.05071

FUN = function(i,j){
  chisq.test(matrix(c(Matriz[i,1], Matriz[i,2],
                     Matriz[j,1], Matriz[j,2]),
                    nrow=2,
                    byrow=TRUE))$ p.value
}

pairwise.table(FUN,
               rownames(Matriz),
               p.adjust.method="none")

# Can adjust p-values;
# see ?p.adjust for options

      Selenium   Vitamin.E Selenium.and.E
Vitamin.E    0.1772113         NA         NA
Selenium.and.E 0.6277621 0.062588260         NA
Placebo      0.1973435 0.007705529      0.4398677

#      #      #
```

Assumptions

See the *Handbook* for information on this topic.

Examples

Chi-square test of independence with continuity correction and without correction

```
#### -----
#### Helmet example, Chi-square independence, p. 63
#### -----

Input =("
PSE      Head.injury  other.injury
Helemt    372         4715
No.helmet 267         1391
")

Matriz = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
```

```

row.names=1))

Matriz

chisq.test(Matriz,
           correct=TRUE)      # Continuity correction for 2 x 2
                             # table

Pearson's Chi-squared test with Yates' continuity correction
X-squared = 111.6569, df = 1, p-value < 2.2e-16

chisq.test(Matriz,
           correct=FALSE)    # No continuity correction for 2 x 2
                             # table

Pearson's Chi-squared test
X-squared = 112.6796, df = 1, p-value < 2.2e-16

#      #      #

```

Chi-square test of independence

```

#### -----
#### Gardemann apolipoprotein example, Chi-square independence,
#### p. 63
#### -----

Input =("
Genotype  No.disease  Coronary.disease
'ins/ins'   268         807
'ins/del'   199         759
'del/del'   42          184
")

Matriz = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

Matriz

chisq.test(Matriz)

Pearson's Chi-squared test
X-squared = 7.2594, df = 2, p-value = 0.02652

#      #      #

```

Graphing the results

The first plot below is a bar chart with confidence intervals, with a style typical of the *ggplot2* package. The second plot is somewhat more similar to the style of the plot in the *Handbook*.

For each example, the code calculates proportions or confidence intervals. This code could be skipped if those values were determined manually and put in to a data frame from which the plot could be generated.

Sometimes factors will need to have the order of their levels specified for *ggplot2* to put them in the correct order on the plot. Otherwise R will alphabetize levels.

Simple bar plot with error bars showing confidence intervals

```
### -----
### Plot example, herons and egrets, Chi-square test of association,
### pp. 63-64
### -----

Input =("
Supplement      No.cancer  Cancer
'Selenium'      8177      575
'Vitamin E'     8117      620
'Selenium+E'    8147      555
'Placebo'       8167      529
")

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

### Add sums and confidence intervals

library(dplyr)

Prostate =
mutate(Prostate,
      Sum = No.cancer + Cancer)

Prostate =
mutate(Prostate,
      Prop = Cancer / Sum,
      low.ci = apply(Prostate[c("Cancer", "Sum")], 1,
        function(y) binom.test(y['Cancer'], y['Sum'])$ conf.int[1]),
      high.ci = apply(Prostate[c("Cancer", "Sum")], 1,
        function(y) binom.test(y['Cancer'], y['Sum'])$ conf.int[2]))

Prostate
```

	Supplement	No.cancer	Cancer	Sum	Prop	low.ci	high.ci
1	Selenium	8177	575	8752	0.06569927	0.06059677	0.07109314
2	Vitamin E	8117	620	8737	0.07096257	0.06566518	0.07654816
3	Selenium+E	8147	555	8702	0.06377844	0.05873360	0.06911770
4	Placebo	8167	529	8696	0.06083257	0.05589912	0.06606271

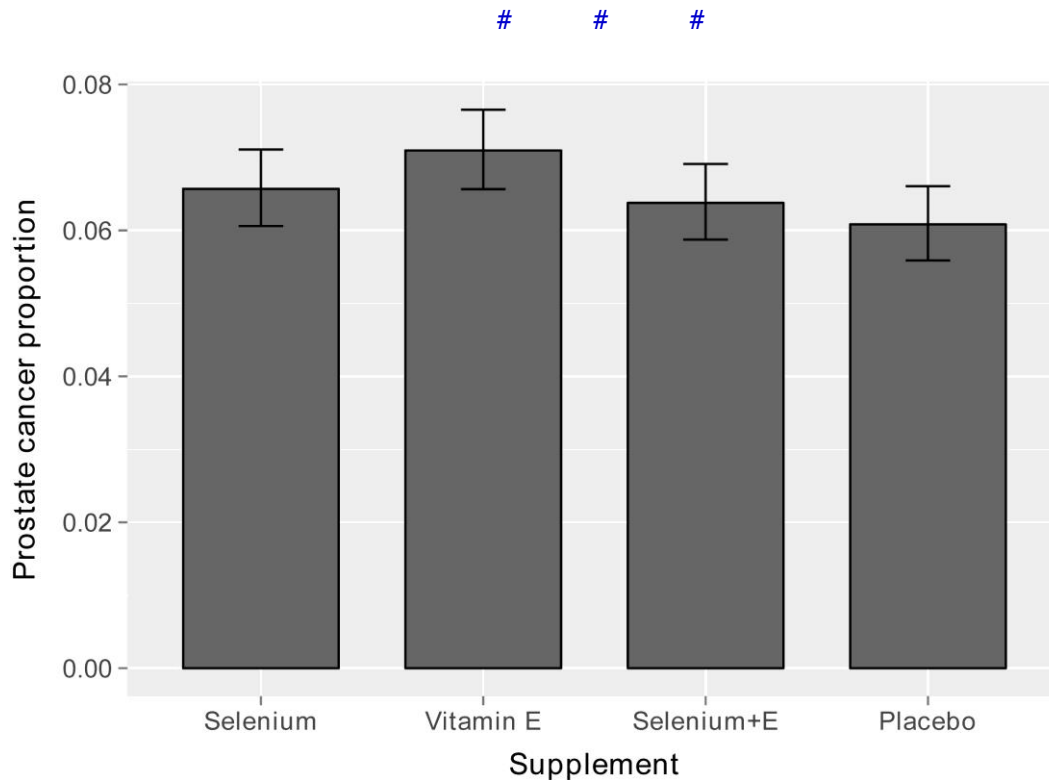
```
### Plot (Bar chart plot)

library(ggplot2)
```

```

ggplot(Prostate,
  aes(x=Supplement, y=Prop)) +
  geom_bar(stat="identity", fill="gray40",
    colour="black", size=0.5,
    width=0.7) +
  geom_errorbar(aes(ymin=low.ci, ymax=high.ci),
    width=0.2, size=0.5, color="black") +
  xlab("Supplement") +
  ylab("Prostate cancer proportion") +
  scale_x_discrete(labels=c("Selenium", "Vitamin E",
    "Selenium+E", "Placebo")) +
  ## ggtitle("Main title") +
  theme(axis.title=element_text(size=14, color="black",
    face="bold", vjust=3)) +
  theme(axis.text = element_text(size=12, color = "gray25",
    face="bold")) +
  theme(axis.title.y = element_text(vjust= 1.8)) +
  theme(axis.title.x = element_text(vjust= -0.5))

```



Bar plot of proportions vs. categories. Error bars indicate 95% confidence intervals for observed proportion.

Bar plot with categories and no error bars

```

#### -----
#### Plot example, herons and egrets, Chi-square independence,
#### p. 64
#### -----

```

```

Input =("
Habitat      Bird      Count
Vegetation   Heron      15
Shoreline    Heron      20
Water        Heron      14
Structures   Heron       6
Vegetation   Egret       8
Shoreline    Egret       5
Water        Egret       7
Structures   Egret       1
")

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

### Specify the order of factor levels

library(dplyr)

Birds=
mutate(Birds,
       Habitat = factor(Habitat,levels=unique(Habitat)),
       Bird = factor(Bird,levels=unique(Bird)))

### Add sums and proportions

Birds$ Sum[Birds$ Bird == 'Heron'] =
  sum(Birds$ Count[Birds$ Bird == 'Heron'])

Birds$ Sum[Birds$ Bird == 'Egret'] =
  sum(Birds$ Count[Birds$ Bird == 'Egret'])

Birds=
mutate(Birds,
       prop = Count / Sum)

Birds

  Habitat Bird Count Sum      prop
1 Vegetation Heron   15  55 0.27272727
2 Shoreline Heron   20  55 0.36363636
3      Water Heron   14  55 0.25454545
4 Structures Heron    6  55 0.10909091
5 Vegetation Egret    8  21 0.38095238
6 Shoreline Egret    5  21 0.23809524
7      Water Egret    7  21 0.33333333
8 Structures Egret    1  21 0.04761905

### Plot adapted from:
### shinyapps.stat.ubc.ca/r-graph-catalog/

library(ggplot2)

```

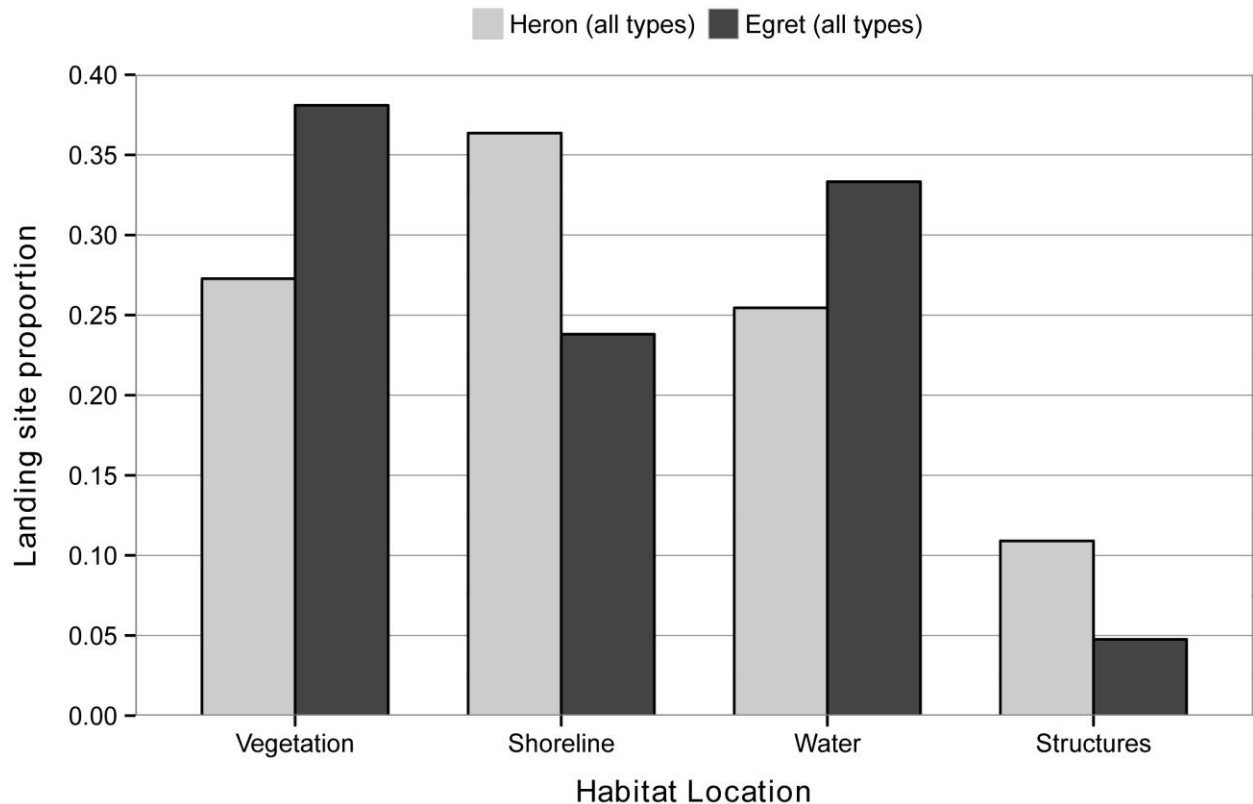
```

library(grid)

ggplot(Birds,
  aes(x = Habitat, y = prop, fill = Bird, ymax=0.40, ymin=0)) +
  geom_bar(stat="identity", position = "dodge", width = 0.7) +
  geom_bar(stat="identity", position = "dodge", colour = "black",
    width = 0.7, show_guide = FALSE) +
  scale_y_continuous(breaks = seq(0, 0.40, 0.05),
    limits = c(0, 0.40),
    expand = c(0, 0)) +
  scale_fill_manual(name = "Bird type" ,
    values = c('grey80', 'grey30'),
    labels = c("Heron (all types)",
      "Egret (all types)")) +
  ## geom_errorbar(position=position_dodge(width=0.7),
  ##               width=0.0, size=0.5, color="black") +
  labs(x = "Habitat Location", y = "Landing site proportion") +
  ## ggtitle("Main title") +
  theme_bw() +
  theme(panel.grid.major.x = element_blank(),
    panel.grid.major.y = element_line(colour = "grey50"),
    plot.title = element_text(size = rel(1.5),
      face = "bold", vjust = 1.5),
    axis.title = element_text(face = "bold"),
    legend.position = "top",
    legend.title = element_blank(),
    legend.key.size = unit(0.4, "cm"),
    legend.key = element_rect(fill = "black"),
    axis.title.y = element_text(vjust= 1.8),
    axis.title.x = element_text(vjust= -0.5))

# # #

```

Similar tests

Chi-square vs. G-test

See the Handbook for information on these topics. *Fisher's exact test*, *G-test*, and *McNemar's test* are discussed elsewhere in this book.

How to do the test

Chi-square test of independence with data as a data frame

In the following example for the chi-square test of independence, the data is read in as a data frame, not as a matrix as in previous examples. This allows more flexibility with how data are entered. For example you could have counts for same *genotype* and *health* distributed among several lines, or have a count of 1 for each row, with a separate row for each individual observation. The *xtabs* function is used to tabulate the data and convert them to a contingency table.

```
### -----
### Gardemann apolipoprotein example, Chi-square independence,
### SAS example, pp. 65-66
### Example using cross-tabulation
### -----

Input =("
Genotype Health Count
ins-ins no_disease 268
ins-ins disease 807
ins-del no_disease 199
```

```

ins-del    disease    759
del-del    no_disease  42
del-del    disease    184
")

```

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

```
### Cross-tabulate the data
```

```
Data.xtabs = xtabs(Count ~ Genotype + Health,
                   data=Data.frame)
```

```
Data.xtabs
```

```

      Health
Genotype disease no_disease
del-del    184         42
ins-del    759        199
ins-ins    807        268

```

```
summary(Data.xtabs) # includes N and factors
```

```

Number of cases in table: 2259
Number of factors: 2

```

```
### Chi-square test of independence
```

```
chisq.test(Data.xtabs)
```

```
x-squared = 7.2594, df = 2, p-value = 0.02652
```

```
# # #
```

Power analysis

Power analysis for chi-square test of independence

```

### -----
### Power analysis, chi-square independence, pp. 66-67
### -----

```

```

# This example assumes you are using a Chi-square test of
# independence. The example in the Handbook appears to use
# a Chi-square goodness-of-fit test

```

```

# In the pwr package, for the Chi-square test of independence,
# the table probabilities should sum to 1

```

```

Input =("
Genotype No.cancer Cancer
GG        0.18      0.165

```

```
GA      0.24      0.225
AA      0.08      0.110
")
```

```
P = as.matrix(read.table(textConnection(Input),
                          header=TRUE,
                          row.names=1))
```

```
P
      No.cancer Cancer
GG      0.18  0.165
GA      0.24  0.225
AA      0.08  0.110
```

```
sum(P)      # Sum of values in the P matrix
```

```
[1] 1
```

```
library(pwr)
```

```
effect.size = ES.w2(P)
```

```
degrees = (nrow(P)-1)*(ncol(P)-1) # Calculate degrees of freedom
```

```
pwr.chisq.test(
  w=effect.size,
  N=NULL,      # Total number of observations
  df=degrees,
  power=0.80,  # 1 minus Type II probability
  sig.level=0.05) # Type I probability
```

```
      w = 0.07663476 # Answer differs significantly
      N = 1640.537   # from Handbook
      df = 2         # Total observations
sig.level = 0.05
power = 0.8
```

```
#      #      #
```

G-test of Independence

There are a few different options for performing G-tests of independence in R. One is the *G.test* function in the package *RVAideMemoire*. Another is the *GTest* function in the package *DescTools*.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Association Tests for Nominal Variables](#)

Packages used in this chapter

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

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

When to use it

G-test example with functions in DescTools and RVAideMemoire

```
### -----
### Vaccination example, G-test of independence, pp. 68-69
### -----

Input =("
Injection.area  No.severe  Severe
Thigh           4788       30
Arm             8916       76
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

library(DescTools)

GTest(Matriz,
      correct="none")          # "none" "williams" "yates"

Log likelihood ratio (G-test) test of independence without correction

G = 2.1087, x-squared df = 1, p-value = 0.1465

library(RVAideMemoire)

G.test(Matriz)

G = 2.1087, df = 1, p-value = 0.1465 # Note values differ from
                                     # the Handbook
                                     # for this example

# # #
```

Null hypothesis

How the test works

See the [Handbook](#) for information on these topics.

Post-hoc tests

For the following example of post-hoc pairwise testing, we'll use the *pairwise.G.test* function from the package *RVAideMemoire* to make the task easier. Then we'll use *pairwise.table* in the native *stats* package as an alternative.

Post-hoc pairwise G-tests with RVAideMemoire

```

#### -----
#### Post-hoc example, G-test of independence, pp. 69-70
#### -----

Input =("
Supplement      No.cancer  Cancer
'Selenium'      8177       575
'Vitamin E'     8117       620
'Selenium+E'    8147       555
'Placebo'       8167       529
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

library(RVAideMemoire)

G.test(Matriz)

      G = 7.7325, df = 3, p-value = 0.05188

library(RVAideMemoire)

pairwise.G.test(Matriz,
                p.method = "none")           # Can adjust p-values;
                                           # see ?p.adjust for options

           Selenium Vitamin E Selenium+E
Vitamin E  0.168      -         -
Selenium+E 0.606     0.058      -
Placebo    0.187     0.007     0.422

```

Post-hoc pairwise G-tests with pairwise.table

As is, this function works on a matrix with two columns, and compares rows.

```

#### -----
#### Post-hoc example, G-test of independence, pp. 69-70
#### -----

Input =("
Supplement      No.cancer  Cancer
'Selenium'      8177       575
'Vitamin E'     8117       620
'Selenium+E'    8147       555
'Placebo'       8167       529
")

```

```

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

library(DescTools)

GTest(Matriz,
      correct="none")

Log likelihood ratio (G-test) test of independence without correction

G = 7.7325, X-squared df = 3, p-value = 0.05188

FUN = function(i,j){
  GTest(matrix(c(Matriz[i,1], Matriz[i,2],
                 Matriz[j,1], Matriz[j,2])),
           nrow=2,
           byrow=TRUE),
        correct="none")$ p.value   # "none" "williams" "yates"
}

pairwise.table(FUN,
               rownames(Matriz),
               p.adjust.method="none")      # Can adjust p-values
                                           # See ?p.adjust for options

      Selenium   Vitamin E Selenium+E
Vitamin E 0.1677388      NA      NA
Selenium+E 0.6060951 0.058385135      NA
Placebo    0.1866826 0.007004601 0.4215013

      #      #      #

```

Assumptions

See the *Handbook* for information on this topic.

Examples

G-tests with DescTools and RVAideMemoire

```

#### -----
#### Helmet example, G-test of independence, p. 72
#### -----

Input =("
PSE      Head.injury  Other.injury
Helemt    372         4715
No.helmet 267         1391
")

Matriz = as.matrix(read.table(textConnection(Input),

```

```

        header=TRUE,
        row.names=1))

Matriz

library(DescTools)

GTest(Matriz,
      correct="none")          # "none" "williams" "yates"

  Log likelihood ratio (G-test) test of independence without correction

  G = 101.54, x-squared df = 1, p-value < 2.2e-16

library(RVAideMemoire)

G.test(Matriz)

  G = 101.5437, df = 1, p-value < 2.2e-16

                                #      #      #

### -----
### Gardemann apolipoprotein example, G-test of independence,
###   p. 72
### -----

Input =("
Genotype  No.disease  Coronary.disease
ins.ins   268         807
ins.del   199         759
del.del   42          184
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

library(DescTools)

GTest(Matriz,
      correct="none")          # "none" "williams" "yates"

  Log likelihood ratio (G-test) test of independence without correction

  G = 7.3008, x-squared df = 2, p-value = 0.02598

library(RVAideMemoire)

G.test(Matriz)

```

$G = 7.3008$, $df = 2$, $p\text{-value} = 0.02598$

#

Graphing the results

Graphing is discussed above in the “Chi-square Test of Independence” section.

Similar tests

Chi-square vs. G-test

See the *Handbook* for information on these topics. *Fisher’s exact test*, *chi-square test*, and *McNemar’s test* are discussed elsewhere in this book.

How to do the test

G-test of independence with data as a data frame

In the following example, the data is read in as a data frame, and the *xtabs* function is used to tabulate the data and convert them to a contingency table.

```
### -----
### Gardemann apolipoprotein example, G-test of independence,
### SAS example, pp. 74–75
### Example using cross-tabulation
### -----
```

```
Input =("
Genotype Health Count
ins-ins no_disease 268
ins-ins disease 807
ins-del no_disease 199
ins-del disease 759
del-del no_disease 42
del-del disease 184
")
```

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

```
### Cross-tabulate the data
```

```
Data.xtabs = xtabs(Count ~ Genotype + Health,
                    data=Data.frame)
```

```
Data.xtabs
```

	Health	
Genotype	disease	no_disease
del-del	184	42
ins-del	759	199
ins-ins	807	268


```
summary(Data.xtabs)                # includes N and factors

Number of cases in table: 2259
Number of factors: 2

### G-tests

library(DescTools)

GTest(Data.xtabs,
       correct="none")              # "none" "williams" "yates"

Log likelihood ratio (G-test) test of independence without correction

G = 7.3008, x-squared df = 2, p-value = 0.02598

library(RVAideMemoire)

G.test(Data.xtabs)

G = 7.3008, df = 2, p-value = 0.02598

#      #      #
```

Power analysis

To calculate power or required samples, follow examples in the “Chi-square Test of Independence” section.

Fisher's Exact Test of Independence

Examples in *Summary and Analysis of Extension Program Evaluation*
[SAEPPER: Association Tests for Nominal Variables](#)

Packages used in this chapter

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

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

When to use it

Null hypothesis

How the test works

See the [Handbook](#) for information on these topics.

Post-hoc tests

For the following example of post-hoc pairwise testing, we'll use the *pairwiseNominalIndependence* function from the package *rcompanion* to make the task easier.

Post-hoc pairwise Fisher's exact tests with RVAideMemoire

```
### -----
### Post-hoc example, Fisher's exact test, p. 79
### -----

Input =("
Frequency  Damaged  Undamaged
Daily      1        24
Weekly     5        20
Monthly    14        11
Quarterly  11        14
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

fisher.test(Matriz,
            alternative="two.sided")

p-value = 0.0001228
alternative hypothesis: two.sided

library(rcompanion)

PT = pairwiseNominalIndependence(Matriz,
                                fisher = TRUE,
                                gtest  = FALSE,
                                chisq  = FALSE,
                                digits = 3)

PT

      Comparison p.Fisher p.adj.Fisher
1   Daily : Weekly 0.189000   0.227000
2   Daily : Monthly 0.000102   0.000612
3   Daily : Quarterly 0.001920   0.005760
4   Weekly : Monthly 0.018600   0.037200
5   Weekly : Quarterly 0.128000   0.192000
6   Monthly : Quarterly 0.572000   0.572000

library(rcompanion)

cldList(comparison = PT$Comparison,
```

```
p.value = PT$p.adj.Fisher,
threshold = 0.05)
```

```
      Group Letter MonoLetter
1   Daily      a           a
2  weekly     ab          ab
3 Monthly      c           c
4 Quarterly   bc          bc
```

Summary of results

<u>Frequency</u>	<u>Damaged</u>	<u>Letter</u>
Daily	4%	a
weekly	20%	ab
Quarterly	44%	bc
Monthly	56%	c

Groups sharing a letter are not significantly different (alpha = 0.05)

```
#      #      #
```

Assumptions

See the *Handbook* for information on this topic.

Examples

Examples of Fisher's exact test with data in a matrix

```
### -----
### Chipmunk example, Fisher's exact test, p. 80
### -----
```

```
Input =("
Distance  Trill  No.trill
10m       16     8
100m      3     18
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))
```

```
Matriz
```

```
fisher.test(Matriz,
            alternative="two.sided")
```

```
p-value = 0.0006862
```

```
#      #      #
```

```
### -----
### Drosophila example, Fisher's exact test, p. 81
### -----
```

```
Input =("
Variation      Synonymous Replacement
'Polymorphisms' 43          2
'Fixed differences' 17        7
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))
```

```
Matriz
```

```
fisher.test(Matriz,
             alternative="two.sided")
```

```
p-value = 0.006653
```

```
#      #      #
```

```
### -----
### King penguin example, Fisher's exact test, p. 81
### -----
```

```
Input =("
Site      Alive Dead
Lower     43    7
Middle    44    6
Upper     49    1
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))
```

```
Matriz
```

```
fisher.test(Matriz,
             alternative="two.sided")
```

```
p-value = 0.08963
alternative hypothesis: two.sided
```

```
#      #      #
```

```
### -----
### Moray eel example, Fisher's exact test, pp. 81-82
### -----
```

```

Input =("
Site      G.moringa  G.vicinus
Grass     127        116
Sand      99         67
Border    264        161
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

fisher.test(Matriz,
            alternative="two.sided")

p-value = 0.04438
alternative hypothesis: two.sided

#      #      #

```

```

### -----
### Herons example, Fisher's exact test, p. 82
### -----

```

```

Input =("
Site      Heron  Egret
Vegetation 15     8
Shoreline  20     5
Water      14     7
Structures  6     1
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

fisher.test(Matriz,
            alternative="two.sided")

p-value = 0.5491
alternative hypothesis: two.sided

#      #      #

```

Graphing the results

Graphing is discussed above in the “Chi-square Test of Independence” section.

Similar tests – McNemar's test

Care is needed in setting up the data for McNemar's test. For a before-and-after test, the contingency table is set-up as before and after as row and column headings, or vice-versa. Note that the total observations in the contingency table is equal to the number of experimental units. That is, in the following example there are 62 men, and the sum of the counts in the contingency table is 62. If you set up the table incorrectly, you might end with double this number, and this will not yield the correct results.

McNemar's test with data in a matrix

```
### -----
### Dysfunction example, McNemar test, pp. 82-83
### -----

Input =("
Row      After.no  After.yes
Before.no 46        10
Before.yes 0         6
")

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

mcnemar.test(Matriz, correct=FALSE)

McNemar's chi-squared = 10, df = 1, p-value = 0.001565

#      #      #
```

McNemar's test with data in a data frame

```
### -----
### Dysfunction example, McNemar test, pp. 82-83
### Example using cross-tabulation
### -----

Input =("
ED.before ED.after  Count
no        no      46
no        yes     10
yes       no       0
yes       yes      6
")

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

Data.xtabs = xtabs(Count ~ ED.before + ED.after, data=Data)

Data.xtabs
```

		ED.after	
ED.before	no	46	10
	yes	0	6

```
mcnemar.test(Data.xtabs, correct=FALSE)
```

```
McNemar's chi-squared = 10, df = 1, p-value = 0.001565
```

```
# # #
```

How to do the test

Fisher's exact test with data as a data frame

```
#### -----
### Chipmunk example, Fisher's exact test, SAS example, p. 83
### Example using cross-tabulation
### -----
```

```
Input =("
Distance  Sound  Count
10m       trill  16
10m       notrill 8
100m      trill  3
100m      notrill 18
")
```

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

```
Data.xtabs = xtabs(Count ~ Distance + Sound, data=Data)
```

```
Data.xtabs
```

	Sound	
Distance	notrill	trill
100m	18	3
10m	8	16

```
summary(Data.xtabs)
```

```
### Fisher's exact test of independence
```

```
fisher.test(Data.xtabs,
             alternative="two.sided")
```

```
p-value = 0.0006862
```

```
# # #
```

```
#### -----
### Bird example, Fisher's exact test, SAS example, p. 84
```

```
### Example using cross-tabulation
```

```
### -----
```

```
Input =("
Bird   Substrate   Count
heron  vegetation   15
heron  shoreline   20
heron  water       14
heron  structures   6
egret  vegetation   8
egret  shoreline   5
egret  water       7
egret  structures   1
")
```

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

```
Data.xtabs = xtabs(Count ~ Bird + Substrate, data=Data)
```

```
Data.xtabs
```

```
      Substrate
Bird  shoreline structures vegetation water
egret      5         1         8       7
heron     20         6        15      14
```

```
summary(Data.xtabs)
```

```
### Fisher's exact test of independence
```

```
fisher.test(Data.xtabs,
             alternative="two.sided")
```

```
p-value = 0.5491
alternative hypothesis: two.sided
```

```
#      #      #
```

Power analysis

To calculate power or required samples, follow examples in the “Chi-square Test of Independence” section.

There, the result was

```
N = 1640.537 # Total observations
```

compared with the value in the *Handbook* of $N_{\text{total}} = 1523$ for this section.

Small Numbers in Chi-square and G-tests

The problem with small numbers

See the [Handbook](#) for information on these topics.

Yates' and William's corrections in R

The following table lists the continuity corrections available for the Chi-square tests and G-tests discussed in this book.

Test	Function	Package	Correction	Option	Default	Notes
Chi-square	chisq.test	stats	Yates	correct=TRUE	TRUE	2 x 2 table only
G	G.test	RVAideMemoire	(none)			
G	GTest	DescTools	Yates	correct="yates"	"none"	
			Williams	correct="williams"		

Pooling

Recommendation

See the *Handbook* for information on these topics.

Repeated G-tests of Goodness-of-Fit

These examples use the *G.test* function in the *RVAideMemoire* package, but the *GTest* function in the *DescTools* package could be used in the same manner.

Packages used in this chapter

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

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

When to use it

Null hypothesis

See the *Handbook* for information on these topics.

How to do the test

Repeated G-tests of goodness-of-fit example

```
#### -----
### Arm crossing example, Repeated G-tests of goodness-of-fit,
### pp. 91-93
### -----
```

```
Input =("
Ethnic.group R    L
Yemen       168  174
Djerba      132  195
Kurdistan   167  204
Libya       162  212
Berber      143  194
Cochin      153  174
")
```

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

Individual G-tests

```
library(RVAideMemoire)
```

```
Fun.G = function (Q){
  G.test(x=c(Q["R"], Q["L"]),
         p=c(0.5, 0.5)
        )$statistic
}
# Functions
# to calculate
# individual G's,
# df's, and p-values
```

```
Fun.df = function (Q){
  G.test(x=c(Q["R"], Q["L"]),
         p=c(0.5, 0.5)
        )$parameter
}
```

```
Fun.p = function (Q){
  G.test(x=c(Q["R"], Q["L"]),
         p=c(0.5, 0.5)
        )$p.value
}
```

```
library(dplyr)
```

```
Data=
mutate(Data,
  Prop.R = R / (R + L),
  G =      apply(Data[c("R", "L")], 1, Fun.G),
  df =      apply(Data[c("R", "L")], 1, Fun.df),
  p.Value = apply(Data[c("R", "L")], 1, Fun.p))
# calculate proportion
# of right arms
```

Data

	Ethnic.group	R	L	Prop.R	G	df	p.Value
1	Yemen	168	174	0.4912281	0.1052686	1	0.745596489
2	Djerba	132	195	0.4036697	12.2138397	1	0.000474363
3	Kurdistan	167	204	0.4501348	3.6961684	1	0.054537574
4	Libya	162	212	0.4331551	6.7045477	1	0.009616732
5	Berber	143	194	0.4243323	7.7478346	1	0.005377698
6	Cochin	153	174	0.4678899	1.3495524	1	0.245356383

Heterogeneity G-test

```
Data.matrix = as.matrix(Data[c("R", "L"]]) # We need a data matrix
# to run G-test
Data.matrix # for heterogeneity
```

```
      R  L
[1,] 168 174
[2,] 132 195
[3,] 167 204
[4,] 162 212
[5,] 143 194
[6,] 153 174
```

```
G.test(Data.matrix) # Heterogeneity
```

```
G-test
G = 6.7504, df = 5, p-value = 0.2399
```

Pooled G-test

```
Total.R = sum(Data$R) # Set up data for pooled
Total.L = sum(Data$L) # G-test
```

```
observed = c(Total.R, Total.L)
expected = c(0.5, 0.5)
```

```
G.test(x=observed,
      p=expected)
```

```
G-test for given probabilities
G = 25.0668, df = 1, p-value = 5.538e-07
```

Total G-test

```
Total.G = sum(Data$G) # Set up data for total
# G-test
```

```
Total.df = sum(Data$df)
```

```
Total.G # Total
```

```
[1] 31.81721
```

```
Total.df
```

```
[1] 6
```

```
pchisq(Total.G,
        df= Total.df,
        lower.tail=FALSE)
```

```
[1] 1.768815e-05
```

```
# # #
```

Example

Repeated G-tests of goodness-of-fit example

```
### -----
### Drosophila example, Repeated G-tests of goodness-of-fit,
### p. 93
### -----
```

```
Input =("
Trial      D      S
'Trial 1'   296   366
'Trial 2'    78    72
'Trial 3'   417   467
")
```

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

Individual G-tests

```
library(RVAideMemoire)
```

```
Fun.G = function (Q){
  G.test(x=c(Q["D"], Q["S"]),
         p=c(0.5, 0.5)
  )$statistic
}
# Functions
# to calculate
# individual G's and
# p-values
```

```
Fun.df = function (Q){
  G.test(x=c(Q["D"], Q["S"]),
         p=c(0.5, 0.5)
  )$parameter
}
```

```
Fun.p = function (Q){
  G.test(x=c(Q["D"], Q["S"]),
         p=c(0.5, 0.5)
```

```

    )$p.value
  }

library(dplyr)

Data =
mutate(Data,
  G =      apply(Data[c("D", "S")], 1, Fun.G),
  df =      apply(Data[c("D", "S")], 1, Fun.df),
  p.Value = apply(Data[c("D", "S")], 1, Fun.p))

Data

```

```

      Trial   D   S      G df   p.Value
1 Trial 1 296 366 7.415668  1 0.00646583
2 Trial 2   78  72 0.240064  1 0.62415986
3 Trial 3 417 467 2.829564  1 0.09254347

```

Heterogeneity G-test

```

Data.matrix = as.matrix(Data[c("D", "S")]) # We need a data matrix
Data.matrix # to run G-test
Data.matrix # for heterogeneity

```

```

      D   S
[1,] 296 366
[2,]  78  72
[3,] 417 467

```

```

G.test(Data.matrix) # Heterogeneity

G-test
G = 2.8168, df = 2, p-value = 0.2445

```

Pooled G-test

```

Total.D = sum(Data$D) # Set up data for pooled
Total.S = sum(Data$S) # G-test

observed = c(Total.D, Total.S)
expected = c(0.5, 0.5)

G.test(x=observed, # Pooled
  p=expected)

G-test for given probabilities
G = 7.6685, df = 1, p-value = 0.005619

```

Total G-test

```
Total.G = sum(Data$G) # Set up data for total
degrees = 3 # G-test

Total.G = sum(Data$G) # Set up data for total
Total.df = sum(Data$df) # G-test

Total.G # Total

[1] 10.4853

Total.df

[1] 3

pchisq(Total.G,
        df=Total.df,
        lower.tail=FALSE)

[1] 0.01486097

# # #
```

Similar tests

See the *Handbook* for information on these topics.

Cochran–Mantel–Haenszel Test for Repeated Tests of Independence

The Cochran–Mantel–Haenszel test can be performed in R with the *mantelhaen.test* function in the native *stats* package. A few other useful functions come from the package *vcd*. One is *woolf_test*, which performs the Woolf test for homogeneity of the odds ratio across strata levels. This has a similar function to the Breslow-Day test mentioned in the *Handbook*. If this test is significant, the C-M-H test may not be appropriate. The Breslow-Day test itself can be performed with a function in the package *DescTools*. For cautions about using this test, see the documentation for this function, or other appropriate sources.

```
library(DescTools); ?BreslowDayTest
```

There are a couple of different ways to generate the three-way contingency table. The table can be read in with the *read.ftable* function. Note that the columns are the stratum variable.

Caution should be used with the formatting, since *read.ftable* can be fussy. I've noticed that it doesn't like leading spaces in the rows. Certain editors, such as the one in R Studio, may add leading spaces when this code is pasted in. To alleviate this, delete those spaces manually, or paste the code into a plain text editor, save the file as a .R file, and then open that file with R Studio.

Another way to generate the contingency table is beginning with a data frame and tabulating the data using the *xtabs* function. The second example uses this method.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Cochran–Mantel–Haenszel Test for 3-Dimensional Tables](#)

Packages used in this chapter

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

```
if(!require(dplyr)){install.packages("dplyr")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(grid)){install.packages("grid")}
if(!require(vcd)){install.packages("vcd")}
```

When to use it

Null hypothesis

How the test works

Assumptions

See the [Handbook](#) for information on these topics.

Examples

Cochran–Mantel–Haenszel Test with data read by read.ftable

```
### -----
### Handedness example, Cochran–Mantel–Haenszel test, p. 97–98
### Example using read.ftable
### -----

# Note no spaces on lines before row names.
# read.ftable can be fussy about leading spaces.

Input =(
"
      Group w.Child B.adult PA.white w.men G.soldier
whorl   Handed
Clockwise Right      708      136      106      109      801
          Left       50       24       32       22      102
CounterC Right     169       73       17       16      180
          Left      13       14        4       26       25
")

Tabla = as.table(read.ftable(textConnection(Input)))
```

```
fable(Tabla) # Display a flattened table
```

Cochran-Mantel-Haenszel test

```
mantelhaen.test(Tabla)
```

```
Mantel-Haenszel x-squared = 5.9421, df = 1, p-value = 0.01478
```

Woolf test

```
library(vcd)
```

```
oddsratio(Tabla, log=TRUE) # Show log odds for each 2x2
```

```
      w.Child      B.adult      PA.white      w.men      G.soldier
0.08547173  0.08319894 -0.24921579  2.08581324  0.08680711
```

```
library(vcd)
```

```
woolf_test(Tabla) # Woolf test for homogeneity of
                  # odds ratios across strata.
                  # If significant, C-M-H test
                  # is not appropriate
```

```
Woolf-test on Homogeneity of Odds Ratios (no 3-way assoc.)
```

```
x-squared = 22.8165, df = 4, p-value = 0.0001378
```

Breslow-Day test

```
library(DescTools)
```

```
BreslowDayTest(Tabla)
```

```
Breslow-Day Test for Homogeneity of the Odds Ratios
```

```
x-squared = 24.7309, df = 4, p-value = 5.698e-05
```

Individual Fisher exact tests

```
n = dim(Tabla)[3]
```

```
for(i in 1:n){
  Name = dimnames(Tabla)[3]$Group[i]
  P.value = fisher.test(Tabla[, , i])$p.value
  cat(Name, "\n")
}
```



```
cat("Fisher test p-value: ", P.value, "\n")
cat("\n")
}
```

Note: "Group" must be the name of the stratum variable

```
w.Child
Fisher test p-value: 0.7435918
```

```
B.adult
Fisher test p-value: 0.8545009
```

```
PA.white
Fisher test p-value: 0.7859788
```

```
W.men
Fisher test p-value: 6.225227e-08
```

```
G.soldier
Fisher test p-value: 0.7160507
```

#

Cochran-Mantel-Haenszel Test with data entered as a data frame

```
### -----
### Mussel example, Cochran-Mantel-Haenszel test, pp. 98-99
### Example using cross-tabulation of a data frame
### -----
```

```
Input =("
Location  Habitat      Allele      Count
Tillamook marine        94         56
Tillamook estuarine    94         69
Tillamook marine      non-94       40
Tillamook estuarine   non-94       77
Yaquina   marine        94         61
Yaquina   estuarine     94        257
Yaquina   marine      non-94         57
Yaquina   estuarine   non-94       301
Alsea     marine        94         73
Alsea     estuarine     94         65
Alsea     marine      non-94         71
Alsea     estuarine   non-94         79
Umpqua    marine        94         71
Umpqua    estuarine     94         48
Umpqua    marine      non-94         55
Umpqua    estuarine   non-94         48
")
```

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

```
### Specify the order of factor levels
### Otherwise, R will alphabetize them

library(dplyr)

Data =
mutate(Data,
  Location = factor(Location, levels=unique(Location)),
  Habitat = factor(Habitat, levels=unique(Habitat)),
  Allele = factor(Allele, levels=unique(Allele)))

### Cross-tabulate the data
### Note here, Location is stratum variable (is last)
### Habitat x Allele are 2 x 2 tables

Data.xtabs = xtabs(Count ~ Allele + Habitat + Location,
  data=Data)
```

```
fable(Data.xtabs) # Display a flattened table
```

		Location Tillamook	Yaquina	Alsea	Umpqua
Allele	Habitat				
94	marine	56	61	73	71
	estuarine	69	257	65	48
non-94	marine	40	57	71	55
	estuarine	77	301	79	48

Cochran-Mantel-Haenszel test

```
mantelhaen.test(Data.xtabs)

Mantel-Haenszel X-squared = 5.0497, df = 1, p-value = 0.02463
```

Woolf test

```
library(vcd)

oddsratio(Data.xtabs, log=TRUE) # Show log odds for each 2x2

Tillamook Yaquina Alsea Umpqua
0.4461712 0.2258568 0.2228401 0.2553467

library(vcd)

woolf_test(Data.xtabs) # Woolf test for homogeneity of
# odds ratios across strata.
# If significant, C-M-H test
# is not appropriate
```

woolf-test on Homogeneity of Odds Ratios (no 3-way assoc.)

X-squared = 0.5292, df = 3, p-value = 0.9124

Breslow-Day test

```
library(DescTools)
```

```
BreslowDayTest(Data.xtabs)
```

Breslow-Day Test for Homogeneity of the Odds Ratios

X-squared = 0.5295, df = 3, p-value = 0.9124

Individual Fisher exact tests

```
n = dim(Data.xtabs)[3]
```

```
for(i in 1:n){
  Name = dimnames(Data.xtabs)[3]$Location[i]
  P.value = fisher.test(Data.xtabs[,i])$p.value
  cat(Name, "\n")
  cat("Fisher test p-value: ", P.value, "\n")
  cat("\n")
}
```

Note: "Location" must be the name of the stratum variable

Tillamook
Fisher test p-value: 0.1145223

Yaquina
Fisher test p-value: 0.2665712

Alsea
Fisher test p-value: 0.4090355

Umpqua
Fisher test p-value: 0.4151874

#

Cochran-Mantel-Haenszel Test with data read by read.ftable

```
### -----
### Niacin example, Cochran-Mantel-Haenszel test, p. 99
### Example using read.ftable
### -----
```

Note no spaces on lines before row names.

```
# read.ftable can be fussy about leading spaces.

Input =(
"          Study FATS AFREGS ARBITER.2 HATS CLAS.1
Supplement Revasc
Niacin      Yes      2      4        1        1      2
            No      46     67       86       37     92
Placebo     Yes     11     12        4        6      1
            No     41     60       76       32     93
")

Tabla = as.table(read.ftable(textConnection(Input)))

ftable(Tabla) # Display a flattened table
```

Cochran-Mantel-Haenszel test

```
mantelhaen.test(Tabla)

Mantel-Haenszel x-squared = 12.7457, df = 1, p-value = 0.0003568
```

Woolf test

```
library(vcd)

oddsratio(Tabla, log=TRUE) # Show log odds for each 2x2

      FATS      AFREGS  ARBITER.2      HATS      CLAS.1
-1.8198174 -1.2089603 -1.5099083 -1.9369415  0.7039581

library(vcd)

woolf_test(Tabla) # Woolf test for homogeneity of
                  # odds ratios across strata.
                  # If significant, C-M-H test
                  # is not appropriate

Woolf-test on Homogeneity of Odds Ratios (no 3-way assoc.)

x-squared = 3.4512, df = 4, p-value = 0.4853
```

Breslow-Day test

```
library(DescTools)

BreslowDayTest(Tabla)

Breslow-Day Test for Homogeneity of the Odds Ratios

x-squared = 4.4517, df = 4, p-value = 0.3483
```

Individual Fisher exact tests

```
n = dim(Tabla)[3]

for(i in 1:n){
  Name = dimnames(Tabla)[3]$Study[i]
  P.value = fisher.test(Tabla[, ,i])$p.value
  cat(Name, "\n")
  cat("Fisher test p-value: ", P.value, "\n")
  cat("\n")
}

### Note: "Study" must be the name of the stratum variable

FATS
Fisher test p-value: 0.01581505

AFREGS
Fisher test p-value: 0.0607213

ARBITER.2
Fisher test p-value: 0.1948915

HATS
Fisher test p-value: 0.1075169

CLAS.1
Fisher test p-value: 1

# # #
```

Graphing the results

Simple bar plot with categories and no error bars

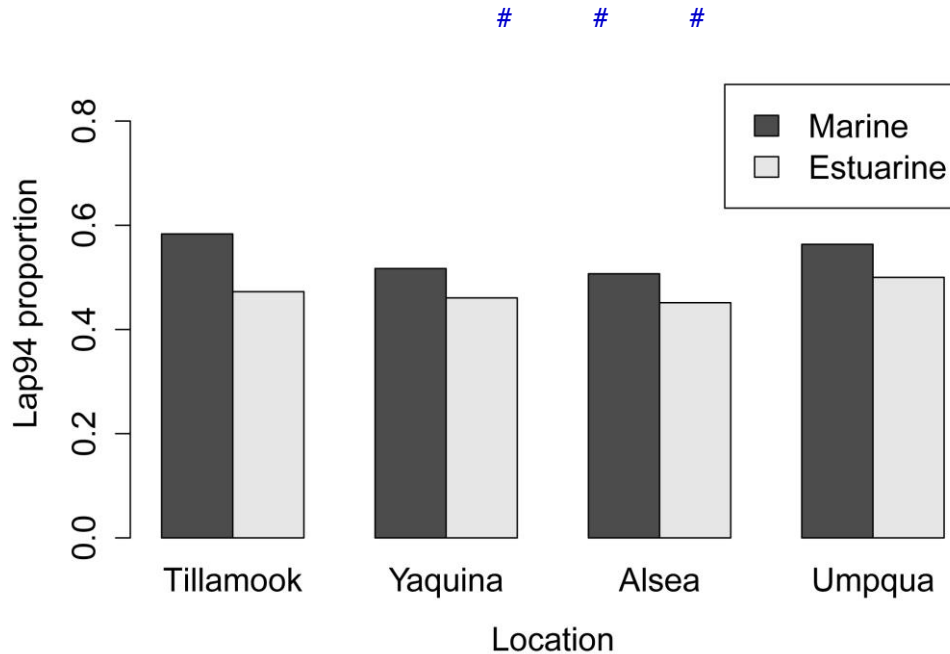
```
### -----
### Simple bar plot of proportions, p. 99
### Uses data in a matrix format
### -----

Input =("
Habitat Tillamook Yaquina Alsea Umpqua
Marine 0.5833 0.5169 0.5069 0.5635
Estuarine 0.4726 0.4606 0.4514 0.5000
")

Matriz = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

Matriz
```

```
barplot(Matriz,
        beside=TRUE,
        legend=TRUE,
        ylim=c(0, 0.9),
        xlab="Location",
        ylab="Lap94 proportion")
```



Bar plot with categories and error bars

This example includes code to calculate the confidence intervals for the error bars and add them to the data frame. This code could be excluded if these values were calculated manually and added to the data frame.

```
#### -----
#### Graph example, bar plot of proportions, p. 99
#### Using ggplot2
#### Plot adapted from:
#### shinyapps.stat.ubc.ca/r-graph-catalog/
#### -----
```

```
Input =("
Location Habitat Allele Count Total Lap.94.Proportion
Tillamook Marine 94 56 96 0.5833
Tillamook Estuarine 94 69 146 0.4726
Yaquina Marine 94 61 118 0.5169
Yaquina Estuarine 94 257 558 0.4606
Alsea Marine 94 73 144 0.5069
Alsea Estuarine 94 65 144 0.4514
Umpqua Marine 94 71 126 0.5635
Umpqua Estuarine 94 48 96 0.5000
")
```

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

### Specify the order of factor levels
### Otherwise, R will alphabetize them

library(dplyr)

Data =
mutate(Data,
  Location = factor(Location, levels=unique(Location)),
  Habitat = factor(Habitat, levels=unique(Habitat)),
  Allele = factor(Allele, levels=unique(Data$ Allele)))

### Add confidence intervals

Fun.low = function (x){
  binom.test(x["Count"], x["Total"],
    0.5)$ conf.int[1]
}

Fun.up = function (x){
  binom.test(x["Count"], x["Total"],
    0.5)$ conf.int[2]
}

Data =
mutate(Data,
  low.ci = apply(Data[c("Count", "Total")], 1, Fun.low),
  upper.ci = apply(Data[c("Count", "Total")], 1, Fun.up))

Data
```

	Location	Habitat	Allele	Count	Total	Lap.94.Proportion	low.ci	upper.ci
1	Tillamook	Marine	94	56	96	0.5833	0.4782322	0.6831506
2	Tillamook	Estuarine	94	69	146	0.4726	0.3894970	0.5568427
3	Yaquina	Marine	94	61	118	0.5169	0.4231343	0.6098931
4	Yaquina	Estuarine	94	257	558	0.4606	0.4186243	0.5029422
5	Alsea	Marine	94	73	144	0.5069	0.4224208	0.5911766
6	Alsea	Estuarine	94	65	144	0.4514	0.3684040	0.5364149
7	Umpqua	Marine	94	71	126	0.5635	0.4723096	0.6516209
8	Umpqua	Estuarine	94	48	96	0.5000	0.3961779	0.6038221

```
### Plot adapted from:
### shinyapps.stat.ubc.ca/r-graph-catalog/

library(ggplot2)
library(grid)

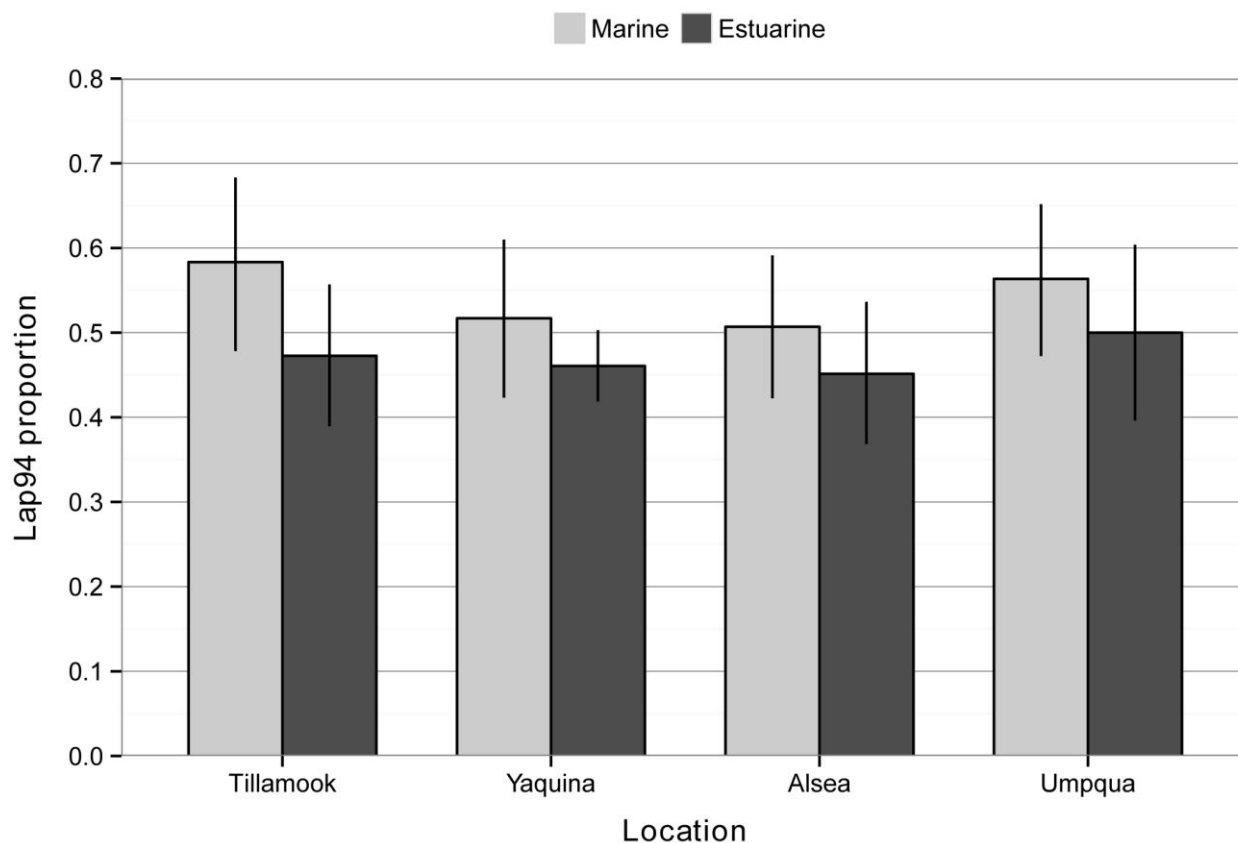
ggplot(Data,
  aes(x = Location, y = Lap.94.Proportion, fill = Habitat,
    ymax=upper.ci, ymin=low.ci)) +
  geom_bar(stat="identity", position = "dodge", width = 0.7) +
```

```

geom_bar(stat="identity", position = "dodge",
         colour = "black", width = 0.7,
         show_guide = FALSE) +
scale_y_continuous(breaks = seq(0, 0.80, 0.1),
                  limits = c(0, 0.80),
                  expand = c(0, 0)) +
scale_fill_manual(name = "Count type" ,
                 values = c('grey80', 'grey30'),
                 labels = c("Marine",
                           "Estuarine")) +
geom_errorbar(position=position_dodge(width=0.7),
              width=0.0, size=0.5, color="black") +
labs(x = "Location",
     y = "Lap94 proportion") +
## ggtitle("Main title") +
theme_bw() +
theme(panel.grid.major.x = element_blank(),
      panel.grid.major.y = element_line(colour = "grey50"),
      plot.title = element_text(size = rel(1.5),
                                face = "bold", vjust = 1.5),
      axis.title = element_text(face = "bold"),
      legend.position = "top",
      legend.title = element_blank(),
      legend.key.size = unit(0.4, "cm"),
      legend.key = element_rect(fill = "black"),
      axis.title.y = element_text(vjust= 1.8),
      axis.title.x = element_text(vjust= -0.5))

# # #

```

Bar plot of proportions vs. categories. Error bars indicate 95% confidence intervals for proportion.

Similar tests

See the *Handbook* for information on this topic.

How to do the test

R code for the SAS example is shown in the “Examples” section above.

Statistics of Central Tendency

Most common statistics of central tendency can be calculated with functions in the native *stats* package. The *psych* and *DescTools* packages add functions for the geometric mean and the harmonic mean. The *describe* function in the *psych* package includes the mean, median, and trimmed mean along with other common statistics. In the native *stats* package, *summary* is a quick way to see the mean, median, and quantiles for numeric variables in a data frame. The mode is not commonly calculated, but can be found in *DescTools*.

Many functions which determine common statistics of central tendency or dispersion will return an *NA* if there are any missing values (NA's) in the analyzed data. In most cases this behavior can be changed with the *na.rm=TRUE* option, which will simply exclude any NA's in the data. The functions shown here either exclude NA's by default or use the *na.rm=TRUE* option.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Descriptive Statistics](#)

Packages used in this chapter

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

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

Introduction

The normal distribution

See the *Handbook* for information on these topics.

Different measures of central tendency

Methods are described in the "Example" section below.

Example

```
### -----
### Central tendency example, pp. 105–106
### -----

Input =("
Stream          Fish
Mill_Creek_1    76
Mill_Creek_2    102
North_Branch_Rock_Creek_1  12
North_Branch_Rock_Creek_2  39
Rock_Creek_1    55
Rock_Creek_2    93
Rock_Creek_3    98
Rock_Creek_4    53
```

```
Turkey_Branch          102  
")
```

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

Arithmetic mean

```
mean(Data$ Fish, na.rm=TRUE)  
  
[1] 70
```

Geometric mean

```
library(psych)  
  
geometric.mean(Data$ Fish)  
  
[1] 59.83515  
  
library(DescTools)  
  
Gmean(Data$ Fish)  
  
[1] 59.83515
```

Harmonic mean

```
library(psych)  
  
harmonic.mean(Data$ Fish)  
  
[1] 45.05709  
  
library(DescTools)  
  
Hmean(Data$ Fish)  
  
[1] 45.05709
```

Median

```
median(Data$ Fish, na.rm=TRUE)  
  
[1] 76
```

Mode

```
library(DescTools)
```

```
Mode(Data$ Fish)
```

```
[1] 102
```

Summary and describe functions for means, medians, and other statistics

The interquartile range (IQR) is $3^{rd} Qu.$ minus $1^{st} Qu.$

```
summary(Data$ Fish)           # Also works on whole data frames
                              # will also report count of NA's
```

```
Min. 1st Qu. Median Mean 3rd Qu. Max.
 12    53    76    70    98    10
```

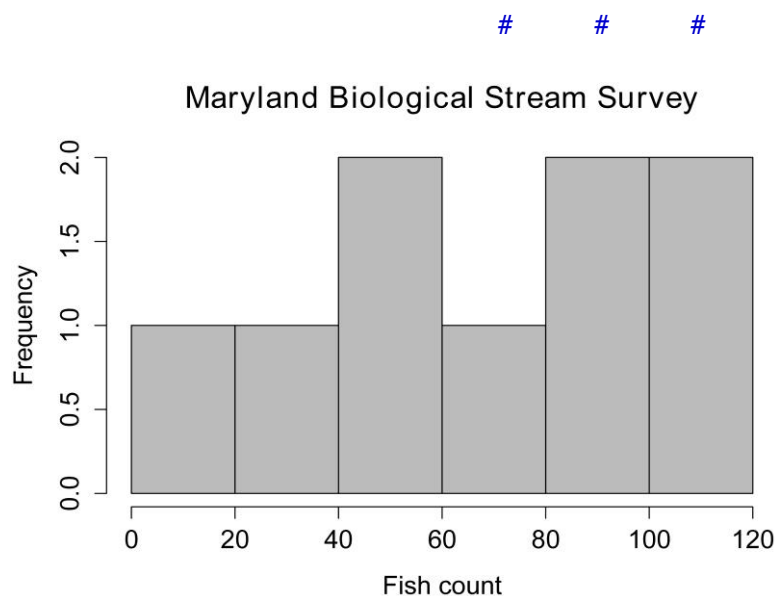
```
library(psych)
```

```
describe(Data$ Fish,         # Also works on whole data frames
           type=2)           # Type of skew and kurtosis
```

```
vars n mean sd median trimmed mad min max range skew kurtosis se
1 1 9 70 32.09 76 70 34.1 12 102 90 -0.65 -0.69 10.7
```

Histogram

```
hist(Data$ Fish,
     col="gray",
     main="Maryland Biological Stream Survey",
     xlab="Fish count")
```



DescTools to produce summary statistics and plots

The *Desc* function in the package *DescTools* produces summary information for individual variables or whole data frames. It has custom output for factor, numeric, integer, and date variables.

```
### -----
### Central tendency example, pp. 105–106
### -----
```

```
Input =("
Stream          Fish
Mill_Creek_1    76
Mill_Creek_2    102
North_Branch_Rock_Creek_1  12
North_Branch_Rock_Creek_2  39
Rock_Creek_1     55
Rock_Creek_2     93
Rock_Creek_3     98
Rock_Creek_4     53
Turkey_Branch   102
")
```

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

```
### Add a numeric variable with the same values as Fish
```

```
Data$Fish.num = as.numeric(Data$Fish)
```

```
### Produce summary statistics and plots
```

```
library(DescTools)
```

```
Desc(Data,
      plotit=TRUE)
```

```
-----
1 - Stream (factor)
```

	length	n	NAs	levels	unique	dupes		
	9	9	0	9	9	n		
				level	freq	perc	cumfreq	cumperc
1				Mill_Creek_1	1	.111	1	.111
2				Mill_Creek_2	1	.111	2	.222
3				North_Branch_Rock_Creek_1	1	.111	3	.333
4				North_Branch_Rock_Creek_2	1	.111	4	.444
5				Rock_Creek_1	1	.111	5	.556
6				Rock_Creek_2	1	.111	6	.667
7				Rock_Creek_3	1	.111	7	.778
8				Rock_Creek_4	1	.111	8	.889
9				Turkey_Branch	1	.111	9	1.000

```
-----
```

.
. < results snipped >
.

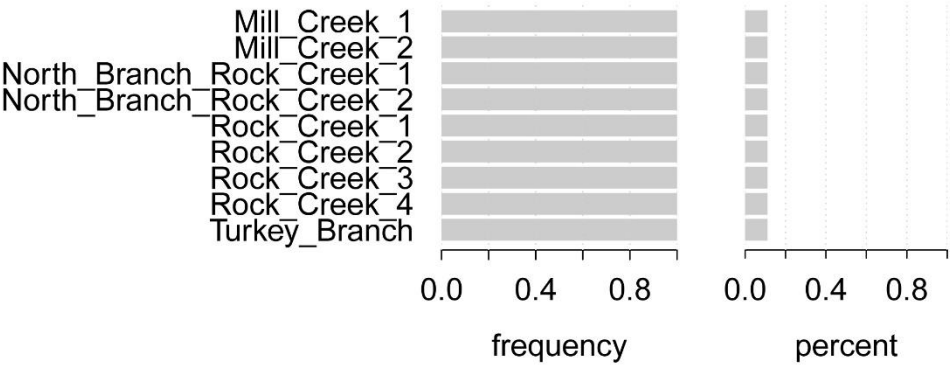
3 - Fish.num (numeric)

length	n	NAs	unique	0s	mean	meanSE
9	9	0	8	0	70	10.695
.05	.10	.25	median	.75	.90	.95
22.800	33.600	53	76	98	102	102
rng	sd	vcoef	mad	IQR	skew	kurt
90	32.086	0.458	34.100	45	-0.448	-1.389

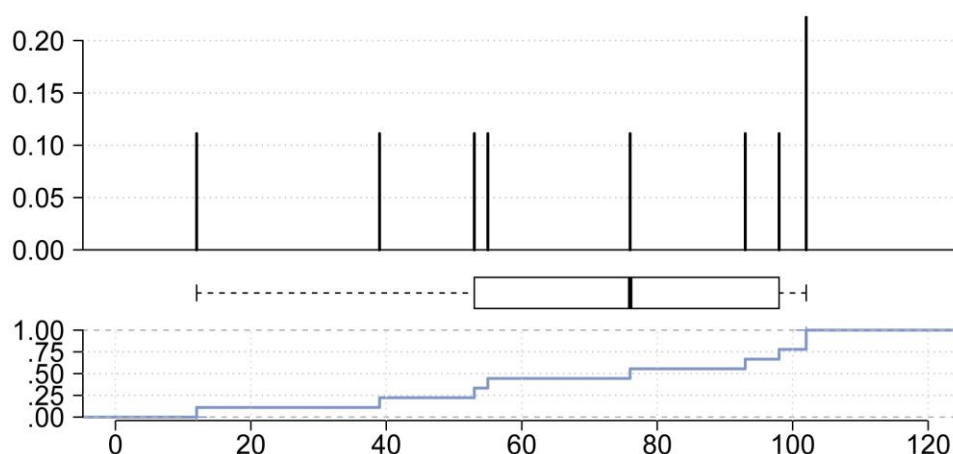
lowest : 12, 39, 53, 55, 76
highest: 55, 76, 93, 98, 102 (2)

Shapiro-wilks normality test p.value : 0.23393

1 - Stream (factor)



3 – Fish.num (numeric)

**DescTools with grouped data**

```
### -----
### Summary statistics with grouped data, hypothetical data
### -----
```

```
Input =("
Stream          Animal  Count
Mill_Creek_1    Fish    76
Mill_Creek_2    Fish   102
North_Branch_Rock_Creek_1 Fish   12
North_Branch_Rock_Creek_2 Fish   39
Rock_Creek_1    Fish    55
Rock_Creek_2    Fish    93
Rock_Creek_3    Fish    98
Rock_Creek_4    Fish    53
Turkey_Branch   Fish   102
```

```
Mill_Creek_1    Insect   28
Mill_Creek_2    Insect   85
North_Branch_Rock_Creek_1 Insect  17
North_Branch_Rock_Creek_2 Insect  20
Rock_Creek_1    Insect   33
Rock_Creek_2    Insect   75
Rock_Creek_3    Insect   78
Rock_Creek_4    Insect   25
Turkey_Branch   Insect   87
")
```

```
D2 = read.table(textConnection(Input),header=TRUE)
```

```
library(DescTools)
```

```
Desc(Count ~ Animal,
     D2,
```

```
digits=1,  
plotit=TRUE)
```

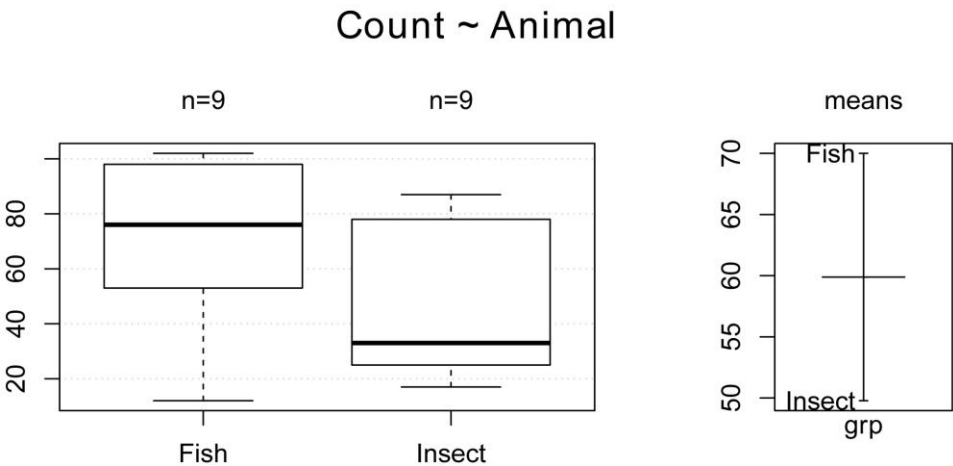
Count ~ Animal

Summary:
n pairs: 18, valid: 18 (100%), missings: 0 (0%), groups: 2

	Fish	Insect
mean	70.0"	49.8'
median	76.0"	33.0'
sd	32.1	30.4
IQR	45.0	53.0
n	9	9
np	0.500	0.500
NAs	0	0
Os	0	0

' min, " max

Kruskal-wallis rank sum test:
Kruskal-wallis chi-squared = 2.125, df = 1, p-value = 0.1449



How to calculate the statistics
Methods are described in the “Example” section above.

Statistics of Dispersion

Measures of dispersion—such as range, variance, standard deviation, and coefficient of variation—can be calculated with standard functions in the native *stats* package. In addition, a function, here called *summary.list*, can be defined to output whichever statistics are of interest.

Introduction

See the *Handbook* for information on this topic.

Example

Statistics of dispersion example

```
### -----
### Statistics of dispersion example, p. 111
### -----

Input =("
Stream                Fish
Mill_Creek_1          76
Mill_Creek_2          102
North_Branch_Rock_Creek_1  12
North_Branch_Rock_Creek_2  39
Rock_Creek_1          55
Rock_Creek_2          93
Rock_Creek_3          98
Rock_Creek_4          53
Turkey_Branch        102
")

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

Range

```
range(Data$ Fish, na.rm=TRUE)

[1] 12 102      # Min and max

max(Data$ Fish, na.rm=TRUE) - min(Data$ Fish, na.rm=TRUE)

[1] 90
```

Sum of squares

Not included here.

Parametric variance

Not included here.

Sample variance

```
var(Data$ Fish, na.rm=TRUE)

[1] 1029.5
```

Standard deviation

```
sd(Data$ Fish, na.rm=TRUE)
```

```
[1] 32.08582
```

Coefficient of variation, as percent

```
sd(Data$ Fish, na.rm=TRUE)/  
mean(Data$ Fish, na.rm=TRUE)*100
```

```
[1] 45.83689
```

Custom function of desired measures of central tendency and dispersion

```
### Note NA's removed in the following function
```

```
summary.list = function(x)list(  
  N.with.NA.removed= length(x[!is.na(x)]),  
  Count.of.NA= length(x[is.na(x)]),  
  Mean=mean(x, na.rm=TRUE),  
  Median=median(x, na.rm=TRUE),  
  Max.Min=range(x, na.rm=TRUE),  
  Range=max(Data$ Fish, na.rm=TRUE) - min(Data$ Fish, na.rm=TRUE),  
  Variance=var(x, na.rm=TRUE),  
  Std.Dev=sd(x, na.rm=TRUE),  
  Coeff.Variation.Prcnt=sd(x, na.rm=TRUE)/mean(x, na.rm=TRUE)*100,  
  Std.Error=sd(x, na.rm=TRUE)/sqrt(length(x[!is.na(x)])),  
  Quantile=quantile(x, na.rm=TRUE)  
)
```

```
summary.list(Data$ Fish)
```

```
$N.with.NA.removed
```

```
[1] 9
```

```
$Count.of.NA
```

```
[1] 0
```

```
$Mean
```

```
[1] 70
```

```
$Median
```

```
[1] 76
```

```
$Range
```

```
[1] 12 102
```

```
$Variance
```

```
[1] 1029.5
```

```
$Std.Dev
```

```
[1] 32.08582

$Coeff.Variation.Prcnt
[1] 45.83689

$Std.Error
[1] 10.69527

$Quantile
 0%  25%  50%  75% 100%
12   53   76   98  102
```

```
#      #      #
```

How to calculate the statistics

Methods are described in the “Example” section above.

Standard Error of the Mean

The standard error of the mean can be calculated with standard functions in the native *stats* package. The *describe* function in the *psych* package includes the standard error of the mean along with other descriptive statistics. This function is useful to summarize multiple variables in a data frame.

Introduction

Similar statistics

See the *Handbook* for information on these topics.

Example

Standard error example

```
### -----
### Standard error example, p. 115
### -----

Input =("
Stream          Fish
Mill_Creek_1    76
Mill_Creek_2   102
North_Branch_Rock_Creek_1 12
North_Branch_Rock_Creek_2 39
Rock_Creek_1    55
Rock_Creek_2    93
Rock_Creek_3    98
Rock_Creek_4    53
Turkey_Branch  102
")
```

```

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

### Calculate standard error manually

sd(Data$ Fish, na.rm=TRUE) /
  sqrt(length(Data$Fish[!is.na(Data$ Fish)]))      # Standard error

[1] 10.69527

### Use describe function from psych package for standard error
### Also works on whole data frames

library(psych)

describe(Data$ Fish,
          type=2)      # Type of skew and kurtosis

      vars n mean      sd median trimmed  mad min max range  skew kurtosis   se
1      1  9   70 32.09      76      70 34.1  12 102    90 -0.65   -0.69 10.7

      #      #      #

```

How to calculate the standard error

Methods are described in the “Example” section above.

Confidence Limits

Introduction

See the [Handbook](#) for information on this topic.

Confidence limits for measurement variables

Methods are described in the “How to calculate confidence limits” section below.

Confidence limits for nominal variables

Examples are given in the “How to calculate confidence limits” section below.

Statistical testing with confidence intervals

Similar statistics

Examples

See the *Handbook* for information on these topics.

How to calculate confidence limits

The confidence limits about the mean—calculated using the t -value discussed in the *Handbook*—can be determined with variety of functions. One is *t.test* in the native *stats* package. Another is the *CI* function in the *Rmisc* package, which also has the function *summarySE* that presents the mean, standard deviation, standard error, and confidence interval for data designated as groups.

The bootstrap method noted in the *Handbook* can be achieved with the *boot* and *boot.ci* functions in the *boot* package.

Confidence intervals for mean with *t.test*, *Rmisc*, and *DescTools*

```
### -----
### Confidence interval for measurement data, blacknose fish , p. 120
### -----

Input =("
Stream          Fish
Mill_Creek_1    76
Mill_Creek_2    102
North_Branch_Rock_Creek_1  12
North_Branch_Rock_Creek_2  39
Rock_Creek_1     55
Rock_Creek_2     93
Rock_Creek_3     98
Rock_Creek_4     53
Turkey_Branch   102
")

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

### Use t.test to produce confidence interval

t.test(Data$ Fish,
       conf.level=0.95)           # Confidence interval of the mean

95 percent confidence interval:
45.33665 94.66335

### Use CI in Rmisc package to produce confidence interval

library(Rmisc)

CI(Data$ Fish,
   ci=0.95)                       # Confidence interval of the mean

      upper      mean      lower
94.66335 70.00000 45.33665

### Use MeanCI in DescTools package to produce confidence interval
```

```
library(DescTools)

MeanCI(Data$ Fish,
       conf.level=0.95)      # Confidence interval of the mean

      mean    lwr.ci    upr.ci
70.00000 45.33665 94.66335

      #      #      #
```

Confidence intervals for means for grouped data

```
### -----
### Confidence interval for grouped data, hypothetical data
### -----

Input =("
Stream                Animal  Count
Mill_Creek_1          Fish    76
Mill_Creek_2          Fish    102
North_Branch_Rock_Creek_1 Fish    12
North_Branch_Rock_Creek_2 Fish    39
Rock_Creek_1          Fish    55
Rock_Creek_2          Fish    93
Rock_Creek_3          Fish    98
Rock_Creek_4          Fish    53
Turkey_Branch         Fish    102

Mill_Creek_1          Insect   76
Mill_Creek_2          Insect  102
North_Branch_Rock_Creek_1 Insect   12
North_Branch_Rock_Creek_2 Insect   39
")

D2 = read.table(textConnection(Input),header=TRUE)

library(Rmisc)

summarySE(data=D2,                # will produce confidence intervals
          measurevar="Count",     # for groups defined by a variable
          groupvars="Animal",
          conf.interval = 0.95)

      Animal N Count      sd      se      ci
1   Fish 9 70.00 32.08582 10.69527 24.66335
2  Insect 4 57.25 39.72719 19.86360 63.21483

      #      #      #
```

Confidence intervals for mean by bootstrap

```
### -----
### Confidence interval for measurement data, blacknose fish , p. 120
### -----
```

```
Input =("
Stream          Fish
Mill_Creek_1    76
Mill_Creek_2    102
North_Branch_Rock_Creek_1  12
North_Branch_Rock_Creek_2  39
Rock_Creek_1    55
Rock_Creek_2    93
Rock_Creek_3    98
Rock_Creek_4    53
Turkey_Branch  102
")

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

Confidence intervals for mean by bootstrap with *DescTools*

```
MeanCI(Data$Fish, method="boot", type="norm", R=10000)

      mean   lwr.ci   upr.ci
70.00000  50.17986  89.84836

# May be different for different iterations

MeanCI(Data$Fish, method="boot", type="basic", R=10000)

      mean   lwr.ci   upr.ci
70.00000  51.44444  90.66667

# May be different for different iterations
```

Confidence intervals for mean by bootstrap with *boot* package

```
library(boot)

Fun = function(x, index) {
  return(c(mean(x[index]),
            var(x[index]) / length(index)))
}

Boot = boot(data=Data$Fish,
            statistic=Fun,
            R=10000)

mean(Boot$t[,1])

[1] 70.01229      # Mean by bootstrap
                  # May be different for different iterations
```

```
boot.ci(Boot,
        conf=0.95)

Intervals :
Level      Normal          Basic          Studentized
95%   (50.22, 89.76 )   (51.11, 90.44 )   (38.85, 91.72 )

Level      Percentile          BCa
95%   (49.56, 88.89 )   (47.44, 87.22 )
Calculations and Intervals on Original Scale

# Note that the bootstrapped confidence limits vary from
# the calculated ones above because the original data set has
# few values and is not necessarily normally distributed.

#      #      #
```

Confidence interval for proportions

The confidence interval for a proportion can be determined with the *binom.test* function, and more options are available in the *BinomCI* function and *MultinomCI* function in the *DescTools* package. More advanced techniques for confidence intervals on proportions and differences in proportions can be found in the *PropCIs* package.

```
#### -----
### Confidence interval for nominal data, colorblind example, p. 118
### -----

binom.test(2, 20, 0.5,
           alternative="two.sided",
           conf.level=0.95)

95 percent confidence interval:
0.01234853 0.31698271

#      #      #

#### -----
### Confidence interval for nominal data, Gus data, p. 121
### -----

Input =("
Paw
right
left
right
right
right
right
right
left
right
right
```



```

right
")

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

Successes = sum(Gus$ Paw == "left")      # Note the == operator
Failures  = sum(Gus$ Paw == "right")

Total = Successes + Failures

Expected = 0.5

binom.test(Successes, Total, Expected,
            alternative="two.sided",
            conf.level=0.95)

95 percent confidence interval:
0.02521073 0.55609546

### Agrees with exact confidence interval from SAS

# # #

```

Confidence interval for proportions using DescTools

Confidence interval for single proportion

```

### -----
### Confidence intervals for nominal data, colorblind example, p. 118
### -----

library(DescTools)

BinomCI(2, 20,
        conf.level = 0.95,
        method = "modified wilson")

### Other methods: "wilson", "wald", "agresti-coull", "jeffreys",
### "modified wilson", "modified jeffreys",
### "clopper-pearson", "arcsine", "logit", "witting"

      est      lwr.ci    upr.ci
[1,] 0.1 0.01776808 0.3010336

# # #

```

Confidence interval for multinomial proportion

```

### -----
### Confidence intervals for multinomial proportions, p. 33
### -----

observed = c(35,74,22,69)

```

```
library(DescTools)

MultinomCI(observed, conf.level=0.95, method="goodman")

### Other methods: "sisonglaz", "cplus1"

      est      lwr.ci      upr.ci
[1,] 0.175 0.11253215 0.2619106
[2,] 0.370 0.28113643 0.4686407
[3,] 0.110 0.06224338 0.1870880
[4,] 0.345 0.25846198 0.4431954

      #      #      #
```

Tests for One Measurement Variable

Student's t -test for One Sample

Introduction

When to use it

Null hypothesis

How the test works

Assumptions

See the *Handbook* for information on these topics.

Example

One sample t-test with observations as vector

```
### -----
### One-sample t-test, transferrin example, pp. 124
### -----

observed = c(0.52, 0.20, 0.59, 0.62, 0.60)
theoretical = 0

t.test(observed,
       mu = theoretical,
       conf.int = 0.95)

One Sample t-test

t = 6.4596, df = 4, p-value = 0.002958

      #      #      #
```

Graphing the results

See the *Handbook* for information on this topic.

Similar tests

The *paired t-test* and *two-sample t-test* are presented elsewhere in this book.

How to do the test

One sample t-test with observations in data frame

```
### -----
### One-sample t-test, SAS example, pp. 125
### -----

Input =("
Angle
120.6
116.4
117.2
118.1
114.1
116.9
113.3
121.1
116.9
117.0
")

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

observed    = Data$ Angle
theoretical = 50

t.test(observed,
       mu = theoretical,
       conf.int=0.95)

One Sample t-test

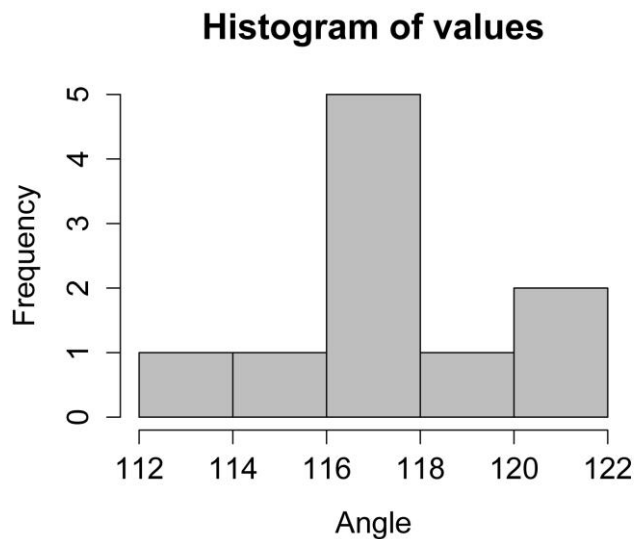
t = 87.3166, df = 9, p-value = 1.718e-14

### Does not agree with Handbook. The Handbook results are incorrect.
### The SAS code produces the following result.
```

	T-Tests		
Variable	DF	t Value	Pr > t
angle	9	87.32	<.0001

Histogram

```
hist(Data$ Angle,
     col="gray",
     main="Histogram of values",
     xlab="Angle")
```



Histogram of data in a single population from a one-sample t-test. Distribution of these values should be approximately normal.

#

Power analysis

Power analysis for one-sample t-test

```
### -----
### Power analysis, t-test, one-sample,
### hip joint example, pp. 125-126
### -----
```

```
M1 = 70          # Theoretical mean
M2 = 71          # Mean to detect
S1 = 2.4         # Standard deviation
S2 = 2.4         # Standard deviation
```

```
cohen.d = (M1 - M2)/sqrt(((S1^2) + (S2^2))/2)
```

```
library(pwr)
```

```
pwr.t.test(
  n = NULL,          # Observations
  d = cohen.d,
  sig.level = 0.05,  # Type I probability
  power = 0.90,      # 1 minus Type II probability
  type = "one.sample", # Change for one- or two-sample
  alternative = "two.sided")
```

One-sample t test power calculation

```
n = 62.47518
```

#

Student’s t-test for Two Samples

Introduction

When to use it

Null hypothesis

How the test works

Assumptions

See the *Handbook* for information on these topics.

Example

Two-sample t-test, independent (unpaired) observations

```
### -----  
### Two-sample t-test, biological data analysis class, pp. 128–129  
### -----
```

```
Input =(  
Group Value  
2pm      69  
2pm      70  
2pm      66  
2pm      63  
2pm      68  
2pm      70  
2pm      69  
2pm      67  
2pm      62  
2pm      63  
2pm      76  
2pm      59  
2pm      62  
2pm      62  
2pm      75  
2pm      62  
2pm      72  
2pm      63  
5pm      68  
5pm      62  
5pm      67  
5pm      68  
5pm      69  
5pm      67  
5pm      61  
5pm      59  
5pm      62  
5pm      61  
5pm      69  
5pm      66  
5pm      62
```

```
5pm    62
5pm    61
5pm    70
")
```

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

```
bartlett.test(Value ~ Group, data=Data)
```

```
### If p-value >= 0.05, use var.equal=TRUE below
```

```
Bartlett's K-squared = 1.2465, df = 1, p-value = 0.2642
```

```
t.test(Value ~ Group, data=Data,
       var.equal=TRUE,
       conf.level=0.95)
```

```
Two Sample t-test
```

```
t = 1.2888, df = 32, p-value = 0.2067
```

```
t.test(Value ~ Group, data=Data,
       var.equal=FALSE,
       conf.level=0.95)
```

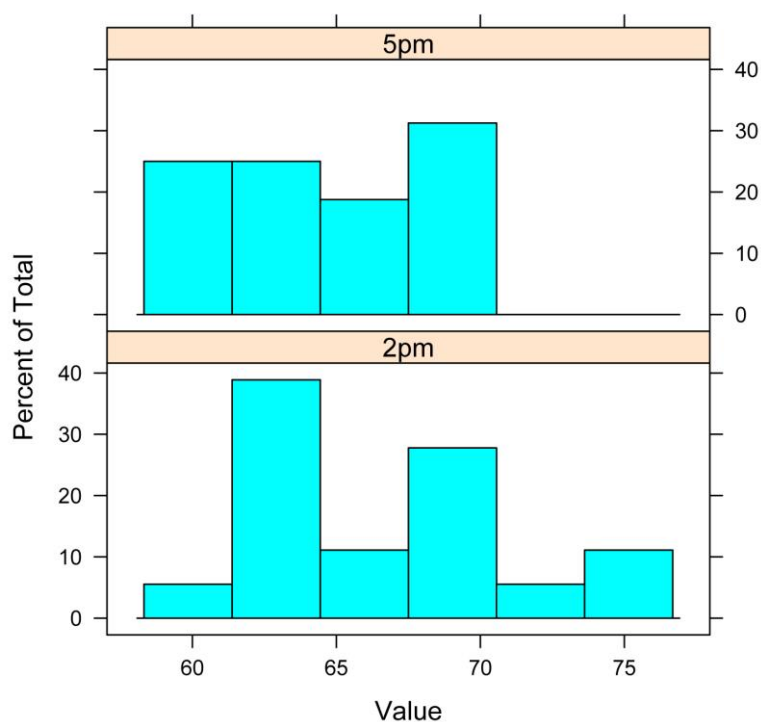
```
Welch Two Sample t-test
```

```
t = 1.3109, df = 31.175, p-value = 0.1995
```

Plot of histograms

```
library(lattice)
```

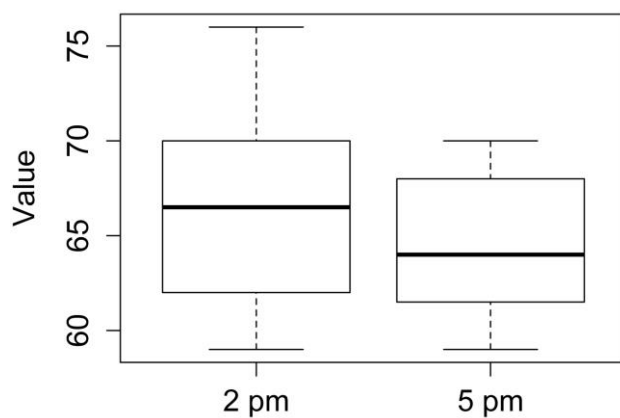
```
histogram(~ Value | Group,
         data=Data,
         layout=c(1,2))    # columns and rows of individual plots
```



Histograms for each population in a two-sample t-test. For the t-test to be valid, the data in each population should be approximately normal. If the distributions are different, minimally Welch's t-test should be used. If the data are not normal or the distributions are different, a non-parametric test like Mann-Whitney U-test or permutation test may be appropriate.

Box plots

```
boxplot(Value ~ Group,
  data = Data,
  names=c("2 pm", "5 pm"),
  ylab="Value")
```



Box plots of two populations from a two-sample t-test.

#

Similar tests

Welch's t-test is discussed below. The *paired t-test* and *signed-rank test* are discussed in this book in their own chapters. *Analysis of variance* (anova) is discussed in several subsequent chapters.

As non-parametric alternatives, the *Mann-Whitney U-test* and the *permutation test* for two independent samples are discussed in the chapter *Mann-Whitney and Two-sample Permutation Test*.

Welch's t-test

Welch's t-test is shown above in the "Example" section ("Two sample unpaired t-test"). It is invoked with the *var.equal=FALSE* option in the *t.test* function.

How to do the test

The SAS example from the *Handbook* is shown above in the "Example" section.

Power analysis

Power analysis for t-test

```
### -----
### Power analysis, t-test, wide feet, p. 131
### -----

M1 = 100.6      # Mean for sample 1
M2 = 103.6      # Mean for sample 2
S1 = 5.26       # Std dev for sample 1
S2 = 5.26       # Std dev for sample 2

Cohen.d = (M1 - M2)/sqrt(((S1^2) + (S2^2))/2)

library(pwr)

pwr.t.test(
  n = NULL,      # Observations in _each_ group
  d = Cohen.d,
  sig.level = 0.05, # Type I probability
  power = 0.90,    # 1 minus Type II probability
  type = "two.sample", # Change for one- or two-sample
  alternative = "two.sided")

Two-sample t test power calculation

      n = 65.57875      # Number for each group
# # #
```


Mann–Whitney and Two-sample Permutation Test

The Mann–Whitney U-test is a nonparametric test, also called the Mann–Whitney–Wilcoxon test. It tests for a difference in central tendency of two groups, or, with certain assumptions, for the difference in medians. It is conducted with the *wilcox.test* function in the native *stats* package. It can be used with continuous or ordinal measurements.

As another non-parametric alternative to t-tests, a permutation test can be used. An example is shown in the “Permutation test for independent samples” section of this chapter.

Mann–Whitney U-test

```
### -----
### Mann-Whitney U-test, biological data analysis class, pp. 128–129
### -----

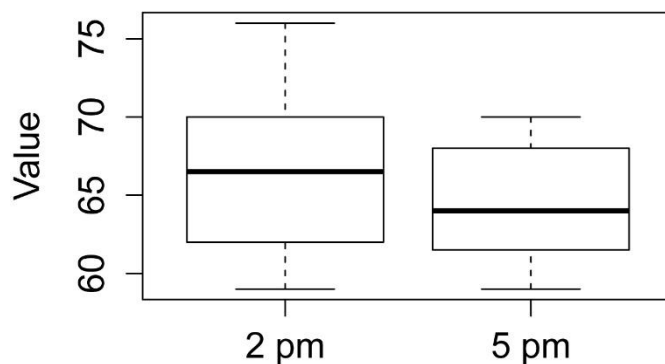
Input =(
Group value
2pm      69
2pm      70
2pm      66
2pm      63
2pm      68
2pm      70
2pm      69
2pm      67
2pm      62
2pm      63
2pm      76
2pm      59
2pm      62
2pm      62
2pm      75
2pm      62
2pm      72
2pm      63
5pm      68
5pm      62
5pm      67
5pm      68
5pm      69
5pm      67
5pm      61
5pm      59
5pm      62
5pm      61
5pm      69
5pm      66
5pm      62
```

```
5pm 62
5pm 61
5pm 70
")
```

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

Box plots

```
boxplot(Value ~ Group,
        data = Data,
        names=c("2 pm", "5 pm"),
        ylab="Value")
```



```
wilcox.test(Value ~ Group, data=Data)
```

```
Wilcoxon rank sum test with continuity correction
```

```
w = 186, p-value = 0.1485
```

```
# # #
```

Permutation test for independent samples

Permutation tests are nonparametric tests, and can be performed with the *coin* package. The permutation test compares values across groups, and can also be used to compare ranks or counts. This test is analogous to a nonparametric t-test. Normality is not assumed but the test may require that distributions have similar variance or shape to be interpreted as a test of means.

```
### -----
### Two-sample permutation test, biological data analysis class,
### pp. 128-129
### -----
```

```
Input =("
Group Value
2pm 69
2pm 70
```

```
2pm 66
2pm 63
2pm 68
2pm 70
2pm 69
2pm 67
2pm 62
2pm 63
2pm 76
2pm 59
2pm 62
2pm 62
2pm 75
2pm 62
2pm 72
2pm 63
5pm 68
5pm 62
5pm 67
5pm 68
5pm 69
5pm 67
5pm 61
5pm 59
5pm 62
5pm 61
5pm 69
5pm 66
5pm 62
5pm 62
5pm 61
5pm 70
")

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

library(coin)

independence_test(Value ~ Group,
                  data = Data)

Asymptotic General Independence Test

Z = 1.2761, p-value = 0.2019

# # #
```

Chapters Not Covered in This Book

Introduction
Step-by-step analysis of biological data

Types of biological variables**Probability****Basic concepts of hypothesis testing****Confounding variables****Independence****Normality****Data transformations**

See the *Handbook* for information on these topics.

Homoscedasticity and heteroscedasticity

Bartlett's test is performed with the *bartlett.test* function. Levene's test can be invoked with the *leveneTest* function in the *car* package. This test can also be used for a model with two independent variables. They are used in the chapter on *One-way anova*.

Type I, II, and III Sums of Squares

An in-depth discussion of Type I, II, and III sum of squares is beyond the scope of this book, but readers should at least be aware of them. They come into play in analysis of variance (anova) tables, when calculating sum of squares, F-values, and p-values.

Perhaps most salient point for beginners is that SAS tends to use Type III by default whereas R will use Type I with the *anova* function. In R, Type II and Type III tests are accessed through *Anova* in the *car* package, as well as through some other functions for other types of analyses. However, for Type III tests to be correct, the way R codes factors has to be changed from its default with the *options(contrasts = ...)* function. Changing this will not affect Type I or Type II tests.

```
options(contrasts = c("contr.sum", "contr.poly"))

### needed for type III tests

### Default is: options(contrasts = c("contr.treatment", "contr.poly"))
```

Type I sum of squares are “sequential.” In essence the factors are tested in the order they are listed in the model. Type III are “partial.” In essence, every term in the model is tested in light of every other term in the model. That means that main effects are tested in light of interaction terms as well as in light of other main effects. Type II are similar to Type III, except that they preserve the principle of marginality. This means that main factors are tested in light of one another, but not in light of the interaction term.

When data are balanced and the design is simple, types I, II, and III will give the same results. But readers should be aware that results will differ for unbalanced data or more complex designs. The code below gives an example of this.

There are disagreements as to which type should be used routinely in analysis of variance. In reality, the user should understand what hypothesis she wants to test, and then choose the appropriate tests. As general advice, I would recommend not using Type I except in cases where you intend to have the effects assessed sequentially. Beyond that, probably a majority of those in the R community recommend Type II tests, while SAS users are more likely to consider Type III tests.

Some experimental designs will call for using a specified type of sum of squares, for example when you see “/SS1” or “HTYPE=1” in SAS code.

A couple of online resources may provide some more clarity:

Falk Scholer. ANOVA (and R). goanna.cs.rmit.edu.au/~fscholer/anova.php.

Daniel Wollschläger. Sum of Squares Type I, II, III: the underlying hypotheses, model comparisons, and their calculation in R. www.uni-kiel.de/psychologie/dwoll/r/ssTypes.php.

As a final note, readers should not confuse these sums of squares with “Type I error”, which refers to rejecting a null hypothesis when it is actually true (a false positive), and “Type II error”, which is failing to reject null hypothesis when it actually false (a false negative).

```
### -----
### Example of different results for Type I, II, III SS
### -----

options(contrasts = c("contr.sum", "contr.poly"))

### needed for type III tests

A      = c("a", "a", "a", "a", "b", "b", "b", "b", "b", "b", "b", "b")
B      = c("x", "y", "x", "y", "x", "y", "x", "y", "x", "x", "x", "x")
C      = c("l", "l", "m", "m", "l", "l", "m", "m", "l", "l", "l", "l")
response = c( 14,  30,  15,  35,  50,  51,  30,  32,  51,  55,  53,  55)

model = lm(response ~ A + B + C + A:B + A:C + B:C)

anova(model)           # Type I tests

library(car)

Anova(model, type="II") # Type II tests

Anova(model, type="III") # Type III tests

#      #      #
```

Effects and p -values from a hypothetical linear model. While in this example the p -values are relatively similar, the B effect would not

be significant with Type I sum of squares at the $\alpha = 0.05$ level, while it would be with Type II or Type III tests.

Effect	Type I p -value	Type II p -value	Type III p -value
A	< 0.0001	< 0.0001	< 0.0001
B	0.09	0.002	0.001
C	0.0002	0.0004	0.001
A:B	0.0004	0.001	0.001
A:C	0.0003	0.0003	0.0003
B:C	0.2	0.2	0.2

One-way Anova

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Introduction to Parametric Tests](#)

[SAEPPER: One-way ANOVA](#)

[SAEPPER: What are Least Square Means?](#)

Packages used in this chapter

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

```
if(!require(dplyr)){install.packages("dplyr")}
if(!require(FSA)){install.packages("FSA")}
if(!require(car)){install.packages("car")}
if(!require(agricolae)){install.packages("agricolae")}
if(!require(multcomp)){install.packages("multcomp")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(lsmmeans)){install.packages("lsmmeans")}
if(!require(multcompView)){install.packages("multcompView")}
if(!require(Rmisc)){install.packages("Rmisc")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(pwr)){install.packages("pwr")}
```

When to use it

Analysis for this example is described below in the “How to do the test” section below.

Null hypothesis

How the test works

Assumptions

Additional analyses

See the [Handbook](#) for information on these topics.

Tukey-Kramer test

The Tukey mean separation tests and others are shown below in the “How to do the test” section.

Partitioning variance

This topic is not covered here.

Example

Code for this example is not included here. An example is covered below in the “How to do the test” section.

Graphing the results

Graphing of the results is shown below in the “How to do the test” section.

Similar tests

Two-sample t-test, *Two-way anova*, *Nested anova*, *Welch’s anova*, and *Kruskal–Wallis* are presented elsewhere in this book.

A *permutation test*, presented in the *One-way Analysis with Permutation Test* chapter, can also be employed as a nonparametric alternative.

How to do the test

The *lm* function in the native *stats* package fits a linear model by least squares, and can be used for a variety of analyses such as regression, analysis of variance, and analysis of covariance. The analysis of variance is then conducted either with the *Anova* function in the *car* package for Type II or Type III sum of squares, or with the *anova* function in the native *stats* package for Type I sum of squares.

If the analysis of variance indicates a significant effect of the independent variable, multiple comparisons among the levels of this factor can be conducted using Tukey or Least Significant Difference (LSD) procedures. The problem of inflating the Type I Error Rate when making multiple comparisons is discussed in the *Multiple Comparisons* chapter in the *Handbook*. R functions which make multiple comparisons usually allow for adjusting p-values. In R, the “BH”, or “fdr”, procedure is the Benjamini–Hochberg procedure discussed in the *Handbook*. See *?p.adjust* for more information.

One-way anova example

```
### -----
### One-way anova, SAS example, pp. 155-156
### -----

Input =(
Location  Aam
Tillamook 0.0571
Tillamook 0.0813
Tillamook 0.0831
Tillamook 0.0976
Tillamook 0.0817
Tillamook 0.0859
Tillamook 0.0735
```

```

Tillamook 0.0659
Tillamook 0.0923
Tillamook 0.0836
Newport 0.0873
Newport 0.0662
Newport 0.0672
Newport 0.0819
Newport 0.0749
Newport 0.0649
Newport 0.0835
Newport 0.0725
Petersburg 0.0974
Petersburg 0.1352
Petersburg 0.0817
Petersburg 0.1016
Petersburg 0.0968
Petersburg 0.1064
Petersburg 0.1050
Magadan 0.1033
Magadan 0.0915
Magadan 0.0781
Magadan 0.0685
Magadan 0.0677
Magadan 0.0697
Magadan 0.0764
Magadan 0.0689
Tvarminne 0.0703
Tvarminne 0.1026
Tvarminne 0.0956
Tvarminne 0.0973
Tvarminne 0.1039
Tvarminne 0.1045
")

```

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

Specify the order of factor levels for plots and Dunnett comparison

```
library(dplyr)
```

```

Data =
mutate(Data,
        Location = factor(Location, levels=unique(Location)))

```

Produce summary statistics

```
library(FSA)
```

```

Summarize(Aam ~ Location,
          data=Data,
          digits=3)

```

	Location	n	mean	sd	min	Q1	median	Q3	max
1	Tillamook	10	0.080	0.012	0.057	0.075	0.082	0.085	0.098
2	Newport	8	0.075	0.009	0.065	0.067	0.074	0.082	0.087
3	Petersburg	7	0.103	0.016	0.082	0.097	0.102	0.106	0.135
4	Magadan	8	0.078	0.013	0.068	0.069	0.073	0.081	0.103
5	Tvarminne	6	0.096	0.013	0.070	0.096	0.100	0.104	0.104

Fit the linear model and conduct ANOVA

```

model = lm(Aam ~ Location,
           data=Data)

library(car)

Anova(model, type="II")                                # Can use type="III"

### If you use type="III", you need the following line before the analysis
### options(contrasts = c("contr.sum", "contr.poly"))

              Sum Sq Df F value    Pr(>F)
Location    0.0045197  4    7.121 0.0002812 ***
Residuals   0.0053949 34

anova(model)                                           # Produces type I sum of squares

              Df    Sum Sq   Mean Sq F value    Pr(>F)
Location      4 0.0045197 0.00112992    7.121 0.0002812 ***
Residuals    34 0.0053949 0.00015867

summary(model)    # Produces r-square, overall p-value, parameter estimates

Multiple R-squared:  0.4559, Adjusted R-squared:  0.3918
F-statistic: 7.121 on 4 and 34 DF,  p-value: 0.0002812

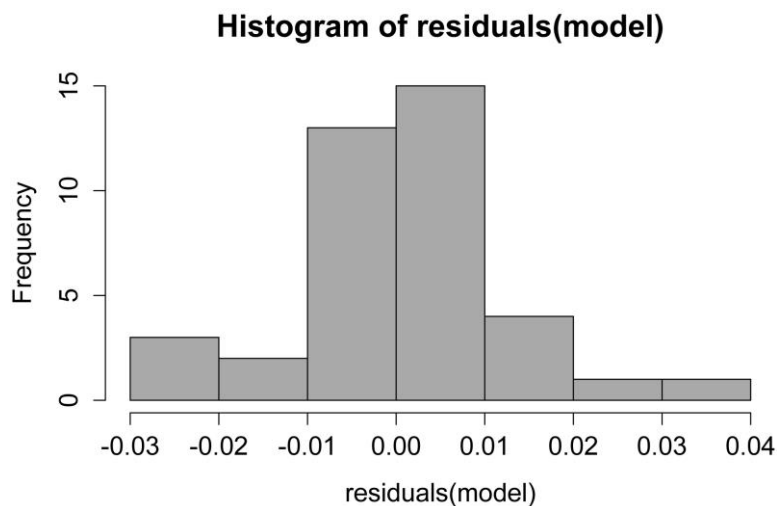
```

Checking assumptions of the model

```

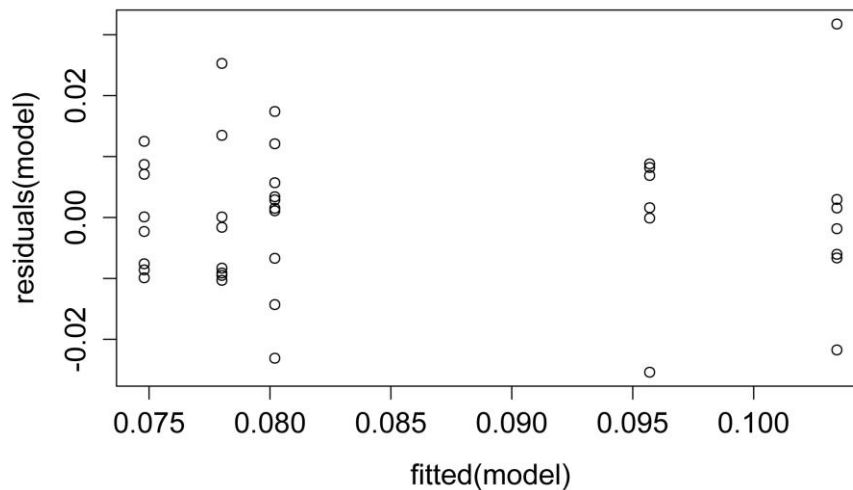
hist(residuals(model),
     col="darkgray")

```



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model),
     residuals(model))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model)
### alternative: library(FSA); residPlot(model)
```

Tukey and Least Significant Difference mean separation tests (pairwise comparisons)

Tukey and other multiple comparison tests can be performed with a handful of functions. The functions *TukeyHSD*, *HSD.test*, and *LSD.test* are probably not appropriate for cases where there are unbalanced data or unequal variances among levels of the factor, though *TukeyHSD* does make an adjustment for mildly unbalanced data. It is my understanding that the *multcomp* and *lsmeans* packages are more appropriate for unbalanced data. Another alternative is the *DTK* package that performs mean separation tests on data with unequal sample sizes and no assumption of equal variances.

Tukey comparisons in *agricolae* package

```
library(agricolae)

(HSD.test(model, "Location"))           # outer parentheses print result

      trt      means M
1 Petersburg 0.1034429 a
2 Tvarminne  0.0957000 ab
3 Tillamook  0.0802000 bc
```

```
4 Magadan      0.0780125 bc
5 Newport      0.0748000 c
```

```
# Means sharing the same letter are not significantly different
```

LSD comparisons in *agricolae* package

```
library(agricolae)
```

```
(LSD.test(model, "Location", # outer parentheses print result
          alpha = 0.05,
          p.adj="none"))     # see ?p.adjust for options
```

```
      trt      means M
1 Petersburg 0.1034429 a
2 Tvarminne  0.0957000 a
3 Tillamook  0.0802000 b
4 Magadan    0.0780125 b
5 Newport    0.0748000 b
```

```
# Means sharing the same letter are not significantly different
```

Multiple comparisons in *multcomp* package

Note that “Tukey” here does not mean Tukey-adjusted comparisons. It just sets up a matrix to compare each mean to each other mean.

```
library(multcomp)
```

```
mc = glht(model,
           mcp(Location = "Tukey"))
```

```
mcs = summary(mc, test=adjusted("single-step"))
```

```
mcs
```

```
### Adjustment options: "none", "single-step", "Shaffer",
###                    "westfall", "free", "holm", "hochberg",
###                    "hommel", "bonferroni", "BH", "BY", "fdr"
```

Linear Hypotheses:

	Estimate	Std. Error	t value	Pr(> t)
Newport - Tillamook == 0	-0.005400	0.005975	-0.904	0.89303
Petersburg - Tillamook == 0	0.023243	0.006208	3.744	0.00555 **
Magadan - Tillamook == 0	-0.002188	0.005975	-0.366	0.99596
Tvarminne - Tillamook == 0	0.015500	0.006505	2.383	0.14413
Petersburg - Newport == 0	0.028643	0.006519	4.394	< 0.001 ***
Magadan - Newport == 0	0.003213	0.006298	0.510	0.98573
Tvarminne - Newport == 0	0.020900	0.006803	3.072	0.03153 *
Magadan - Petersburg == 0	-0.025430	0.006519	-3.901	0.00376 **
Tvarminne - Petersburg == 0	-0.007743	0.007008	-1.105	0.80211
Tvarminne - Magadan == 0	0.017688	0.006803	2.600	0.09254 .

```
cld(mcs,
    level=0.05,
    decreasing=TRUE)
```

```
Tillamook    Newport Petersburg    Magadan Tvarminne
    "bc"        "c"          "a"        "bc"      "ab"
```

```
### Means sharing a letter are not significantly different
```

Multiple comparisons to a control in *multcomp* package

```
### Control is the first level of the factor
```

```
library(multcomp)
```

```
mc = glht(model,
           mcp(Location = "Dunnett"))
```

```
summary(mc, test=adjusted("single-step"))
```

```
### Adjustment options: "none", "single-step", "Shaffer",
###                      "westfall", "free", "holm", "hochberg",
###                      "hommel", "bonferroni", "BH", "BY", "fdr"
```

```
Linear Hypotheses:
```

	Estimate	Std. Error	t value	Pr(> t)
Newport - Tillamook == 0	-0.005400	0.005975	-0.904	0.79587
Petersburg - Tillamook == 0	0.023243	0.006208	3.744	0.00252 **
Magadan - Tillamook == 0	-0.002188	0.005975	-0.366	0.98989
Tvarminne - Tillamook == 0	0.015500	0.006505	2.383	0.07794 .

Multiple comparisons to a control with Dunnett Test

```
### The control group can be specified with the control option,
### or will be the first level of the factor
```

```
library(DescTools)
```

```
DunnettTest(Aam ~ Location,
            data = Data)
```

```
Dunnett's test for comparing several treatments with a control :
95% family-wise confidence level
```

	diff	lwr.ci	upr.ci	pval
Newport-Tillamook	-0.00540000	-0.020830113	0.01003011	0.7958
Petersburg-Tillamook	0.02324286	0.007212127	0.03927359	0.0026 **
Magadan-Tillamook	-0.00218750	-0.017617613	0.01324261	0.9899
Tvarminne-Tillamook	0.01550000	-0.001298180	0.03229818	0.0778 .

Multiple comparisons with least square means

Least square means can be calculated for each group. Here a Tukey adjustment is applied for multiple comparisons among group least square means. The multiple comparisons can be displayed as a compact letter display.

```
library(lsmmeans)
library(multcompView)

leastsquare = lsmmeans(model,
                        pairwise ~ Location,
                        adjust = "tukey")

$contrasts

```

contrast	estimate	SE	df	t.ratio	p.value
Tillamook - Newport	0.005400000	0.005975080	34	0.904	0.8935
Tillamook - Petersburg	-0.023242857	0.006207660	34	-3.744	0.0057
Tillamook - Magadan	0.002187500	0.005975080	34	0.366	0.9960
Tillamook - Tvarminne	-0.015500000	0.006504843	34	-2.383	0.1447
Newport - Petersburg	-0.028642857	0.006519347	34	-4.394	0.0009
Newport - Magadan	-0.003212500	0.006298288	34	-0.510	0.9858
Newport - Tvarminne	-0.020900000	0.006802928	34	-3.072	0.0317
Petersburg - Magadan	0.025430357	0.006519347	34	3.901	0.0037
Petersburg - Tvarminne	0.007742857	0.007008087	34	1.105	0.8028
Magadan - Tvarminne	-0.017687500	0.006802928	34	-2.600	0.0929

```

P value adjustment: tukey method for comparing a family of 5 estimates

cld(leastsquare,
    alpha = 0.05,
    Letters = letters,
    adjust="tukey")

```

Location	lsmean	SE	df	lower.CL	upper.CL	.group
Newport	0.0748000	0.004453562	34	0.06268565	0.08691435	a
Magadan	0.0780125	0.004453562	34	0.06589815	0.09012685	ab
Tillamook	0.0802000	0.003983387	34	0.06936459	0.09103541	ab
Tvarminne	0.0957000	0.005142530	34	0.08171155	0.10968845	bc
Petersburg	0.1034429	0.004761058	34	0.09049207	0.11639365	c

```

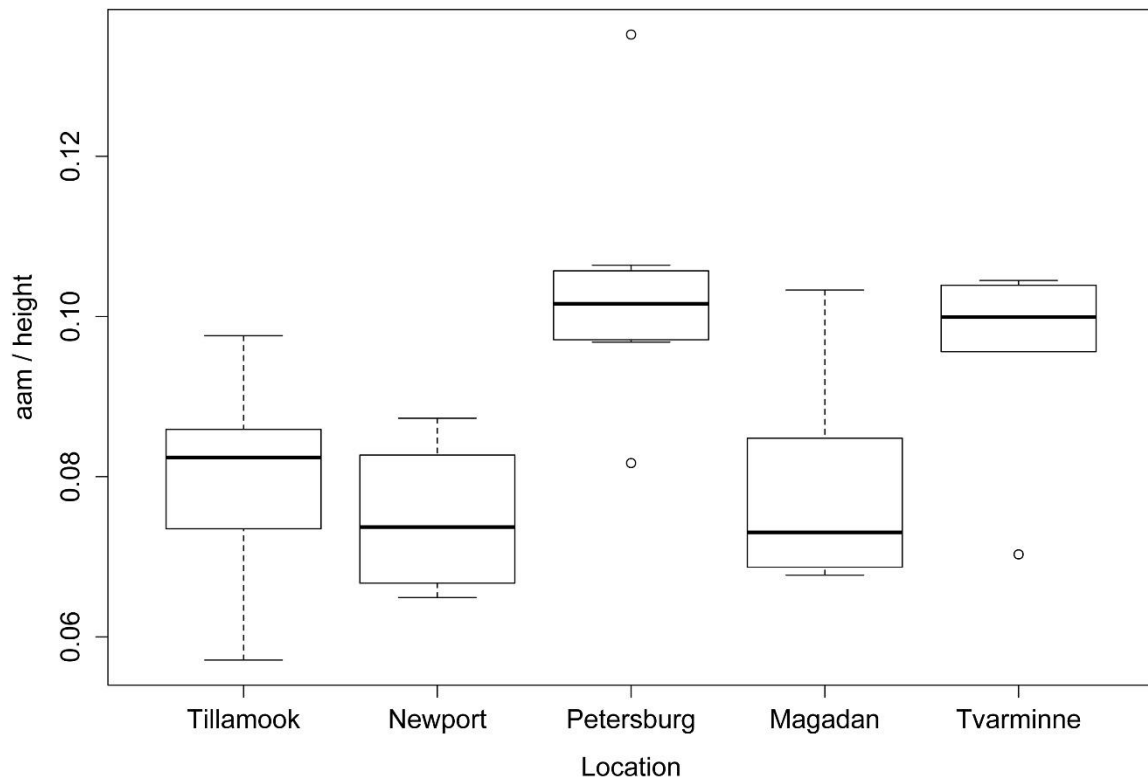
Confidence level used: 0.95
Conf-level adjustment: sidak method for 5 estimates
P value adjustment: tukey method for comparing a family of 5 estimates
significance level used: alpha = 0.05

```

Graphing the resultsSimple box plots of values across groups

```
boxplot(Aam ~ Location,
        data = Data,
```

```
ylab="aam / height",
xlab="Location")
```



Box plots of values for each level of the independent variable for a one-way analysis of variance (ANOVA).

Simple bar plot of means across groups

```
### Summarize the data frame (Data) into a table
```

```
library(Rmisc)
```

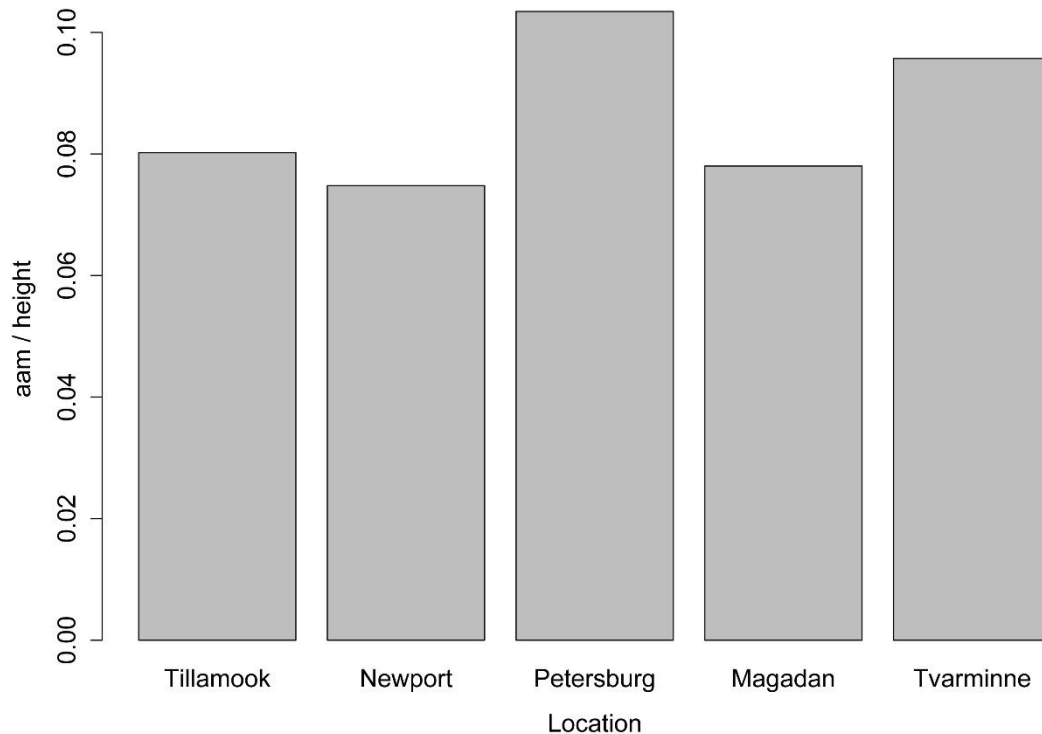
```
Data2 = summarySE(data=Data,
  "Aam",
  groupvars="Location",
  conf.interval = 0.95)
```

```
Tabla = as.table(Data2$Aam)
rownames(Tabla) = Data2$Location
```

```
Tabla
```

```
Tillamook    Newport Petersburg    Magadan    Tvarminne
0.0802000    0.0748000    0.1034429    0.0780125    0.0957000
```

```
barplot(Tabla,
        ylab="aam / height",
        xlab="Location")
```



Bar plot of means for each level of the independent variable for a one-way analysis of variance (ANOVA).

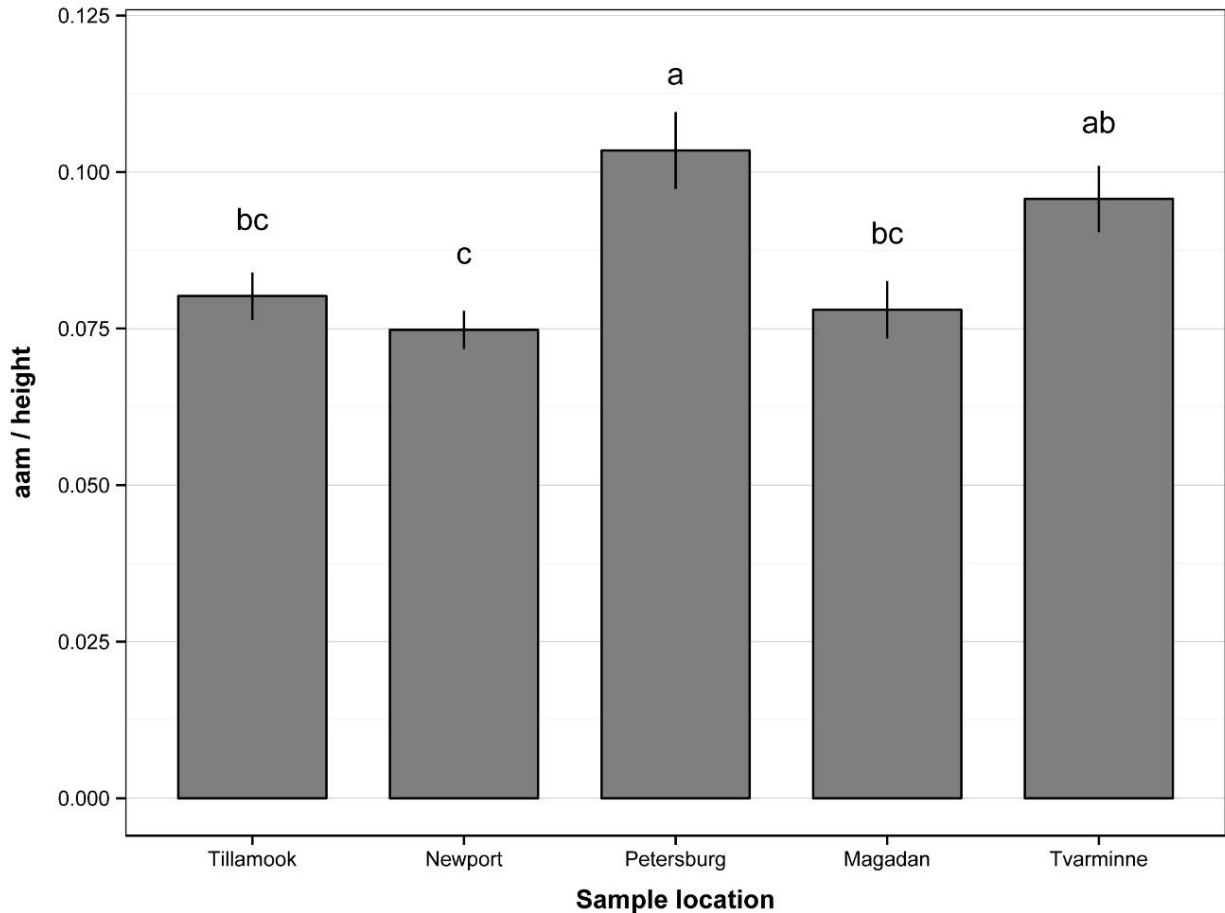
Bar plot of means with error bars across groups

```
library(ggplot2)

offset.v = -3      # offsets for mean letters
offset.h = 0.5

ggplot(Data2,
        aes(x = Location, y = Aam,
            ymax=0.12, ymin=0.0)) +
  geom_bar(stat="identity", fill="gray50",
          colour = "black", width = 0.7) +
  geom_errorbar(aes(ymax=Aam+se, ymin=Aam-se,
                  width=0.0, size=0.5, color="black")) +
  geom_text(aes(label=c("bc", "c", "a", "bc", "ab"),
                  hjust=offset.h, vjust=offset.v)) +
  labs(x = "Sample location",
       y = "aam / height") +
  ## ggtitle("Main title") +
  theme_bw() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line(colour = "grey80"),
        plot.title = element_text(size = rel(1.5),
                                   face = "bold", vjust = 1.5),
        axis.title = element_text(face = "bold"),
        axis.title.y = element_text(vjust= 1.8),
        axis.title.x = element_text(vjust= -0.5),
```

```
panel.border = element_rect(colour="black")
)
```



Bar plot of means for each level of the independent variable of a one-way analysis of variance (ANOVA). Error indicates standard error of the mean. Bars sharing the same letter are not significantly different according to Tukey's HSD test.

Welch's anova

Bartlett's test and Levene's test can be used to check the homoscedasticity of groups from a one-way anova. A significant result for these tests ($p < 0.05$) suggests that groups are heteroscedastic. One approach with heteroscedastic data in a one way anova is to use the Welch correction with the *oneway.test* function in the native *stats* package. A more versatile approach is to use the *white.adjust=TRUE* option in the *Anova* function from the *car* package.

```
### Bartlett test for homogeneity of variance
```

```
bartlett.test(Aam ~ Location,
  data = Data)
```

```
Bartlett test of homogeneity of variances
```

```
Bartlett's K-squared = 2.4341, df = 4, p-value = 0.6565
```



```
### Levene test for homogeneity of variance
```

```
library(car)
```

```
leveneTest(Aam ~ Location,
           data = Data)
```

```
Levene's Test for Homogeneity of Variance (center = median)
```

```
      Df F value Pr(>F)
group  4    0.12 0.9744
      34
```

```
### welch's anova for unequal variances
```

```
oneway.test(Aam ~ Location,
            data=Data,
            var.equal=FALSE)
```

```
One-way analysis of means (not assuming equal variances)
```

```
F = 5.6645, num df = 4.000, denom df = 15.695, p-value = 0.00508
```

```
### white-adjusted anova for heteroscedasticity
```

```
model = lm(Aam ~ Location,
           data=Data)
```

```
library(car)
```

```
Anova(model, Type="II",
      white.adjust=TRUE)
```

```
      Df      F    Pr(>F)
Location  4 5.4617 0.001659 **
Residuals 34
```

```
#      #      #
```

Power analysis

Power analysis for one-way anova

```
### -----
### Power analysis for anova, pp. 157
### -----
```

```
library(pwr)
```

```
groups = 5
means = c(10, 10, 15, 15, 15)
sd = 12
```

```
grand.mean = mean(means)
Cohen.f = sqrt( sum( (1/groups) * (means-grand.mean)^2 ) ) /sd

pwr.anova.test(k = groups,
               n = NULL,
               f = Cohen.f,
               sig.level = 0.05,
               power = 0.80)
```

Balanced one-way analysis of variance power calculation

n = 58.24599

NOTE: n is number in each group

#

Kruskal-Wallis Test

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Kruskal-Wallis Test](#)

Packages used in this chapter

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

```
if(!require(dplyr)){install.packages("dplyr")}
if(!require(FSA)){install.packages("FSA")}
if(!require(DescTools)){install.packages("DescTools")}
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(multcompView)){install.packages("multcompView")}
```

When to use it

See the [Handbook](#) for information on this topic.

Null hypothesis

This example shows just summary statistics, histograms by group, and the Kruskal-Wallis test. An example with plots, post-hoc tests, and alternative tests is shown in the “Example” section below.

Kruskal-Wallis test example

```
### -----
### Kruskal-wallis test, hypothetical example, p. 159
### -----

Input =("
Group      Value
```

```

Group.1      1
Group.1      2
Group.1      3
Group.1      4
Group.1      5
Group.1      6
Group.1      7
Group.1      8
Group.1      9
Group.1     46
Group.1     47
Group.1     48
Group.1     49
Group.1     50
Group.1     51
Group.1     52
Group.1     53
Group.1    342
Group.2     10
Group.2     11
Group.2     12
Group.2     13
Group.2     14
Group.2     15
Group.2     16
Group.2     17
Group.2     18
Group.2     37
Group.2     58
Group.2     59
Group.2     60
Group.2     61
Group.2     62
Group.2     63
Group.2     64
Group.2    193
Group.3     19
Group.3     20
Group.3     21
Group.3     22
Group.3     23
Group.3     24
Group.3     25
Group.3     26
Group.3     27
Group.3     28
Group.3     65
Group.3     66
Group.3     67
Group.3     68
Group.3     69
Group.3     70
Group.3     71
Group.3     72
")

```

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

```
### Specify the order of factor levels
```

```
library(dplyr)
```

```
Data =  
mutate(Data,  
  Group = factor(Group, levels=unique(Group)))
```

Medians and descriptive statistics

As noted in the *Handbook*, each group has identical medians and means.

```
library(FSA)
```

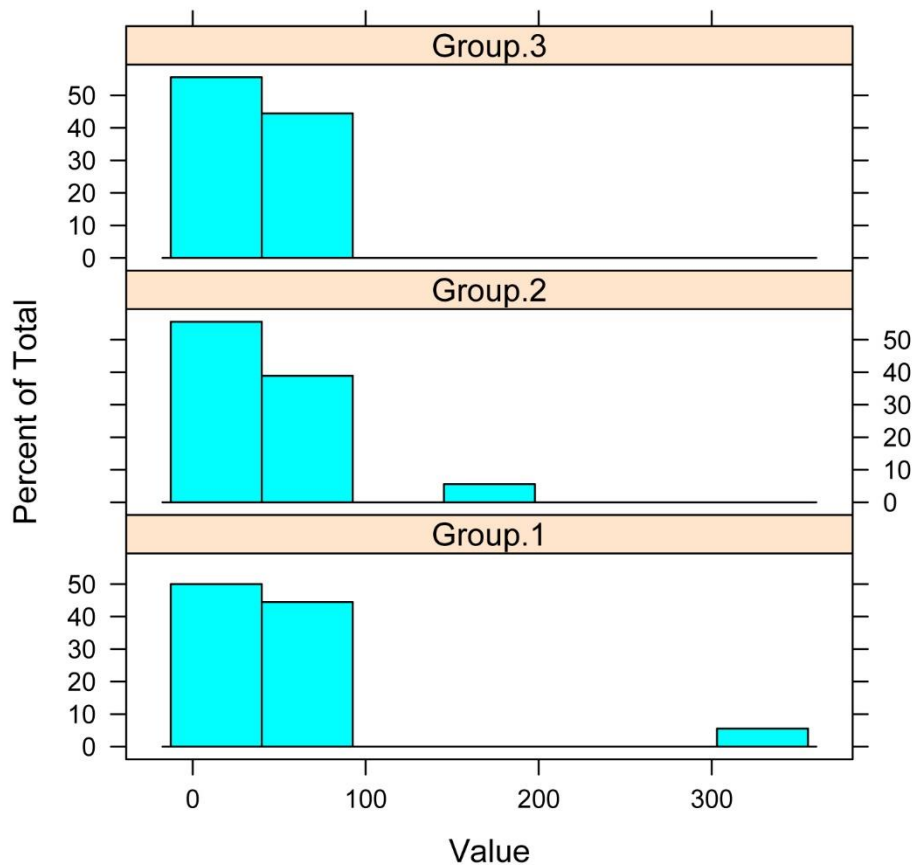
```
Summarize(Value ~ Group,  
  data = Data)
```

	Group	n	mean	sd	min	Q1	median	Q3	max
1	Group.1	18	43.5	77.77513	1	5.25	27.5	49.75	342
2	Group.2	18	43.5	43.69446	10	14.25	27.5	60.75	193
3	Group.3	18	43.5	23.16755	19	23.25	27.5	67.75	72

Histograms for each group

```
library(lattice)
```

```
histogram(~ Value | Group,  
  data=Data,  
  layout=c(1,3)) # columns and rows of individual plots
```



Kruskal-Wallis test

In this case, there is a significant difference in the distributions of values among groups, as is evident both from the histograms and from the significant Kruskal-Wallis test. Only in cases where the distributions in each group are similar can a significant Kruskal-Wallis test be interpreted as a difference in medians.

```
kruskal.test(Value ~ Group,
             data = Data)
```

kruskal-wallis chi-squared = 7.3553, df = 2, p-value = 0.02528

#

How the test works

Assumptions

See the [Handbook](#) for information on these topics.

Example

The Kruskal-Wallis test is performed on a data frame with the *kruskal.test* function in the native *stats* package. Shown first is a complete example with plots, post-hoc tests, and alternative methods, for the example used in R help. It is data measuring if the mucociliary efficiency in the rate of dust removal is different among normal subjects, subjects with obstructive airway

disease, and subjects with asbestosis. For the original citation, use the `?kruskal.test` command. For both the submissive dog example and the oyster DNA example from the *Handbook*, a Kruskal-Wallis test is shown later in this chapter.

Kruskal-Wallis test example

```
### -----
### kruskal-wallis test, asbestosis example from R help for
### kruskal.test
### -----
```

```
Input =("
Obs Health      Efficiency
1   Normal      2.9
2   Normal      3.0
3   Normal      2.5
4   Normal      2.6
5   Normal      3.2
6   OAD         3.8
7   OAD         2.7
8   OAD         4.0
9   OAD         2.4
10  Asbestosis  2.8
11  Asbestosis  3.4
12  Asbestosis  3.7
13  Asbestosis  2.2
14  Asbestosis  2.0
")
```

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

```
### Specify the order of factor levels
```

```
library(dplyr)
```

```
Data =
mutate(Data,
       Health = factor(Health, levels=unique(Health)))
```

Medians and descriptive statistics

```
library(FSA)
```

```
Summarize(Efficiency ~ Health,
          data = Data)
```

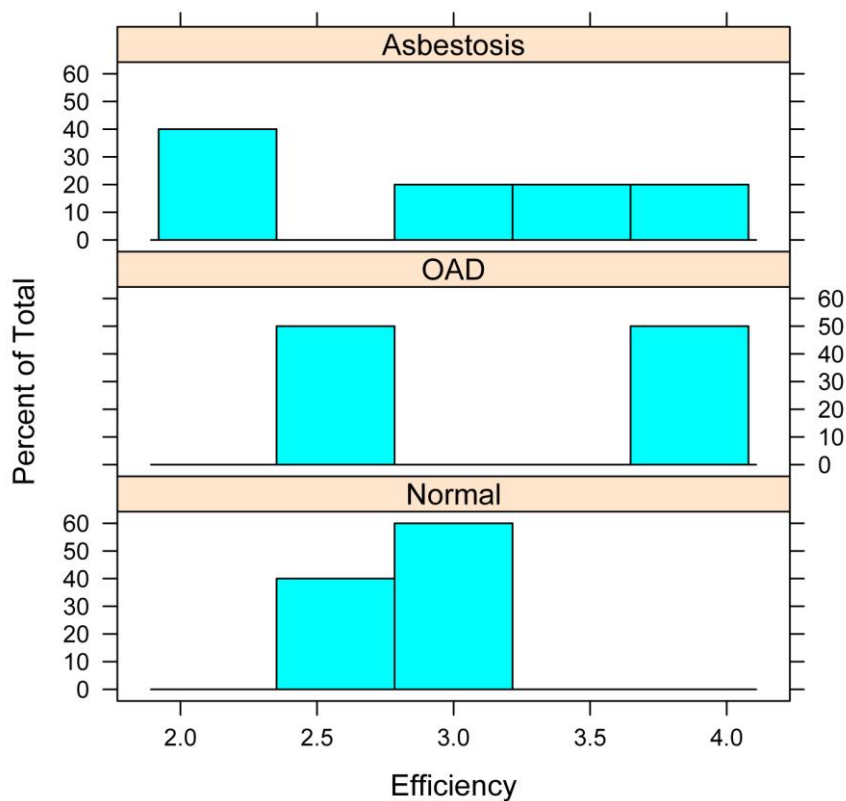
	Health	n	mean	sd	min	Q1	median	Q3	max
1	Normal	5	2.840	0.2880972	2.5	2.600	2.90	3.00	3.2
2	OAD	4	3.225	0.7932003	2.4	2.625	3.25	3.85	4.0
3	Asbestosis	5	2.820	0.7362065	2.0	2.200	2.80	3.40	3.7

Graphing the results

Stacked histograms of values across groups

```
library(lattice)

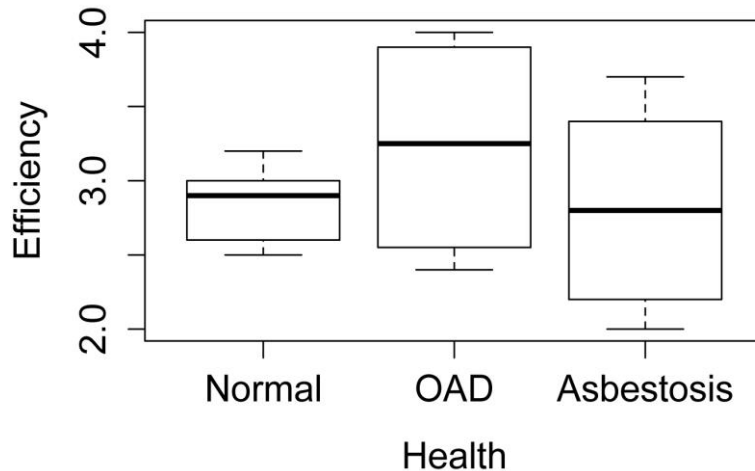
histogram(~ Efficiency | Health,
          data=Data,
          layout=c(1,3))      # columns and rows of individual plots
```



Stacked histograms for each group in a Kruskal-Wallis test. If the distributions are similar, then the Kruskal-Wallis test will test for a difference in medians.

Simple box plots of values across groups

```
boxplot(Efficiency ~ Health,
        data = Data,
        ylab="Efficiency",
        xlab="Health")
```



Kruskal-Wallis test

```
kruskal.test(Efficiency ~ Health,
             data = Data)
```

```
kruskal-wallis chi-squared = 0.7714, df = 2, p-value = 0.68
```

Dunn test for multiple comparisons

If the Kruskal-Wallis test is significant, a post-hoc analysis can be performed to determine which levels of the independent variable differ from each other level. Probably the most popular test for this is the Dunn test, which is performed with the *dunnTest* function in the *FSA* package. Adjustments to the p-values could be made using the *method* option to control the familywise error rate or to control the false discovery rate. See *?p.adjust* for details.

Zar (2010) states that the Dunn test is appropriate for groups with unequal numbers of observations.

If there are several values to compare, it can be beneficial to have R convert this table to a compact letter display for you. The *cldList* function in the *rcompanion* package can do this.

```
### Order groups by median

Data$Health = factor(Data$Health,
                     levels=c("OAD", "Normal", "Asbestosis"))

### Dunn test

library(FSA)

PT = dunnTest(Efficiency ~ Health,
             data=Data,
             method="bh")    # Can adjust p-values;
                             # See ?p.adjust for options

PT
```


Dunn (1964) Kruskal-wallis multiple comparison
p-values adjusted with the False Discovery Rate method.

	Comparison	Z	P.unadj	P.adj
1	OAD - Normal	0.6414270	0.5212453	0.7818680
2	OAD - Asbestosis	0.8552360	0.3924205	1.0000000
3	Normal - Asbestosis	0.2267787	0.8205958	0.8205958

```
PT = PT$res
```

```
PT
```

```
library(rcompanion)
```

```
cldList(comparison = PT$Comparison,
        p.value     = PT$P.adj,
        threshold   = 0.05)
```

Error: No significant differences.

Nemenyi test for multiple comparisons

Zar (2010) suggests that the Nemenyi test is not appropriate for groups with unequal numbers of observations.

```
library(DescTools)
```

```
PT = NemenyiTest(x = Data$Efficiency,
                 g = Data$Health,
                 dist="tukey")
```

```
PT
```

Nemenyi's test of multiple comparisons for independent samples (tukey)

	mean.rank.diff	pval
OAD-Normal	1.8	0.7972
Asbestosis-Normal	-0.6	0.9720
Asbestosis-OAD	-2.4	0.6686

```
library(rcompanion)
```

```
cldList(comparison = PT$Comparison,
        p.value     = PT$P.adj,
        threshold   = 0.05)
```

Error: No significant differences.

Pairwise Mann-Whitney U-tests

Another post-hoc approach is to use pairwise Mann-Whitney U-tests. To prevent the inflation of type I error rates, adjustments to the p-values can be made using the *p.adjust.method* option to control the familywise error rate or to control the false discovery rate. See *?p.adjust* for details.

If there are several values to compare, it can be beneficial to have R convert this table to a compact letter display for you. The *multcompLetters* function in the *multcompView* package can do this, but first the table of p-values must be converted to a full table.

```
PT = pairwise.wilcox.test(Data$Efficiency,
                          Data$Health,
                          p.adjust.method="none")
      # Can adjust p-values;
      # See ?p.adjust for options
```

```
PT
```

```
Pairwise comparisons using wilcoxon rank sum test
```

```
      Normal OAD
OAD      0.73  -
Asbestosis 1.00 0.41
```

```
PT = PT$p.value
```

```
library(rcompanion)
```

```
PT1 = fullPTable(PT)
```

```
PT1
```

```
      Normal      OAD Asbestosis
Normal  1.0000000 0.7301587 1.0000000
OAD      0.7301587 1.0000000 0.4126984
Asbestosis 1.0000000 0.4126984 1.0000000
```

```
library(multcompView)
```

```
multcompLetters(PT1,
                compare="<",
                threshold=0.05,
                Letters=letters,
                reversed = FALSE)
```

```
      Normal      OAD Asbestosis
      "a"      "a"      "a"
```

```
### values sharing the same letter are not significantly different
```

#

Kruskal-Wallis test example

```
#### -----
#### Kruskal-wallis test, submissive dog example, pp. 161-162
#### -----
```

```
Input =("
Dog      Sex      Rank
Merlino   Male      1
Gastone   Male      2
Pippo     Male      3
Leon     Male      4
Golia     Male      5
Lancillotto Male    6
Mamy      Female    7
Nanà      Female    8
Isotta    Female    9
Diana     Female   10
Simba     Male     11
Pongo     Male     12
Semola    Male     13
Kimba     Male     14
Morgana    Female   15
Stella    Female   16
Hansel    Male     17
Cucciola  Male     18
Mammolo   Male     19
Dotto     Male     20
Gongolo   Male     21
Gretel    Female   22
Brontolo  Female   23
Eolo      Female   24
Mag        Female   25
Emy        Female   26
Pisola    Female   27
")
```

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

```
kruskal.test(Rank ~ Sex,
              data = Data)
```

```
Kruskal-wallis chi-squared = 4.6095, df = 1, p-value = 0.03179
```

#

Graphing the results

Graphing of the results is shown above in the “Example” section.

Similar tests

One-way anova is presented elsewhere in this book.

How to do the test

Kruskal-Wallis test example

```
### -----
### kruskal-wallis test, oyster DNA example, pp. 163-164
### -----
```

```
Input =("
Markername Markertype fst
CVB1        DNA        -0.005
CVB2m       DNA         0.116
CVJ5        DNA        -0.006
CVJ6        DNA         0.095
CVL1        DNA         0.053
CVL3        DNA         0.003
6Pg         protein    -0.005
Aat-2       protein     0.016
Acp-3       protein     0.041
Adk-1       protein     0.016
Ap-1        protein     0.066
Est-1       protein     0.163
Est-3       protein     0.004
Lap-1       protein     0.049
Lap-2       protein     0.006
Mpi-2       protein     0.058
Pgi         protein    -0.002
Pgm-1       protein     0.015
Pgm-2       protein     0.044
Sdh         protein     0.024
")
```

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

```
kruskal.test(fst ~ Markertype,
             data = Data)
```

```
kruskal-wallis chi-squared = 0.0426, df = 1, p-value = 0.8365
```

```
#      #      #
```

Power Analysis

See the *Handbook* for information on this topic.

References

Zar, J.H. 2010. Biostatistical Analysis, 5th ed. Pearson Prentice Hall: Upper Saddle River, NJ.

One-way Analysis with Permutation Test

Permutation tests are non-parametric tests that do not assume normally-distributed errors. However, these tests may assume that distributions have similar variance or shape to be interpreted as a test of means.

A one-way anova using permutation tests can be performed with the *coin* package. A post-hoc analysis can be conducted with pairwise permutation tests analagous to pairwise t-tests. This can be accomplished with the functions *pairwisePermutationTest* and *pairwisePermutationMatrix* in the *rcompanion* package, which rely on the *independence_test* function in the *coin* package.

For more information on permutation tests available in the *coin* package, see:

```
help(package="coin")
```

Consult the chapters on *One-way Anova* and *Kruskal–Wallis Test* for general consideration about conducting analysis of variance.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Permutation Test of Independence](#)

Packages used in this chapter

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

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

Permutation test for one-way analysis

```
### -----
### One-way permutation test, hypothetical data
### -----

Input =("
Factor Response
A      4.6
A      5.5
A      3.4
A      5.0
A      3.9
A      4.5
B      3.6
B      4.5
B      2.4
B      4.0
B      2.9
B      3.5
```

```

C      2.6
C      3.5
C      1.4
C      3.0
C      1.9
C      2.5
D      4.7
D      5.6
D      3.5
D      5.1
D      4.0
D      4.6
")

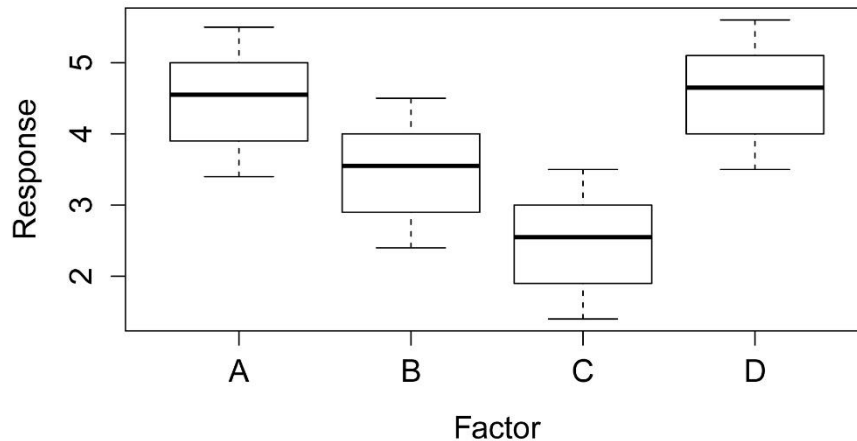
```

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

```
Data$Factor = factor(Data$Factor,
                      ordered=FALSE,
                      levels=unique(Data$Factor))
```

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

```
boxplot(Response ~ Factor,
         data = Data,
         ylab = "Response",
         xlab = "Factor")
```



Permutation test

```
library(coin)
```

```
independence_test(Response ~ Factor,
                  data = Data)
```

```
Asymptotic General Independence Test
```

```
maxT = 3.2251, p-value = 0.005183
```

Pairwise permutation tests

Pairwise permutation tests could be used as a post-hoc test for a significant permutation test. If no p-value adjustment is made, then the type I error rate may be inflated due to multiple comparisons. Here, the “fdr” p-value adjustment method is used to control the false discovery rate.

Table output with *pairwisePermutationTest*

```
#### Order groups by median

Data$Factor = factor(Data$Factor,
                      levels = c("D", "A", "B", "C"))

library(FSA)

headtail(Data)

#### Pairwise tests

library(rcompanion)

PT = pairwisePermutationTest(Response ~ Factor,
                             data = Data,
                             method="fdr")

PT
```

	Comparison	Stat	p.value	p.adjust
1	D - A = 0	-0.2409	0.8096	0.80960
2	D - B = 0	-2.074	0.03812	0.06106
3	D - C = 0	-2.776	0.005505	0.01876
4	A - B = 0	1.952	0.05088	0.06106
5	A - C = 0	2.734	0.006253	0.01876
6	B - C = 0	1.952	0.05088	0.06106

```
library(rcompanion)

cldList(p.adjust ~ Comparison,
        data      = PT,
        threshold = 0.05)
```

	Group	Letter	MonoLetter
1	D	a	a
2	A	a	a
3	B	ab	ab
4	C	b	b

Compact letter display output with *pairwisePermutationMatrix*

```
#### Order groups by median
```

```
Data$Factor = factor(Data$Factor,
                      levels = c("D", "A", "B", "C"))

library(FSA)

headtail(Data)

### Pairwise tests

library(rcompanion)

PM = pairwisePermutationMatrix(Response ~ Factor,
                               data = Data,
                               method="fdr")
```

```
PM
```

```
$Unadjusted
      D      A      B      C
D NA 0.8096 0.03812 0.005505
A NA      NA 0.05088 0.006253
B NA      NA      NA 0.050880
C NA      NA      NA      NA

$Method
[1] "fdr"

$Adjusted
      D      A      B      C
D 1.00000 0.80960 0.06106 0.01876
A 0.80960 1.00000 0.06106 0.01876
B 0.06106 0.06106 1.00000 0.06106
C 0.01876 0.01876 0.06106 1.00000
```

```
library(multcompView)

multcompLetters(PM$Adjusted,
                compare="<",
                threshold=0.05,
                Letters=letters,
                reversed = FALSE)
```

```
      D      A      B      C
"a"  "a"  "ab"  "b"
```

```
#      #      #
```


Nested Anova

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Using Random Effects in Models](#)

[SAEPPER: What are Least Square Means?](#)

[SAEPPER: One-way ANOVA with Random Blocks](#)

Packages used in this chapter

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

```
if(!require(nlme)){install.packages("nlme")}
if(!require(multcomp)){install.packages("multcomp")}
if(!require(multcompView)){install.packages("multcompView")}
if(!require(lsmmeans)){install.packages("lsmmeans")}
if(!require(lme4)){install.packages("lme4")}
if(!require(lmerTest)){install.packages("lmerTest")}
if(!require(TukeyC)){install.packages("TukeyC")}
```

When to use it

Null hypotheses

How the test works

Partitioning variance and optimal allocation of resources

Unequal sample sizes

Assumptions

Example

Graphing the results

Similar tests

See the [Handbook](#) for information on these topics.

How to do the test

Nested anova example with mixed effects model (nlme)

One approach to fit a nested anova is to use a mixed effects model. Here *Tech* is being treated as a fixed effect, while *Rat* is treated as a random effect. Note that the F-value and p-value for the test on *Tech* agree with the values in the *Handbook*. The effect of *Rat* will be tested by comparing this model to a model without the *Rat* term. The model is fit using the *lme* function in *nlme*.

```
### -----
### Nested anova, SAS example, pp. 171-173
### -----

Input =("
Tech Rat Protein
Janet 1 1.119
Janet 1 1.2996
Janet 1 1.5407
Janet 1 1.5084
```

Janet	1	1.6181
Janet	1	1.5962
Janet	1	1.2617
Janet	1	1.2288
Janet	1	1.3471
Janet	1	1.0206
Janet	2	1.045
Janet	2	1.1418
Janet	2	1.2569
Janet	2	0.6191
Janet	2	1.4823
Janet	2	0.8991
Janet	2	0.8365
Janet	2	1.2898
Janet	2	1.1821
Janet	2	0.9177
Janet	3	0.9873
Janet	3	0.9873
Janet	3	0.8714
Janet	3	0.9452
Janet	3	1.1186
Janet	3	1.2909
Janet	3	1.1502
Janet	3	1.1635
Janet	3	1.151
Janet	3	0.9367
Brad	5	1.3883
Brad	5	1.104
Brad	5	1.1581
Brad	5	1.319
Brad	5	1.1803
Brad	5	0.8738
Brad	5	1.387
Brad	5	1.301
Brad	5	1.3925
Brad	5	1.0832
Brad	6	1.3952
Brad	6	0.9714
Brad	6	1.3972
Brad	6	1.5369
Brad	6	1.3727
Brad	6	1.2909
Brad	6	1.1874
Brad	6	1.1374
Brad	6	1.0647
Brad	6	0.9486
Brad	7	1.2574
Brad	7	1.0295
Brad	7	1.1941
Brad	7	1.0759
Brad	7	1.3249
Brad	7	0.9494
Brad	7	1.1041
Brad	7	1.1575
Brad	7	1.294

```

Brad 7 1.4543
")

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

### Since Rat is read in as an integer variable, convert it to factor

Data$Rat = as.factor(Data$Rat)

library(nlme)

model = lme(Protein ~ Tech, random=~1|Rat,
            data=Data,
            method="REML")

anova.lme(model,
           type="sequential",
           adjustSigma = FALSE)

              numDF denDF  F-value p-value
(Intercept)      1    54 587.8664 <.0001
Tech              1     4  0.2677 0.6322

```

Post-hoc comparison of means

Note that “Tukey” here instructs the *glht* function to compare all means, not to perform a Tukey adjustment of multiple comparisons.

```

library(multcomp)

posthoc = glht(model,
               linfct = mcp(Tech="Tukey"))

mcs = summary(posthoc,
              test=adjusted("single-step"))

mcs

### Adjustment options: "none", "single-step", "Shaffer",
###                      "westfall", "free", "holm", "hochberg",
###                      "hommel", "bonferroni", "BH", "BY", "fdr"

Linear Hypotheses:
              Estimate Std. Error z value Pr(>|z|)
Janet - Brad == 0 -0.05060    0.09781  -0.517   0.605

cld(mcs,
    level=0.05,
    decreasing=TRUE)

Brad Janet
"a"      "a"

### Means sharing a letter are not significantly different

```

Post-hoc comparison of least-square means

Least squares means are adjusted for other terms in the model. If the experimental design is unbalanced or there is missing data, the least square means may differ significantly from arithmetic means for treatments, but are generally more representative of the population means than the arithmetic means would be.

Note that the adjustments for multiple comparisons (`adjust="tukey"`) appears in both the `lsmeans` and `cld` functions.

```
library(multcompView)
library(lsmeans)

leastsquare = lsmeans(model,
                      pairwise ~ Tech,
                      adjust="tukey")      ### Tukey-adjusted comparisons

leastsquare

$lsmeans
  Tech    lsmean      SE df lower.CL upper.CL
Brad  1.211023 0.06916055  5 1.0332405 1.388806
Janet  1.160420 0.06916055  4 0.9683995 1.352440

Confidence level used: 0.95

$contrasts
  contrast      estimate      SE df t.ratio p.value
Brad - Janet 0.05060333 0.09780778  4   0.517  0.6322

cld(leastsquare,
    alpha=0.05,
    Letters=letters,      ### Use lower-case letters for .group
    adjust="tukey")      ### Tukey-adjusted comparisons

Tech    lsmean      SE df asymp.LCL asymp.UCL .group
Janet  1.160420 0.06916018 NA  1.005745  1.315095  a
Brad   1.211023 0.06916018 NA  1.056348  1.365698  a

### Means sharing a letter in .group are not significantly different
```

Test the significance of the random effect in the mixed effects model

In order to test the significance of the random effect from our model (*Rat*), we can fit a new model with only the fixed effects from the model. For this we use the `gl`s function in the *nlme* package. We then compare the two models with the `anova` function. Note that the p-value does not agree with p-value from the *Handbook*, because the technique is different, though in this case the conclusion is the same. As a general precaution, if your models are fit with "REML" (restricted maximum likelihood) estimation, then you should compare only models with the

same fixed effects. If you need to compare models with different fixed effects, use “ML” as the estimation method for all models.

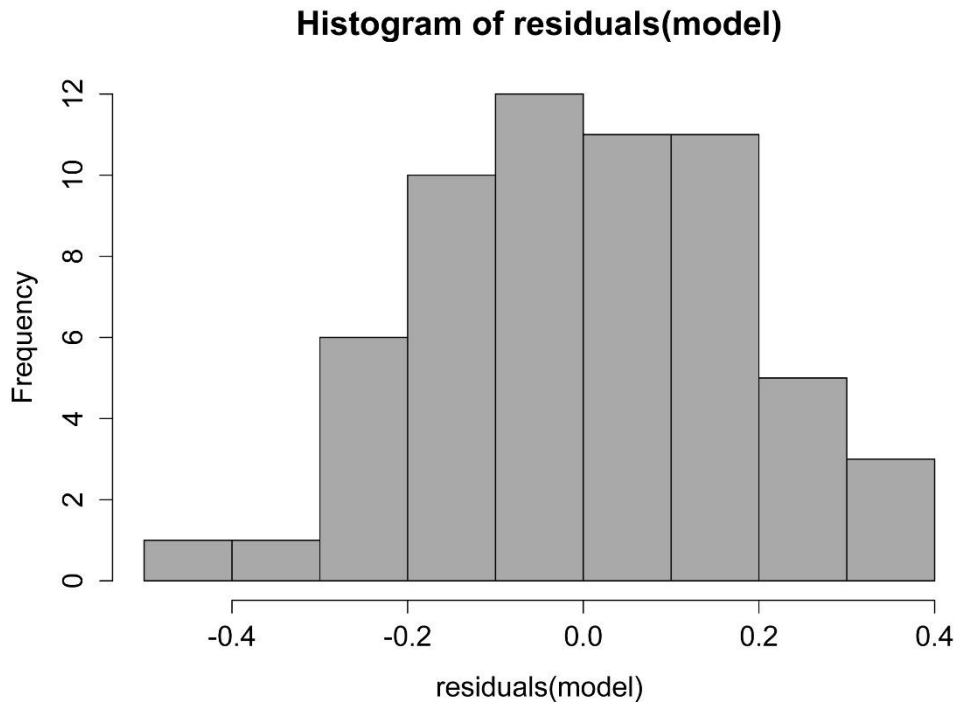
```
model.fixed = gls(Protein ~ Tech,
                  data=Data,
                  method="REML")
```

```
anova(model,
       model.fixed)
```

	Model	df	AIC	BIC	logLik	Test	L.Ratio	p-value
model	1	4	-7.819054	0.4227176	7.909527			
model.fixed	2	3	-4.499342	1.6819872	5.249671	1 vs 2	5.319713	0.0211

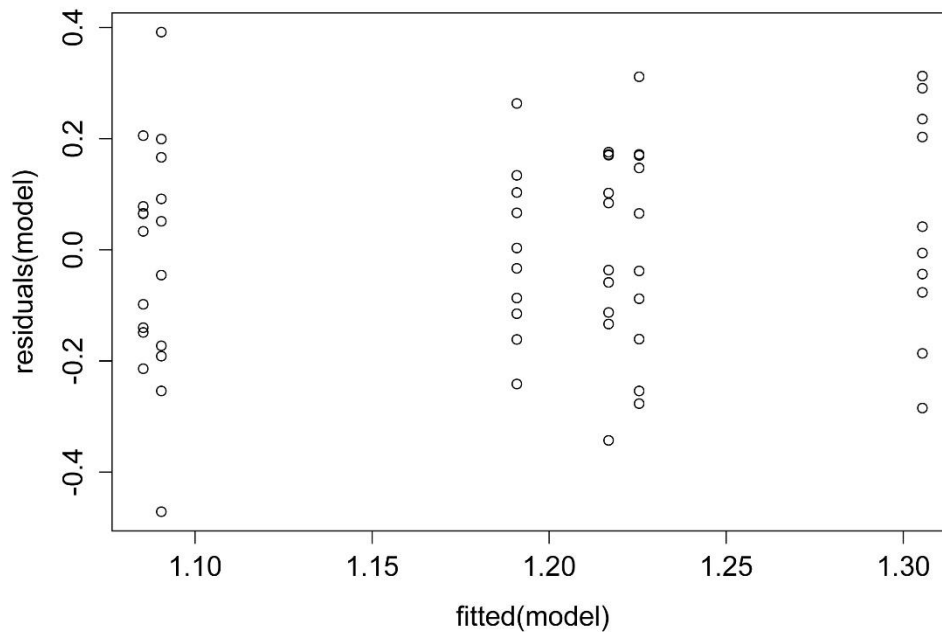
Checking assumptions of the model

```
hist(residuals(model),
     col="darkgray")
```



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model),
     residuals(model))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

additional model checking plots with: `plot(model)`

#

Mixed effects model with lmer

The following is an abbreviated example of a nested anova using the *lmer* function in the *lme4* package. See the previous example in this chapter for explanation and model-checking.

```
### -----
### Nested anova, SAS example, pp. 171-173
### -----
```

```
Input =("
Tech Rat Protein
Janet 1 1.119
Janet 1 1.2996
Janet 1 1.5407
Janet 1 1.5084
Janet 1 1.6181
Janet 1 1.5962
Janet 1 1.2617
Janet 1 1.2288
Janet 1 1.3471
Janet 1 1.0206
Janet 2 1.045
Janet 2 1.1418
Janet 2 1.2569
Janet 2 0.6191
```

```

Janet 2 1.4823
Janet 2 0.8991
Janet 2 0.8365
Janet 2 1.2898
Janet 2 1.1821
Janet 2 0.9177
Janet 3 0.9873
Janet 3 0.9873
Janet 3 0.8714
Janet 3 0.9452
Janet 3 1.1186
Janet 3 1.2909
Janet 3 1.1502
Janet 3 1.1635
Janet 3 1.151
Janet 3 0.9367
Brad 5 1.3883
Brad 5 1.104
Brad 5 1.1581
Brad 5 1.319
Brad 5 1.1803
Brad 5 0.8738
Brad 5 1.387
Brad 5 1.301
Brad 5 1.3925
Brad 5 1.0832
Brad 6 1.3952
Brad 6 0.9714
Brad 6 1.3972
Brad 6 1.5369
Brad 6 1.3727
Brad 6 1.2909
Brad 6 1.1874
Brad 6 1.1374
Brad 6 1.0647
Brad 6 0.9486
Brad 7 1.2574
Brad 7 1.0295
Brad 7 1.1941
Brad 7 1.0759
Brad 7 1.3249
Brad 7 0.9494
Brad 7 1.1041
Brad 7 1.1575
Brad 7 1.294
Brad 7 1.4543
")

```

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

```
Data$Rat = as.factor(Data$Rat)
```

```
library(lme4)
```

```
library(lmerTest)
```

```
model = lmer(Protein ~ Tech + (1|Rat),
             data=Data,
             REML=TRUE)
```

```
anova(model)
```

Analysis of Variance Table of type III with Satterthwaite approximation for degrees of freedom

	Sum Sq	Mean Sq	NumDF	DenDF	F.value	Pr(>F)
Tech	0.0096465	0.0096465	1	4	0.26768	0.6322

```
rand(model)
```

Analysis of Random effects Table:

	Chi.sq	Chi.DF	p.value
Rat	5.32	1	0.02 *

```
diff1smeans(model,
             test.efs="Tech")
```

Differences of LSMEANS:

	Estimate	Standard Error	DF	t-value	Lower CI	Upper CI	p-value
Tech Brad - Janet	0.1	0.0978	4.0	0.52	-0.221	0.322	0.6

```
library(multcomp)
```

```
posthoc = glht(model,
               linfct = mcp(Tech="Tukey"))
```

```
mcs = summary(posthoc,
               test=adjusted("single-step"))
```

```
mcs
```

Linear Hypotheses:

	Estimate	Std. Error	z value	Pr(> z)
Janet - Brad == 0	-0.05060	0.09781	-0.517	0.605

(Adjusted p values reported -- single-step method)

```
cld(mcs,
    level=0.05,
    decreasing=TRUE)
```

Brad	Janet
"a"	"a"

#

Nested anova example with the aov function

```
### -----
```



```
### Nested anova, SAS example, pp. 171-173
```

```
### -----
```

```
Input =(
Tech  Rat  Protein
Janet 1    1.119
Janet 1    1.2996
Janet 1    1.5407
Janet 1    1.5084
Janet 1    1.6181
Janet 1    1.5962
Janet 1    1.2617
Janet 1    1.2288
Janet 1    1.3471
Janet 1    1.0206
Janet 2    1.045
Janet 2    1.1418
Janet 2    1.2569
Janet 2    0.6191
Janet 2    1.4823
Janet 2    0.8991
Janet 2    0.8365
Janet 2    1.2898
Janet 2    1.1821
Janet 2    0.9177
Janet 3    0.9873
Janet 3    0.9873
Janet 3    0.8714
Janet 3    0.9452
Janet 3    1.1186
Janet 3    1.2909
Janet 3    1.1502
Janet 3    1.1635
Janet 3    1.151
Janet 3    0.9367
Brad  5    1.3883
Brad  5    1.104
Brad  5    1.1581
Brad  5    1.319
Brad  5    1.1803
Brad  5    0.8738
Brad  5    1.387
Brad  5    1.301
Brad  5    1.3925
Brad  5    1.0832
Brad  6    1.3952
Brad  6    0.9714
Brad  6    1.3972
Brad  6    1.5369
Brad  6    1.3727
Brad  6    1.2909
Brad  6    1.1874
Brad  6    1.1374
Brad  6    1.0647
Brad  6    0.9486
```

```
Brad 7 1.2574
Brad 7 1.0295
Brad 7 1.1941
Brad 7 1.0759
Brad 7 1.3249
Brad 7 0.9494
Brad 7 1.1041
Brad 7 1.1575
Brad 7 1.294
Brad 7 1.4543
")
```

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

```
### Since Rat is read in as an integer variable, convert it to factor
```

```
Data$Rat = as.factor(Data$Rat)
```

Using the *aov* function for a nested anova

The *aov* function in the native stats package allows you to specify an error component to the model. When formulating this model in R, the correct error is *Rat*, not *Tech/Rat* (Rat within Tech) as used in the SAS example. The SAS model will tolerate *Rat* or *Rat(Tech)*.

The summary of the *aov* will produce the correct test for *Tech*. The test for *Rat* can be performed by manually calculating the p-value for the F-test using the output for *Error:Rat* and *Error:Within*.

See the rattlesnake example in the *Two-way anova* chapter for designating an error term in a repeated-measures model.

```
fit = aov(Protein ~ Tech + Error(Rat), data=Data)
summary(fit)
```

```
Error: Rat
      Df Sum Sq Mean Sq F value Pr(>F)
Tech    1 0.0384  0.03841    0.268  0.632
Residuals 4 0.5740  0.14349
```

```
### This matches "use for groups" in the Handbook
```

Using Mean Sq and Df values to get p-value for H = Tech and Error = Rat

```
pf(q=0.03841/0.14349,
   df1=1,
   df2=4,
   lower.tail=FALSE)
```

```
[1] 0.6321845
```

```
### Note: This is same test as summary(fit)
```

Using Mean Sq and Df values to get p-value for H = Rat and Error = Within

```
summary(fit)

Error: Within
      Df Sum Sq Mean Sq F value Pr(>F)
Residuals 54  1.946  0.03604

pf(q=0.14349/0.03604,
  df1=4,
  df2=54,
  lower.tail=F)

[1] 0.006663615

### Matches "use for subgroups" in the Handbook
```

Post-hoc comparison of means with Tukey

The *aov* function with an *Error* component produces an object of *aovlist* type, which unfortunately isn't handled by many post-hoc testing functions. However, in the *TukeyC* package, you can specify a model and error term. For unbalanced data, the *dispersion* parameter may need to be modified.

```
library(TukeyC)

tuk = TukeyC(Data,
  model = 'Protein ~ Tech + Error(Rat)',
  error = 'Rat',
  which = 'Tech',
  fl1=1,
  sig.level = 0.05)

summary(tuk)

Groups of means at sig.level = 0.05
      Means G1
Brad   1.21  a
Janet  1.16  a
```

Two-way Anova

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Two-way ANOVA](#)

[SAEPPER: Using Random Effects in Models](#)

[SAEPPER: What are Least Square Means?](#)

Packages used in this chapter

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

```
if(!require(FSA)){install.packages("FSA")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(car)){install.packages("car")}
if(!require(multcompView)){install.packages("multcompView")}
if(!require(lsmmeans)){install.packages("lsmmeans")}
if(!require(grid)){install.packages("grid")}
if(!require(nlme)){install.packages("nlme")}
if(!require(lme4)){install.packages("lme4")}
if(!require(lmerTest)){install.packages("lmerTest")}
if(!require(rcompanion)){install.packages("rcompanion")}
```

When to use it

Null hypotheses

How the test works

Assumptions

See the [Handbook](#) for information on these topics.

Examples

The rattlesnake example is shown at the end of the “How to do the test” section.

How to do the test

For notes on linear models and conducting anova, see the “How to do the test” section in the *One-way anova* chapter of this book. For two-way anova with robust regression, see the chapter on *Two-way Anova with Robust Estimation*.

Two-way anova example

```
### -----
### Two-way anova, SAS example, pp. 179–180
### -----

Input = (
  id Sex      Genotype Activity
  1 male     ff        1.884
  2 male     ff        2.283
  3 male     fs        2.396
  4 female   ff        2.838
  5 male     fs        2.956
  6 female   ff        4.216
  7 female   ss        3.620
  8 female   ff        2.889
  9 female   fs        3.550
  10 male    fs        3.105
```

```

11 female fs 4.556
12 female fs 3.087
13 male ff 4.939
14 male ff 3.486
15 female ss 3.079
16 male fs 2.649
17 female fs 1.943
19 female ff 4.198
20 female ff 2.473
22 female ff 2.033
24 female fs 2.200
25 female fs 2.157
26 male ss 2.801
28 male ss 3.421
29 female ff 1.811
30 female fs 4.281
32 female fs 4.772
34 female ss 3.586
36 female ff 3.944
38 female ss 2.669
39 female ss 3.050
41 male ss 4.275
43 female ss 2.963
46 female ss 3.236
48 female ss 3.673
49 male ss 3.110
")

```

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

Means and summary statistics by group

```
library(FSA)
```

```
Sum = Summarize(Activity ~ Sex + Genotype,
                 data = Data)
```

```
Sum
```

	Sex	Genotype	n	mean	sd	min	Q1	median	Q3	max
1	female	ff	8	3.05025	0.9599032	1.811	2.363	2.864	4.008	4.216
2	male	ff	4	3.14800	1.3745115	1.884	2.183	2.884	3.849	4.939
3	female	fs	8	3.31825	1.1445388	1.943	2.189	3.318	4.350	4.772
4	male	fs	4	2.77650	0.3168433	2.396	2.586	2.802	2.993	3.105
5	female	ss	8	3.23450	0.3617754	2.669	3.028	3.158	3.594	3.673
6	male	ss	4	3.40175	0.6348109	2.801	3.033	3.266	3.634	4.275

```
### Add standard error
```

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

```
Sum
```

	Sex	Genotype	n	mean	sd	min	Q1	median	Q3	max	se
1	female	ff	8	3.05025	0.9599032	1.811	2.363	2.864	4.008	4.216	0.3393770
2	male	ff	4	3.14800	1.3745115	1.884	2.183	2.884	3.849	4.939	0.6872558
3	female	fs	8	3.31825	1.1445388	1.943	2.189	3.318	4.350	4.772	0.4046556
4	male	fs	4	2.77650	0.3168433	2.396	2.586	2.802	2.993	3.105	0.1584216
5	female	ss	8	3.23450	0.3617754	2.669	3.028	3.158	3.594	3.673	0.1279069
6	male	ss	4	3.40175	0.6348109	2.801	3.033	3.266	3.634	4.275	0.3174054

Interaction plot using summary statistics

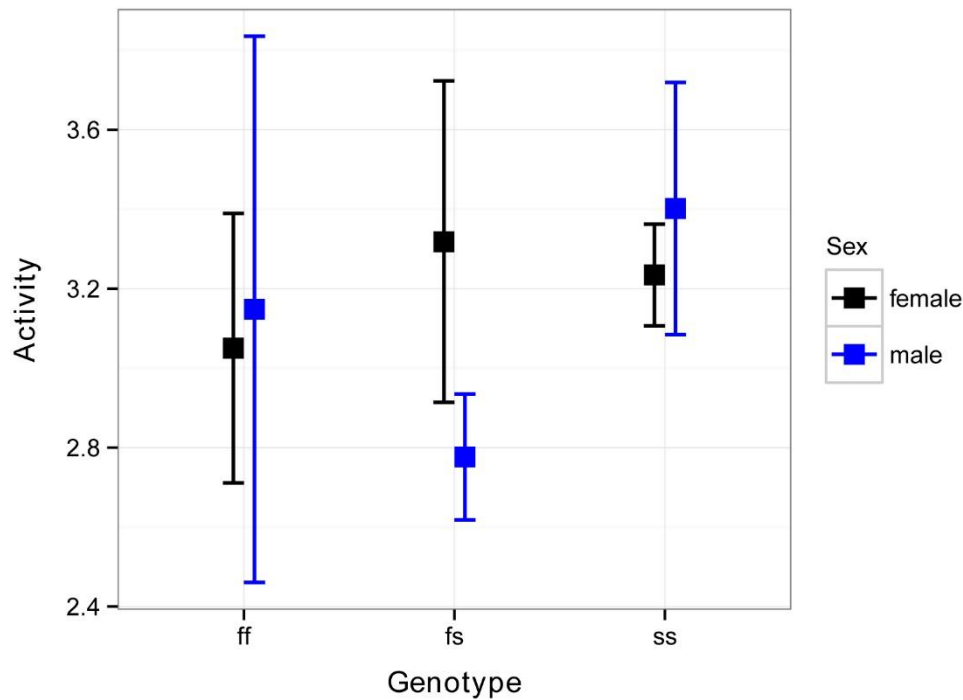
```
library(ggplot2)

pd = position_dodge(.2)

ggplot(Sum, aes(x=Genotype,
                y=mean,
                color=Sex)) +
  geom_errorbar(aes(ymin=mean-se,
                  ymax=mean+se),
               width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() +
  theme(axis.title.y = element_text(vjust= 1.8),
        axis.title.x = element_text(vjust= -0.5),
        axis.title = element_text(face = "bold")) +
  scale_color_manual(values = c("black", "blue"))+

  ylab("Activity")

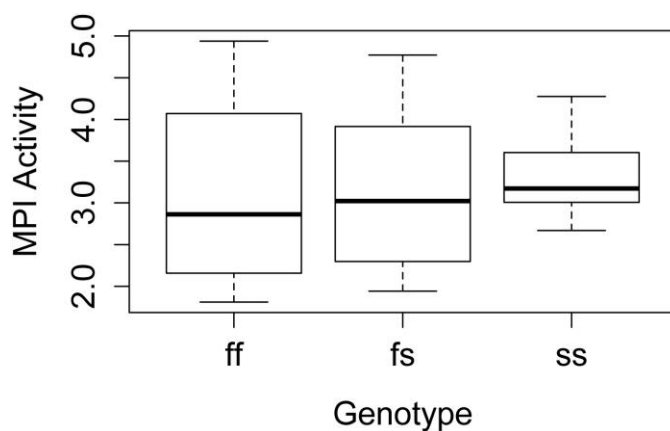
### You may see an error, "ymax not defined"
### In this case, it does not appear to affect anything
```



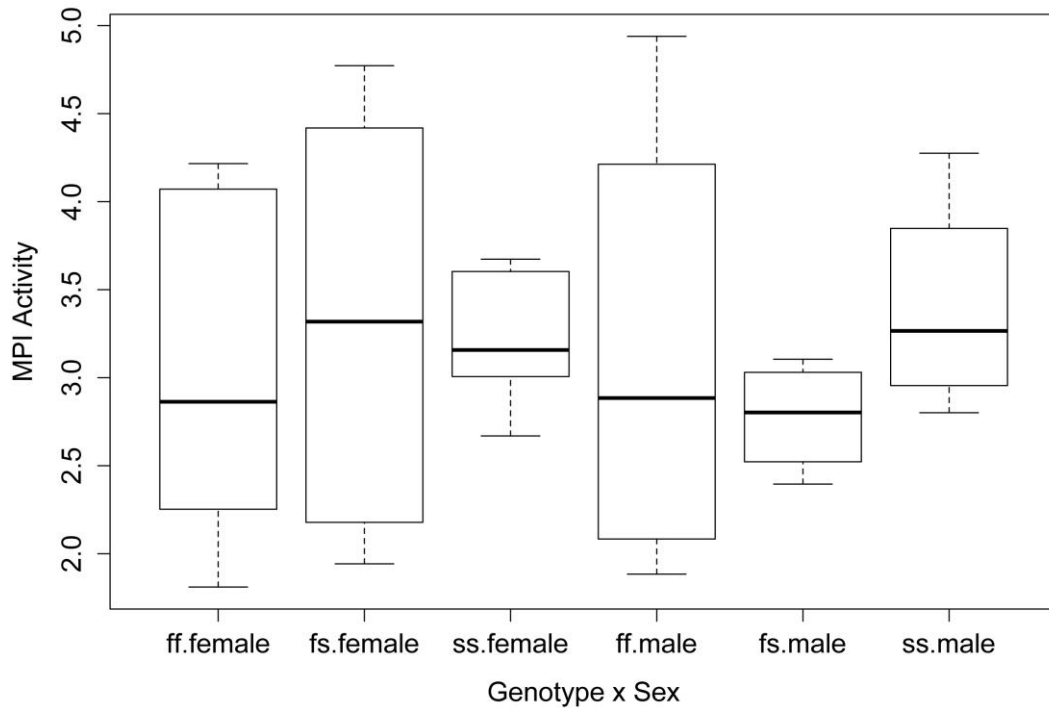
Interaction plot for a two-way anova. Square points represent means for groups, and error bars indicate standard errors of the mean.

Simple box plot of main effect and interaction

```
boxplot(Activity ~ Genotype,
  data = Data,
  xlab = "Genotype",
  ylab = "MPI Activity")
```



```
boxplot(Activity ~ Genotype:Sex,
  data = Data,
  xlab = "Genotype x Sex",
  ylab = "MPI Activity")
```



Fit the linear model and conduct ANOVA

```
model = lm(Activity ~ Sex + Genotype + Sex:Genotype,
            data=Data)
```

```
library(car)
```

```
Anova(model,
       type="II")    ### Type II sum of squares
```

```
### If you use type="III", you need the following line before the analysis
### options(contrasts = c("contr.sum", "contr.poly"))
```

	Sum Sq	Df	F value	Pr(>F)
Sex	0.0681	1	0.0861	0.7712
Genotype	0.2772	2	0.1754	0.8400
Sex:Genotype	0.8146	2	0.5153	0.6025
Residuals	23.7138	30		

```
anova(model)        # Produces type I sum of squares
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Sex	1	0.0681	0.06808	0.0861	0.7712
Genotype	2	0.2772	0.13862	0.1754	0.8400
Sex:Genotype	2	0.8146	0.40732	0.5153	0.6025
Residuals	30	23.7138	0.79046		

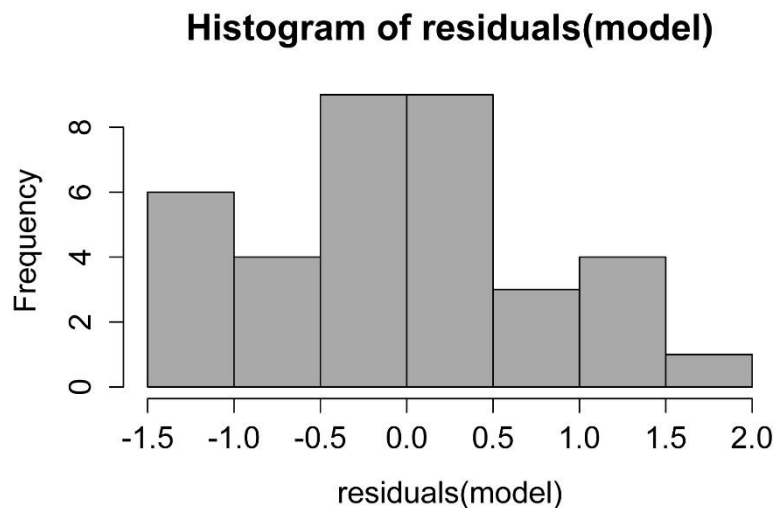
```
summary(model)      # Produces r-square, overall p-value, parameter estimates
```


Multiple R-squared: 0.04663, Adjusted R-squared: -0.1123

F-statistic: 0.2935 on 5 and 30 DF, p-value: 0.9128

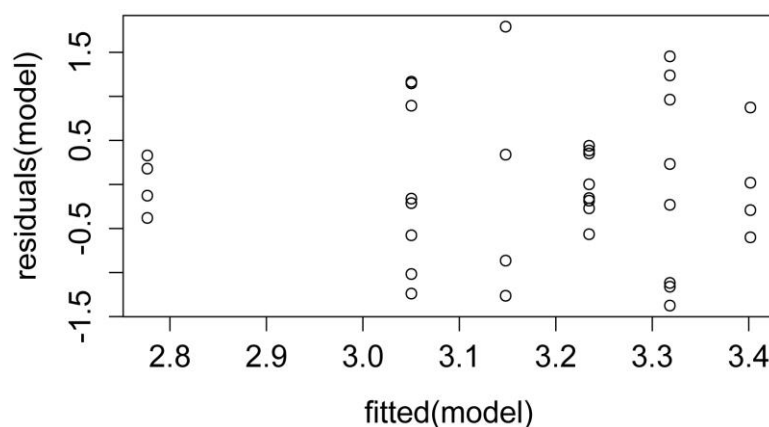
Checking assumptions of the model

```
hist(residuals(model),
     col="darkgray")
```



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model),
     residuals(model))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University:

condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

additional model checking plots with: `plot(model)`

```
### alternative: library(FSA); residPlot(model)
```

Post-hoc comparison of least-square means

For notes on least-square means, see the “Post-hoc comparison of least-square means” section in the *Nested anova* chapter in this book.

One advantage of the using the *lsmeans* package for post-hoc tests is that it can produce comparisons for interaction effects.

In general, if the interaction effect is significant, you will want to look at comparisons of means for the interactions. If the interaction effect is not significant but a main effect is, it is appropriate to look at comparisons among the means for that main effect. In this case, because no effect of *Sex*, *Genotype*, or *Sex:Genotype* was significant, we would not actually perform any mean separation test.

Mean separations for main factor with *lsmeans*

```
library(multcompView)
library(lsmeans)

lsmeans = lsmeans::lsmeans ### Uses the lsmeans function
                           ### from the lsmeans package,
                           ### not from the lmerTest package

leastsquare = lsmeans(model,
                       pairwise ~ Genotype,
                       adjust="tukey")

cld(leastsquare,
     alpha=.05,
     Letters=letters,
     adjust="tukey")

Genotype    lsmean      SE df lower.CL upper.CL .group
fs          3.047375 0.2722236 30 2.359065 3.735685 a
ff          3.099125 0.2722236 30 2.410815 3.787435 a
ss          3.318125 0.2722236 30 2.629815 4.006435 a

### Means sharing a letter in .group are not significantly different
```

Mean separations for interaction effect with *lsmeans*

```
library(multcompView)
library(lsmeans)

lsmeans = lsmeans::lsmeans ### Uses the lsmeans function
                           ### from the lsmeans package,
                           ### not from the lmerTest package

leastsquare = lsmeans(model,
```

```

pairwise ~ Sex:Genotype,
adjust="tukey")

cld(leastsquare,
    alpha=.05,
    Letters=letters,
    adjust="tukey")

Sex      Genotype  lsmmean      SE df lower.CL upper.CL .group
male    fs        2.77650  0.4445393 30 1.524666 4.028334 a
female  ff        3.05025  0.3143368 30 2.165069 3.935431 a
male    ff        3.14800  0.4445393 30 1.896166 4.399834 a
female  ss        3.23450  0.3143368 30 2.349319 4.119681 a
female  fs        3.31825  0.3143368 30 2.433069 4.203431 a
male    ss        3.40175  0.4445393 30 2.149916 4.653584 a

### Note that means are listed from low to high,
### not in the same order as Summarize

```

Graphing the results

Simple bar plot with categories and no error bars

```

### Re-enter data as matrix

Input =("
Sex      ff      fs      ss
Female  3.05025  3.31825  3.23450
Male    3.14800  2.77650  3.40175
")

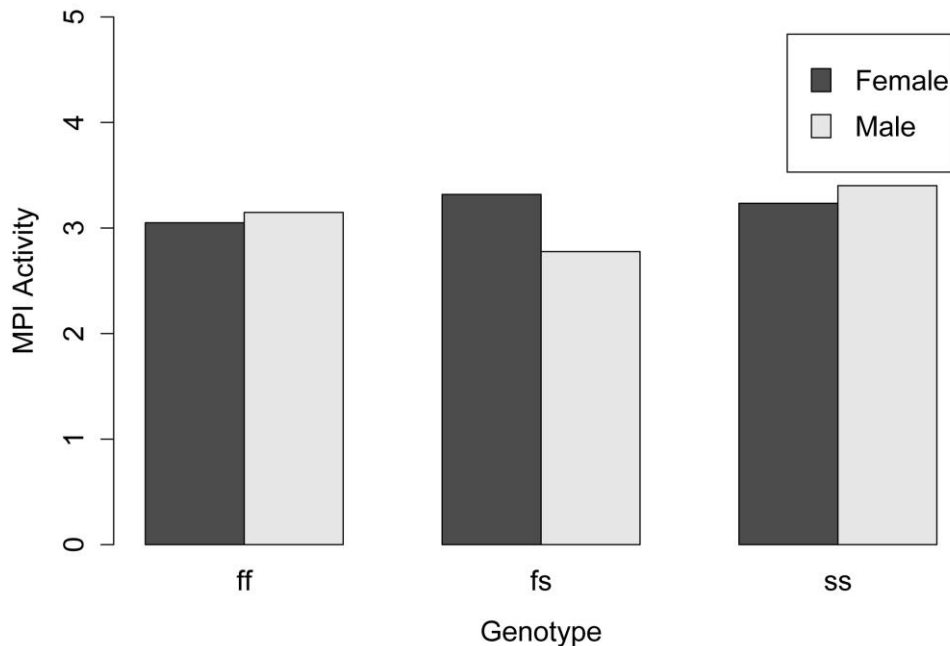
Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

           ff      fs      ss
Female 3.05025 3.31825 3.23450
Male   3.14800 2.77650 3.40175

barplot(Matriz,
        beside=TRUE,
        legend=TRUE,
        ylim=c(0, 5),
        xlab="Genotype",
        ylab="MPI Activity")

```



Bar plot with error bars with *ggplot2*

This plot uses the data frame created by *Summarize* in *FSA*. Error bars indicate standard error of the means (*se* in the data frame).

```
library(FSA)
```

```
Sum = Summarize(Activity ~ Sex + Genotype,  
                data = Data)
```

```
Sum
```

```
### Add standard error
```

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

```
Sum
```

	Sex	Genotype	n	mean	sd	min	Q1	median	Q3	max	se
1	female	ff	8	3.05025	0.9599032	1.811	2.363	2.864	4.008	4.216	0.3393770
2	male	ff	4	3.14800	1.3745115	1.884	2.183	2.884	3.849	4.939	0.6872558
3	female	fs	8	3.31825	1.1445388	1.943	2.189	3.318	4.350	4.772	0.4046556
4	male	fs	4	2.77650	0.3168433	2.396	2.586	2.802	2.993	3.105	0.1584216
5	female	ss	8	3.23450	0.3617754	2.669	3.028	3.158	3.594	3.673	0.1279069
6	male	ss	4	3.40175	0.6348109	2.801	3.033	3.266	3.634	4.275	0.3174054

```
### Plot adapted from:
```

```
### shinyapps.stat.ubc.ca/r-graph-catalog/
```

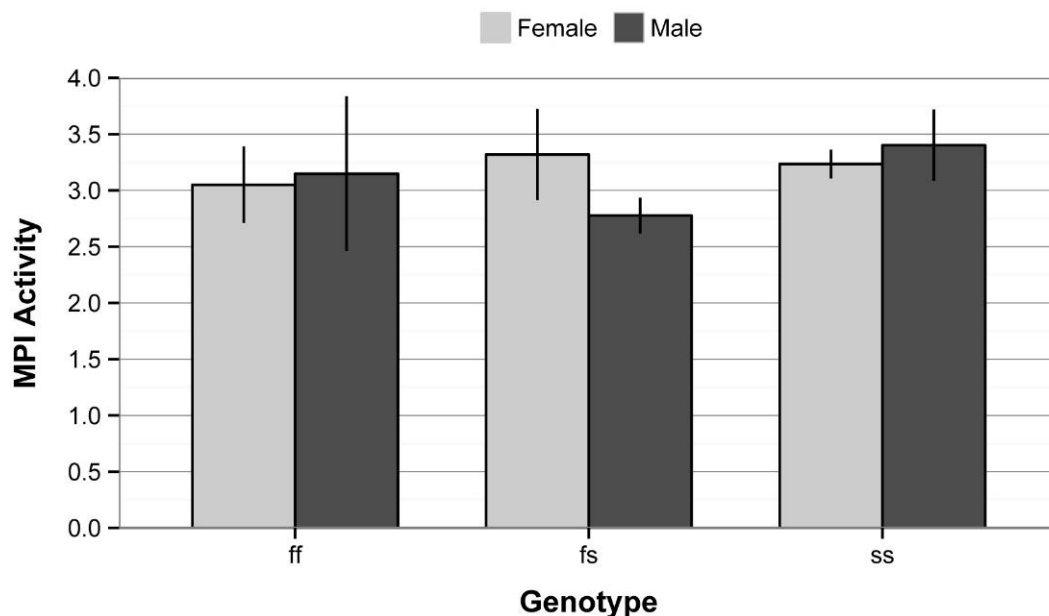
```
library(ggplot2)
```

```
library(grid)
```

```

ggplot(Sum,
  aes(x = Genotype,
      y = mean,
      fill = Sex,
      ymax=mean+se,
      ymin=mean-se)) +
  geom_bar(stat="identity", position = "dodge", width = 0.7) +
  geom_bar(stat="identity", position = "dodge",
          colour = "black", width = 0.7,
          show.legend = FALSE) +
  scale_y_continuous(breaks = seq(0, 4, 0.5),
                    limits = c(0, 4),
                    expand = c(0, 0)) +
  scale_fill_manual(name = "Count type",
                   values = c('grey80', 'grey30'),
                   labels = c("Female",
                              "Male")) +
  geom_errorbar(position=position_dodge(width=0.7),
               width=0.0, size=0.5, color="black") +
  labs(x = "Genotype",
       y = "MPI Activity") +
  ## ggtitle("Main title") +
  theme_bw() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line(colour = "grey50"),
        plot.title = element_text(size = rel(1.5),
                                   face = "bold", vjust = 1.5),
        axis.title = element_text(face = "bold"),
        legend.position = "top",
        legend.title = element_blank(),
        legend.key.size = unit(0.4, "cm"),
        legend.key = element_rect(fill = "black"),
        axis.title.y = element_text(vjust= 1.8),
        axis.title.x = element_text(vjust= -0.5)
)

```



Bar plot for a two-way anova. Bar heights represent means for groups, and error bars indicate standard errors of the mean.

#

Rattlesnake example – two-way anova without replication, repeated measures

This example could be interpreted as two-way anova without replication or as a one-way repeated measures experiment. Below it is analyzed as a two-way fixed effects model using the *lm* function, and as a mixed effects model using the *nlme* package and *lme4* packages.

```
### -----
### Two-way anova, rattlesnake example, pp. 177–178
### -----
```

```
Input = ("
Day  Snake  Openings
1    D1      85
1    D3     107
1    D5      61
1    D8      22
1    D11     40
1    D12     65
2    D1      58
2    D3      51
2    D5      60
2    D8      41
2    D11     45
2    D12     27
3    D1      15
3    D3      30
3    D5      68
3    D8      63
3    D11     28
3    D12      3
4    D1      57
4    D3      12
4    D5      36
4    D8      21
4    D11     10
4    D12     16
")
```

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

```
### Treat Day as a factor variable
```

```
Data$Day = as.factor(Data$Day)
```

Using two-way fixed effects model

Means and summary statistics by group

```
library(FSA)
```

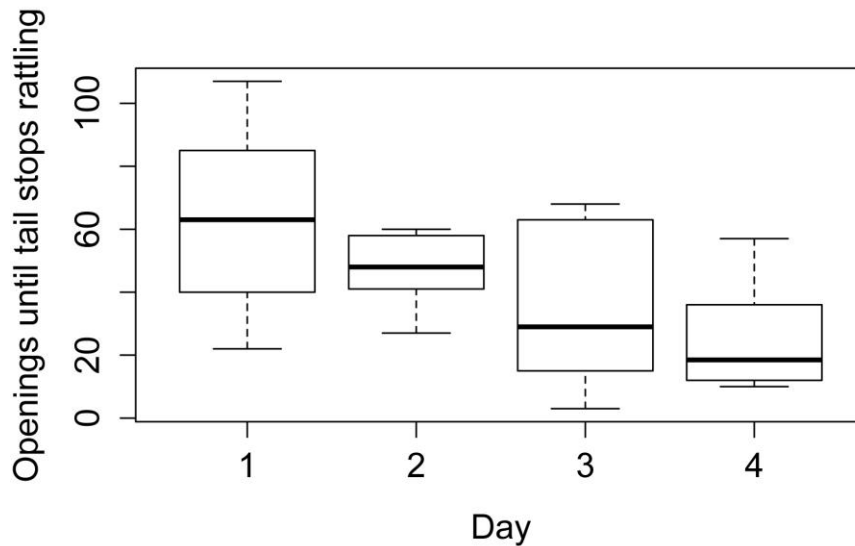
```
Sum = Summarize(Openings ~ Day,
                data = Data)
```

```
Sum
```

	Day	n	mean	sd	min	Q1	median	Q3	max
1	1	6	63.33333	30.45434	22	45.25	63.0	80.00	107
2	2	6	47.00000	12.21475	27	42.00	48.0	56.25	60
3	3	6	34.50000	25.95958	3	18.25	29.0	54.75	68
4	4	6	25.33333	18.08498	10	13.00	18.5	32.25	57

Simple box plots

```
boxplot(Openings ~ Day,
        data = Data,
        xlab = "Day",
        ylab = "Openings until tail stops rattling")
```

Fit the linear model and conduct ANOVA

```
model = lm(Openings ~ Day + Snake,
            data=Data)
```

```
library(car)
```

```
Anova(model, type="II")      # Type II sum of squares
```

```
### If you use type="III", you need the following line before the analysi
### options(contrasts = c("contr.sum", "contr.poly"))
```

	Sum Sq	Df	F value	Pr(>F)
Day	4877.8	3	3.3201	0.04866 *
Snake	3042.2	5	1.2424	0.33818
Residuals	7346.0	15		

```
anova(model) # Produces type I sum of squares
```

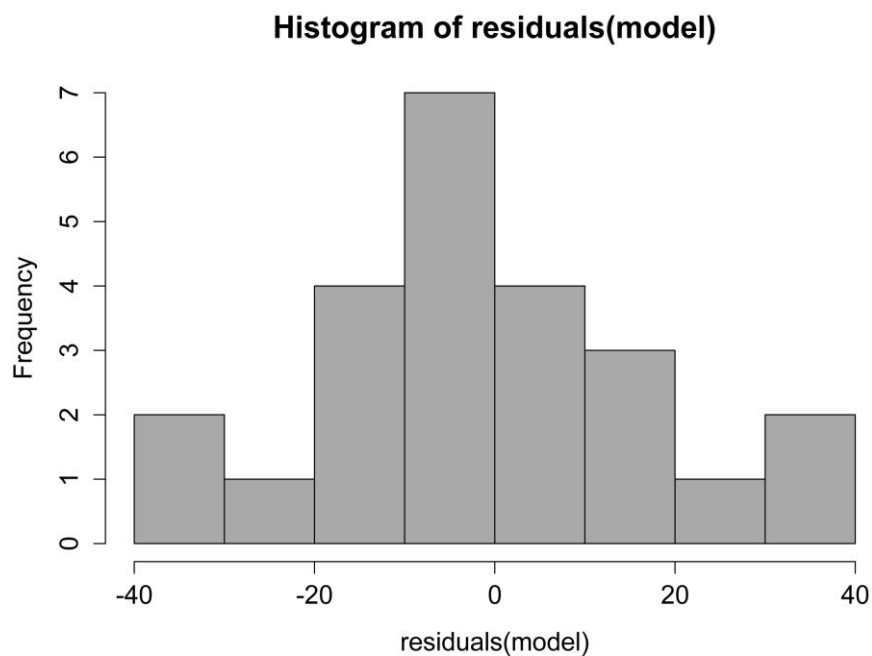
	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Day	3	4877.8	1625.93	3.3201	0.04866 *
Snake	5	3042.2	608.44	1.2424	0.33818
Residuals	15	7346.0	489.73		

```
summary(model) # Produces r-square, overall p-value,
                # parameter estimates
```

Multiple R-squared: 0.5188, Adjusted R-squared: 0.2622
 F-statistic: 2.022 on 8 and 15 DF, p-value: 0.1142

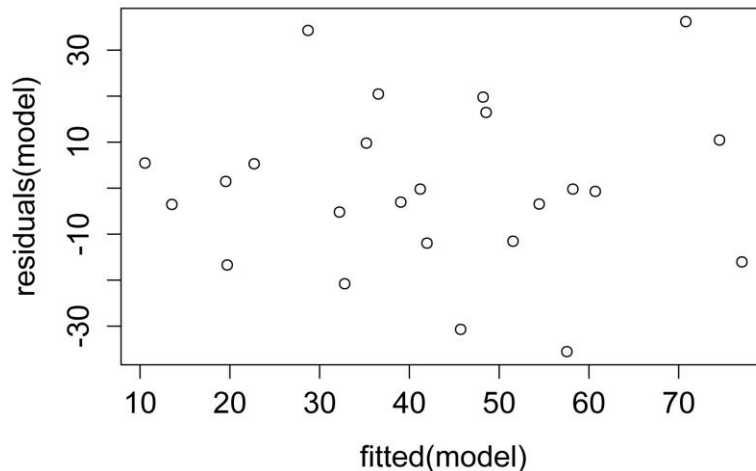
Checking assumptions of the model

```
hist(residuals(model),
     col="darkgray")
```



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model),
     residuals(model))
```

A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University:
condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model)
### alternative: library(FSA); residPlot(model)
```

Mean separations for main factor with *lsmeans*

For notes on least-square means, see the “Post-hoc comparison of least-square” means section in the *Nested anova* chapter in this book. For other mean separation techniques for a main factor in anova, see “Tukey and Least Significant Difference mean separation tests (pairwise comparisons)” section in the *One-way anova* chapter.

```
library(multcompView)
library(lsmeans)

lsmeans = lsmeans::lsmeans ### Uses the lsmeans function
                           ### from the lsmeans package,
                           ### not from the lmerTest package

leastsquare = lsmeans(model,
                      pairwise ~ Day,
                      adjust="tukey")

cld(leastsquare,
     alpha=.05,
     Letters=letters,
     adjust="tukey")
```

Day	lsmean	SE	df	lower.CL	upper.CL	.group
4	25.33333	9.034476	15	-0.2085871	50.87525	a
3	34.50000	9.034476	15	8.9580796	60.04192	ab
2	47.00000	9.034476	15	21.4580796	72.54192	ab
1	63.33333	9.034476	15	37.7914129	88.87525	b

Means sharing a letter in .group are not significantly different

Using mixed effects model with nlme

This is an abbreviated example using the *lme* function in the *nlme* package.

```
library(nlme)

model = lme(Openings ~ Day, random=~1|Snake,
            data=Data,
            method="REML")

anova.lme(model,
           type="sequential",
           adjustSigma = FALSE)
```

	numDF	denDF	F-value	p-value
(Intercept)	1	15	71.38736	<.0001
Day	3	15	3.32005	0.0487

```
library(multcompView)
library(lsmmeans)
```

```
lsmmeans = lsmmeans::lsmmeans ### Uses the lsmmeans function
                                ### from the lsmmeans package,
                                ### not from the lmerTest package
```

```
leastsquare = lsmmeans(model,
                       pairwise ~ Day,
                       alpha=.05,
                       adjust="tukey")
```

```
cld(leastsquare,
    alpha=.05,
    Letters=letters,
    adjust="tukey")
```

Day	lsmmean	SE	df	Lower.CL	upper.CL	.group
4	25.33333	9.304196	5	-9.9416542	60.60832	a
3	34.50000	9.304196	5	-0.7749876	69.77499	ab
2	47.00000	9.304196	5	11.7250124	82.27499	ab
1	63.33333	9.304196	5	28.0583458	98.60832	b

Means sharing a letter in .group are not significantly different

Using mixed effects model with lmer

This is an abbreviated example using the *lmer* function in the *lme4* package.

```
library(lme4)
library(lmerTest)

model = lmer(Openings ~ Day + (1|Snake),
```

```
data=Data,
REML=TRUE)
```

```
anova(model)
```

```
Analysis of Variance Table of type III with Satterthwaite
approximation for degrees of freedom
```

```
Sum Sq Mean Sq NumDF DenDF F.value Pr(>F)
Day 4877.8 1625.9 3 15 3.3201 0.04866 *
```

```
rand(model)
```

```
Analysis of Random effects Table:
Chi.sq Chi.DF p.value
Snake 0.0915 1 0.8
```

Least square means with the *lsmeans* package

```
library(multcompView)
library(lsmeans)
```

```
lsmeans = lsmeans::lsmeans ### Uses the lsmeans function
                             ### from the lsmeans package,
                             ### not from the lmerTest package
```

```
leastsquare = lsmeans(model,
                      pairwise ~ Day,
                      alpha=.05,
                      adjust="tukey")
```

```
cld(leastsquare,
    alpha=.05,
    Letters=letters,
    adjust="tukey")
```

```
Day    lsmean      SE    df  lower.CL upper.CL .group
4      25.33333 9.304196 19.81 -0.1441779 50.81084 a
3      34.50000 9.304196 19.81  9.0224887 59.97751 ab
2      47.00000 9.304196 19.81 21.5224887 72.47751 ab
1      63.33333 9.304196 19.81 37.8558221 88.81084 b
```

```
Degrees-of-freedom method: satterthwaite
Confidence level used: 0.95
Conf-level adjustment: sidak method for 4 estimates
P value adjustment: tukey method for comparing a family of 4 estimates
significance level used: alpha = 0.05
```

```
### Means sharing a letter in .group are not significantly different
```

Least square means using the *lmerTest* package

```
lsmeans = lmerTest::lsmeans ### Uses the lsmeans function
                                ### from the lmerTest package,
                                ### not from the lsmeans package
```

```
LT = lsmeans(model,
              test.effs = "Day")
```

```
LT
```

```
Least Squares Means table:
```

	Day	Estimate	Standard Error	DF	t-value	Lower CI	Upper CI	p-value
Day 1	1.0	63.33	9.30	19.8	6.81	43.91	82.8	<2e-16 ***
Day 2	2.0	47.00	9.30	19.8	5.05	27.58	66.4	1e-04 ***
Day 3	3.0	34.50	9.30	19.8	3.71	15.08	53.9	0.001 **
Day 4	4.0	25.33	9.30	19.8	2.72	5.91	44.8	0.013 *

```
PT = diffLSmeans(model,
                  test.effs="Day")
```

```
PT
```

```
Differences of LSMEANS:
```

	Estimate	Standard Error	DF	t-value	Lower CI	Upper CI	p-value
Day 1 - 2	16.3	12.78	15.0	1.28	-10.90	43.6	0.220
Day 1 - 3	28.8	12.78	15.0	2.26	1.60	56.1	0.039 *
Day 1 - 4	38.0	12.78	15.0	2.97	10.77	65.2	0.009 **
Day 2 - 3	12.5	12.78	15.0	0.98	-14.73	39.7	0.343
Day 2 - 4	21.7	12.78	15.0	1.70	-5.57	48.9	0.111
Day 3 - 4	9.2	12.78	15.0	0.72	-18.07	36.4	0.484

```
### Extract lsmeans table
```

```
Sum = PT$diffs.lsmeans.table
```

```
### Extract comparisons and p-values
```

```
Comparison = rownames(Sum)
```

```
P.value = Sum$'p-value'
```

```
### Adjust p-values
```

```
P.value.adj = p.adjust(P.value,
                        method = "none")
```

```
### Fix names of comparisons
```

```
Comparison = gsub("-", "- Day", Comparison)
```

```
### Produce compact letter display
```

```
library(rcompanion)
```

```
cldList(comparison = Comparison,
        p.value     = P.value.adj,
        threshold = 0.05)
```

	Group	Letter	MonoLetter
1	Day1	a	a
2	Day2	ab	ab
3	Day3	b	b
4	Day4	b	b

```
# # #
```

Two-way Anova with Robust Estimation

A two-way anova using robust estimators can be performed with the *WRS2* package. Options for estimators are M-estimators, trimmed means, and medians. This type of analysis is resistant to deviations from the assumptions of the traditional ordinary-least-squares anova, and are robust to outliers. However, it may not be appropriate for data that deviate too widely from parametric assumptions.

The main analysis using M-estimators for a two-way anova is conducted with the *pbad2way* function in the *WRS2* package. Post-hoc tests can be performed with the *mcp2a* function in the *WRS2* package or with my custom functions *pairwiseRobustTest* and *pairwiseRobustMatrix*, which rely on the *pb2gen* function in *WRS2*.

My custom function *groupwiseHuber* uses the *HuberM* function in the *DescTools* package to determine the Huber M-estimators across groups in a data frame.

For more information on robust tests available in the *WRS2* package, see:

```
help(package="WRS2")
```

Consult the chapter on *Two-way Anova* for general consideration about conducting analysis of variance.

Packages used in this chapter

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

```
if(!require(rcompanion)){install.packages("rcompanion")}
if(!require(ggplot2)){install.packages("ggplot2")}
if(!require(WRS2)){install.packages("WRS2")}
if(!require(multcompView)){install.packages("multcompView")}
if(!require(psych)){install.packages("psych")}
```

Example

```
### -----
### Two-way anova with robust estimators, hypothetical data
### Using WRS2 package
### -----

Input = ("
Factor.A Factor.B Response
1        x        0.9
1        y        1.4
1        x        1.3
1        y        2.0
1        x        1.6
1        y        2.6
m        x        2.4
m        y        3.6
m        x        2.8
m        y        3.7
m        x        3.2
m        y        3.0
n        x        1.6
n        y        1.2
n        x        2.0
n        y        1.9
n        x        2.7
n        y        0.9
")

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

Produce Huber M-estimators and confidence intervals by group

```
library(rcompanion)

Sum = groupwiseHuber(Response ~ Factor.A + Factor.B,
                      data = Data,
                      conf.level=0.95,
                      conf.type="wald")

Sum
```

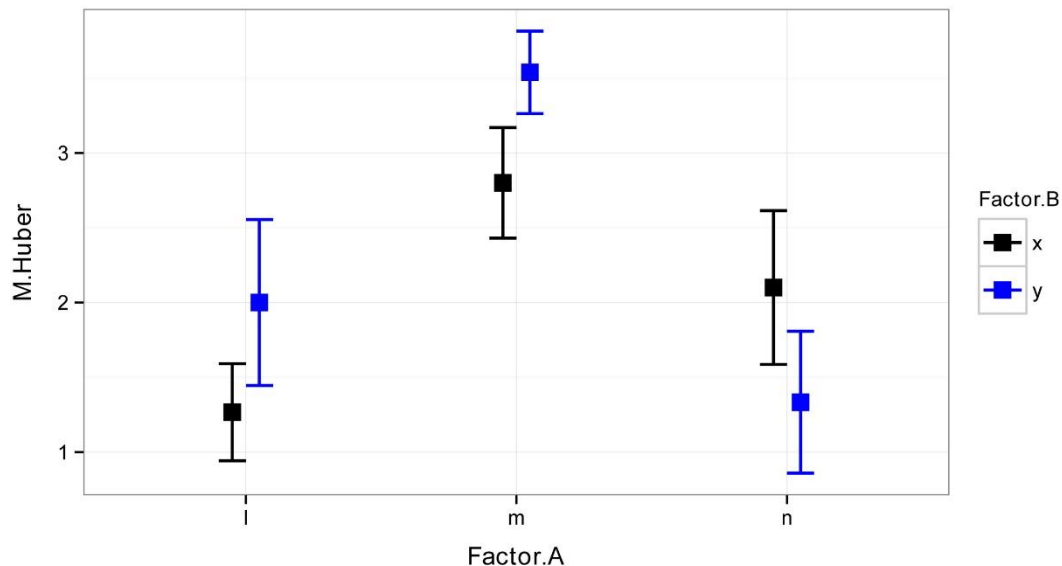
	Factor.A	Factor.B	n	M.Huber	lower.ci	upper.ci
1	1	x	3	1.266667	0.9421910	1.591142
2	1	y	3	2.000000	1.4456385	2.554362
3	m	x	3	2.800000	2.4304256	3.169574
4	m	y	3	3.538805	3.2630383	3.814572
5	n	x	3	2.100000	1.5855743	2.614426
6	n	y	3	1.333333	0.8592063	1.807460

Interaction plot using summary statistics

```
library(ggplot2)

pd = position_dodge(.2)

ggplot(Sum, aes(x=Factor.A,
                y=M.Huber,
                color=Factor.B)) +
  geom_errorbar(aes(ymin=lower.ci,
                  ymax=upper.ci),
              width=.2, size=0.7, position=pd) +
  geom_point(shape=15, size=4, position=pd) +
  theme_bw() +
  theme(
    axis.title.y = element_text(vjust= 1.8),
    axis.title.x = element_text(vjust= -0.5),
    axis.title = element_text(face = "bold")) +
  scale_color_manual(values = c("black", "blue"))
```

***Two-way analysis of variance for M-estimators***

The *est* = "mom" option uses a modified M-estimator for the analysis. To analyze using medians, use the *est*= "median" option in the *pbad2way* function in the *WRS2* package. To analyze using trimmed means, use the *t2way* function in the *WRS2* package.

```
library(WRS2)

pbad2way(Response ~ Factor.A + Factor.B + Factor.A:Factor.B,
  data = Data,
  est = "mom",      # modified M-estimator
  nboot = 5000)    # number of bootstrap samples
                  # a higher number will take longer to compute

pbad2way(formula = Response ~ Factor.A + Factor.B + Factor.A:Factor.B,
```

```

data = Data, est = "mom", nboot = 3000)

p.value
Factor.A      0.0000
Factor.B      0.3403
Factor.A:Factor.B 0.0460

```

Produce post-hoc tests for main effects with mcp2a

```

post = mcp2a(Response ~ Factor.A + Factor.B + Factor.A:Factor.B,
             data = Data,
             est = "mom",      # M-estimator
             nboot = 5000)    # number of bootstrap samples

post$contrasts

post

```

	Factor.A1	Factor.A2	Factor.A3	Factor.B1	Factor.A1: Factor.B1	Factor.A2: Factor.B1	Factor.A3: Factor.B1
l_x	1	1	0	1	1	1	0
l_y	1	1	0	-1	-1	-1	0
m_x	-1	0	1	1	-1	0	1
m_y	-1	0	1	-1	1	0	-1
n_x	0	-1	-1	1	0	-1	-1
n_y	0	-1	-1	-1	0	1	1

	v1	ci.lower	ci.upper	p-value
Factor.A1	-3.18333	-4.20000	-1.60000	0.00000
Factor.A2	-0.16667	-1.70000	1.36667	0.40233
Factor.A3	3.01667	1.40000	4.05000	0.00000
Factor.B1	-0.81667	-2.28333	1.00000	0.22233
Factor.A1:Factor.B1	0.11667	-1.50000	1.16667	0.48033
Factor.A2:Factor.B1	-1.50000	-3.10000	0.00000	0.01767
Factor.A3:Factor.B1	-1.61667	-2.80000	0.00000	0.01433

```

### The Factor.A1 contrast compares l to m; since it is significant,
###   l is significantly different than m.

```

```

### The Factor.A2 contrast compares l to n; since it is not significant,
###   l is not significantly different than n.

```

Produce post-hoc tests for main effects with pairwiseRobustTest or pairwiseRobustMatrix

Table and compact letter display with pairwiseRobustTest

```

### Order groups by median

Data$Factor.A = factor(Data$Factor.A,
                       levels = c("n", "l", "m"))

```



```
### Pairwise robust test
```

```
library(rcompanion)
```

```
PT = pairwiseRobustTest(x      = Data$Response,
                        g      = Data$Factor.A,
                        est    = "mom",
                        nboot  = 5000,
                        method = "fdr")
# adjust p-values; see ?p.adjust for options
```

```
PT
```

	Comparison	Statistic	p.value	p.adjust
1	n - l = 0	0.08333	0.7204	0.7204
2	n - m = 0	-1.4	0.0014	0.0021
3	l - m = 0	-1.483	6e-04	0.0018

```
### p-values may differ
```

```
### Produce compact letter display
```

```
library(rcompanion)
```

```
cldList(comparison = PT$Comparison,
        p.value    = PT$p.adjust,
        threshold  = 0.05)
```

	Group	Letter	MonoLetter
1	n	a	a
2	l	a	a
3	m	b	b

Compact letter display output with *pairwiseRobustMatrix*

```
### Order groups by median
```

```
Data$Factor = factor(Data$Factor,
                     levels = c("n", "l", "m"))
```

```
### Pairwise robust tests
```

```
library(rcompanion)
```

```
PM = pairwiseRobustMatrix(x      = Data$Response,
                          g      = Data$Factor.A,
                          est    = "mom",
                          nboot  = 5000,
                          method = "fdr")
# adjust p-values; see ?p.adjust for options
```

```
PM$Adjusted
```

```
      n      l      m
n 1.0000 0.7128 6e-04
l 0.7128 1.0000 0e+00
m 0.0006 0.0000 1e+00
```

```
### p-values may differ
```

```
library(multcompView)
```

```
multcompLetters(PM$Adjusted,
                 compare="<",
                 threshold=0.05,
                 Letters=letters,
                 reversed = FALSE)
```

```
      n      l      m
"a" "a" "b"
```

Produce post-hoc tests for interaction effect

```
### Create a factor which is the interaction of Factor.A and Factor.B
```

```
Data$Factor.int = interaction (Data$Factor.A, Data$Factor.B)
```

```
### Order groups by median
```

```
Data$Factor.int = factor(Data$Factor.int,
                         levels = c("m.y", "m.x", "n.x", "l.y", "n.y", "l.x"))
```

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

```
library(psych)
```

```
headTail(Data)
```

	Factor.A	Factor.B	Response	Factor.int
8	m	y	3.6	m.y
10	m	y	3.7	m.y
12	m	y	3	m.y
7	m	x	2.4	m.x
...	<NA>	<NA>	...	<NA>
18	n	y	0.9	n.y
1	l	x	0.9	l.x
3	l	x	1.3	l.x
5	l	x	1.6	l.x

Table and compact letter display with *pairwiseRobustTest*

```
library(rcompanion)

PT = pairwiseRobustTest(x      = Data$Response,
                        g      = Data$Factor.int,
                        est    = "mom",
                        nboot  = 5000,
                        method = "fdr")
# adjust p-values; see ?p.adjust for options
```

```
PT
```

	Comparison	Statistic	p.value	p.adjust
1	m.y - m.x = 0	-0.85	0.1348	0.1615
2	m.y - n.x = 0	-1.55	0	0.0000
3	m.y - l.y = 0	-1.65	0	0.0000
4	m.y - n.y = 0	-2.317	0	0.0000
5	m.y - l.x = 0	-2.383	0	0.0000
6	m.x - n.x = 0	-0.7	0.1312	0.1615
7	m.x - l.y = 0	0.8	0.1228	0.1615
8	m.x - n.y = 0	1.467	0	0.0000
9	m.x - l.x = 0	-1.533	0	0.0000
10	n.x - l.y = 0	0.1	0.7798	0.8355
11	n.x - n.y = 0	0.7667	0.1344	0.1615
12	n.x - l.x = 0	0.8333	0.0664	0.1423
13	l.y - n.y = 0	-0.6667	0.14	0.1615
14	l.y - l.x = 0	-0.7333	0.1296	0.1615
15	n.y - l.x = 0	-0.06667	0.944	0.9440

```
### p-values may differ
```

```
### Produce compact letter display
```

```
library(rcompanion)

cldList(comparison = PT$Comparison,
        p.value    = PT$p.adjust,
        threshold  = 0.05)
```

	Group	Letter	MonoLetter
1	m.y	a	a
2	m.x	ab	ab
3	n.x	bc	bc
4	l.y	bc	bc
5	n.y	c	c
6	l.x	c	c

Compact letter display output with *pairwiseRobustMatrix*

```
### Order groups by median
```

```
Data$Factor.int = factor(Data$Factor.int,
```

```

levels = c("m.y", "m.x", "n.x", "l.y", "n.y", "l.x"))

### Pairwise robust tests

library(rcompanion)

PM = pairwiseRobustMatrix(x      = Data$Response,
                          g      = Data$Factor.int,
                          est    = "mom",
                          nboot  = 5000,
                          method = "fdr")
# adjust p-values; see ?p.adjust for options

PM

$Unadjusted
      m.y    m.x    n.x    l.y    n.y    l.x
m.y NA 0.1312 0.000 0.0000 0.0000 0.0000
m.x NA      NA 0.126 0.1320 0.0000 0.0000
n.x NA      NA      NA 0.7638 0.1328 0.0680
l.y NA      NA      NA      NA 0.1304 0.1408
n.y NA      NA      NA      NA      NA 0.9318
l.x NA      NA      NA      NA      NA      NA

$Method
[1] "fdr"

$Adjusted
      m.y    m.x    n.x    l.y    n.y    l.x
m.y 1.0000 0.1625 0.0000 0.0000 0.0000 0.0000
m.x 0.1625 1.0000 0.1625 0.1625 0.0000 0.0000
n.x 0.0000 0.1625 1.0000 0.8184 0.1625 0.1457
l.y 0.0000 0.1625 0.8184 1.0000 0.1625 0.1625
n.y 0.0000 0.0000 0.1625 0.1625 1.0000 0.9318
l.x 0.0000 0.0000 0.1457 0.1625 0.9318 1.0000

### p-values may differ

library(multcompView)

multcompLetters(PM$Adjusted,
                compare="<",
                threshold=0.05,
                Letters=letters,
                reversed = FALSE)

m.y m.x n.x l.y n.y l.x
"a" "ab" "bc" "bc" "c" "c"

### Note, means are not ordered from largest to smallest

```

#

Paired t-test

Paired t-tests can be conducted with the `t.test` function in the native `stats` package using the `paired=TRUE` option. Data can be in long format or short format. Examples of each are shown in this chapter.

As a non-parametric alternative to paired t-tests, a permutation test can be used. An example is shown in the “Permutation test for dependent samples” section of this chapter.

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Paired t-test](#)

[SAEPPER: One-way Permutation Test of Symmetry for Paired Ordinal Data](#)

Packages used in this chapter

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

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

When to use it

The horseshoe crab example is shown at the end of the “How to do the test” section.

Null hypothesis

Assumption

How the test works

See the [Handbook](#) for information on these topics.

Examples

The flicker feather example is shown in the “How to do the test” section.

Graphing the results

Plots are shown in the “How to do the test” section.

How to do the test

Paired t-test, data in wide format, flicker feather example

```
### -----
### Paired t-test, Flicker feather example, p. 185
### -----

Input = (
  Bird   Typical   odd
A       -0.255    -0.324
```

```
B      -0.213   -0.185
C      -0.190   -0.299
D      -0.185   -0.144
E      -0.045   -0.027
F      -0.025   -0.039
G      -0.015   -0.264
H       0.003   -0.077
I       0.015   -0.017
J       0.020   -0.169
K       0.023   -0.096
L       0.040   -0.330
M       0.040   -0.346
N       0.050   -0.191
O       0.055   -0.128
P       0.058   -0.182
")
```

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

Paired t-test

```
t.test(Data$Typical,
       Data$Odd,
       paired=TRUE,
       conf.level=0.95)
```

```
t = 4.0647, df = 15, p-value = 0.001017
```

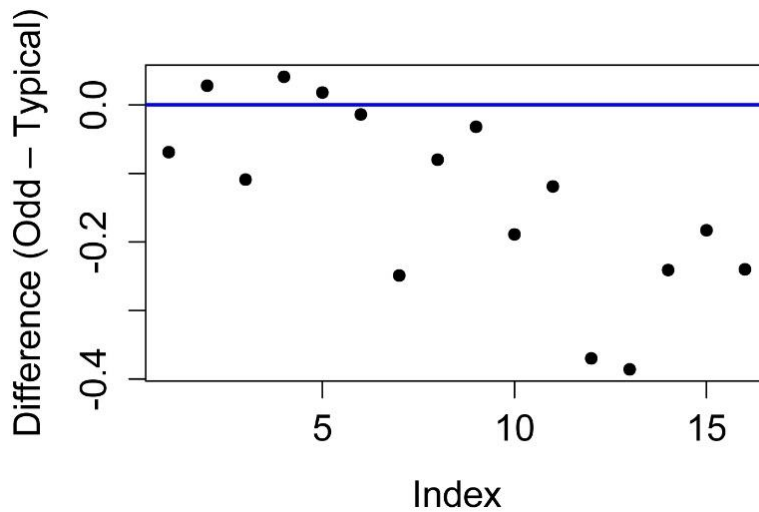
```
mean of the differences
0.137125
```

Simple plot of differences

```
Difference = Data$Odd - Data$Typical

plot(Difference,
     pch = 16,
     ylab="Difference (Odd - Typical)")

abline(0,0, col="blue", lwd=2)
```

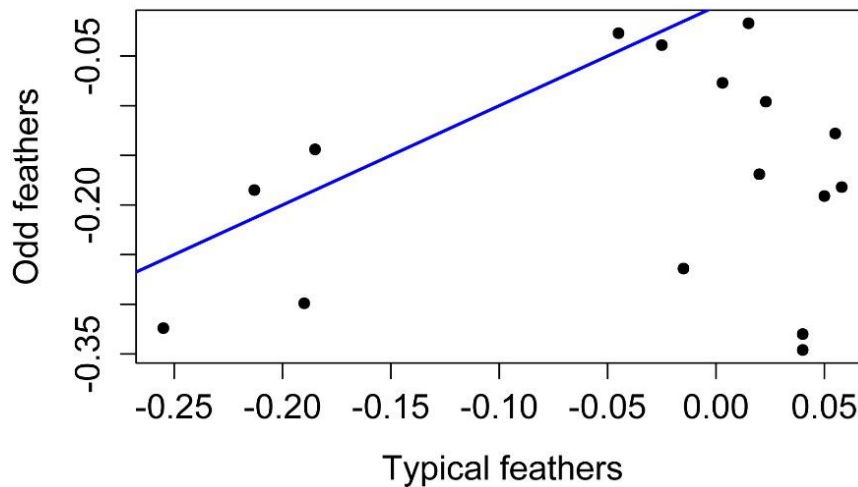


A simple plot of differences between one sample and the other. Points below the blue line indicate observations where *Typical* is greater than *Odd*, that is where $(Odd - Typical)$ is negative.

Simple 1-to-1 plot of values

```
plot(Data$Typical, Data$Odd,
     pch = 16,
     xlab="Typical feathers",
     ylab="Odd feathers")
```

```
abline(0,1, col="blue", lwd=2)
```

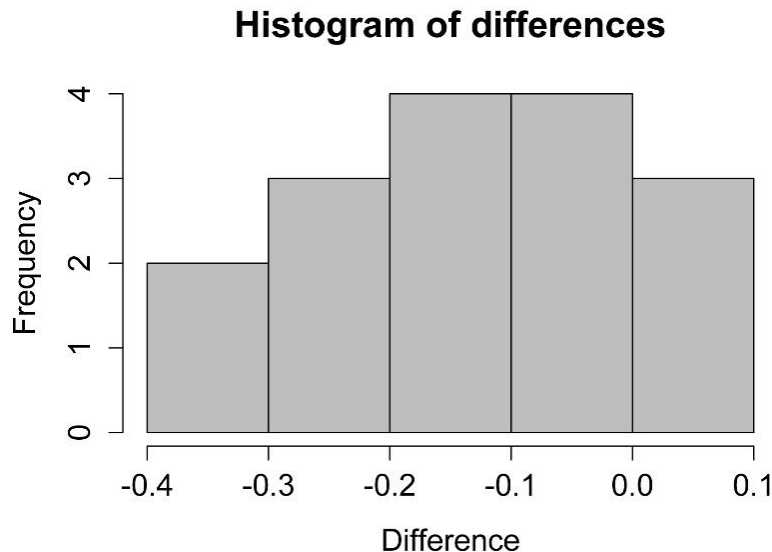


Plot of paired samples from a paired t-test. Circles below or to the right of the blue one-to-one line indicate observations with a higher value for *Typical* than for *Odd*.

Checking assumptions of the model

```
Difference = Data$Odd - Data$Typical

hist(Difference,
     col="gray",
     main="Histogram of differences",
     xlab="Difference")
```



Histogram of differences of two populations from a paired t-test. Distribution of differences should be approximately normal. Bins with negative values indicate observations with a higher value for Typical than for Odd.

Graphing the results

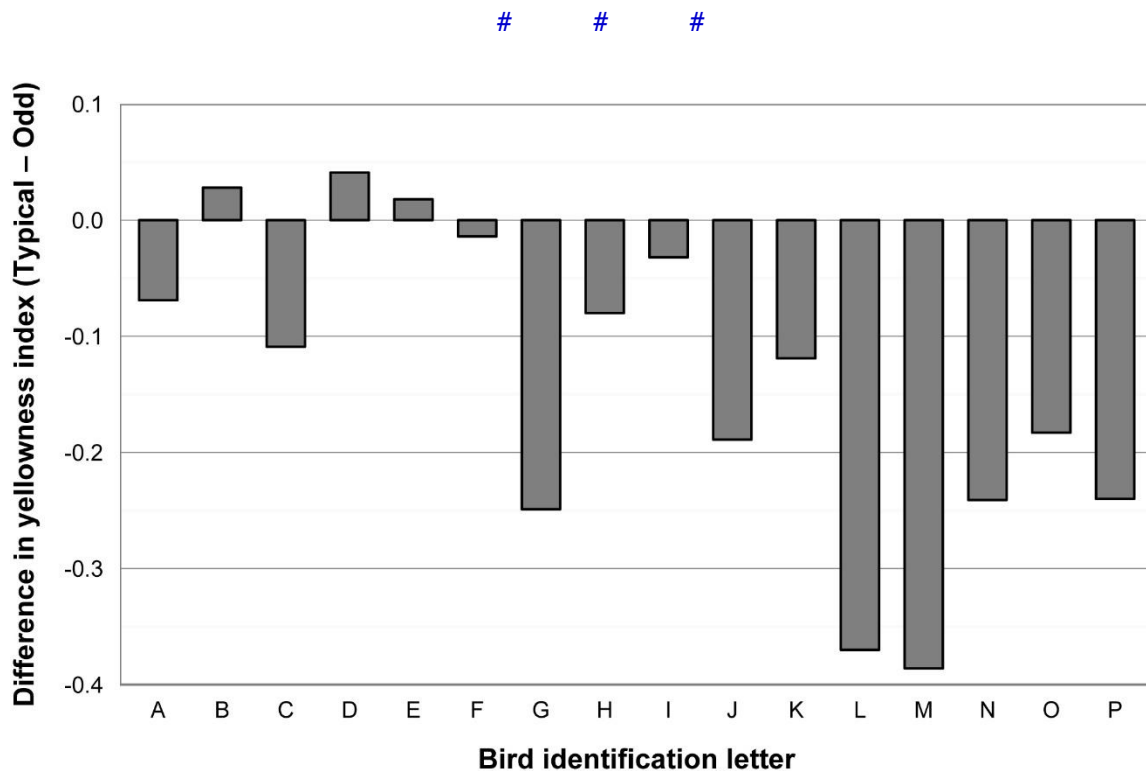
```
Data$Difference = Data$Odd - Data$Typical

library(ggplot2)

ggplot(Data,
       aes(x = Bird,
           y = Difference)) +
  geom_bar(stat = "identity",
          fill = "grey50",
          colour = "black",
          width = 0.6) +
  scale_y_continuous(breaks = seq(-0.4, 0.1, 0.1),
                    limits = c(-0.4, 0.1),
                    expand = c(0, 0)) +
  #ggtitle("Chart title") +
  labs(x = "Bird identification letter",
       y = "Difference in yellowness index (Typical - Odd)") +
  theme_bw() +
  theme(panel.grid.major.x = element_blank(),
```



```
panel.grid.major.y = element_line(colour = "grey50"),
plot.title = element_text(size = rel(1.5),
                           face = "bold", vjust = 1.5),
axis.ticks.x = element_blank(),
axis.ticks.y = element_blank(),
axis.title.y = element_text(face = "bold",
                             vjust= 1.8),
axis.title.x = element_text(face = "bold",
                             vjust= -0.8)
)
```



Paired t-test, data in wide format, horseshoe crab example

```
### -----
### Paired t-test, Horseshoe crab example, pp. 181-182
### -----

# Note, if you use "2011" as a variable name,
# the read.table function will convert it to "x2011"

Input = ("
Beach          Year.2011  Year.2012
'Bennetts Pier' 35282    21814
'Big Stone'     359350   83500
'Broadkill'     45705    13290
'Cape Henlopen' 49005    30150
'Fortescue'     68978    125190
'Fowler'        8700     4620
'Gandys'        18780    88926
```

```
'Higbees'      13622      1205
'Highs'        24936      29800
'Kimbles'      17620      53640
'Kitts Hummock' 117360     68400
'Norburys Landing' 102425     74552
'North Bowers'  59566     36790
'North Cape May' 32610     4350
'Pickering'     137250    110550
'Pierces Point' 38003     43435
'Primehook'     101300    20580
'Reeds'         62179     81503
'Slaughter'     203070    53940
'South Bowers'  135309    87055
'South CSL'     150656    112266
'Ted Harvey'    115090    90670
'Townbank'      44022     21942
'Villas'        56260     32140
'Woodland'      125       1260
")
```

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

Paired t-test

```
t.test(Data$Year.2011,
       Data$Year.2012,
       paired=TRUE,
       conf.level=0.95)
```

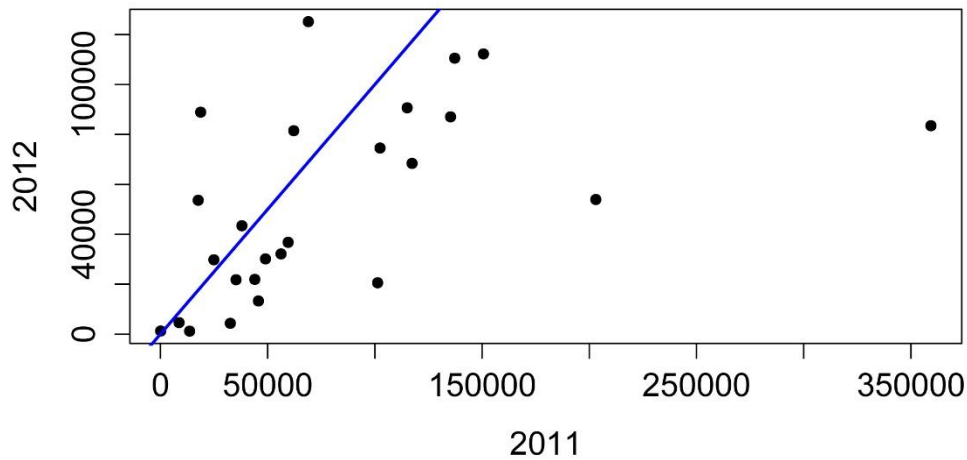
```
t = 2.1119, df = 24, p-value = 0.04529
```

```
mean of the differences
28225.4
```

Simple 1-to-1 plot of values

```
plot(Data$Year.2011, Data$Year.2012,
     pch = 16,
     xlab="2011",
     ylab="2012")
```

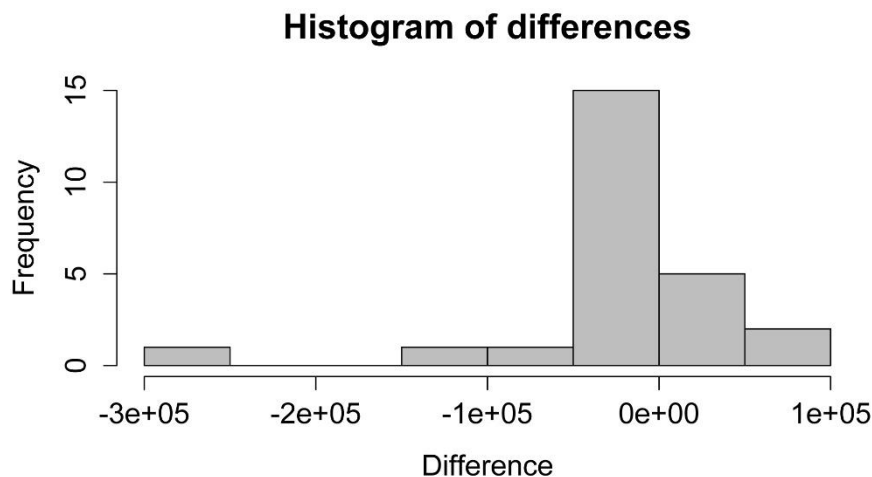
```
abline(0,1, col="blue", lwd=2)
```



Plot of paired samples from a paired t-test. Circles below and to the right of the blue one-to-one line indicate observations with a higher value for 2011 than for 2012.

```
Difference = Data$Year.2012 - Data$Year.2011
```

```
hist(Difference,
     col="gray",
     main="Histogram of differences",
     xlab="Difference")
```



Histogram of differences in two populations from paired t-test. Distribution of differences should be approximately normal. Bins with negative values indicate observations with a higher score for 2011 than for 2012.

#

Paired t-test, data in long format

```
### -----
### Paired t-test, long format data, Flicker feather example, p. 185
### -----
```

```
Input = ("
Bird    Feather    Length
A       Typical    -0.255
B       Typical    -0.213
C       Typical    -0.19
D       Typical    -0.185
E       Typical    -0.045
F       Typical    -0.025
G       Typical    -0.015
H       Typical     0.003
I       Typical     0.015
J       Typical     0.02
K       Typical     0.023
L       Typical     0.04
M       Typical     0.04
N       Typical     0.05
O       Typical     0.055
P       Typical     0.058
A       Odd        -0.324
B       Odd        -0.185
C       Odd        -0.299
D       Odd        -0.144
E       Odd        -0.027
F       Odd        -0.039
G       Odd        -0.264
H       Odd        -0.077
I       Odd        -0.017
J       Odd        -0.169
K       Odd        -0.096
L       Odd        -0.33
M       Odd        -0.346
N       Odd        -0.191
O       Odd        -0.128
P       Odd        -0.182
")
```

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

```
### Note: data must be ordered so that the first observation of Group 1
### is the same subject as the first observation of Group 2
```

```
t.test(Length ~ Feather,
      data=Data,
      paired=TRUE,
      conf.level=0.95)
```

```
t = -4.0647, df = 15, p-value = 0.001017
```

```
mean of the differences
-0.137125
```

#

Permutation test for dependent samples

This permutation test is analogous to a paired t-test.

```
### -----
### Paired two-sample permutation test, long format data
### Flicker feather example, p. 185
### -----
```

```
Input = ("
Bird    Feather    Length
A       Typical    -0.255
B       Typical    -0.213
C       Typical    -0.19
D       Typical    -0.185
E       Typical    -0.045
F       Typical    -0.025
G       Typical    -0.015
H       Typical     0.003
I       Typical     0.015
J       Typical     0.02
K       Typical     0.023
L       Typical     0.04
M       Typical     0.04
N       Typical     0.05
O       Typical     0.055
P       Typical     0.058
A       Odd        -0.324
B       Odd        -0.185
C       Odd        -0.299
D       Odd        -0.144
E       Odd        -0.027
F       Odd        -0.039
G       Odd        -0.264
H       Odd        -0.077
I       Odd        -0.017
J       Odd        -0.169
K       Odd        -0.096
L       Odd        -0.33
M       Odd        -0.346
N       Odd        -0.191
O       Odd        -0.128
P       Odd        -0.182
")
```

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

```
library(coin)
```

```
independence_test(Length ~ Feather | Bird,
                  data = Data)
```

Asymptotic General Independence Test

$z = -2.8959$, $p\text{-value} = 0.003781$

#

Power analysis

Power analysis for paired t-test

```
#### -----
#### Power analysis, paired t-test, pp. 185–186
#### -----

Detect = 0.1          # Difference in means to detect
SD      = 0.135       # Standard deviation of differences

Cohen.d = Detect/SD

library(pwr)

pwr.t.test(
  n = NULL,          # Number of _pairs_ of observations
  d = Cohen.d,
  sig.level = 0.05,  # Type I probability
  power = 0.90,      # 1 minus Type II probability
  type = "paired",   # paired t-test
  alternative = "two.sided")

Paired t test power calculation

n = 21.16434

NOTE: n is number of *pairs*

# # #
```

Wilcoxon Signed-rank Test

Examples in *Summary and Analysis of Extension Program Evaluation*

[SAEPPER: Two-sample Paired Rank-sum Test](#)

[SAEPPER: Sign Test for Two-sample Paired Data](#)

Packages used in this chapter

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

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

When to use it

The poplar example is shown below in the “How to do the test” section.

Null hypothesis**How it works****Examples****Graphing the results**

See the *Handbook* for information on these topics.

Similar tests

Paired t-test and permutation test are described in the *Paired t-test* chapter. The sign test is described below.

How to do the test***Wilcoxon signed-rank test example***

```
### -----
### wilcoxon signed-rank test, poplar example, p. 189
### -----

Input = ("
Clone      August  November
Balsam_Spire    8.1    11.2
Beaupre        10.0    16.3
Hazendans       16.5    15.3
Hoogvorst      13.6    15.6
Raspalje        9.5    10.5
Unal           8.3    15.5
Columbia_River 18.3    12.7
Fritzi_Pauley   13.3    11.1
Trichobel       7.9    19.9
Gaver           8.1    20.4
Gibecq          8.9    14.2
Primo           12.6    12.7
Wolterson      13.4    36.8
")

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

wilcox.test(Data$August,
             Data$November,
             paired=TRUE)

wilcoxon signed rank test

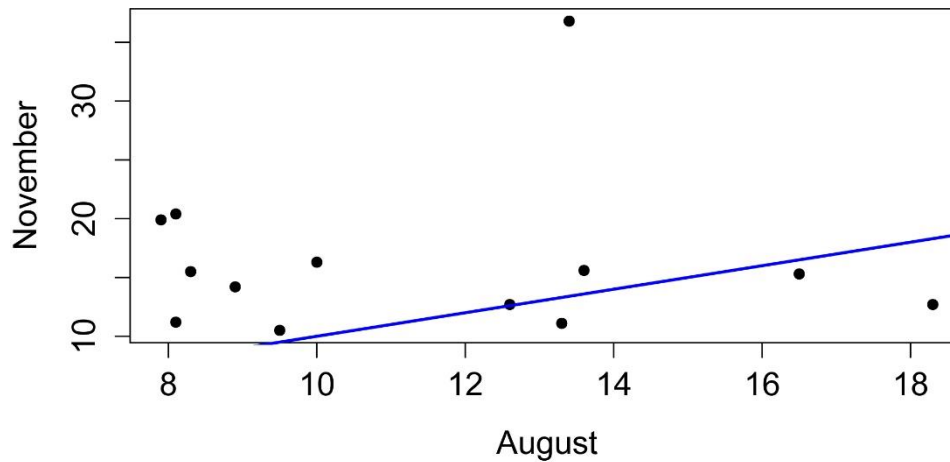
V = 16, p-value = 0.03979

### Matches "Signed Rank" p-value in SAS output
```

Simple 1-to-1 plot of values

```
plot(Data$August, Data$November,
     pch = 16,
     xlab="August",
```

```
ylab="November")
abline(0,1, col="blue", lwd=2)
```



Plot of paired samples from a Wilcoxon signed-rank test. Circles above and to the left of the blue one-to-one line indicate observations with a higher value for November than for August.

#

Sign test example

The following is an example of the two-sample dependent-samples sign test. The data are arranged as a data frame in which each row contains the values for both measurements being compared for each experimental unit. This is sometimes called “wide format” data. The *SIGN.test* function in the *BSDA* package is used. The option *md=0* indicates that the expected difference in the medians is 0 (null hypothesis). This function can also perform a one-sample sign test.

```
### -----
### Two-sample sign test, poplar example, p. 189
### -----
```

```
Input = ("
Clone      August  November
Balsam_Spire  8.1    11.2
Beaupre     10.0    16.3
Hazendans   16.5    15.3
Hoogvorst   13.6    15.6
Raspalje    9.5    10.5
Unal        8.3    15.5
Columbia_River 18.3    12.7
Fritzi_Pauley 13.3    11.1
Trichobel   7.9    19.9
Gaver       8.1    20.4
Gibecq      8.9    14.2
Primo       12.6    12.7
```



```
wolterson      13.4    36.8
")
```

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

```
library(BSDA)
```

```
SIGN.test(x = Data$ August,
          y = Data$ November,
          md = 0,
          alternative = "two.sided",
          conf.level = 0.95)
```

Dependent-samples Sign-Test

S = 3, p-value = 0.09229

Matches "Sign" p-value in SAS output

#

Correlation and Linear Regression

Introduction

The amphipod egg example is shown below in the “How to do the test” section.

When to use them

Correlation versus linear regression

Correlation and causation

Null hypothesis

Independent vs. dependent variables

How the test works

Assumptions

See the *Handbook* for information on these topics.

Examples

The species diversity example is shown below in the “How to do the test” section.

Graphing the results

Similar tests

How to do the test

Correlation and linear regression example

```
### -----
### Correlation and linear regression, species diversity example
### pp. 207–208
### -----

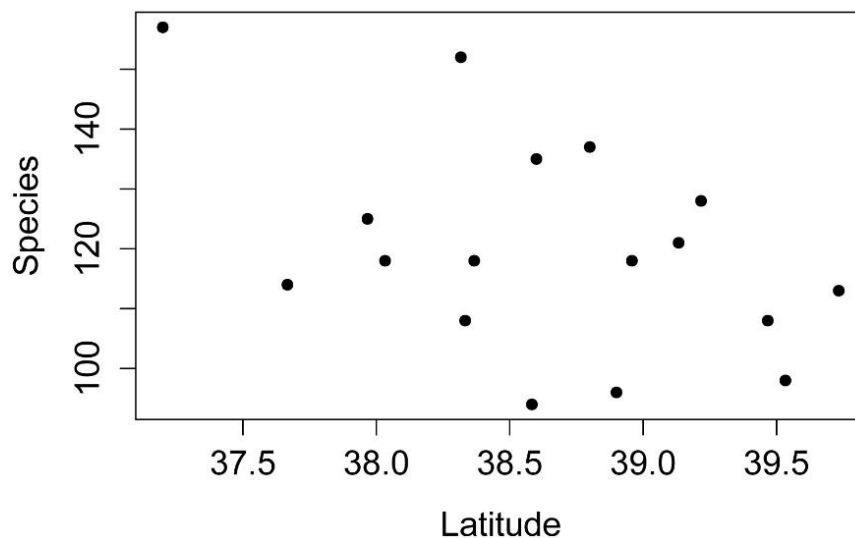
Input = ("
Town      State  Latitude  Species
'Bombay Hook'    DE      39.217    128
'Cape Henlopen'  DE      38.800    137
'Middletown'     DE      39.467    108
'Milford'        DE      38.958    118
'Rehoboth'       DE      38.600    135
'Seafood-Nanticoke' DE    38.583     94
'Wilmington'    DE      39.733    113
'Crisfield'     MD      38.033    118
'Denton'        MD      38.900     96
'Elkton'        MD      39.533     98
'Lower Kent County' MD    39.133    121
'Ocean City'    MD      38.317    152
'Salisbury'     MD      38.333    108
'S Dorchester County' MD    38.367    118
```

```
'Cape Charles'      VA      37.200      157
'Chincoteague'      VA      37.967      125
'Wachapreague'      VA      37.667      114
")
```

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

Simple plot of the data

```
plot(Species ~ Latitude,
     data=Data,
     pch=16,
     xlab = "Latitude",
     ylab = "Species")
```



Correlation

Correlation can be performed with the *cor.test* function in the native *stats* package. It can perform Pearson, Kendall, and Spearman correlation procedures. Methods for multiple correlation of several variables simultaneously are discussed in the *Multiple regression* chapter.

Pearson correlation

Pearson correlation is the most common form of correlation. It is a parametric test, and assumes that the data are linearly related and that the residuals are normally distributed.

```
cor.test(~ Species + Latitude,
        data=Data,
        method = "pearson",
        conf.level = 0.95)
```

Pearson's product-moment correlation

t = -2.0225, df = 15, p-value = 0.06134

cor

-0.4628844

Kendall correlation

Kendall rank correlation is a non-parametric test that does not assume a distribution of the data or that the data are linearly related. It ranks the data to determine the degree of correlation.

```
cor.test( ~ Species + Latitude,
         data=Data,
         method = "kendall",
         continuity = FALSE,
         conf.level = 0.95)
```

Kendall's rank correlation tau

z = -1.3234, p-value = 0.1857

tau
-0.2388326

Spearman correlation

Spearman rank correlation is a non-parametric test that does not assume a distribution of the data or that the data are linearly related. It ranks the data to determine the degree of correlation, and is appropriate for ordinal measurements.

```
cor.test( ~ Species + Latitude,
         data=Data,
         method = "spearman",
         continuity = FALSE,
         conf.level = 0.95)
```

Spearman's rank correlation rho

S = 1111.908, p-value = 0.1526

rho
-0.3626323

Linear regression

Linear regression can be performed with the *lm* function in the native *stats* package. A robust regression can be performed with the *lmrob* function in the *robustbase* package.

```
model = lm(Species ~ Latitude,
           data = Data)
```

```
summary(model)           # shows parameter estimates,
                          # p-value for model, r-square
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	585.145	230.024	2.544	0.0225 *
Latitude	-12.039	5.953	-2.022	0.0613 .

Multiple R-squared: 0.2143, Adjusted R-squared: 0.1619
 F-statistic: 4.09 on 1 and 15 DF, p-value: 0.06134

```
library(car)
```

```
Anova(model, type="II") # shows p-value for effects in model
```

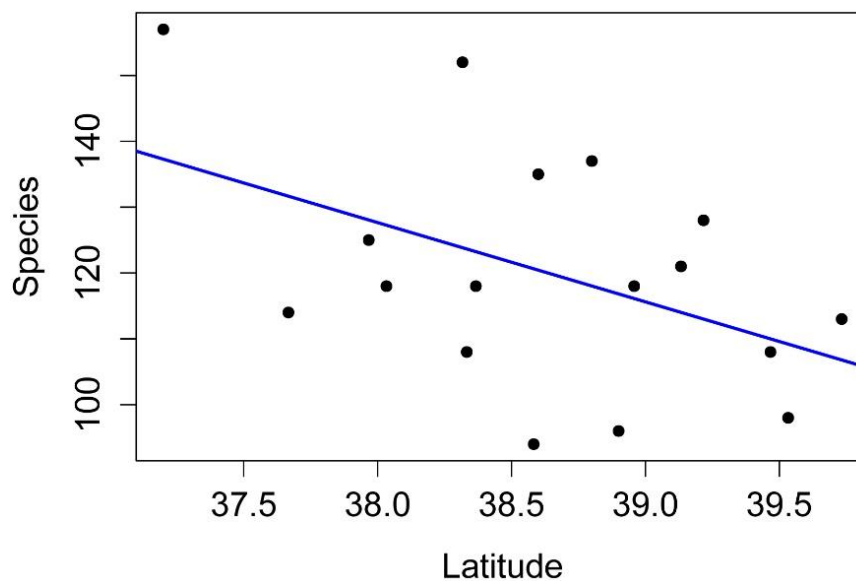
```
Response: Species
      Sum Sq Df F value    Pr(>F)
Latitude  1096.6   1   4.0903 0.06134 .
Residuals 4021.4  15
```

Plot linear regression

```
int = model$coefficient["(Intercept)"]
slope = model$coefficient["Latitude"]

plot(Species ~ Latitude,
     data = Data,
     pch=16,
     xlab = "Latitude",
     ylab = "Species")

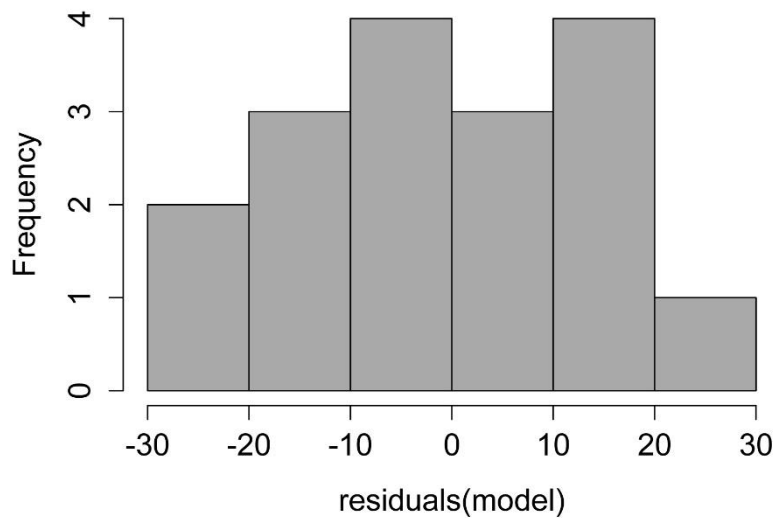
abline(int, slope,
       lty=1, lwd=2, col="blue") # style and color of line
```



Checking assumptions of the model

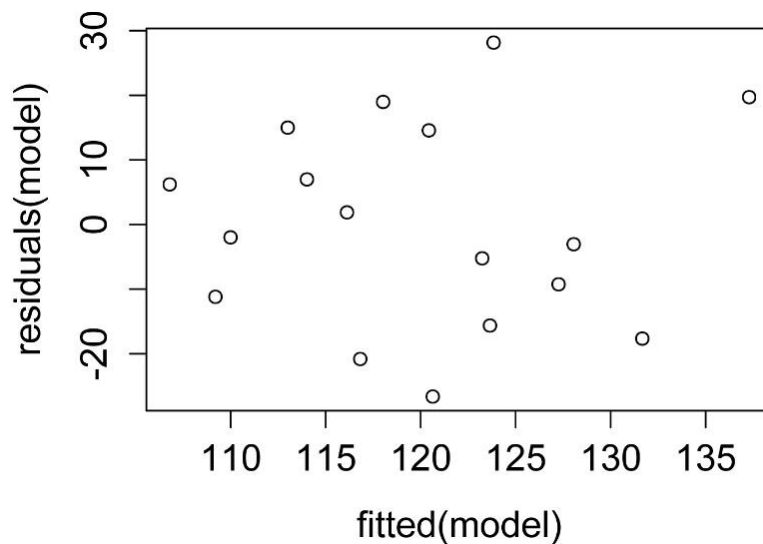
```
hist(residuals(model),
     col="darkgray")
```

Histogram of residuals(model)



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model),
     residuals(model))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model)
### alternative: library(FSA); residPlot(model)
```

Robust regression

The *lmrob* function in the *robustbase* package produces a linear regression which is not sensitive to outliers in the response variable. It uses MM-estimation.

```
library(robustbase)

model = lmrob(Species ~ Latitude,
              data = Data)

summary(model)                                # shows parameter estimates, r-square
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	568.830	230.203	2.471	0.0259 *
Latitude	-11.619	5.912	-1.966	0.0681 .

```
Multiple R-squared:  0.1846, Adjusted R-squared:  0.1302

model.null = lmrob(Species ~ 1,
                  data = Data)

anova(model, model.null)                      # shows p-value for model
```

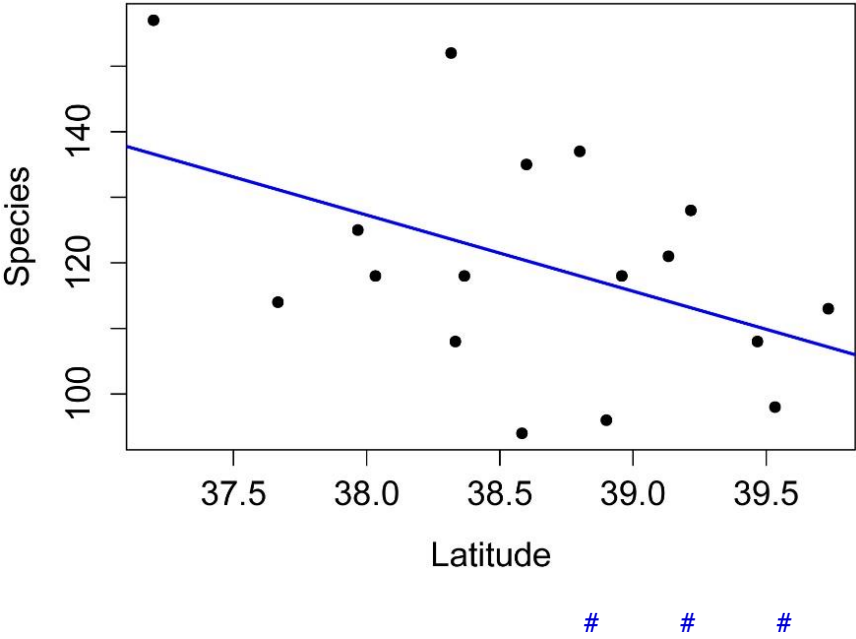
	pseudoDf	Test.Stat	Df	Pr(>chisq)
1	15			
2	16	3.8634	1	0.04935 *

Plot the model

```
int = model$coefficient["(Intercept)"]
slope = model$coefficient["Latitude"]

plot(Species ~ Latitude,
     data = Data,
     pch=16,
     xlab = "Latitude",
     ylab = "Species")

abline(int, slope,
       lty=1, lwd=2, col="blue")           # style and color of line
```



Linear regression example

```
### -----
### Linear regression, amphipod eggs example
### pp. 191-193
### -----
```

```
Input = ("
Weight Eggs
5.38 29
7.36 23
6.13 22
4.75 20
8.10 25
8.62 25
6.30 17
7.44 24
7.26 20
7.17 27
7.78 24
6.23 21
5.42 22
7.87 22
5.25 23
7.37 35
8.01 27
4.92 23
7.03 25
6.45 24
5.06 19
6.72 21
7.00 20
9.39 33
```



```
6.49 17
6.34 21
6.16 25
5.74 22
")
```

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

```
model = lm(Eggs ~ Weight,
           data = Data)
```

```
summary(model)           # shows parameter estimates,
                          # p-value for model, r-square
```

```
Coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  12.6890     4.2009   3.021  0.0056 **
Weight        1.6017     0.6176   2.593  0.0154 *
```

```
Multiple R-squared:  0.2055, Adjusted R-squared:  0.175
F-statistic: 6.726 on 1 and 26 DF,  p-value: 0.0154
```

```
### Neither the r-squared nor the p-value agrees with what is reported
### in the Handbook.
```

```
library(car)
```

```
Anova(model, type="II")           # shows p-value for effects in model
```

```
              Sum Sq Df F value Pr(>F)
Weight        93.89  1  6.7258 0.0154 *
Residuals    362.96 26
```

```
#      #      #
```

Power analysis

Power analysis for correlation

```
### -----
### Power analysis, correlation, p. 208
### -----
```

```
pwr.r.test(n = NULL,
           r = 0.500,
           sig.level = 0.05,
           power = 0.80,
           alternative = "two.sided")
```

```
approximate correlation power calculation (arctangh transformation)
```

```
n = 28.87376 # answer is somewhat different than in Handbook
```

```
#      #      #
```

Spearman Rank Correlation

When to use it

Null hypothesis

Assumption

How the test works

See the *Handbook* for information on these topics.

Example

Example of Spearman rank correlation

```
### -----
### Spearman rank correlation, frigatebird example
### p. 212
### -----
```

```
Input = ("
Volume  Pitch
1760    529
2040    566
2440    473
2550    461
2730    465
2740    532
3010    484
3080    527
3370    488
3740    485
4910    478
5090    434
5090    468
5380    449
5850    425
6730    389
6990    421
7960    416
")
```

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

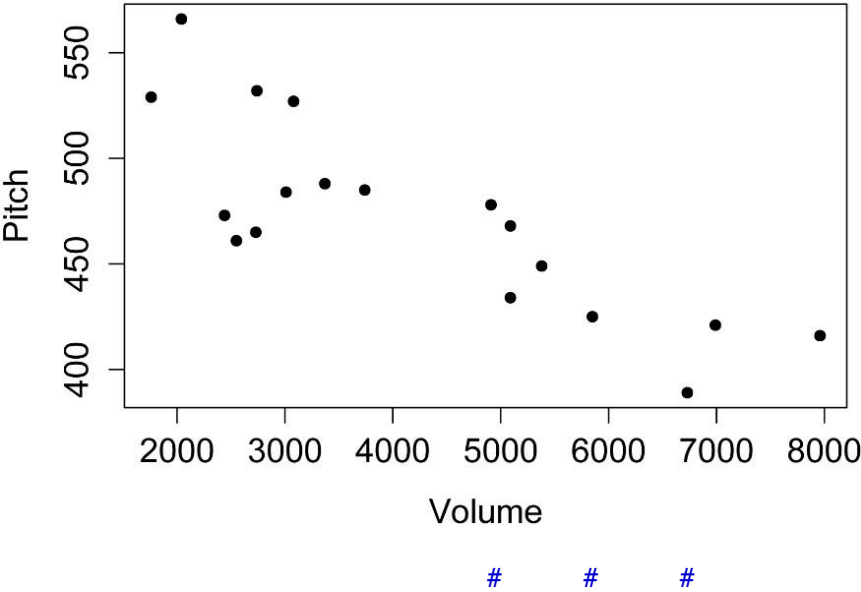
```
cor.test( ~ Pitch + Volume,
          data=Data,
          method = "spearman",
          continuity = FALSE,
          conf.level = 0.95)
```

Spearman's rank correlation rho

S = 1708.382, p-value = 0.0002302
sample estimates:
rho
-0.7630357

Simple plot of the data

```
plot(Pitch ~ Volume,  
     data=Data,  
     pch=16)
```



Graphing the results

See the *Handbook* for information on this topic.

How to do the test

Example of Spearman rank correlation

```
### -----  
### Spearman rank correlation, species diversity example  
### p. 214  
### -----  
  
Input = ("  
Town          State  Latitude  Species  
'Bombay Hook'  DE      39.217   128  
'Cape Henlopen' DE      38.800   137  
'Middletown'   DE      39.467   108  
'Milford'      DE      38.958   118  
'Rehoboth'     DE      38.600   135  
'Seaford-Nanticoke' DE    38.583    94  
'Wilmington'  DE      39.733   113  
'Crisfield'    MD      38.033   118
```

```
'Denton'           MD  38.900    96
'Elkton'           MD  39.533    98
'Lower Kent County' MD  39.133   121
'Ocean City'       MD  38.317   152
'Salisbury'        MD  38.333   108
'S Dorchester County' MD 38.367   118
'Cape Charles'     VA  37.200   157
'Chincoteague'     VA  37.967   125
'Wachapreague'     VA  37.667   114
")
```

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

```
cor.test( ~ Species + Latitude,
          data=Data,
          method = "spearman",
          continuity = FALSE,
          conf.level = 0.95)
```

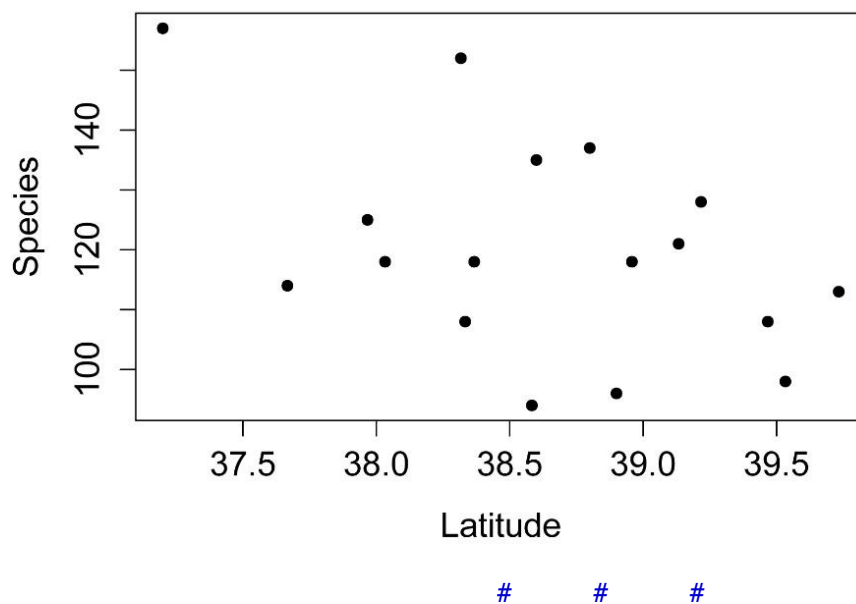
```
Spearman's rank correlation rho
```

```
s = 1111.908, p-value = 0.1526
```

```
rho
-0.3626323
```

Simple plot of the data

```
plot(Species ~ Latitude,
     data=Data,
     pch=16)
```



Curvilinear Regression

When to use it

Null hypotheses

Assumptions

How the test works

Examples

Graphing the results

Similar tests

See the *Handbook* for information on these topics.

How to do the test

This chapter will fit models to curvilinear data using three methods: 1) Polynomial regression; 2) B-spline regression with polynomial splines; and 3) Nonlinear regression with the *nls* function. In this example, each of these three will find essentially the same best-fit curve with very similar p-values and R-squared values.

Polynomial regression

Polynomial regression is really just a special case of multiple regression, which is covered in the *Multiple regression* chapter. In this example we will fit a few models, as the *Handbook* does, and then compare the models with the extra sum of squares test, the Akaike information criterion (AIC), and the adjusted R-squared as model fit criteria.

For a linear model (*lm*), the adjusted R-squared is included with the output of the *summary(model)* statement. The AIC is produced with its own function call, *AIC(model)*. The extra sum of squares test is conducted with the *anova* function applied to two models.

For AIC, smaller is better. For adjusted R-squared, larger is better. A non-significant p-value for the extra sum of squares test comparing model *a* to model *b* indicates that the model with the extra terms does not significantly reduce the error sum of squares over the reduced model. Which is to say, a non-significant p-value suggests the model with the additional terms is not better than the reduced model.

```
### -----
### Polynomial regression, turtle carapace example
### pp. 220–221
### -----
```

```
Input = ("
Length Clutch
284      3
290      2
290      7
290      7
298     11
299     12
302     10
```

```

306      8
306      8
309      9
310     10
311     13
317      7
317      9
320      6
323     13
334      2
334      8
")

```

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

```

### Change Length from integer to numeric variable
### otherwise, we will get an integer overflow error on big numbers

```

```
Data$Length = as.numeric(Data$Length)
```

```
### Create quadratic, cubic, quartic variables
```

```
library(dplyr)
```

```

Data =
mutate(Data,
      Length2 = Length*Length,
      Length3 = Length*Length*Length,
      Length4 = Length*Length*Length*Length)

```

```
library(FSA)
```

```
headtail(Data)
```

	Length	Clutch	Length2	Length3	Length4
1	284	3	80656	22906304	6505390336
2	290	2	84100	24389000	7072810000
3	290	7	84100	24389000	7072810000
16	323	13	104329	33698267	10884540241
17	334	2	111556	37259704	12444741136
18	334	8	111556	37259704	12444741136

Define the models to compare

```

model.1 = lm (Clutch ~ Length,                      data=Data)
model.2 = lm (Clutch ~ Length + Length2,            data=Data)
model.3 = lm (Clutch ~ Length + Length2 + Length3,  data=Data)
model.4 = lm (Clutch ~ Length + Length2 + Length3 + Length4, data=Data)

```

Generate the model selection criteria statistics for these models

```
summary(model.1)
```

```
Coefficients:
```

```
      Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.4353     17.3499  -0.03    0.98
Length       0.0276      0.0563   0.49    0.63
```

```
Multiple R-squared:  0.0148, Adjusted R-squared:  -0.0468
F-statistic: 0.24 on 1 and 16 DF,  p-value: 0.631
```

```
AIC(model.1)
```

```
[1] 99.133
```

```
summary(model.2)
```

```
Coefficients:
```

```
      Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.00e+02   2.70e+02  -3.33   0.0046 **
Length       5.86e+00   1.75e+00   3.35   0.0044 **
Length2      -9.42e-03   2.83e-03  -3.33   0.0045 **
```

```
Multiple R-squared:  0.434, Adjusted R-squared:  0.358
F-statistic: 5.75 on 2 and 15 DF,  p-value: 0.014
```

```
AIC(model.2)
```

```
[1] 91.16157
```

```
anova(model.1, model.2)
```

```
Analysis of Variance Table
```

```
    Res.Df    RSS Df Sum of Sq    F Pr(>F)
1      16 186.15
2      15 106.97  1    79.178 11.102 0.00455 **
```

```
### Continue this process for the remainder of the models
```

Model selection criteria for four polynomial models. Model 2 has the lowest AIC, suggesting it is the best model from this list for these data. Likewise model 2 shows the largest adjusted R-squared. Finally, the extra SS test shows model 2 to be better than model 1, but that model 3 is not better than model 2. All this evidence indicates selecting model 2.

Model	AIC	Adjusted R-squared	p-value for extra SS from previous model
-------	-----	--------------------	--

1	99.1	- 0.047	
2	91.2	0.36	0.0045
3	92.7	0.33	0.55
4	94.4	0.29	0.64

Compare models with *compareLM* and *anova*

This process can be automated somewhat by using my *compareLM* function and by passing multiple models to the *anova* function. Any of AIC, AICc, or BIC can be minimized to select the best model. If you have no preference, I might recommend using AICc.

```
model.1 = lm (Clutch ~ Length, data=Data)
model.2 = lm (Clutch ~ Length + Length2, data=Data)
model.3 = lm (Clutch ~ Length + Length2 + Length3, data=Data)
model.4 = lm (Clutch ~ Length + Length2 + Length3 + Length4, data=Data)
```

```
library(rcompanion)
```

```
compareLM(model.1, model.2, model.3, model.4)
```

```
$Fit.criteria
  Rank Df.res   AIC   AICc   BIC R.squared Adj.R.sq p.value Shapiro.W Shapiro.p
1     2    16 99.13 100.80 101.80  0.01478  -0.0468 0.63080   0.9559   0.5253
2     3    15 91.16  94.24  94.72  0.43380   0.3583 0.01403   0.9605   0.6116
3     4    14 92.68  97.68  97.14  0.44860   0.3305 0.03496   0.9762   0.9025
4     5    13 94.37 102.00  99.71  0.45810   0.2914 0.07413   0.9797   0.9474
```

```
anova(model.1, model.2, model.3, model.4)
```

```
  Res.Df   RSS Df Sum of Sq    F    Pr(>F)
1      16 186.15
2      15 106.97  1    79.178 10.0535 0.007372 ** ## Compares m.2 to m.1
3      14 104.18  1     2.797  0.3551 0.561448 ## Compares m.3 to m.2
4      13 102.38  1     1.792  0.2276 0.641254 ## Compares m.4 to m.3
```

Investigate the final model

```
model.final = lm (Clutch ~ Length + Length2,
                  data=Data)
```

```
summary(model.final)           # Shows coefficients,
                               # overall p-value for model, R-squared
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -9.00e+02   2.70e+02  -3.33   0.0046 **
Length       5.86e+00   1.75e+00   3.35   0.0044 **
Length2      -9.42e-03   2.83e-03  -3.33   0.0045 **
```

```
Multiple R-squared:  0.434,    Adjusted R-squared:  0.358
F-statistic: 5.75 on 2 and 15 DF,  p-value: 0.014
```



```
library(car)
```

```
Anova(model.final, type="II")
```

```
# Shows p-values for individual terms
```

```
Anova Table (Type II tests)
```

```
Response: Clutch
```

	Sum Sq	Df	F value	Pr(>F)
Length	79.9	1	11.2	0.0044 **
Length2	79.2	1	11.1	0.0045 **
Residuals	107.0	15		

Simple plot of model

```
plot(Clutch ~ Length,
     data = Data,
     pch=16,
     xlab = "Carapace length",
     ylab = "Clutch")
```

```
i = seq(min(Data$Length), max(Data$Length), len=100)
```

```
# x-values for line
```

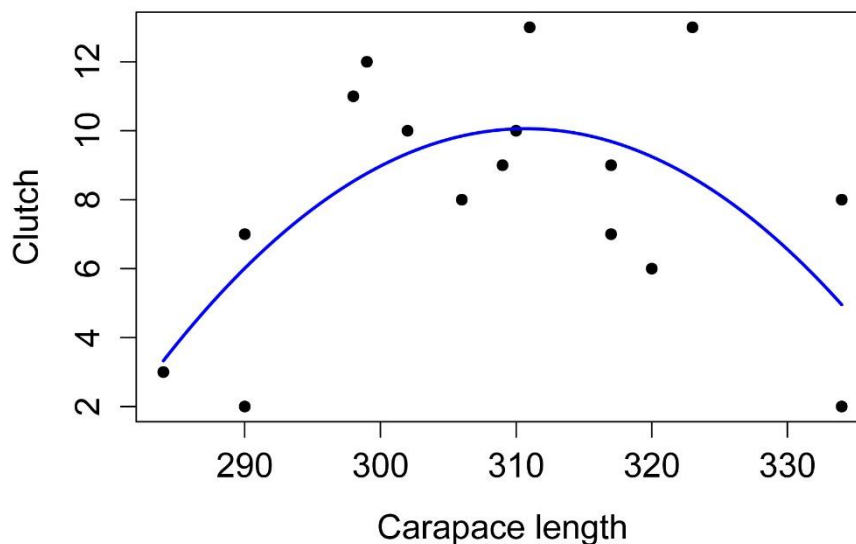
```
predy = predict(model.final,
                 data.frame(Length=i, Length2=i*i))
```

```
# fitted values
```

```
lines(i, predy,
      lty=1, lwd=2, col="blue")
```

```
# spline curve
```

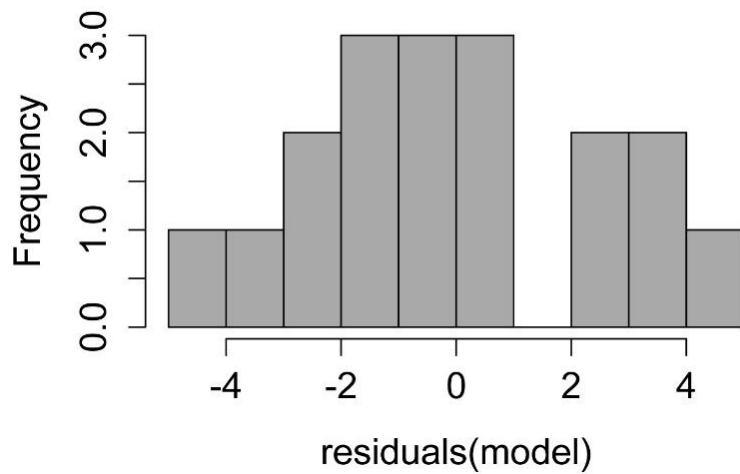
```
# style and color
```



Checking assumptions of the model

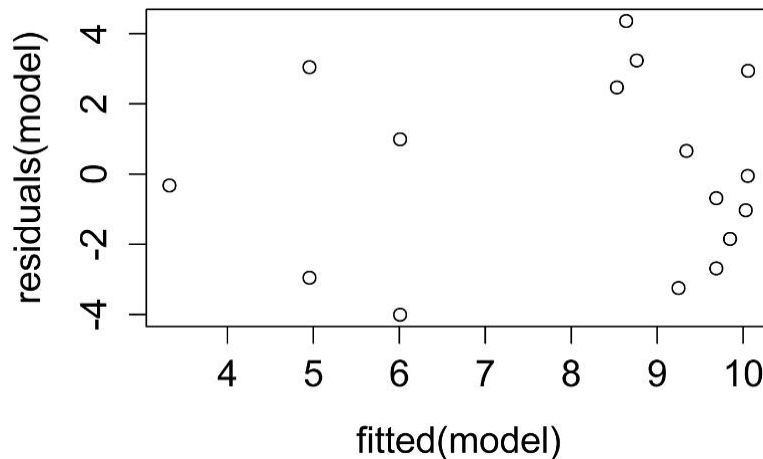
```
hist(residuals(model.final),
     col="darkgray")
```

Histogram of residuals(model)



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model.final),
     residuals(model.final))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model.final)
```

```
# # #
```

B-spline regression with polynomial splines

B-spline regression uses smaller segments of linear or polynomial regression which are stitched together to make a single model. It is useful to fit a curve to data when you don't have a theoretical model to use (e.g. neither linear, nor polynomial, nor nonlinear). It does not assume a linear relationship between the variables, but the residuals should still be normal and independent. The model may be influenced by outliers.

```
#### -----
#### B-spline regression, turtle carapace example
#### pp. 220-221
#### -----

Input = ("
Length  Clutch
284      3
290      2
290      7
290      7
298     11
299     12
302     10
306      8
306      8
309      9
310     10
311     13
317      7
317      9
320      6
323     13
334      2
334      8
")

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

library(splines)

model = lm(Clutch ~ bs(Length,
                        knots = 5,      # How many internal segment nodes?
                        degree = 2),    # 1=local linear fits, 2=quadratic
            data = Data)

summary(model)                                # Display p-value and R-squared

Residual standard error: 2.671 on 15 degrees of freedom
Multiple R-squared:  0.4338, Adjusted R-squared:  0.3583
F-statistic: 5.747 on 2 and 15 DF,  p-value: 0.01403
```

Simple plot of model

```
plot(Clutch ~ Length,
     data = Data,
```

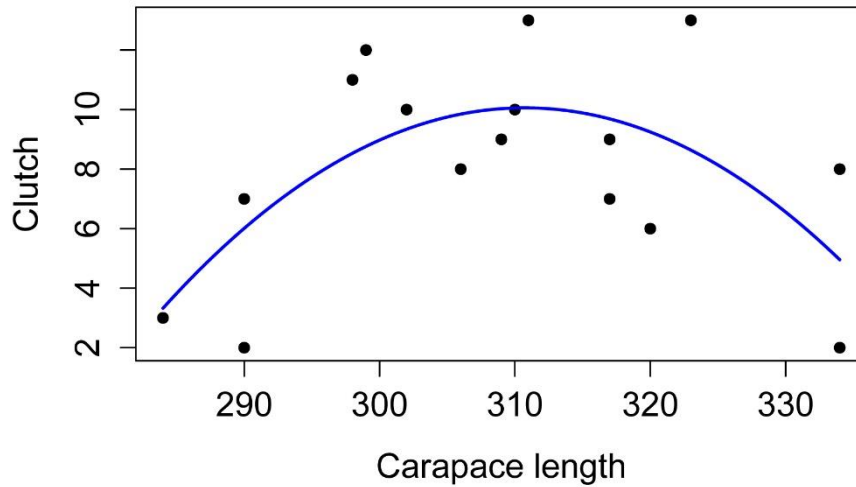
```

pch=16,
xlab = "Carapace length",
ylab = "Clutch")

i = seq(min(Data$Length), max(Data$Length), len=100)
predy = predict(model, data.frame(Length=i))
lines(i, predy,
      lty=1, lwd=2, col="blue")

```

x-values for line
fitted values
spline curve
style and color



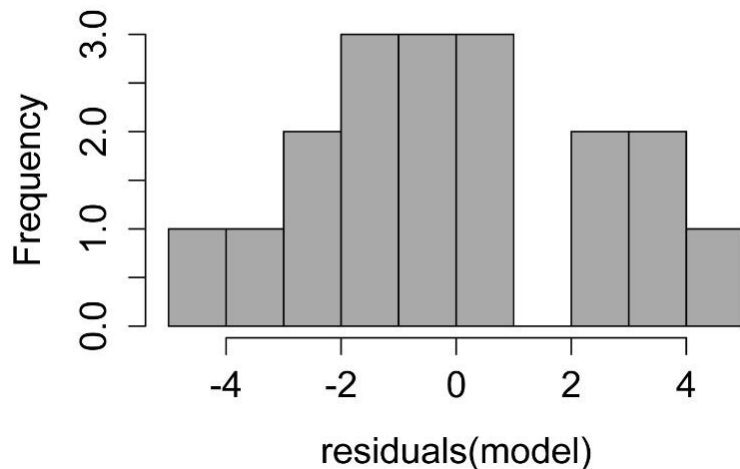
Checking assumptions of the model

```

hist(residuals(model),
     col="darkgray")

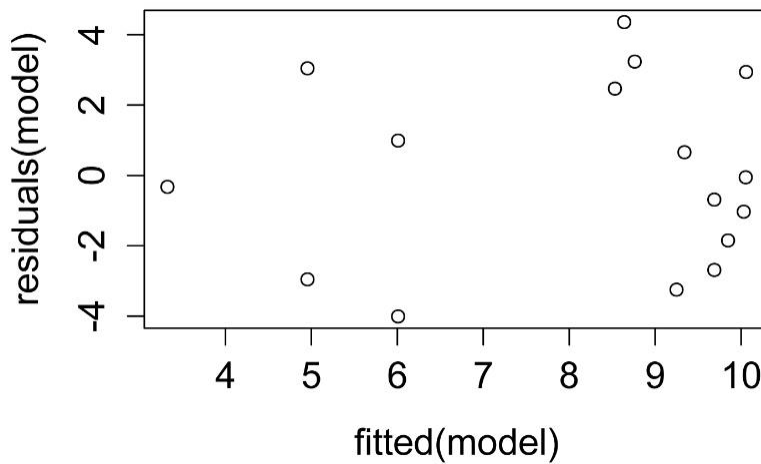
```

Histogram of residuals(model)



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model),
     residuals(model))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model)
```

```
# # #
```

Nonlinear regression

Nonlinear regression can fit various nonlinear models to a data set. These model might include exponential models, logarithmic models, decay curves, or growth curves. The *nls* function works by an iterative process, starting with user supplied estimates for the parameters in the model, and finding successively better parameter estimates until certain convergence criteria are met.

In this example, we assume that we want to fit a parabola to our data, but we'll use the vertex form of the equation ($y = a \cdot (x-h) + k$). This form is handy because the point (h, k) indicates the vertex of the parabola.

Note in the formula in the *nls* call below, that there are variables from our data (*Clutch* and *Length*), and parameters we want to estimate (*Lcenter*, *Cmax*, and *a*).

There's no set process for choosing starting estimates for the parameters. Often, the parameters will be meaningful. For example, here, *Lcenter* is the x-coordinate of the vertex and *Cmax* is the y-coordinate of the vertex. So we can guess at reasonable values for these. The parameter *a* would be difficult to guess at, though we know it should be negative because the parabola opens downward.

Because *nls* uses an iterative process based on initial estimates of the parameters, it fails to find a solution if the estimates are too far off, or it may return a set of parameter estimates that don't fit the data well. It is important to plot the solution and make sure it is reasonable. I have seen *nls*

have difficulty with models that have more than three parameters. The package *nlmrt* uses a different process for determining the iterations, and may be better to fit difficult models.

If you wish to have an overall p-value for the model and a pseudo-R-squared for the model, the model will need to be compared with a null model. Technically for this to be valid, the null model must be nested within the fitted model. That means that the null model is a special case of the fitted model. In our example, if we were to force a to be zero, that would leave a model $Clutch \sim constant$, where $constant$ would be a parameter that estimates the mean of the *Clutch* variable. Many theoretical models do not have this property; that is, they don't have a constant or linear term. They are therefore considered nonlinear models. In these cases, *nls* can still be used to fit the model, but the extra steps determining the model's overall p-value and pseudo-R-squared are technically not valid. In these cases, models could be compared with the Akaike information criterion (AIC).

The p-value for the model, relative to the null model, is determined with the extra SS (F) test (*anova* function) or likelihood ratio test (*lrtest* in the package *lmtest*).

There are various pseudo-R-squared values that have been developed for models without R-squared defined. My function *nagelkerke* calculates the McFadden, the Cox and Snell, and the Nagelkerke pseudo-R-squared. For *nls* models, a null model must be explicitly defined and passed to the function. The Nagelkerke is a modification of the Cox and Snell so that it has a maximum of 1. I find the Nagelkerke to usually be satisfactory for *nls*, *lme*, and *gls* models. As a technical note, for *gls* and *lme* models, my function uses the likelihood for the model with ML fitting (REML = FALSE).

Pseudo-R-squared values are not directly comparable to multiple R-squared values, though in the examples in this chapter, the Nagelkerke is reasonably close to the multiple R-squared for the quadratic parabola model.

```
### -----
### Nonlinear regression, turtle carapace example
### pp. 220-221
### -----
```

```
Input = ("
Length  Clutch
284      3
290      2
290      7
290      7
298     11
299     12
302     10
306      8
306      8
309      9
310     10
311     13
317      7
317      9
```

```

320      6
323     13
334      2
334      8
")

```

```

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

model = nls(Clutch ~ a * (Length - Lcenter)^2 + Cmax,
            data = Data,
            start = c(Lcenter = 310,
                      Cmax = 12,
                      a = -1),
            trace = FALSE,
            nls.control(maxiter = 1000)
            )

summary(model)

```

```

Parameters:
      Estimate Std. Error t value Pr(>|t|)
Lcenter 310.72865    2.37976  130.57 < 2e-16 ***
Cmax     10.05879    0.86359   11.65 6.5e-09 ***
a        -0.00942    0.00283   -3.33 0.0045 **

```

Determine overall p-value and pseudo-R-squared

```

model.null = nls(Clutch ~ I,
                 data = Data,
                 start = c(I = 8),
                 trace = FALSE)

anova(model, model.null)

```

	Res.Df	Res.Sum Sq	Df	Sum Sq	F value	Pr(>F)
1	15	106.97				
2	17	188.94	-2	-81.971	5.747	0.01403 *

```

library(rcompanion)

nagelkerke(fit = model,
           null = model.null)

$Pseudo.R.squared.for.model.vs.null
              Pseudo.R.squared
McFadden          0.109631
Cox and Snell (ML)  0.433836
Nagelkerke (Cragg and Uhler) 0.436269

```

Determine confidence intervals for parameters

```
library(nlstools)

confint2(model,
          level = 0.95,
          method = "asymptotic")
```

	2.5 %	97.5 %
Lcenter	305.6563154	315.800988774
Cmax	8.2180886	11.899483768
a	-0.0154538	-0.003395949

```
Boot=nlsBoot(model)
```

```
summary(Boot)
```

```
-----
Bootstrap statistics
      Estimate Std. error
Lcenter 311.07998936 2.872859816
Cmax    10.13306941 0.764154661
a        -0.00938236 0.002599385

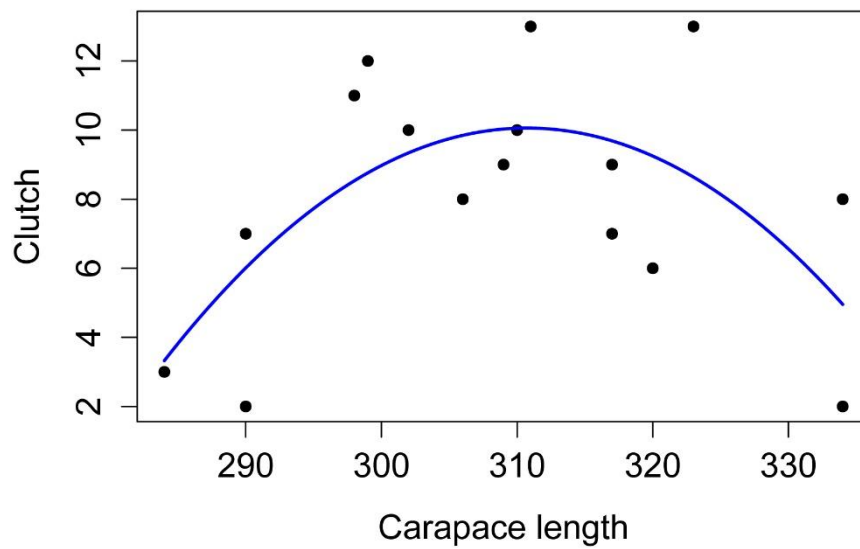
-----
Median of bootstrap estimates and percentile confidence intervals
      Median      2.5%      97.5%
Lcenter 310.770796703 306.78718266 316.153528168
Cmax    10.157560932  8.58974408  11.583719723
a        -0.009402318 -0.01432593 -0.004265714
```

Simple plot of model

```
plot(Clutch ~ Length,
     data = Data,
     pch=16,
     xlab = "Carapace length",
     ylab = "Clutch")

i = seq(min(Data$Length), max(Data$Length), len=100)
predy = predict(model, data.frame(Length=i))
lines(i, predy,
      lty=1, lwd=2, col="blue")
```

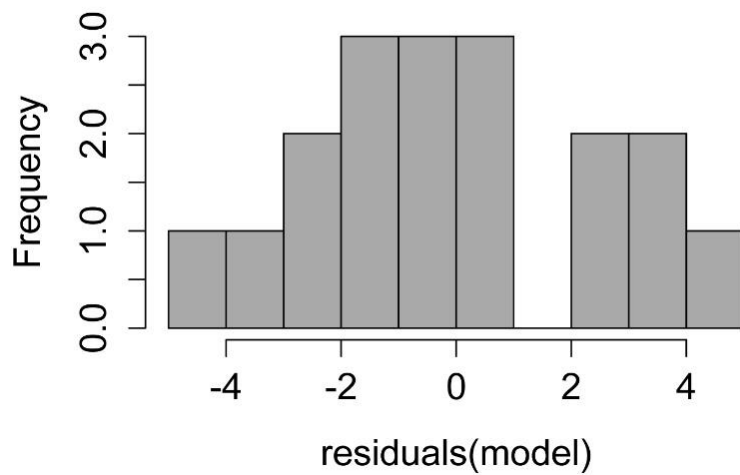
x-values for line
fitted values
spline curve
style and color



Checking assumptions of the model

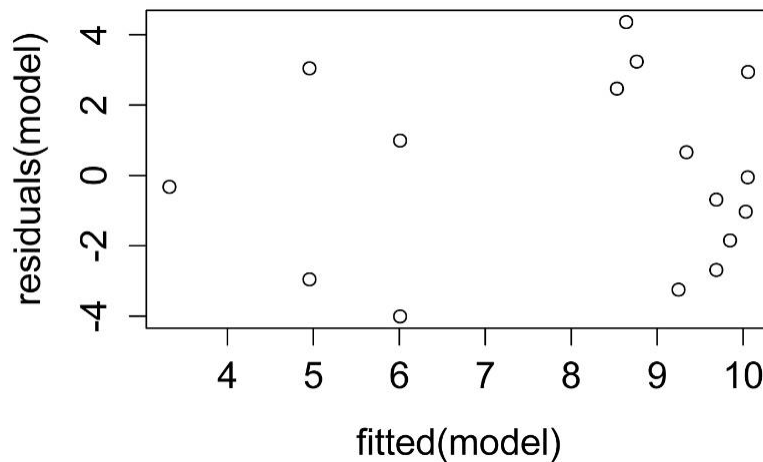
```
hist(residuals(model),
     col="darkgray")
```

Histogram of residuals(model)



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model),
     residuals(model))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

#

Analysis of Covariance

When to use it

The cricket example is shown in the “How to do the test” section.

Null hypotheses

Assumptions

How the test works

Examples

Graphing the results

Similar tests

See the *Handbook* for information on these topics.

How to do the test

Analysis of covariance example with two categories and type II sum of squares

This example uses type II sum of squares, but otherwise follows the example in the *Handbook*.

The parameter estimates are calculated differently in R, so the calculation of the intercepts of the lines is slightly different.

```
### -----
### Analysis of covariance, cricket example
### pp. 228–229
### -----
```

```
Input = ("
Species Temp Pulse
```

```

ex      20.8    67.9
ex      20.8    65.1
ex      24      77.3
ex      24      78.7
ex      24      79.4
ex      24      80.4
ex      26.2    85.8
ex      26.2    86.6
ex      26.2    87.5
ex      26.2    89.1
ex      28.4    98.6
ex      29      100.8
ex      30.4    99.3
ex      30.4    101.7
niv     17.2    44.3
niv     18.3    47.2
niv     18.3    47.6
niv     18.3    49.6
niv     18.9    50.3
niv     18.9    51.8
niv     20.4    60
niv     21      58.5
niv     21      58.9
niv     22.1    60.7
niv     23.5    69.8
niv     24.2    70.9
niv     25.9    76.2
niv     26.5    76.1
niv     26.5    77
niv     26.5    77.7
niv     28.6    84.7
")

```

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

Simple plot

```

plot(x   = Data$Temp,
     y   = Data$Pulse,
     col = Data$Species,
     pch = 16,
     xlab = "Temperature",
     ylab = "Pulse")

legend('bottomright',
      legend = levels(Data$Species),
      col = 1:2,
      cex = 1,
      pch = 16)

```

Analysis of covariance

```
options(contrasts = c("contr.treatment", "contr.poly"))
```

```
### These are the default contrasts in R
```

```
model.1 = lm (Pulse ~ Temp + Species + Temp:Species,
              data = Data)
```

```
library(car)
```

```
Anova(model.1, type="II")
```

```
Anova Table (Type II tests)
```

	Sum Sq	Df	F value	Pr(>F)
Temp	4376.1	1	1388.839	< 2.2e-16 ***
Species	598.0	1	189.789	9.907e-14 ***
Temp:Species	4.3	1	1.357	0.2542

```
### Interaction is not significant, so the slope across groups
### is not different.
```

```
model.2 = lm (Pulse ~ Temp + Species,
              data = Data)
```

```
library(car)
```

```
Anova(model.2, type="II")
```

```
Anova Table (Type II tests)
```

	Sum Sq	Df	F value	Pr(>F)
Temp	4376.1	1	1371.4	< 2.2e-16 ***
Species	598.0	1	187.4	6.272e-14 ***

```
### The category variable (Species) is significant,
### so the intercepts among groups are different
```

```
summary(model.2)
```

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-7.21091	2.55094	-2.827	0.00858 **
Temp	3.60275	0.09729	37.032	< 2e-16 ***
Speciesniv	-10.06529	0.73526	-13.689	6.27e-14 ***

```
### Note that these estimates are different than in the Handbook,
### but the calculated results will be identical.
### The slope estimate is the same.
### The intercept for species 1 (ex) is (intercept).
### The intercept for species 2 (niv) is (intercept) + Speciesniv.
### This is determined from the contrast coding of the Species
### variable shown below, and the fact that Speciesniv is shown in
```

```
### coefficient table above.
```

```
contrasts(Data$Species)
```

```
      niv
ex      0
niv     1
```

Simple plot with fitted lines

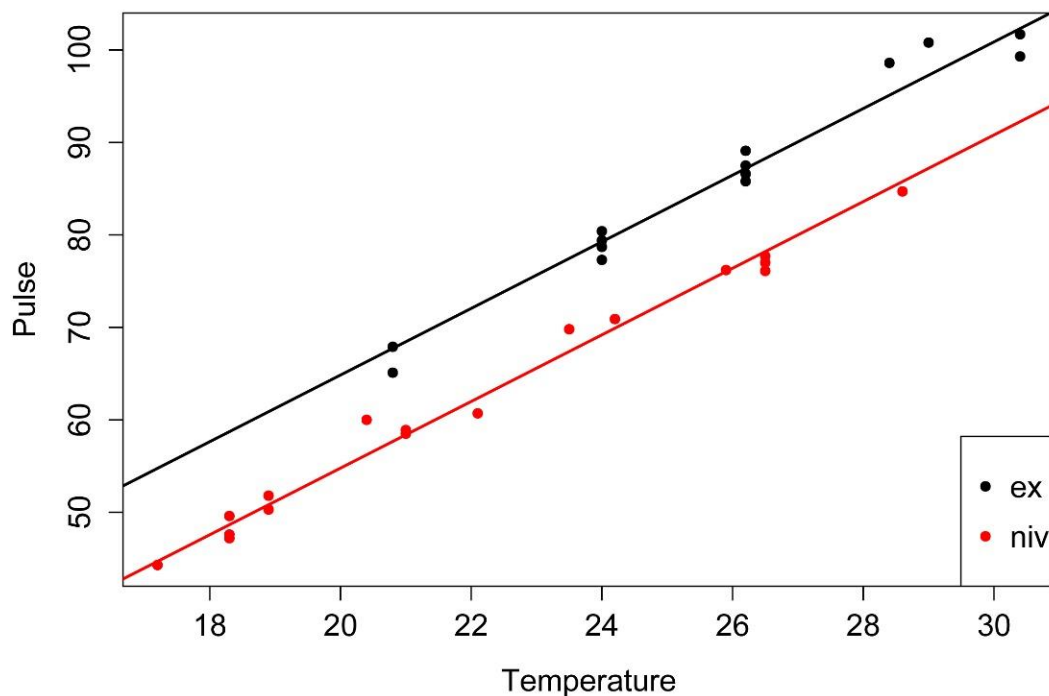
```
I.nought = -7.21091
I1 = I.nought + 0
I2 = I.nought + -10.06529
B = 3.60275

plot(x = Data$Temp,
     y = Data$Pulse,
     col = Data$Species,
     pch = 16,
     xlab = "Temperature",
     ylab = "Pulse")

legend('bottomright',
      legend = levels(Data$Species),
      col = 1:2,
      cex = 1,
      pch = 16)

abline(I1, B,
      lty=1, lwd=2, col = 1)

abline(I2, B,
      lty=1, lwd=2, col = 2)
```



p-value and R-squared of combined model

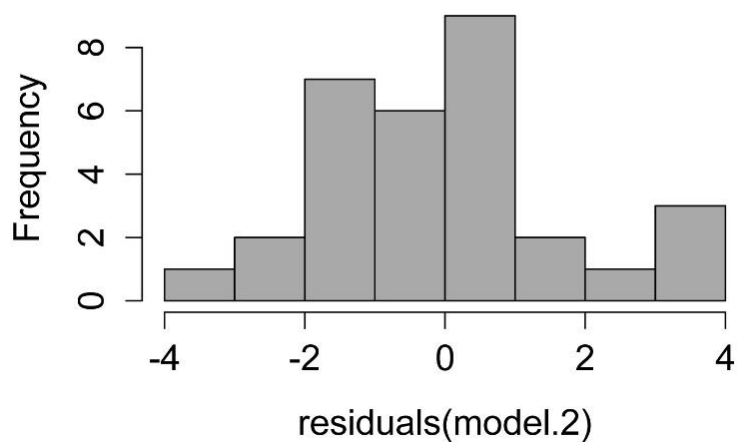
```
summary(model.2)
```

Multiple R-squared: 0.9896, Adjusted R-squared: 0.9888
F-statistic: 1331 on 2 and 28 DF, p-value: < 2.2e-16

Checking assumptions of the model

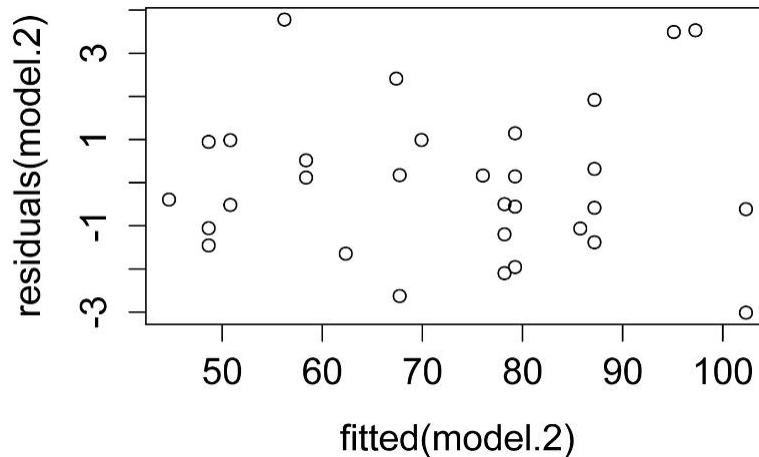
```
hist(residuals(model.2),  
     col="darkgray")
```

Histogram of residuals(model.2)



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model.2),
     residuals(model.2))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model.2)
### alternative: library(FSA); residPlot(model.2)
```

```
# # #
```

Analysis of covariance example with three categories and type II sum of squares

This example uses type II sum of squares, and considers a case with three groups.

```
### -----
### Analysis of covariance, hypothetical data
### -----
```

```
Input = ("
Species Temp Pulse
ex      20.8 67.9
ex      20.8 65.1
ex      24   77.3
ex      24   78.7
ex      24   79.4
ex      24   80.4
ex      26.2 85.8
ex      26.2 86.6
ex      26.2 87.5
ex      26.2 89.1
ex      28.4 98.6
```

```

ex      29      100.8
ex      30.4     99.3
ex      30.4    101.7
niv     17.2     44.3
niv     18.3     47.2
niv     18.3     47.6
niv     18.3     49.6
niv     18.9     50.3
niv     18.9     51.8
niv     20.4     60
niv     21       58.5
niv     21       58.9
niv     22.1     60.7
niv     23.5     69.8
niv     24.2     70.9
niv     25.9     76.2
niv     26.5     76.1
niv     26.5     77
niv     26.5     77.7
niv     28.6     84.7
fake    17.2     74.3
fake    18.3     77.2
fake    18.3     77.6
fake    18.3     79.6
fake    18.9     80.3
fake    18.9     81.8
fake    20.4     90
fake    21       88.5
fake    21       88.9
fake    22.1     90.7
fake    23.5     99.8
fake    24.2    100.9
fake    25.9    106.2
fake    26.5    106.1
fake    26.5    107
fake    26.5    107.7
fake    28.6    114.7
")

```

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

Simple plot

```

plot(x   = Data$Temp,
     y   = Data$Pulse,
     col = Data$Species,
     pch = 16,
     xlab = "Temperature",
     ylab = "Pulse")

legend('bottomright',
      legend = levels(Data$Species),
      col = 1:3,

```



```
cex = 1,
pch = 16)
```

Analysis of covariance

```
options(contrasts = c("contr.treatment", "contr.poly"))
```

```
### These are the default contrasts in R
```

```
model.1 = lm (Pulse ~ Temp + Species + Temp:Species,
              data = Data)
```

```
library(car)
```

```
Anova(model.1, type="II")
```

	Sum Sq	Df	F value	Pr(>F)	
Temp	7026.0	1	2452.4187	<2e-16	***
Species	7835.7	2	1367.5377	<2e-16	***
Temp:Species	5.2	2	0.9126	0.4093	

```
### Interaction is not significant, so the slope among groups
### is not different.
```

```
model.2 = lm (Pulse ~ Temp + Species,
              data = Data)
```

```
library(car)
```

```
Anova(model.2, type="II")
```

	Sum Sq	Df	F value	Pr(>F)	
Temp	7026.0	1	2462.2	< 2.2e-16	***
Species	7835.7	2	1373.0	< 2.2e-16	***
Residuals	125.6	44			

```
### The category variable (Species) is significant,
### so the intercepts among groups are different
```

```
summary(model.2)
```

```
Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-6.35729	1.90713	-3.333	0.00175	**
Temp	3.56961	0.07194	49.621	< 2e-16	***
Speciesfake	19.81429	0.66333	29.871	< 2e-16	***
Speciesniv	-10.18571	0.66333	-15.355	< 2e-16	***

```
### The slope estimate is the Temp coefficient.
### The intercept for species 1 (ex) is (intercept).
### The intercept for species 2 (fake) is (intercept) + Speciesfake.
```

```
### The intercept for species 3 (niv) is (intercept) + Speciesniv.
### This is determined from the contrast coding of the Species
### variable shown below.
```

```
contrasts(Data$Species)
```

```
      fake niv
ex      0   0
fake    1   0
niv     0   1
```

Simple plot with fitted lines

```
I.nought = -6.35729
I1 = I.nought + 0
I2 = I.nought + 19.81429
I3 = I.nought + -10.18571
B = 3.56961

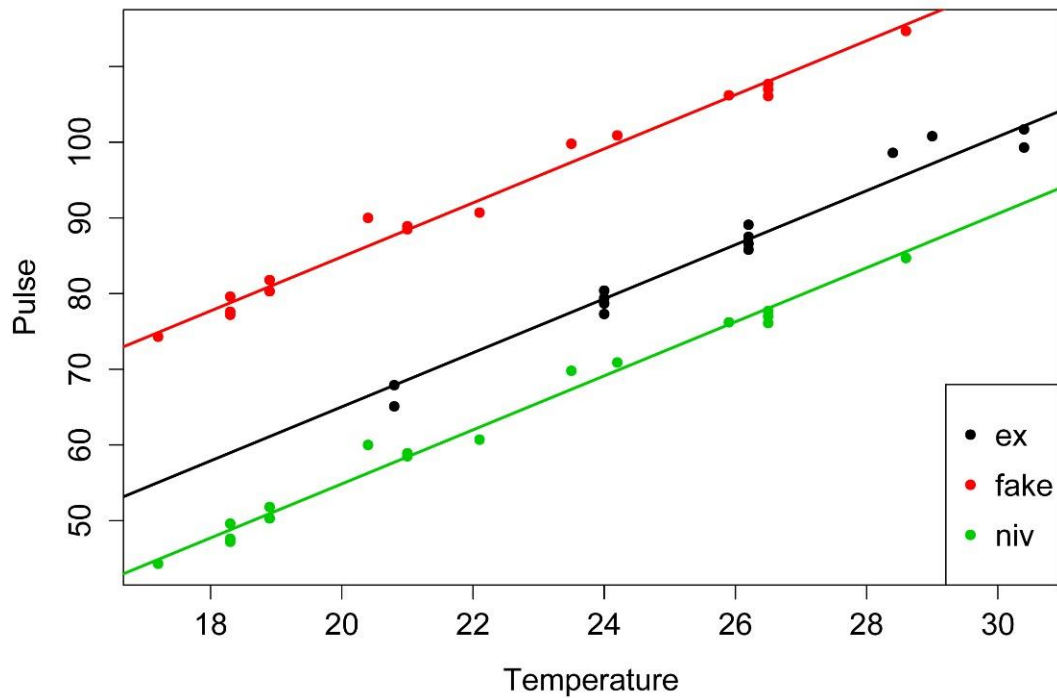
plot(x = Data$Temp,
     y = Data$Pulse,
     col = Data$Species,
     pch = 16,
     xlab = "Temperature",
     ylab = "Pulse")

legend('bottomright',
      legend = levels(Data$Species),
      col = 1:3,
      cex = 1,
      pch = 16)

abline(I1, B,
      lty=1, lwd=2, col = 1)

abline(I2, B,
      lty=1, lwd=2, col = 2)

abline(I3, B,
      lty=1, lwd=2, col = 3)
```



p-value and R-squared of combined model

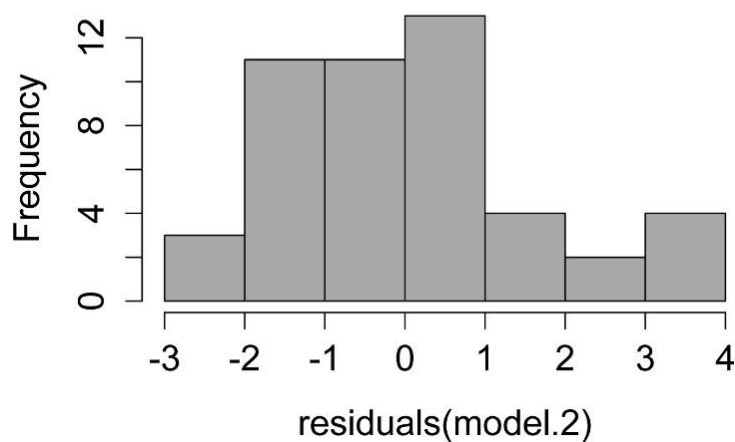
```
summary(model.2)
```

Multiple R-squared: 0.9919, Adjusted R-squared: 0.9913
F-statistic: 1791 on 3 and 44 DF, p-value: < 2.2e-16

Checking assumptions of the model

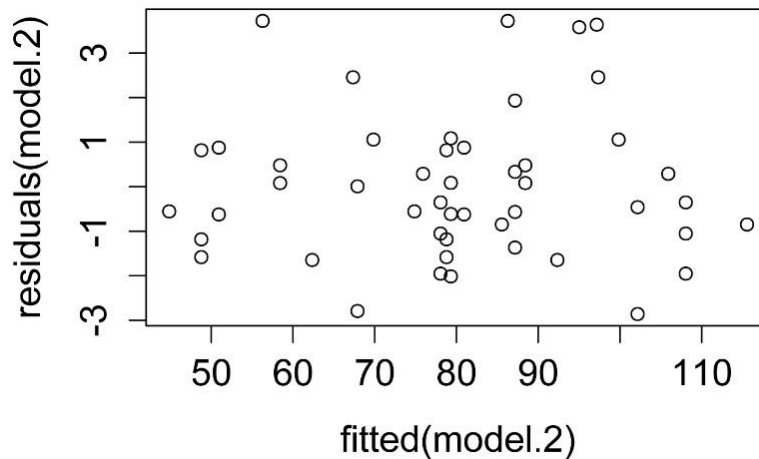
```
hist(residuals(model.2),  
     col="darkgray")
```

Histogram of residuals(model.2)



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model.2),
     residuals(model.2))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model.2)
### alternative: library(FSA); residPlot(model.2)
```

```
# # #
```

Power analysis

See the *Handbook* for information on this topic.

Multiple Regression

When to use it

Null hypothesis

How it works

Using nominal variables in a multiple regression

Selecting variables in multiple regression

Assumptions

See the *Handbook* for information on these topics.

Example

The Maryland Biological Stream Survey example is shown in the “How to do the multiple regression” section.

Graphing the results

Similar tests

See the *Handbook* for information on these topics.

How to do multiple regression

Multiple correlation

Whenever you have a dataset with multiple numeric variables, it is a good idea to look at the correlations among these variables. One reason is that if you have a dependent variable, you can easily see which independent variables correlate with that dependent variable. A second reason is that if you will be constructing a multiple regression model, adding an independent variable that is strongly correlated with an independent variable already in the model is unlikely to improve the model much, and you may have a good reason to chose one variable over another.

Finally, it is worthwhile to look at the distribution of the numeric variables. If the distributions differ greatly, using Kendall or Spearman correlations may be more appropriate. Also, if independent variables differ in distribution from the dependent variable, the independent variables may need to be transformed. In this example, *Longnose*, *Acreage*, *Maxdepth*, *NO3*, and *SO4* are relatively log-normally distributed, while *DO2* and *Temp* are relatively normal in distribution. It may be advisable in this case to transform these variable so that they all have similar distributions (not shown here).

With the *corr.test* function in the *psych* package, the “Correlation matrix” shows r-values and the “Probability values” table shows p-values. The *PerformanceAnalytics* plot shows r-values, with asterisks indicating significance, as well as a histogram of the individual variables. Either of these indicates that *Longnose* is significantly correlated with *Acreage*, *Maxdepth*, and *NO3*.

```
### -----
### Multiple correlation and regression, stream survey example
### pp. 236–237
### -----
```

```
Input = ("
Stream          Longnose  Acreage  DO2   Maxdepth  NO3   SO4   Temp
BASIN_RUN       13       2528    9.6   80        2.28  16.75  15.3
BEAR_BR         12       3333    8.5   83        5.34   7.74  19.4
BEAR_CR         54      19611    8.3   96        0.99  10.92  19.5
BEAVER_DAM_CR   19       3570    9.2   56        5.44  16.53  17
BEAVER_RUN      37       1722    8.1   43        5.66   5.91  19.3
BENNETT_CR      2        583    9.2   51        2.26   8.81  12.9
BIG_BR          72       4790    9.4   91        4.1    5.65  16.7
BIG_ELK_CR     164      35971   10.2  81        3.2   17.53  13.8
BIG_PIPE_CR     18      25440    7.5  120        3.53   8.2   13.7
BLUE_LICK_RUN    1       2217    8.5   46        1.2   10.85  14.3
BROAD_RUN       53       1971   11.9  56        3.25  11.12  22.2
BUFFALO_RUN     16      12620    8.3   37        0.61  18.87  16.8
BUSH_CR         32      19046    8.3  120        2.93  11.31  18
CABIN_JOHN_CR   21       8612    8.2  103        1.57  16.09  15
CARROLL_BR      23       3896   10.4  105        2.77  12.79  18.4
```

COLLIER_RUN	18	6298	8.6	42	0.26	17.63	18.2
CONOWINGO_CR	112	27350	8.5	65	6.95	14.94	24.1
DEAD_RUN	25	4145	8.7	51	0.34	44.93	23
DEEP_RUN	5	1175	7.7	57	1.3	21.68	21.8
DEER_CR	26	8297	9.9	60	5.26	6.36	19.1
DORSEY_RUN	8	7814	6.8	160	0.44	20.24	22.6
FALLS_RUN	15	1745	9.4	48	2.19	10.27	14.3
FISHING_CR	11	5046	7.6	109	0.73	7.1	19
FLINTSTONE_CR	11	18943	9.2	50	0.25	14.21	18.5
GREAT_SENECA_CR	87	8624	8.6	78	3.37	7.51	21.3
GREENE_BR	33	2225	9.1	41	2.3	9.72	20.5
GUNPOWDER_FALLS	22	12659	9.7	65	3.3	5.98	18
HAINES_BR	98	1967	8.6	50	7.71	26.44	16.8
HAWLINGS_R	1	1172	8.3	73	2.62	4.64	20.5
HAY_MEADOW_BR	5	639	9.5	26	3.53	4.46	20.1
HERRINGTON_RUN	1	7056	6.4	60	0.25	9.82	24.5
HOLLANDS_BR	38	1934	10.5	85	2.34	11.44	12
ISRAEL_CR	30	6260	9.5	133	2.41	13.77	21
LIBERTY_RES	12	424	8.3	62	3.49	5.82	20.2
LITTLE_ANTIETAM_CR	24	3488	9.3	44	2.11	13.37	24
LITTLE_BEAR_CR	6	3330	9.1	67	0.81	8.16	14.9
LITTLE_CONOCOCHIEGUE_CR	15	2227	6.8	54	0.33	7.6	24
LITTLE_DEER_CR	38	8115	9.6	110	3.4	9.22	20.5
LITTLE_FALLS	84	1600	10.2	56	3.54	5.69	19.5
LITTLE_GUNPOWDER_R	3	15305	9.7	85	2.6	6.96	17.5
LITTLE_HUNTING_CR	18	7121	9.5	58	0.51	7.41	16
LITTLE_PAINT_BR	63	5794	9.4	34	1.19	12.27	17.5
MAINSTEM_PATUXENT_R	239	8636	8.4	150	3.31	5.95	18.1
MEADOW_BR	234	4803	8.5	93	5.01	10.98	24.3
MILL_CR	6	1097	8.3	53	1.71	15.77	13.1
MORGAN_RUN	76	9765	9.3	130	4.38	5.74	16.9
MUDDY_BR	25	4266	8.9	68	2.05	12.77	17
MUDLICK_RUN	8	1507	7.4	51	0.84	16.3	21
NORTH_BR	23	3836	8.3	121	1.32	7.36	18.5
NORTH_BR_CASSELMAN_R	16	17419	7.4	48	0.29	2.5	18
NORTHWEST_BR	6	8735	8.2	63	1.56	13.22	20.8
NORTHWEST_BR_ANACOSTIA_R	100	22550	8.4	107	1.41	14.45	23
OWENS_CR	80	9961	8.6	79	1.02	9.07	21.8
PATAPSCO_R	28	4706	8.9	61	4.06	9.9	19.7
PINEY_BR	48	4011	8.3	52	4.7	5.38	18.9
PINEY_CR	18	6949	9.3	100	4.57	17.84	18.6
PINEY_RUN	36	11405	9.2	70	2.17	10.17	23.6
PRETTYBOY_BR	19	904	9.8	39	6.81	9.2	19.2
RED_RUN	32	3332	8.4	73	2.09	5.5	17.7
ROCK_CR	3	575	6.8	33	2.47	7.61	18
SAVAGE_R	106	29708	7.7	73	0.63	12.28	21.4
SECOND_MINE_BR	62	2511	10.2	60	4.17	10.75	17.7
SENECA_CR	23	18422	9.9	45	1.58	8.37	20.1
SOUTH_BR_CASSELMAN_R	2	6311	7.6	46	0.64	21.16	18.5
SOUTH_BR_PATAPSCO	26	1450	7.9	60	2.96	8.84	18.6
SOUTH_FORK_LINGANORE_CR	20	4106	10.0	96	2.62	5.45	15.4
TUSCARORA_CR	38	10274	9.3	90	5.45	24.76	15
WATTS_BR	19	510	6.7	82	5.25	14.19	26.5

")

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

```
### Create a new data frame with only the numeric variables.
```

```
### This is required for corr.test and chart.Correlation
```

```
library(dplyr)
```

```
Data.num =
  select(Data,
         Longnose,
         Acerage,
         DO2,
         Maxdepth,
         NO3,
         SO4,
         Temp)
```

```
library(FSA)
```

```
headtail(Data.num)
```

	Longnose	Acerage	DO2	Maxdepth	NO3	SO4	Temp
1	13	2528	9.6	80	2.28	16.75	15.3
2	12	3333	8.5	83	5.34	7.74	19.4
3	54	19611	8.3	96	0.99	10.92	19.5
66	20	4106	10.0	96	2.62	5.45	15.4
67	38	10274	9.3	90	5.45	24.76	15.0
68	19	510	6.7	82	5.25	14.19	26.5

```
library(psych)
```

```
corr.test(Data.num,
          use = "pairwise",
          method="pearson",
          adjust="none",      # Can adjust p-values; see ?p.adjust for options
          alpha=.05)
```

Correlation matrix

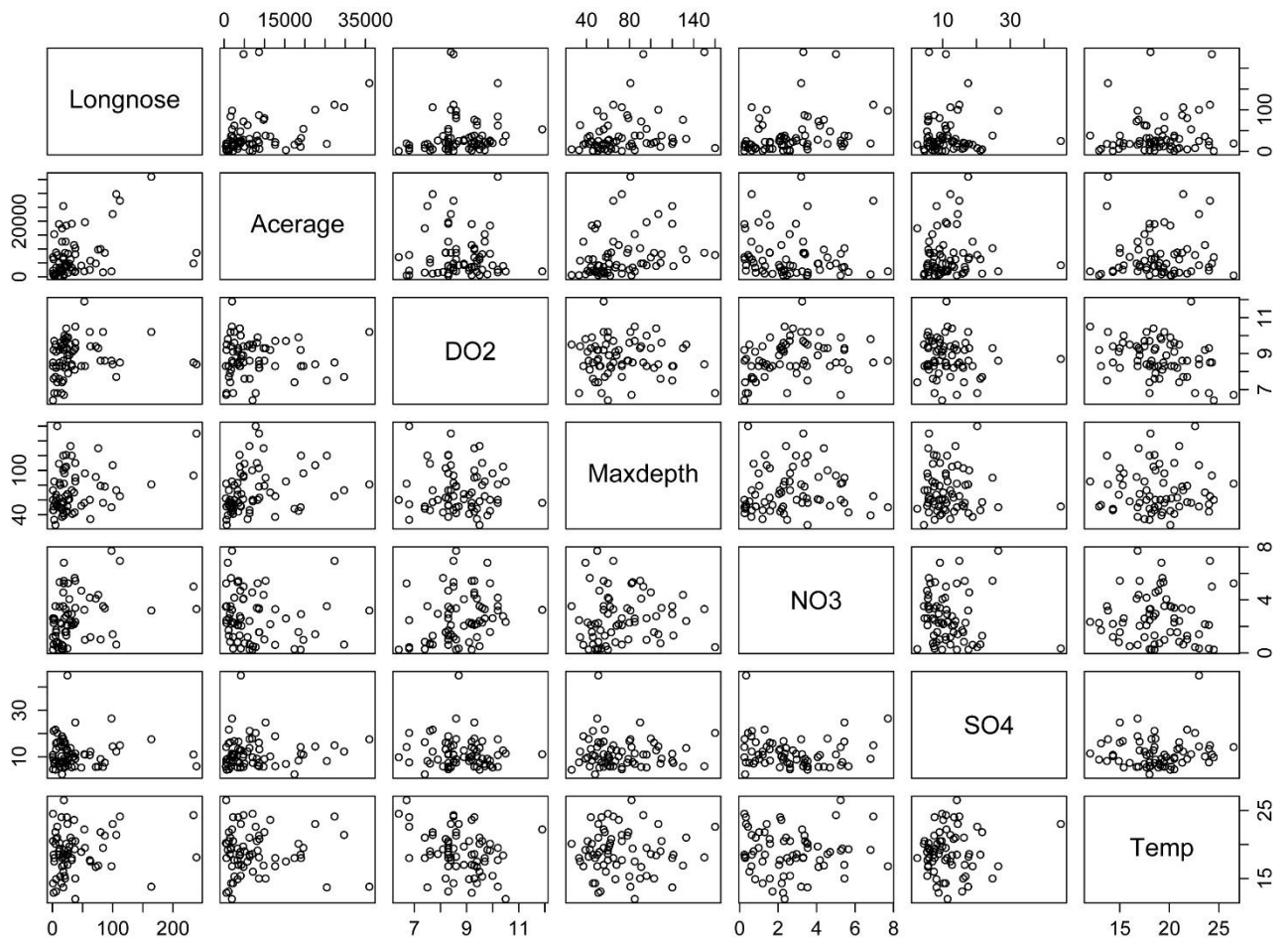
	Longnose	Acerage	DO2	Maxdepth	NO3	SO4	Temp
Longnose	1.00	0.35	0.14	0.30	0.31	-0.02	0.14
Acerage	0.35	1.00	-0.02	0.26	-0.10	0.05	0.00
DO2	0.14	-0.02	1.00	-0.06	0.27	-0.07	-0.32
Maxdepth	0.30	0.26	-0.06	1.00	0.04	-0.05	0.00
NO3	0.31	-0.10	0.27	0.04	1.00	-0.09	0.00
SO4	-0.02	0.05	-0.07	-0.05	-0.09	1.00	0.08
Temp	0.14	0.00	-0.32	0.00	0.00	0.08	1.00

Sample Size

Probability values (Entries above the diagonal are adjusted for multiple tests.)

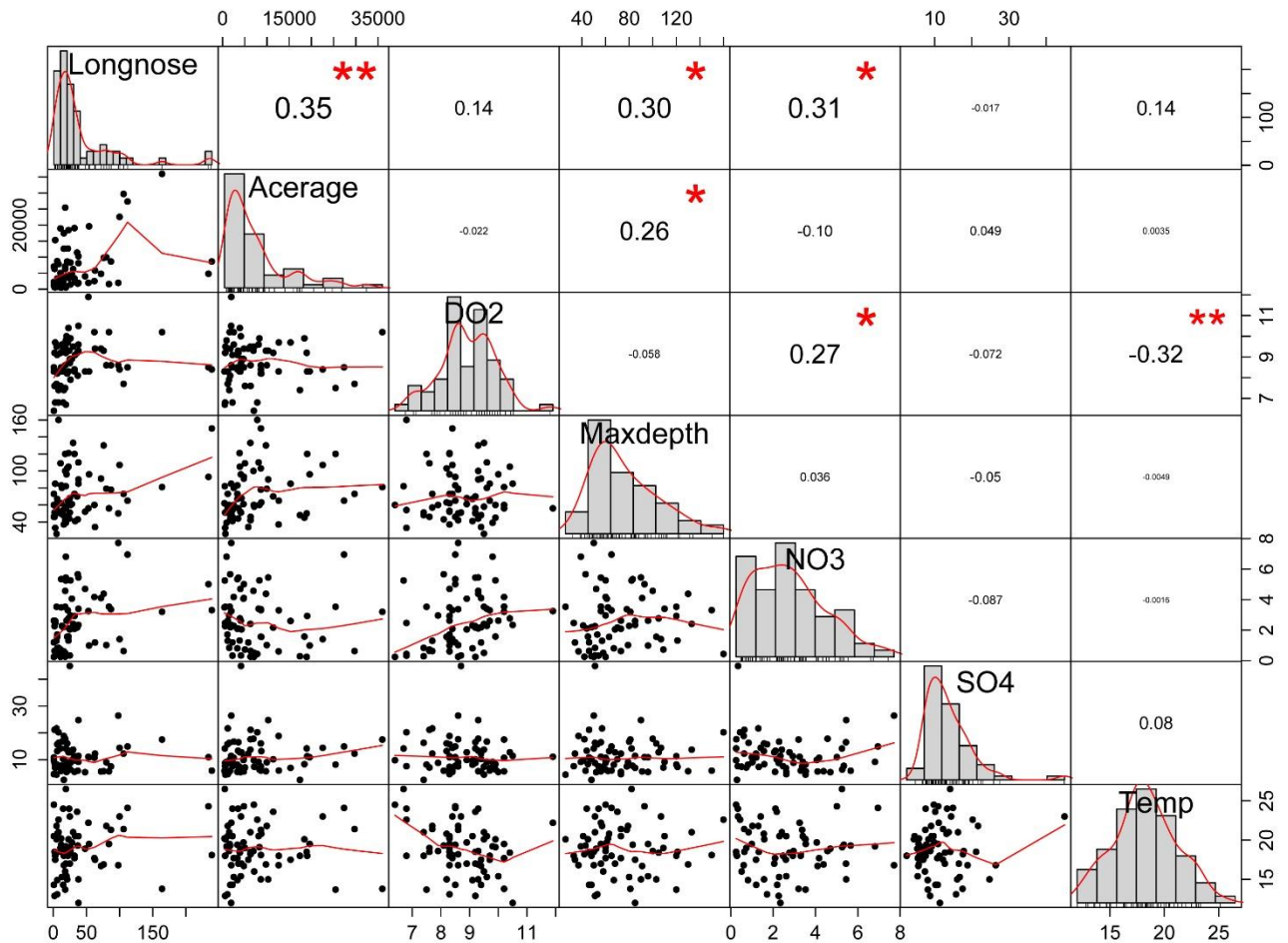
	Longnose	Acerage	DO2	Maxdepth	NO3	SO4	Temp
Longnose	0.00	0.00	0.27	0.01	0.01	0.89	0.26
Acerage	0.00	0.00	0.86	0.03	0.42	0.69	0.98
DO2	0.27	0.86	0.00	0.64	0.02	0.56	0.01
Maxdepth	0.01	0.03	0.64	0.00	0.77	0.69	0.97
NO3	0.01	0.42	0.02	0.77	0.00	0.48	0.99
SO4	0.89	0.69	0.56	0.69	0.48	0.00	0.52
Temp	0.26	0.98	0.01	0.97	0.99	0.52	0.00

```
pairs(data=Data,
      ~ Longnose + Acerage + DO2 + Maxdepth + NO3 + SO4 + Temp)
```



```
library(PerformanceAnalytics)

chart.Correlation(Data.num,
                  method="pearson",
                  histogram=TRUE,
                  pch=16)
```

Multiple regression

Model selection using the *step* function

The *step* function has options to add terms to a model (*direction="forward"*), remove terms from a model (*direction="backward"*), or to use a process that both adds and removes terms (*direction="both"*). It uses AIC (Akaike information criterion) as a selection criterion. You can use the option $k = \log(n)$ to use BIC instead.

You can add the *test="F"* option to see the p-value for adding or removing terms, but the test will still follow the AIC statistic. If you use this, however, note that a significant p-value essentially argues for the term being included in the model, whether it's its addition or its removal that's being considered.

A full model and a null are defined, and then the function will follow a procedure to find the model with the lowest AIC. The final model is shown at the end of the output, with the *Call*: indication, and lists the coefficients for that model.

Stepwise procedure

```
model.null = lm(Longnose ~ 1,
                 data=Data)
```

```
model.full = lm(Longnose ~ Acerage + DO2 + Maxdepth + NO3 + SO4 + Temp,
                 data=Data)
```

```
step(model.null,
      scope = list(upper=model.full),
      direction="both",
      data=Data)
```

```
Longnose ~ 1
```

	Df	Sum of Sq	RSS	AIC
+ Acerage	1	17989.6	131841	518.75
+ NO3	1	14327.5	135503	520.61
+ Maxdepth	1	13936.1	135894	520.81
<none>			149831	525.45
+ Temp	1	2931.0	146899	526.10
+ DO2	1	2777.7	147053	526.17
+ SO4	1	45.3	149785	527.43

```
.
.
< snip... more steps >
```

```
.
.
Longnose ~ Acerage + NO3 + Maxdepth
```

	Df	Sum of Sq	RSS	AIC
<none>			107904	509.13
+ Temp	1	2948.0	104956	509.24
+ DO2	1	669.6	107234	510.70
- Maxdepth	1	6058.4	113962	510.84
+ SO4	1	5.9	107898	511.12
- Acerage	1	14652.0	122556	515.78
- NO3	1	16489.3	124393	516.80

```
Call:
```

```
lm(formula = Longnose ~ Acerage + NO3 + Maxdepth, data = Data)
```

```
Coefficients:
```

(Intercept)	Acerage	NO3	Maxdepth
-23.829067	0.001988	8.673044	0.336605

Define final model

```
model.final = lm(Longnose ~ Acerage + Maxdepth + NO3,
                  data=Data)
```

```
summary(model.final)      # Show coefficients, R-squared, and overall p-value
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-2.383e+01	1.527e+01	-1.560	0.12367
Acerage	1.988e-03	6.742e-04	2.948	0.00446 **
Maxdepth	3.366e-01	1.776e-01	1.896	0.06253 .
NO3	8.673e+00	2.773e+00	3.127	0.00265 **

Multiple R-squared: 0.2798, Adjusted R-squared: 0.2461
 F-statistic: 8.289 on 3 and 64 DF, p-value: 9.717e-05

Analysis of variance for individual terms

```
library(car)
```

```
Anova(model.final,  
      Type="II")
```

Anova Table (Type II tests)

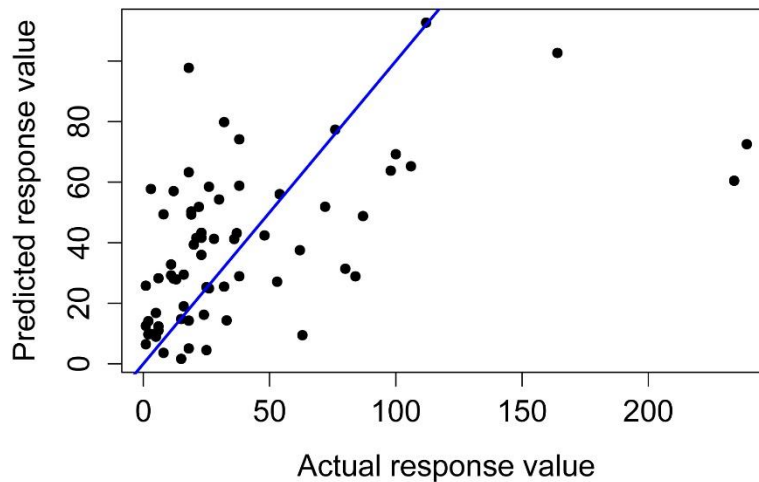
```
Response: Longnose
      Sum Sq Df F value    Pr(>F)
Acerage   14652  1  8.6904 0.004461 **
Maxdepth   6058  1  3.5933 0.062529 .
NO3        16489  1  9.7802 0.002654 **
Residuals 107904 64
```

Simple plot of predicted values with 1-to-1 line

```
Data$predy = predict(model.final)
```

```
plot(predy ~ Longnose,  
     data=Data,  
     pch = 16,  
     xlab="Actual response value",  
     ylab="Predicted response value")
```

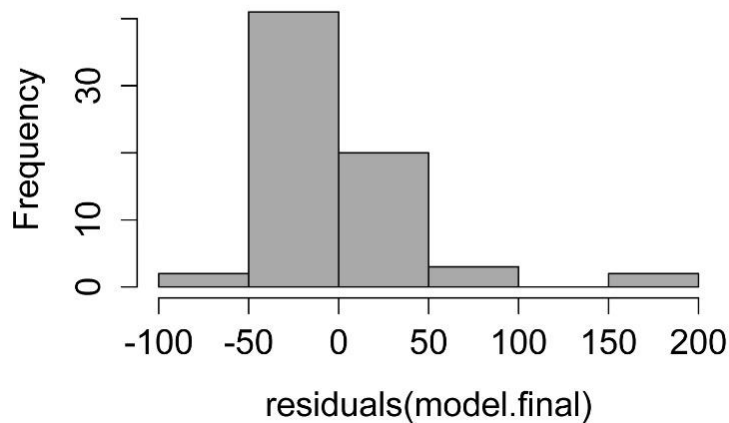
```
abline(0,1, col="blue", lwd=2)
```



Checking assumptions of the model

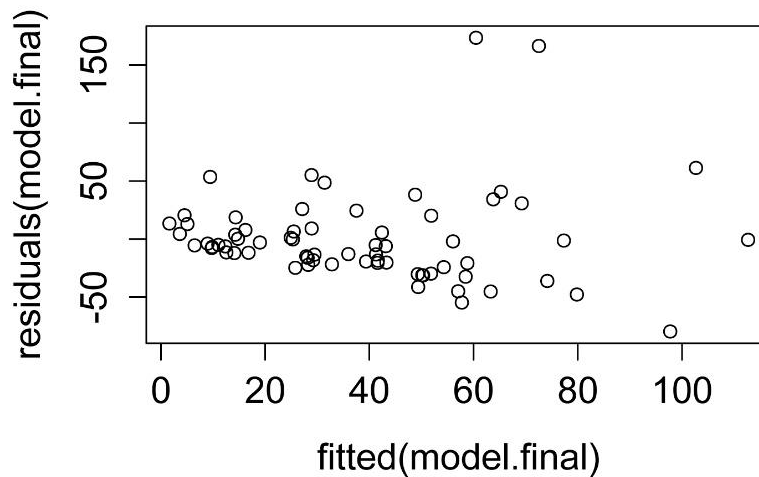
```
hist(residuals(model.final),  
     col="darkgray")
```

Histogram of residuals(model.final)



A histogram of residuals from a linear model. The distribution of these residuals should be approximately normal.

```
plot(fitted(model.final),
     residuals(model.final))
```



A plot of residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model.final)
```

Model fit criteria

Model fit criteria are available to decide which model is most appropriate. The step function uses AIC, or optionally BIC, but there are others. You don't want to use multiple R-squared, because it will continue to improve as more terms are added into the model. Instead, you want to use a criterion that balances the improvement in explanatory power with not adding

extraneous terms to the model. Adjusted R-squared is a modification of R-squared that includes this balance. Larger is better. AIC is based on information theory and measures this balance. AICc is an adjustment to AIC that is more appropriate for data sets with relatively fewer observations. BIC is similar to AIC, but penalizes more for additional terms in the model. Smaller is better for AIC, AICc, and BIC. There are differing opinions on which model fitting criteria is best to use, but if you have no opinion, I would recommend AICc for routine use.

Using the *step* procedure to automatically find an optimal model is an option, but some people caution against using an automated procedure because it might not hone in on the best model. Instead, you can look at the model fit criteria for competing models manually. There may be reasons why you wish to include or exclude some terms in the model, and it may be useful to look at different model selection criteria simultaneously.

In my *compare.lm* function below, *Shapiro.W* and *Shapiro.p* are results from the Shapiro–Wilks test for normality on the model residuals. A higher Shapiro W and a higher Shapiro p indicate that the residuals are more normally distributed. You should be aware, however, that any model with a high number of observation may yield a significant p-value ($p < 0.05$) for the Shapiro–Wilks test. It is best to investigate the residuals visually.

In the following example, we'll look only at the terms that are significantly correlated with *Longnose* (*Acerage*, *Maxdepth*, and *NO3*), and then add in the other terms just to show the decrease in AICc by adding extra terms.

Note that AIC and BIC are calculated differently than in the *step* function.

```
model.1 = lm(Longnose ~ Acerage, data=Data)
model.2 = lm(Longnose ~ Maxdepth, data=Data)
model.3 = lm(Longnose ~ NO3, data=Data)
model.4 = lm(Longnose ~ Acerage + Maxdepth, data=Data)
model.5 = lm(Longnose ~ Acerage + NO3, data=Data)
model.6 = lm(Longnose ~ Maxdepth + NO3, data=Data)
model.7 = lm(Longnose ~ Acerage + Maxdepth + NO3, data=Data)
model.8 = lm(Longnose ~ Acerage + Maxdepth + NO3 + DO2, data=Data)
model.9 = lm(Longnose ~ Acerage + Maxdepth + NO3 + SO4, data=Data)
model.10 = lm(Longnose ~ Acerage + Maxdepth + NO3 + Temp, data=Data)
```

```
library(rcompanion)
```

```
compareLM(model.1, model.2, model.3, model.4, model.5, model.6,
           model.7, model.8, model.9, model.10)
```

```
$Models
  Formula
1 "Longnose ~ Acerage"
2 "Longnose ~ Maxdepth"
3 "Longnose ~ NO3"
4 "Longnose ~ Acerage + Maxdepth"
5 "Longnose ~ Acerage + NO3"
6 "Longnose ~ Maxdepth + NO3"
7 "Longnose ~ Acerage + Maxdepth + NO3"
8 "Longnose ~ Acerage + Maxdepth + NO3 + DO2"
```

```
9 "Longnose ~ Acerage + Maxdepth + NO3 + SO4"
10 "Longnose ~ Acerage + Maxdepth + NO3 + Temp"
```

```
$Fit.criteria
```

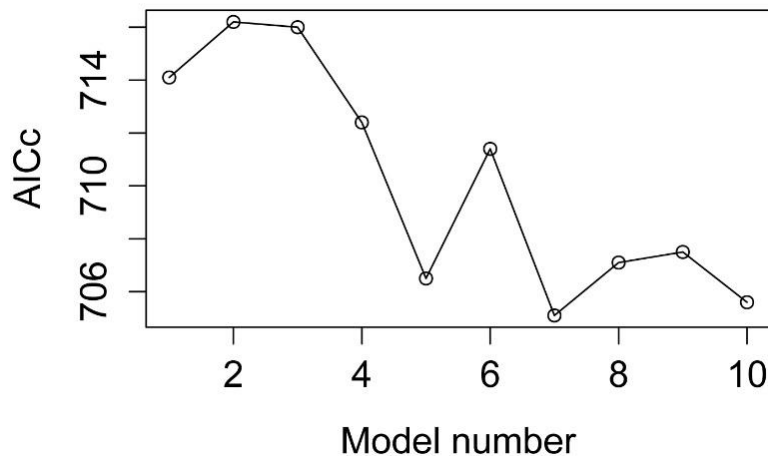
	Rank	Df.res	AIC	AICc	BIC	R.squared	Adj.R.sq	p.value	Shapiro.w	Shapiro.p
1	2	66	713.7	714.1	720.4	0.12010	0.10670	3.796e-03	0.7278	6.460e-10
2	2	66	715.8	716.2	722.4	0.09301	0.07927	1.144e-02	0.7923	2.115e-08
3	2	66	715.6	716.0	722.2	0.09562	0.08192	1.029e-02	0.7361	9.803e-10
4	3	65	711.8	712.4	720.6	0.16980	0.14420	2.365e-03	0.7934	2.250e-08
5	3	65	705.8	706.5	714.7	0.23940	0.21600	1.373e-04	0.7505	2.055e-09
6	3	65	710.8	711.4	719.6	0.18200	0.15690	1.458e-03	0.8149	8.405e-08
7	4	64	704.1	705.1	715.2	0.27980	0.24610	9.717e-05	0.8108	6.511e-08
8	5	63	705.7	707.1	719.0	0.28430	0.23890	2.643e-04	0.8041	4.283e-08
9	5	63	706.1	707.5	719.4	0.27990	0.23410	3.166e-04	0.8104	6.345e-08
10	5	63	704.2	705.6	717.5	0.29950	0.25500	1.409e-04	0.8225	1.371e-07

```
### Model 7 is the model which minimizes AICc, which is the same model
### chosen by the step function
```

```
Result = compare.lm(model.1, model.2, model.3, model.4, model.5, model.6,
                    model.7, model.8, model.9, model.10)
```

```
plot(Result$Fit.criteria$AICc,
     xlab = "Model number",
     ylab = "AICc")
```

```
lines(Result$Fit.criteria$AICc)
```



A plot of AICc (modified Akaike information criterion) of several models. Model 7 minimizes AICc, and is therefore chosen as the best model out of this set.

Comparing models with likelihood ratio test

It may also be helpful to compare models with the extra sum of squares test or likelihood ratio test to see if additional terms significantly reduce the error sum of squares.

One of the compared models should be nested within the other. That is, the one model should be the same as the other, except with additional terms. For example in the set of models below, it is

appropriate to compare *model.7* to *model.4*. Or to compare each of *model.8*, *model.9*, and *model.10* to *model.7*.

For a single comparison, the *anova* function can be used for the Extra SS test, or *lrtest* in *lmtest* can be used for the likelihood ratio test. For multiple comparisons, the *extraSS* and *lrt* functions in the *FSA* package can be used. The *extraSS* function works only for *lm* and *nls* models, whereas the *lrt* function works on a wider range of model objects.

```
model.4 = lm(Longnose ~ Acerage + Maxdepth, data=Data)
model.7 = lm(Longnose ~ Acerage + Maxdepth + NO3, data=Data)
model.8 = lm(Longnose ~ Acerage + Maxdepth + NO3 + DO2, data=Data)
model.9 = lm(Longnose ~ Acerage + Maxdepth + NO3 + SO4, data=Data)
model.10 = lm(Longnose ~ Acerage + Maxdepth + NO3 + Temp, data=Data)
```

```
anova(model.7, model.4)
```

Analysis of Variance Table

```
Model 1: Longnose ~ Acerage + Maxdepth + NO3
Model 2: Longnose ~ Acerage + Maxdepth
```

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	64	107904				
2	65	124393	-1	-16489	9.7802	0.002654 **

```
library(lmtest)
```

```
lrtest(model.7, model.4)
```

Likelihood ratio test

```
Model 1: Longnose ~ Acerage + Maxdepth + NO3
Model 2: Longnose ~ Acerage + Maxdepth
```

	#Df	LogLik	Df	Chisq	Pr(>Chisq)
1	5	-347.05			
2	4	-351.89	-1	9.6701	0.001873 **

```
library(FSA)
```

```
extraSS(model.8, model.9, model.10,
        com=model.7)
```

```
Model 1: Longnose ~ Acerage + Maxdepth + NO3 + DO2
Model 2: Longnose ~ Acerage + Maxdepth + NO3 + SO4
Model 3: Longnose ~ Acerage + Maxdepth + NO3 + Temp
Model A: Longnose ~ Acerage + Maxdepth + NO3
```

	DfO	RSSO	DfA	RSSA	Df	SS	F	Pr(>F)
1vA	63	107234.38	64	107903.97	-1	-669.59	0.3934	0.5328
2vA	63	107898.06	64	107903.97	-1	-5.91	0.0035	0.9533

```
3vA 63 104955.97 64 107903.97 -1 -2948.00 1.7695 0.1882
```

```
lrt(model.8, model.9, model.10,
    com=model.7)
```

```
Model 1: Longnose ~ Acerage + Maxdepth + NO3 + DO2
Model 2: Longnose ~ Acerage + Maxdepth + NO3 + SO4
Model 3: Longnose ~ Acerage + Maxdepth + NO3 + Temp
Model A: Longnose ~ Acerage + Maxdepth + NO3
```

	DfO	logLikO	DfA	logLikA	Df	logLik	Chisq	Pr(>Chisq)
1vA	63	-346.83881	64	-347.05045	-1	0.21164	0.4233	0.5153
2vA	63	-347.04859	64	-347.05045	-1	0.00186	0.0037	0.9513
3vA	63	-346.10863	64	-347.05045	-1	0.94182	1.8836	0.1699

```
# # #
```

Power analysis

See the *Handbook* for information on this topic.

Simple Logistic Regression

When to use it

Null hypothesis

How the test works

Assumptions

See the *Handbook* for information on these topics.

Examples

The Mpi example is shown below in the “How to do the test” section.

Graphing the results

Similar tests

See the *Handbook* for information on these topics.

How to do the test

Logistic regression can be performed in R with the *glm* (generalized linear model) function. This function uses a link function to determine which kind of model to use, such as logistic, probit, or poisson. These are indicated in the *family* and *link* options. See *?glm* and *?family* for more information.

Assumptions

Generalized linear models have fewer assumptions than most common parametric tests. Observations still need to be independent, and the correct link function needs to be specified. So,

for example you should understand when to use a poisson regression, and when to use a logistic regression. However, the normal distribution of data or residuals is not required.

Specifying the counts of “successes” and “failures”

Logistic regression has a dependent variable with two levels. In R, this can be specified in three ways. 1) The dependent variable can be a factor variable where the first level is interpreted as “failure” and the other levels are interpreted as “success”. (As in the second example in this chapter). 2) The dependent variable can be a vector of proportions of successes, with the caveat that the number of observations for each proportion is indicated in the *weights* option. 3) The dependent variable can be a matrix with two columns, with the first column being the number of “successes” and the second being the number of “failures”. (As in the first example in this chapter).

Not all proportions or counts are appropriate for logistic regression analysis

Note that in each of these specifications, both the number of successes and the number of failures is known. You should not perform logistic regression on proportion data where you don’t know (or don’t tell R) how many individuals went into those proportions. In statistics, 75% is different if it means 3 out of 4 rather than 150 out of 200. As another example where logistic regression doesn’t apply, the weight people lose in a diet study expressed as a proportion of initial weight cannot be interpreted as a count of “successes” and “failures”. Here, you might be able to use common parametric methods, provided the model assumptions are met; log or arc-sine transformations may be appropriate. Likewise, if you count the number of people in front of you in line, you can’t interpret this as a percentage of people since you don’t know how many people are *not* in front of you in line. In this case with count data as the dependent variable, you might use poisson regression.

Overdispersion

One potential problem to be aware of when using generalized linear models is overdispersion. This occurs when the residual deviance of the model is high relative to the residual degrees of freedom. It is basically an indication that the model doesn’t fit the data well.

It is my understanding, however, that overdispersion is technically not a problem for a simple logistic regression, that is one with a binomial dependent and a single continuous independent variable. Overdispersion is discussed in the chapter on *Multiple logistic regression*.

Pseudo-R-squared

R does not produce r-squared values for generalized linear models (glm). My function *nagelkerke* will calculate the McFadden, Cox and Snell, and Nagelkerke pseudo-R-squared for glm and other model fits. The Cox and Snell is also called the ML, and the Nagelkerke is also called the Cragg and Uhler. These pseudo-R-squared values compare the maximum likelihood of the model to a nested null model fit with the same method. They should not be thought of as the same as the r-squared from an ordinary-least-squares linear (OLS) model, but instead as a relative measure among similar models. The Cox and Snell for an OLS linear model, however, will be equivalent to r-squared for that model. I have seen it mentioned that a McFadden pseudo-R-squared of 0.2–0.4 indicates a good fit. Whereas, I find that the Nagelkerke usually gives a reasonable indication of the goodness of fit for a model on a scale of 0 to 1. That being said, I have found the Cox and Snell and Nagelkerke to sometimes yield values I wouldn’t expect

for some glm. The function *pR2* in the package *pscl* will also produce these pseudo-R-squared values.

Testing for p-values

Note that testing p-values for a logistic or poisson regression uses Chi-square tests. This is achieved through the *test*="Wald" option in *Anova* to test the significance of each coefficient, and the *test*="Chisq" option in *anova* for the significance of the overall model. A likelihood ratio test can also be used to test the significance of the overall model.

Logistic regression example

```
### -----
### Logistic regression, amphipod example, p. 247
### -----

Input = ("
Location      Latitude  mpi90  mpi100
Port_Townsend,_WA  48.1      47    139
Neskowin,_OR      45.2     177    241
Siuslaw_R.,_OR     44.0    1087   1183
Umpqua_R.,_OR     43.7     187    175
Coos_Bay,_OR      43.5     397    671
San_Francisco,_CA  37.8      40     14
Carmel,_CA        36.6      39     17
Santa_Barbara,_CA  34.3      30      0
")

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

Data$Total = Data$mpi90 + Data$mpi100

Data$Percent = Data$mpi100 / + Data$Total
```

Model fitting

```
Trials = cbind(Data$mpi100, Data$mpi90)          # Successes, Failures

model = glm(Trials ~ Latitude,
            data = Data,
            family = binomial(link="logit"))
```

Coefficients and exponentiated coefficients

```
summary(model)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -7.64686    0.92487  -8.268  <2e-16 ***
Latitude      0.17864    0.02104   8.490  <2e-16 ***
```

```

confint(model)
              2.5 %      97.5 %
(Intercept) -9.5003746 -5.8702453
Latitude     0.1382141  0.2208032

exp(model$coefficients)      # exponentiated coefficients

(Intercept)      Latitude
0.0004775391 1.1955899446

exp(confint(model))          # 95% CI for exponentiated coefficients

              2.5 %      97.5 %
(Intercept) 7.482379e-05 0.002822181
Latitude    1.148221e+00 1.247077992

```

Analysis of variance for individual terms

```

library(car)

Anova(model, type="II", test="wald")

Analysis of Deviance Table (Type II tests)

Response: Trials
      Df  Chisq Pr(>Chisq)
Latitude  1 72.076 < 2.2e-16 ***

```

Pseudo-R-squared

```

library(rcompanion)

nagelkerke(model)

$Models

Model: "glm, Trials ~ Latitude, binomial(link = \"logit\"), Data"
Null:  "glm, Trials ~ 1, binomial(link = \"logit\"), Data"

$Pseudo.R.squared.for.model.vs.null
              Pseudo.R.squared
McFadden          0.425248
Cox and Snell (ML) 0.999970
Nagelkerke (Cragg and Uhler) 0.999970

```

Overall p-value for model

```

anova(model,
       update(model, ~1), # update here produces null model for comparison
       test="Chisq")

```

Analysis of Deviance Table

Model 1: Trials ~ Latitude

Model 2: Trials ~ 1

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
1	6	70.333			
2	7	153.633	-1	-83.301	< 2.2e-16 ***

library(lmtest)

lrtest(model)

Likelihood ratio test

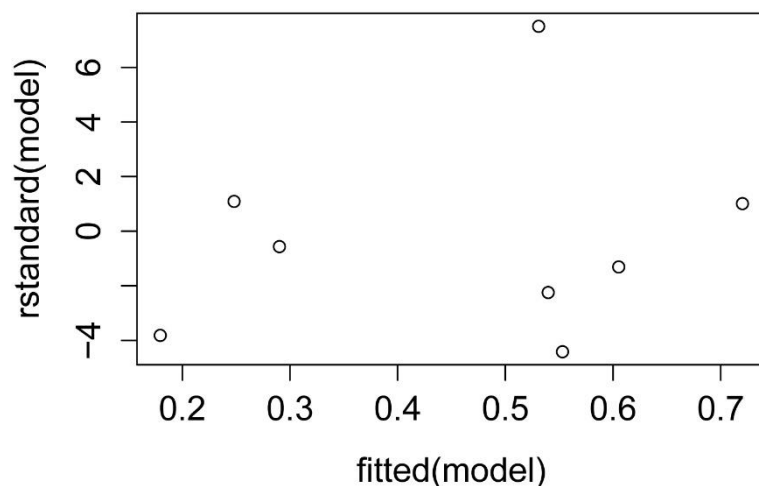
Model 1: Trials ~ Latitude

Model 2: Trials ~ 1

	#Df	LogLik	Df	Chisq	Pr(>Chisq)
1	2	-56.293			
2	1	-97.944	-1	83.301	< 2.2e-16 ***

Plot of standardized residuals

```
plot(fitted(model),
     rstandard(model))
```



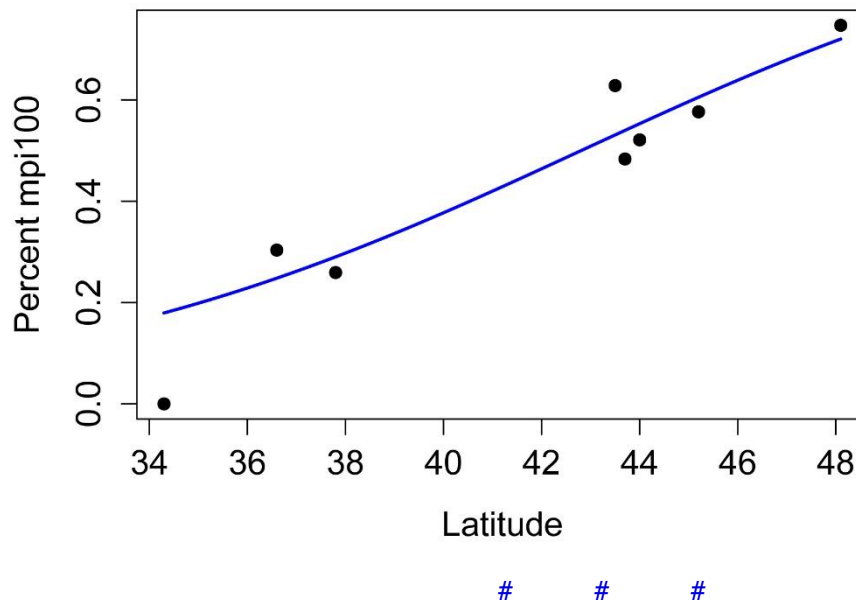
A plot of standardized residuals vs. predicted values. The residuals should be unbiased and homoscedastic. For an illustration of these properties, see this diagram by Steve Jost at DePaul University: condor.depaul.edu/sjost/it223/documents/resid-plots.gif.

```
### additional model checking plots with: plot(model)
```

Plotting the model

```
plot(Percent ~ Latitude,
     data = Data,
     xlab="Latitude",
     ylab="Percent mpi100",
     pch=19)

curve(predict(model,data.frame(Latitude=x),type="response"),
      lty=1, lwd=2, col="blue",
      add=TRUE)
```



Logistic regression example

```
### -----
### Logistic regression, favorite insect example, p. 248
### -----

Input = ("
Height Insect
62 beetle
66 other
61 beetle
67 other
62 other
76 other
66 other
70 beetle
67 other
66 other
70 other
70 other
77 beetle
76 other
72 beetle
76 beetle")
```

```

72      other
70      other
65      other
63      other
63      other
70      other
72      other
70      beetle
74      other
")

```

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

Model fitting

```

model = glm(Insect ~ Height,
            data=Data,
            family = binomial(link="logit"))

```

Coefficients and exponentiated coefficients

```
summary(model)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	4.41379	6.66190	0.663	0.508
Height	-0.05016	0.09577	-0.524	0.600

```
confint(model)
```

	2.5 %	97.5 %
(Intercept)	-8.4723648	18.4667731
Height	-0.2498133	0.1374819

```
exp(model$coefficients)      # exponentiated coefficients
```

	Height
(Intercept)	82.5821122
Height	0.9510757

```
exp(confint(model))          # 95% CI for exponentiated coefficients
```

	2.5 %	97.5 %
(Intercept)	0.0002091697	1.047171e+08
Height	0.7789461738	1.147381e+0

Analysis of variance for individual terms

```
library(car)
```

```
Anova(model, type="II", test="wald")
```

Analysis of Deviance Table (Type II tests)

```

Response: Insect
      Df  Chisq Pr(>Chisq)
Height   1 0.2743    0.6004
Residuals 23

```

Pseudo-R-squared

```
library(rcompanion)
```

```
nagelkerke(model)
```

```

$Pseudo.R.squared.for.model.vs.null
                                Pseudo.R.squared
McFadden                        0.00936978
Cox and Snell (ML)              0.01105020
Nagelkerke (Cragg and Uhler)    0.01591030

```

Overall p-value for model

```

anova(model,
      update(model, ~1), # update here produces null model for comparison
      test="Chisq")

```

Analysis of Deviance Table

```

Model 1: Insect ~ Height
Model 2: Insect ~ 1
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      23      29.370
2      24      29.648 -1 -0.27779  0.5982

```

```
library(lmtest)
```

```
lrtest(model)
```

Likelihood ratio test

```

Model 1: Insect ~ Height
Model 2: Insect ~ 1
  #Df  LogLik Df  Chisq Pr(>Chisq)
1    2 -14.685
2    1 -14.824 -1 0.2778    0.5982

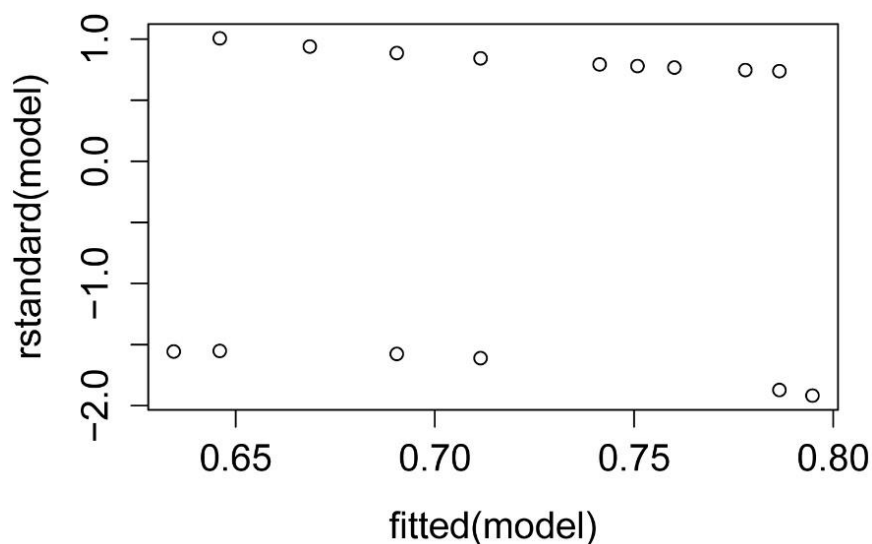
```

Plot of standardized residuals

```

plot(fitted(model),
     rstandard(model))

```



Plotting the model

```
### Convert Insect to a numeric variable, levels 0 and 1
```

```
Data$Insect.num=as.numeric(Data$Insect)-1
```

```
library(FSA)
```

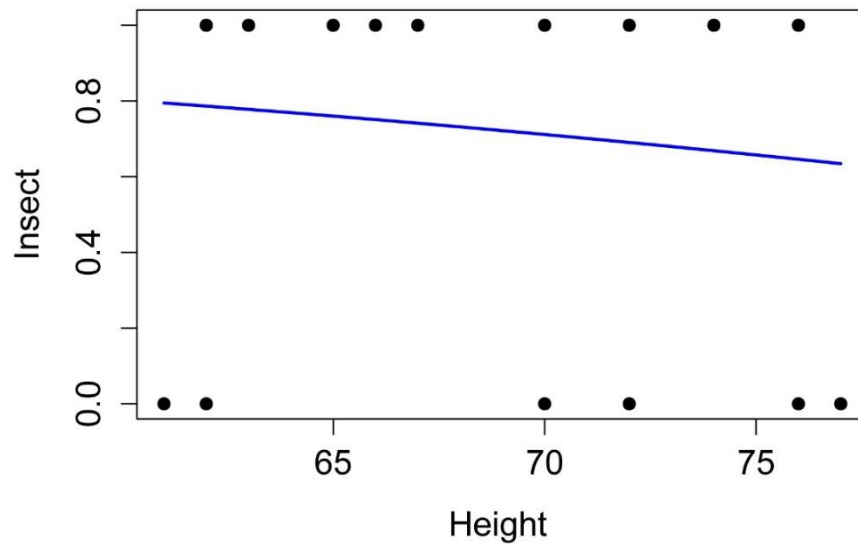
```
headtail(Data)
```

	Height	Insect	Insect.num
1	62	beetle	0
2	66	other	1
3	61	beetle	0
23	72	other	1
24	70	beetle	0
25	74	other	1

```
### Plot
```

```
plot(Insect.num ~ Height,
     data = Data,
     xlab="Height",
     ylab="Insect",
     pch=19)
```

```
curve(predict(model,data.frame(Height=x),type="response"),
      lty=1, lwd=2, col="blue",
      add=TRUE)
```

```
### Convert Insect to a logical variable, levels TRUE and FALSE
```

```
Data$Insect.log=(Data$Insect=="other")
```

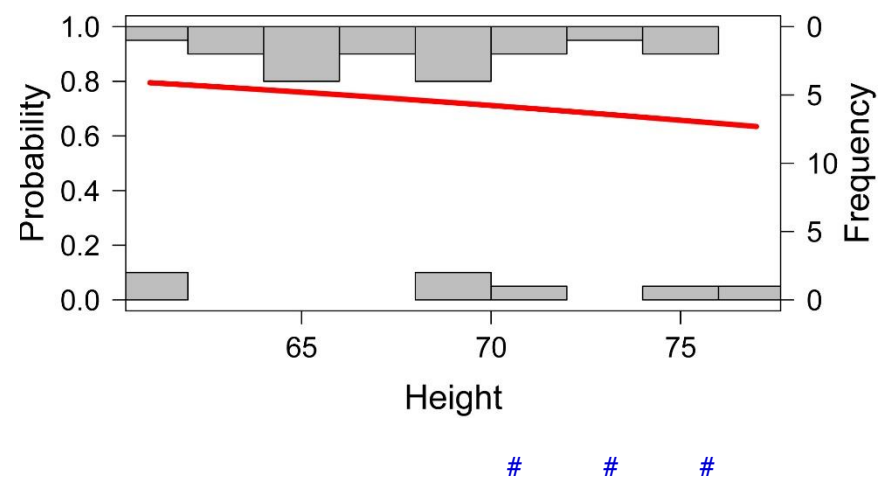
```
library(FSA)
```

```
headtail(Data)
```

	Height	Insect	Insect.num	Insect.log
1	62	beetle	0	FALSE
2	66	other	1	TRUE
3	61	beetle	0	FALSE
23	72	other	1	TRUE
24	70	beetle	0	FALSE
25	74	other	1	TRUE

```
library(popbio)
```

```
logi.hist.plot(Data$Height,
               Data$Insect.log,
               boxp=FALSE,
               type="hist",
               col="gray",
               xlabel="Height")
```



Logistic regression example with significant model and abbreviated code

```
### -----
### Logistic regression, hypothetical example
### Abbreviated code and description
### -----

Input = ("
Continuous  Factor
62          A
63          A
64          A
65          A
66          A
67          A
68          A
69          A
70          A
71          A
72          A
73          A
74          A
75          A
72.5        B
73.5        B
74.5        B
75          B
76          B
77          B
78          B
79          B
80          B
81          B
82          B
83          B
84          B
85          B
86          B
```

```

")

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

model = glm(Factor ~ Continuous,
             data=Data,
             family = binomial(link="logit"))

summary(model)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -66.4981    32.3787  -2.054   0.0400 *
Continuous    0.9027     0.4389   2.056   0.0397 *

library(car)

Anova(model, type="II", test="wald")

Analysis of Deviance Table (Type II tests)

Response: Factor
              Df Chisq Pr(>Chisq)
Continuous    1  4.229   0.03974 *
Residuals    27

library(rcompanion)

nagelkerke(model)

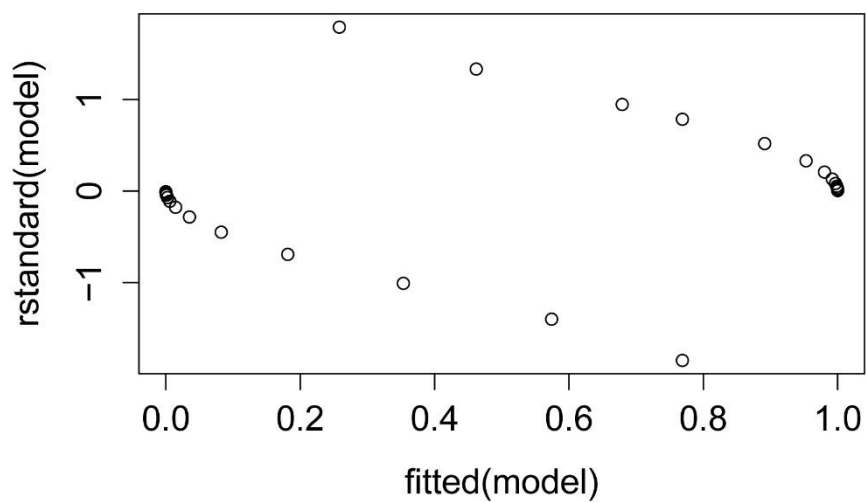
              Pseudo.R.squared
McFadden              0.697579
Cox and Snell (ML)      0.619482
Nagelkerke (Cragg and Uhler) 0.826303

anova(model,
        update(model, ~1),
        test="Chisq")

      Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1         27      12.148
2         28      40.168 -1    -28.02  1.2e-07 ***

plot(fitted(model),
     rstandard(model))

```



```
### Convert Factor to a numeric variable, levels 0 and 1
```

```
Data$Factor.num=as.numeric(Data$Factor)-1
```

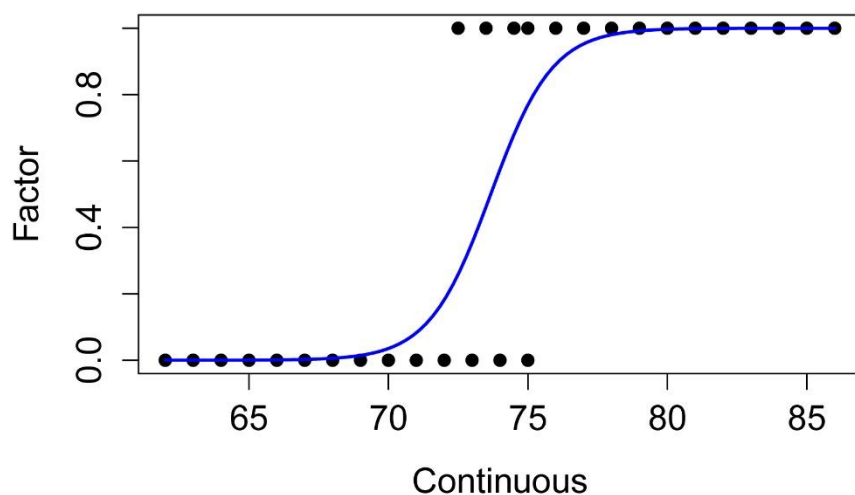
```
library(FSA)
```

```
headtail(Data)
```

	Continuous	Factor	Factor.num
1	62	A	0
2	63	A	0
3	64	A	0
27	84	B	1
28	85	B	1
29	86	B	1

```
plot(Factor.num ~ Continuous,
     data = Data,
     xlab="Continuous",
     ylab="Factor",
     pch=19)
```

```
curve(predict(model,data.frame(Continuous=x),type="response"),
      lty=1, lwd=2, col="blue",
      add=TRUE)
```



```
### Convert Factor to a logical variable, levels TRUE and FALSE
```

```
Data$Factor.log=(Data$Factor=="B")
```

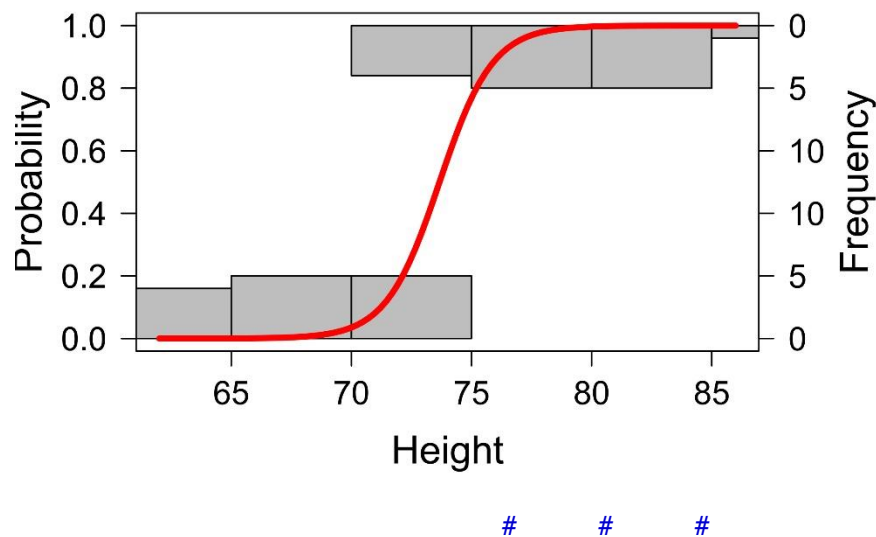
```
library(FSA)
```

```
headtail(Data)
```

	Continuous	Factor	Factor.num	Factor.log
1	62	A	0	FALSE
2	63	A	0	FALSE
3	64	A	0	FALSE
27	84	B	1	TRUE
28	85	B	1	TRUE
29	86	B	1	TRUE

```
library(popbio)
```

```
logi.hist.plot(Data$Continuous,
               Data$Factor.log,
               boxp=FALSE,
               type="hist",
               col="gray",
               xlabel="Height")
```



Power analysis

See the *Handbook* for information on this topic.

Multiple Logistic Regression

When to use it

The bird example is shown in the “How to do multiple logistic regression” section.

Null hypothesis

How it works

Selecting variables in multiple logistic regression

See the *Handbook* for information on these topics.

Assumptions

See the *Handbook* and the “How to do multiple logistic regression” section below for information on this topic.

Example

Graphing the results

Similar tests

See the *Handbook* for information on these topics.

How to do multiple logistic regression

Multiple logistic regression can be determined by a stepwise procedure using the *step* function. This function selects models to minimize AIC, not according to p-values as does the SAS example in the *Handbook*. Note, also, that in this example the *step* function found a different model than did the procedure in the *Handbook*.

It is often advised to not blindly follow a stepwise procedure, but to also compare competing models using fit statistics (AIC, AICc, BIC), or to build a model from available variables that are biologically or scientifically sensible.

Multiple correlation is one tool for investigating the relationship among potential independent variables. For example, if two independent variables are correlated to one another, likely both won't be needed in a final model, but there may be reasons why you would choose one variable over the other.

Multiple correlation

```
#### -----
#### Multiple logistic regression, bird example, p. 254–256
#### -----

#### When using read.table, the column headings need to be on the
#### same line. If the headings will spill over to the next line,
#### be sure to not put an enter or return at the end of the top
#### line. The same holds for each line of data.

Input = ("
Species Status Length Mass Range Migr Insect Diet Clutch Broods Wood Upland Water Release Indiv
Cyg_olor 1 1520 9600 1.21 1 12 2 6 1 0 0 1 6 29
Cyg_atra 1 1250 5000 0.56 1 0 1 6 1 0 0 1 10 85
Cer_nova 1 870 3360 0.07 1 0 1 4 1 0 0 1 3 8
Ans_caer 0 720 2517 1.1 3 12 2 3.8 1 0 0 1 1 10
Ans_anse 0 820 3170 3.45 3 0 1 5.9 1 0 0 1 2 7
Bra_cana 1 770 4390 2.96 2 0 1 5.9 1 0 0 1 10 60
Bra_sand 0 50 1930 0.01 1 0 1 4 2 0 0 0 1 2
Alo_aegy 0 680 2040 2.71 1 NA 2 8.5 1 0 0 1 1 8
Ana_plat 1 570 1020 9.01 2 6 2 12.6 1 0 0 1 17 1539
Ana_acut 0 580 910 7.9 3 6 2 8.3 1 0 0 1 3 102
Ana_pene 0 480 590 4.33 3 0 1 8.7 1 0 0 1 5 32
Aix_spon 0 470 539 1.04 3 12 2 13.5 2 1 0 1 5 10
Ayt_feri 0 450 940 2.17 3 12 2 9.5 1 0 0 1 3 9
Ayt_fuli 0 435 684 4.81 3 12 2 10.1 1 0 0 1 2 5
Ore_pict 0 275 230 0.31 1 3 1 9.5 1 1 1 0 9 398
Lop_cali 1 256 162 0.24 1 3 1 14.2 2 0 0 0 15 1420
Col_virg 1 230 170 0.77 1 3 1 13.7 1 0 0 0 17 1156
Ale_grae 1 330 501 2.23 1 3 1 15.5 1 0 1 0 15 362
Ale_rufa 0 330 439 0.22 1 3 2 11.2 2 0 0 0 2 20
Per_perd 0 300 386 2.4 1 3 1 14.6 1 0 1 0 24 676
Cot_pect 0 182 95 0.33 3 NA 2 7.5 1 0 0 0 3 NA
Cot_aust 1 180 95 0.69 2 12 2 11 1 0 0 1 11 601
Lop_nyct 0 800 1150 0.28 1 12 2 5 1 1 1 0 4 6
Pha_colc 1 710 850 1.25 1 12 2 11.8 1 1 0 0 27 244
Syr_reev 0 750 949 0.2 1 12 2 9.5 1 1 1 0 2 9
Tet_tetr 0 470 900 4.17 1 3 1 7.9 1 1 1 0 2 13
Lag_lago 0 390 517 7.29 1 0 1 7.5 1 1 1 0 2 4
Ped_phas 0 440 815 1.83 1 3 1 12.3 1 1 0 0 1 22
Tym_cupi 0 435 770 0.26 1 4 1 12 1 0 0 0 3 57
Van_vane 0 300 226 3.93 2 12 3 3.8 1 0 0 0 8 124
Plu_squa 0 285 318 1.67 3 12 3 4 1 0 0 1 2 3
Pte_alch 0 350 225 1.21 2 0 1 2.5 2 0 0 0 1 8
Pha_chal 0 320 350 0.6 1 12 2 2 2 1 0 0 8 42
Ocy_loph 0 330 205 0.76 1 0 1 2 7 1 0 1 4 23
Leu_mela 0 372 NA 0.07 1 12 2 2 1 1 0 0 6 34
Ath_noct 1 220 176 4.84 1 12 3 3.6 1 1 0 0 7 221
Tyt_alba 0 340 298 8.9 2 0 3 5.7 2 1 0 0 1 7
Dac_nova 1 460 382 0.34 1 12 3 2 1 1 0 0 7 21

```

```

Lul_arbo 0      150 32.1 1.78 2      4      2      3.9 2      1      0      0      1      5
Ala_arve 1      185 38.9 5.19 2     12      2      3.7 3      0      0      0     11     391
Pru_modu 1      145 20.5 1.95 2     12      2      3.4 2      1      0      0     14     245
Eri_rebe 0      140 15.8 2.31 2     12      2      5    2      1      0      0     11     123
Lus_mega 0      161 19.4 1.88 3     12      2      4.7 2      1      0      0      4      7
Tur_meru 1      255 82.6 3.3  2     12      2      3.8 3      1      0      0     16     596
Tur_phil 1      230 67.3 4.84 2     12      2      4.7 2      1      0      0     12     343
Syl_comm 0      140 12.8 3.39 3     12      2      4.6 2      1      0      0      1      2
Syl_atri 0      142 17.5 2.43 2      5      2      4.6 1      1      0      0      1      5
Man_mela 0      180 NA   0.04 1     12      3      1.9 5      1      0      0      1      2
Man_mela 0      265  59 0.25 1     12      2      2.6 NA   1      0      0      1     80
Gra_cyan 0      275 128 0.83 1     12      3      3    2      1      0      1      1     NA
Gym_tibi 1      400 380 0.82 1     12      3      4    1      1      0      0     15     448
Cor_mone 0      335 203 3.4  2     12      2      4.5 1      1      0      0      2      3
Cor_frug 1      400 425 3.73 1     12      2      3.6 1      1      0      0     10     182
Stu_vulg 1      222 79.8 3.33 2      6      2      4.8 2      1      0      0     14     653
Acr_tris 1      230 111.3 0.56 1     12      2      3.7 1      1      0      0      5      88
Pas_dome 1      149 28.8 6.5  1      6      2      3.9 3      1      0      0     12     416
Pas_mont 0      133  22 6.8  1      6      2      4.7 3      1      0      0      3      14
Aeg_temp 0      120 NA   0.17 1      6      2      4.7 3      1      0      0      3      14
Emb_gutt 0      120  19 0.15 1      4      1      5    3      0      0      0      4     112
Poe_gutt 0      100 12.4 0.75 1      4      1      4.7 3      0      0      0      1      12
Lon_punc 0      110 13.5 1.06 1      0      1      5    3      0      0      0      1      8
Lon_cast 0      100 NA   0.13 1      4      1      5    NA   0      0      1      4     45
Pad_oryz 0      160 NA   0.09 1      0      1      5    NA   0      0      0      2      6
Fri_coel 1      160 23.5 2.61 2     12      2      4.9 2      1      0      0     17     449
Fri_mont 0      146 21.4 3.09 3     10      2      6    NA   1      0      0      7     121
Car_chlo 1      147 29   2.09 2      7      2      4.8 2      1      0      0      6      65
Car_spin 0      117 12   2.09 3      3      1      4    2      1      0      0      3      54
Car_card 1      120 15.5 2.85 2      4      1      4.4 3      1      0      0     14     626
Aca_flam 1      115 11.5 5.54 2      6      1      5    2      1      0      0     10     607
Aca_flavi 0      133 17   1.67 2      0      1      5    3      0      1      0      3      61
Aca_cann 0      136 18.5 2.52 2      6      1      4.7 2      1      0      0     12     209
Pyr_pyrr 0      142 23.5 3.57 1      4      1      4    3      1      0      0      2     NA
Emb_citr 1      160 28.2 4.11 2      8      2      3.3 3      1      0      0     14     656
Emb_hort 0      163 21.6 2.75 3     12      2      5    1      0      0      0      1      6
Emb_cirl 1      160 23.6 0.62 1     12      2      3.5 2      1      0      0      3      29
Emb_scho 0      150 20.7 5.42 1     12      2      5.1 2      0      0      1      2      9
Pir_rubr 0      170 31   0.55 3     12      2      4    NA   1      0      0      1      2
Age_phoe 0      210 36.9 2    2      8      2      3.7 1      0      0      1      1      2
Stu_negl 0      225 106.5 1.2 2     12      2      4.8 2      0      0      0      1      2
")

```

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

Create a data frame of numeric variables

```
### Select only those variables that are numeric or can be made numeric
```

```
library(dplyr)
```

```
Data.num =
  select(Data,
    Status,
    Length,
    Mass,
    Range,
    Migr,
    Insect,
    Diet,
    Clutch,
    Broods,
    wood,
```



```
Upland,
Water,
Release,
Indiv)
```

```
### Covert integer variables to numeric variables
```

```
Data.num$Status = as.numeric(Data.num$Status)
Data.num$Length = as.numeric(Data.num$Length)
Data.num$Migr   = as.numeric(Data.num$Migr)
Data.num$Insect = as.numeric(Data.num$Insect)
Data.num$Diet   = as.numeric(Data.num$Diet)
Data.num$Broods = as.numeric(Data.num$Broods)
Data.num$Wood   = as.numeric(Data.num$Wood)
Data.num$Upland = as.numeric(Data.num$Upland)
Data.num$Water  = as.numeric(Data.num$Water)
Data.num$Release = as.numeric(Data.num$Release)
Data.num$Indiv  = as.numeric(Data.num$Indiv)
```

```
### Examine the new data frame
```

```
library(FSA)
```

```
headtail(Data.num)
```

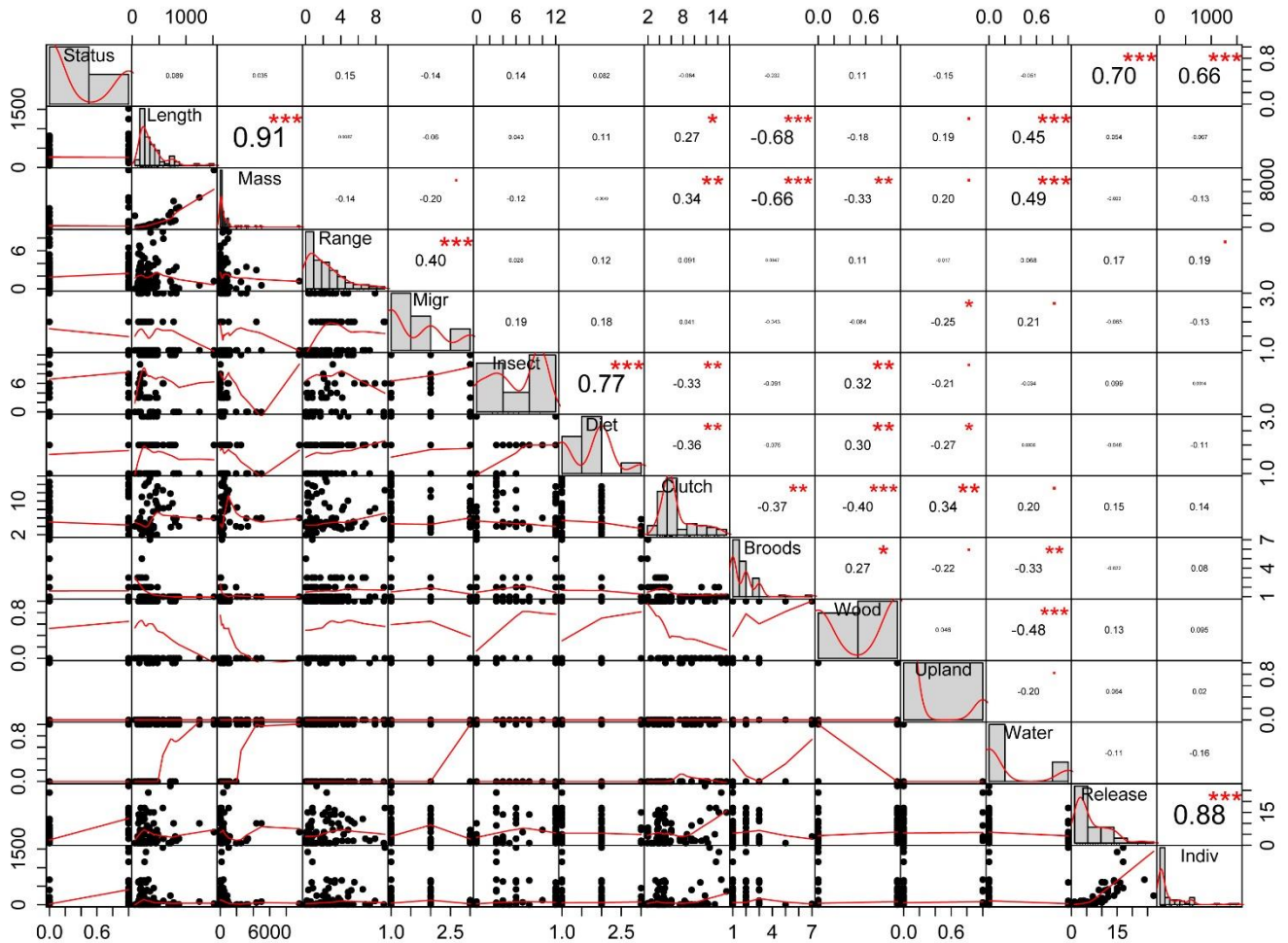
	Status	Length	Mass	Range	Migr	Insect	Diet	Clutch	Broods	wood	Upland	Water	Release	Indiv
1	1	1520	9600.0	1.21	1	12	2	6.0	1	0	0	1	6	29
2	1	1250	5000.0	0.56	1	0	1	6.0	1	0	0	1	10	85
3	1	870	3360.0	0.07	1	0	1	4.0	1	0	0	1	3	8
77	0	170	31.0	0.55	3	12	2	4.0	NA	1	0	0	1	2
78	0	210	36.9	2.00	2	8	2	3.7	1	0	0	1	1	2
79	0	225	106.5	1.20	2	12	2	4.8	2	0	0	0	1	2

Examining correlations among variables

```
### Note I used Spearman correlations here
```

```
library(PerformanceAnalytics)
```

```
chart.Correlation(Data.num,
  method="spearman",
  histogram=TRUE,
  pch=16)
```



```
library(psych)
```

```
corr.test(Data.num,
  use = "pairwise",
  method="spearman",
  adjust="none",      # Can adjust p-values; see ?p.adjust for options
  alpha=.05)
```

```
# # #
```

Multiple logistic regression example

In this example, the data contain missing values. In SAS, missing values are indicated with a period, whereas in R missing values are indicated with *NA*. SAS will often deal with missing values seamlessly. While this makes things easier for the user, it may not ensure that the user understands what is being done with these missing values. In some cases, R requires that user be explicit with how missing values are handled. One method to handle missing values in a multiple regression would be to remove all observations from the data set that have any missing values. This is what we will do prior to the stepwise procedure, creating a data frame called *Data.omit*. However, when we create our final model, we want to exclude only those observations that have missing values in the variables that are actually included in that final

model. For testing the overall p-value of the final model, plotting the final model, or using the *glm.compare* function, we will create a data frame called *Data.final* with only those observations excluded.

There are some cautions about using the *step* procedure with certain glm fits, though models in the binomial and poisson families should be okay. See *?stats::step* for more information.

```
### -----
### Multiple logistic regression, bird example, p. 254-256
### -----

Input = ("
Species Status Length Mass Range Migr Insect Diet Clutch Broods wood Upland Water Release Indiv
Cyg_olor 1 1520 9600 1.21 1 12 2 6 1 0 0 1 6 29
Cyg_atra 1 1250 5000 0.56 1 0 1 6 1 0 0 1 10 85
Cer_nova 1 870 3360 0.07 1 0 1 4 1 0 0 1 3 8
Ans_caer 0 720 2517 1.1 3 12 2 3.8 1 0 0 1 1 10
Ans_anse 0 820 3170 3.45 3 0 1 5.9 1 0 0 1 2 7
Bra_cana 1 770 4390 2.96 2 0 1 5.9 1 0 0 1 10 60
Bra_sand 0 50 1930 0.01 1 0 1 4 2 0 0 0 1 2
Alo_aegy 0 680 2040 2.71 1 NA 2 8.5 1 0 0 1 1 8
Ana_plat 1 570 1020 9.01 2 6 2 12.6 1 0 0 1 17 1539
Ana_acut 0 580 910 7.9 3 6 2 8.3 1 0 0 1 3 102
Ana_pene 0 480 590 4.33 3 0 1 8.7 1 0 0 1 5 32
Aix_spon 0 470 539 1.04 3 12 2 13.5 2 1 0 1 5 10
Ayt_feri 0 450 940 2.17 3 12 2 9.5 1 0 0 1 3 9
Ayt_fuli 0 435 684 4.81 3 12 2 10.1 1 0 0 1 2 5
Ore_pict 0 275 230 0.31 1 3 1 9.5 1 1 1 0 9 398
Lop_cali 1 256 162 0.24 1 3 1 14.2 2 0 0 0 15 1420
Col_virg 1 230 170 0.77 1 3 1 13.7 1 0 0 0 17 1156
Ale_grae 1 330 501 2.23 1 3 1 15.5 1 0 1 0 15 362
Ale_rufa 0 330 439 0.22 1 3 2 11.2 2 0 0 0 2 20
Per_perd 0 300 386 2.4 1 3 1 14.6 1 0 1 0 24 676
Cot_pect 0 182 95 0.33 3 NA 2 7.5 1 0 0 0 3 NA
Cot_aust 1 180 95 0.69 2 12 2 11 1 0 0 1 11 601
Lop_nyct 0 800 1150 0.28 1 12 2 5 1 1 1 0 4 6
Pha_colc 1 710 850 1.25 1 12 2 11.8 1 1 0 0 27 244
Syr_reev 0 750 949 0.2 1 12 2 9.5 1 1 1 0 2 9
Tet_tetr 0 470 900 4.17 1 3 1 7.9 1 1 1 0 2 13
Lag_lago 0 390 517 7.29 1 0 1 7.5 1 1 1 0 2 4
Ped_phas 0 440 815 1.83 1 3 1 12.3 1 1 0 0 1 22
Tym_cupi 0 435 770 0.26 1 4 1 12 1 0 0 0 3 57
Van_vane 0 300 226 3.93 2 12 3 3.8 1 0 0 0 8 124
Plu_squa 0 285 318 1.67 3 12 3 4 1 0 0 1 2 3
Pte_alch 0 350 225 1.21 2 0 1 2.5 2 0 0 0 1 8
Pha_chal 0 320 350 0.6 1 12 2 2 2 1 0 0 8 42
Ocy_loph 0 330 205 0.76 1 0 1 2 7 1 0 1 4 23
Leu_mela 0 372 NA 0.07 1 12 2 2 1 1 0 0 6 34
Ath_noct 1 220 176 4.84 1 12 3 3.6 1 1 0 0 7 221
Tyt_alba 0 340 298 8.9 2 0 3 5.7 2 1 0 0 1 7
Dac_nova 1 460 382 0.34 1 12 3 2 1 1 0 0 7 21
Lul_arbo 0 150 32.1 1.78 2 4 2 3.9 2 1 0 0 1 5
Ala_arve 1 185 38.9 5.19 2 12 2 3.7 3 0 0 0 11 391
Pru_modu 1 145 20.5 1.95 2 12 2 3.4 2 1 0 0 14 245
Eri_rebe 0 140 15.8 2.31 2 12 2 5 2 1 0 0 11 123
Lus_mega 0 161 19.4 1.88 3 12 2 4.7 2 1 0 0 4 7
Tur_meru 1 255 82.6 3.3 2 12 2 3.8 3 1 0 0 16 596
Tur_phil 1 230 67.3 4.84 2 12 2 4.7 2 1 0 0 12 343
Syl_comm 0 140 12.8 3.39 3 12 2 4.6 2 1 0 0 1 2
Syl_atri 0 142 17.5 2.43 2 5 2 4.6 1 1 0 0 1 5
Man_mela 0 180 NA 0.04 1 12 3 1.9 5 1 0 0 1 2
Man_mela 0 265 59 0.25 1 12 2 2.6 NA 1 0 0 1 80
Gra_cyan 0 275 128 0.83 1 12 3 3 2 1 0 1 1 NA
Gym_tibi 1 400 380 0.82 1 12 3 4 1 1 0 0 15 448
Cor_mone 0 335 203 3.4 2 12 2 4.5 1 1 0 0 2 3
```

```

Cor_frug 1      400    425 3.73 1    12    2    3.6 1    1    0    0    10    182
Stu_vulg 1      222   79.8 3.33 2     6    2    4.8 2    1    0    0    14    653
Acr_tris 1      230  111.3 0.56 1    12    2    3.7 1    1    0    0     5     88
Pas_dome 1      149   28.8 6.5  1     6    2    3.9 3    1    0    0    12    416
Pas_mont 0      133    22 6.8  1     6    2    4.7 3    1    0    0     3     14
Aeg_temp 0      120   NA  0.17 1     6    2    4.7 3    1    0    0     3     14
Emb_gutt 0      120    19 0.15 1     4    1     5  3    0    0    0     4    112
Poe_gutt 0      100   12.4 0.75 1     4    1    4.7 3    0    0    0     1     12
Lon_punc 0      110   13.5 1.06 1     0    1     5  3    0    0    0     1      8
Lon_cast 0      100   NA  0.13 1     4    1     5  NA  0    0    1     4     45
Pad_oryz 0      160   NA  0.09 1     0    1     5  NA  0    0    0     2      6
Fri_coel 1      160   23.5 2.61 2    12    2    4.9 2    1    0    0    17    449
Fri_mont 0      146   21.4 3.09 3    10    2     6  NA  1    0    0     7    121
Car_chlo 1      147    29 2.09 2     7    2    4.8 2    1    0    0     6     65
Car_spin 0      117    12 2.09 3     3    1     4  2    1    0    0     3     54
Car_card 1      120   15.5 2.85 2     4    1    4.4 3    1    0    0    14    626
Aca_flam 1      115   11.5 5.54 2     6    1     5  2    1    0    0    10    607
Aca_flavi 0      133    17 1.67 2     0    1     5  3    0    1    0     3     61
Aca_cann 0      136   18.5 2.52 2     6    1    4.7 2    1    0    0    12    209
Pyr_pyrr 0      142   23.5 3.57 1     4    1     4  3    1    0    0     2     NA
Emb_citr 1      160   28.2 4.11 2     8    2    3.3 3    1    0    0    14    656
Emb_hort 0      163   21.6 2.75 3    12    2     5  1    0    0    0     1      6
Emb_cirl 1      160   23.6 0.62 1    12    2    3.5 2    1    0    0     3     29
Emb_scho 0      150   20.7 5.42 1    12    2    5.1 2    0    0    1     2      9
Pir_rubr 0      170    31 0.55 3    12    2     4  NA  1    0    0     1      2
Age_phoe 0      210   36.9 2     2     8    2    3.7 1    0    0    1     1      2
Stu_negl 0      225  106.5 1.2  2    12    2    4.8 2    0    0    0     1      2
")

```

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

Determining model with step procedure

```
### Create new data frame with all missing values removed (NA's)
```

```
Data.omit = na.omit(Data)
```

```
### Define full and null models and do step procedure
```

```
model.null = glm(Status ~ 1,
                  data=Data.omit,
                  family = binomial(link="logit"))
```

```
model.full = glm(Status ~ Length + Mass + Range + Migr + Insect + Diet +
                  Clutch + Broods + wood + Upland + water +
                  Release + Indiv,
                  data=Data.omit,
                  family = binomial(link="logit")
)
```

```
step(model.null,
      scope = list(upper=model.full),
      direction="both",
      test="Chisq",
      data=Data
)
```

```

Start:  AIC=92.34
Status ~ 1
      Df Deviance    AIC    LRT  Pr(>Chi)
+ Release  1   56.130  60.130 34.213 4.940e-09 ***
+ Individ  1   60.692  64.692 29.651 5.172e-08 ***
+ Migr     1   85.704  89.704  4.639  0.03125 *
+ Upland   1   86.987  90.987  3.356  0.06696 .
+ Insect   1   88.231  92.231  2.112  0.14614
<none>          90.343  92.343
+ Mass     1   88.380  92.380  1.963  0.16121
+ Wood     1   88.781  92.781  1.562  0.21133
+ Diet     1   89.195  93.195  1.148  0.28394
+ Length   1   89.372  93.372  0.972  0.32430
+ Water    1   90.104  94.104  0.240  0.62448
+ Broods   1   90.223  94.223  0.120  0.72898
+ Range    1   90.255  94.255  0.088  0.76676
+ Clutch   1   90.332  94.332  0.012  0.91420
.
.
< several more steps >
.
.
Step:  AIC=42.03
Status ~ Upland + Migr + Mass + Individ + Insect + Wood
      Df Deviance    AIC    LRT  Pr(>Chi)
<none>          28.031  42.031
- Wood     1   30.710  42.710  2.679  0.101686
+ Diet     1   26.960  42.960  1.071  0.300673
+ Length   1   27.965  43.965  0.066  0.796641
+ Water    1   27.970  43.970  0.062  0.803670
+ Broods   1   27.983  43.983  0.048  0.825974
+ Clutch   1   28.005  44.005  0.027  0.870592
+ Release  1   28.009  44.009  0.022  0.881631
+ Range    1   28.031  44.031  0.000  0.999964
- Insect   1   32.369  44.369  4.338  0.037276 *
- Migr     1   35.169  47.169  7.137  0.007550 **
- Upland   1   38.302  50.302 10.270  0.001352 **
- Mass     1   43.402  55.402 15.371  8.833e-05 ***
- Individ  1   71.250  83.250 43.219  4.894e-11 ***

```

Final model

```

model.final = glm(Status ~ Upland + Migr + Mass + Individ + Insect + Wood,
                  data=Data,
                  family = binomial(link="logit"),
                  na.action(na.omit)
                  )

```

```
summary(model.final)
```

```

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.5496482   2.0827400  -1.704  0.088322 .
Upland       -4.5484289   2.0712502  -2.196  0.028093 *

```

Migr	-1.8184049	0.8325702	-2.184	0.028956	*
Mass	0.0019029	0.0007048	2.700	0.006940	**
Indiv	0.0137061	0.0038703	3.541	0.000398	***
Insect	0.2394720	0.1373456	1.744	0.081234	.
Wood	1.8134445	1.3105911	1.384	0.166455	

Analysis of variance for individual terms

```
library(car)

Anova(model.final, type="II", test="wald")
```

Pseudo-R-squared

```
library(rcompanion)

nagelkerke(model.final)

$Pseudo.R.squared.for.model.vs.null
Pseudo.R.squared
McFadden 0.700475
Cox and Snell (ML) 0.637732
Nagelkerke (Cragg and Uhler) 0.833284
```

Overall p-value for model

```
### Create data frame with variables in final model and NA's omitted
```

```
library(dplyr)
```

```
Data.final =
  select(Data,
    Status,
    Upland,
    Migr,
    Mass,
    Indiv,
    Insect,
    Wood)
```

```
Data.final = na.omit(Data.final)
```

```
### Define null models and compare to final model
```

```
model.null = glm(Status ~ 1,
  data=Data.final,
  family = binomial(link="logit")
)
```

```
anova(model.final,
```

```
model.null,
test="Chisq")
```

Analysis of Deviance Table

Model 1: Status ~ Upland + Migr + Mass + Indiv + Insect + Wood

Model 2: Status ~ 1

	Resid. Df	Resid. Dev	Df	Deviance	Pr(>Chi)
1	63	30.392			
2	69	93.351	-6	-62.959	1.125e-11 ***

```
library(lmtest)
```

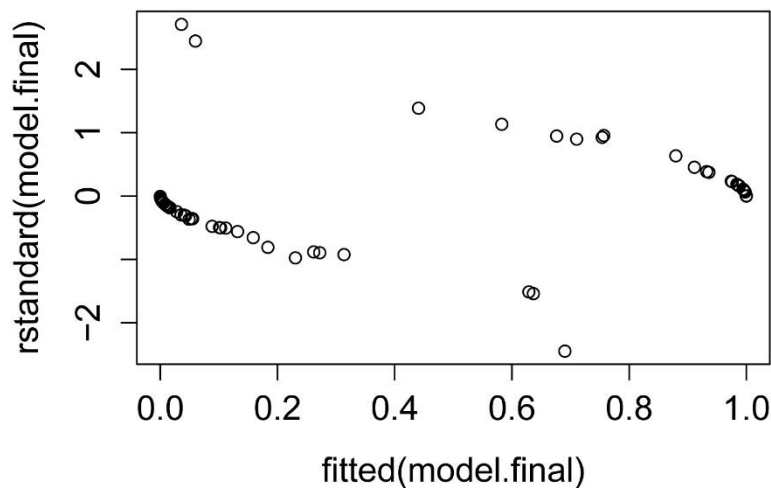
```
lrtest(model.final)
```

Likelihood ratio test

	#Df	LogLik	Df	Chisq	Pr(>Chisq)
1	7	-15.196			
2	1	-46.675	-6	62.959	1.125e-11 ***

Plot of standardized residuals

```
plot(fitted(model.final),
     rstandard(model.final))
```



Simple plot of predicted values

Create data frame with variables in final model and NA's omitted

```
library(dplyr)
```

```
Data.final =
  select(Data,
         Status,
         Upland,
```

```

Migr,
Mass,
Indiv,
Insect,
wood)

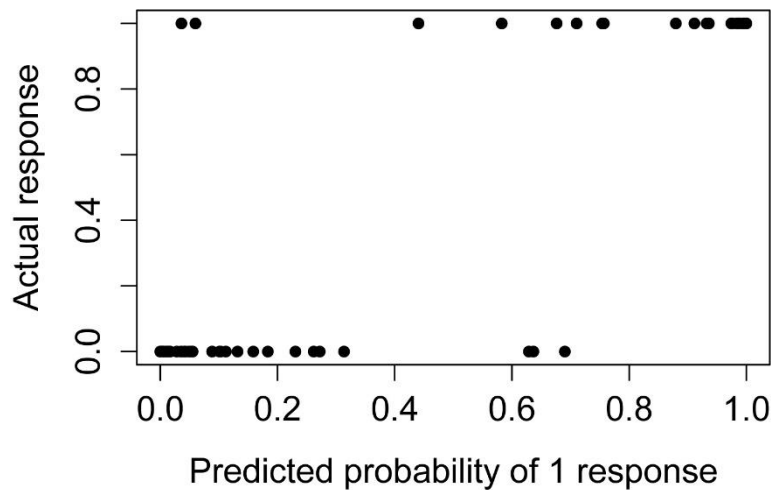
Data.final = na.omit(Data.final)

Data.final$predy = predict(model.final,
                           type="response")

### Plot

plot(Status ~ predy,
     data = Data.final,
     pch = 16,
     xlab="Predicted probability of 1 response",
     ylab="Actual response")

```



Check for overdispersion

Overdispersion is a situation where the residual deviance of the glm is large relative to the residual degrees of freedom. These values are shown in the *summary* of the model. One guideline is that if the ratio of the residual deviance to the residual degrees of freedom exceeds 1.5, then the model is overdispersed. Overdispersion indicates that the model doesn't fit the data well: the explanatory variables may not well describe the dependent variable or the model may not be specified correctly for these data. If there is overdispersion, one potential solution is to use the quasibinomial *family* option in *glm*.

```

summary(model)

Null deviance: 93.351  on 69  degrees of freedom
Residual deviance: 30.392  on 63  degrees of freedom

summary(model.final)$deviance / summary(model.final)$df.residual

```


[1] 0.482417

Alternative to assess models: using *compare.glm*

An alternative to, or a supplement to, using a stepwise procedure is comparing competing models with fit statistics. My *compare.glm* function will display AIC, AICc, BIC, and pseudo-R-squared for glm models. The models used should all be fit to the same data. That is, caution should be used if different variables in the data set contain missing values. If you don't have any preference on which fit statistic to use, I might recommend AICc, or BIC if you'd rather aim for having fewer terms in the final model.

A series of models can be compared with the standard *anova* function. Models should be nested within the previous model or the next model in the list in the *anova* function; and models should be fit to the same data. When comparing multiple regression models, a p-value to include a new term is often relaxed is 0.10 or 0.15.

In the following example, the models chosen with the stepwise procedure are used. Note that while model 9 minimizes AIC and AICc, model 8 minimizes BIC. The anova results suggest that model 8 is not a significant improvement to model 7. These results give support for selecting any of model 7, 8, or 9. Note that the SAS example in the *Handbook* selected model 4.

```
### Create data frame with just final terms and no NA's
```

```
library(dplyr)
```

```
Data.final =  
  select(Data,  
    Status,  
    Upland,  
    Migr,  
    Mass,  
    Indiv,  
    Insect,  
    wood)
```

```
Data.final = na.omit(Data.final)
```

```
### Define models to compare.
```

```
model.1=glm(Status ~ 1,  
            data=Data.omit, family=binomial())  
model.2=glm(Status ~ Release,  
            data=Data.omit, family=binomial())  
model.3=glm(Status ~ Release + Upland,  
            data=Data.omit, family=binomial())  
model.4=glm(Status ~ Release + Upland + Migr,  
            data=Data.omit, family=binomial())  
model.5=glm(Status ~ Release + Upland + Migr + Mass,  
            data=Data.omit, family=binomial())  
model.6=glm(Status ~ Release + Upland + Migr + Mass + Indiv,
```

```

      data=Data.omit, family=binomial())
model.7=glm(Status ~ Release + Upland + Migr + Mass + Indiv + Insect,
      data=Data.omit, family=binomial())
model.8=glm(Status ~ Upland + Migr + Mass + Indiv + Insect,
      data=Data.omit, family=binomial())
model.9=glm(Status ~ Upland + Migr + Mass + Indiv + Insect + Wood,
      data=Data.omit, family=binomial())

```

Use compare.glm to assess fit statistics.

```
library(rcompanion)
```

```
compareGLM(model.1, model.2, model.3, model.4, model.5, model.6,
      model.7, model.8, model.9)
```

\$Models

Formula

```

1 "Status ~ 1"
2 "Status ~ Release"
3 "Status ~ Release + Upland"
4 "Status ~ Release + Upland + Migr"
5 "Status ~ Release + Upland + Migr + Mass"
6 "Status ~ Release + Upland + Migr + Mass + Indiv"
7 "Status ~ Release + Upland + Migr + Mass + Indiv + Insect"
8 "Status ~ Upland + Migr + Mass + Indiv + Insect"
9 "Status ~ Upland + Migr + Mass + Indiv + Insect + Wood"

```

\$Fit.criteria

	Rank	Df.res	AIC	AICC	BIC	McFadden	Cox.and.Snell	Nagelkerke	p.value
1	1	66	94.34	94.53	98.75	0.0000	0.0000	0.0000	Inf
2	2	65	62.13	62.51	68.74	0.3787	0.3999	0.5401	2.538e-09
3	3	64	56.02	56.67	64.84	0.4684	0.4683	0.6325	3.232e-10
4	4	63	51.63	52.61	62.65	0.5392	0.5167	0.6979	7.363e-11
5	5	62	50.64	52.04	63.87	0.5723	0.5377	0.7263	7.672e-11
6	6	61	49.07	50.97	64.50	0.6118	0.5618	0.7588	5.434e-11
7	7	60	46.42	48.90	64.05	0.6633	0.5912	0.7985	2.177e-11
8	6	61	44.71	46.61	60.14	0.6601	0.5894	0.7961	6.885e-12
9	7	60	44.03	46.51	61.67	0.6897	0.6055	0.8178	7.148e-12

Use anova to compare each model to the previous one.

```

anova(model.1, model.2, model.3,model.4, model.5, model.6,
      model.7, model.8, model.9,
      test="Chisq")

```

Analysis of Deviance Table

```

Model 1: Status ~ 1
Model 2: Status ~ Release
Model 3: Status ~ Release + Upland
Model 4: Status ~ Release + Upland + Migr
Model 5: Status ~ Release + Upland + Migr + Mass
Model 6: Status ~ Release + Upland + Migr + Mass + Indiv

```

```
Model 7: Status ~ Release + Upland + Migr + Mass + Indiv + Insect
Model 8: Status ~ Upland + Migr + Mass + Indiv + Insect
Model 9: Status ~ Upland + Migr + Mass + Indiv + Insect + Wood
```

	Resid.	Df	Resid.	Dev	Df	Deviance	Pr(>Chi)	
1		66		90.343				
2		65		56.130	1	34.213	4.94e-09	***
3		64		48.024	1	8.106	0.004412	**
4		63		41.631	1	6.393	0.011458	*
5		62		38.643	1	2.988	0.083872	.
6		61		35.070	1	3.573	0.058721	.
7		60		30.415	1	4.655	0.030970	*
8		61		30.710	-1	-0.295	0.587066	
9		60		28.031	1	2.679	0.101686	

#

Power analysis

See the *Handbook* for information on this topic.

Multiple Comparisons

The problem with multiple comparisons

See the *Handbook* for information on this topic. Also see sections of this book with the terms “multiple comparisons”, “Tukey”, “pairwise”, “post-hoc”, “p.adj”, “p.adjust”, “p.method”, or “adjust”.

Controlling the familywise error rate: Bonferroni correction

Example is shown below in the “How to do the tests” section

Controlling the false discovery rate: Benjamini–Hochberg procedure

Example is shown below in the “How to do the tests” section

Assumption

When not to correct for multiple comparisons

See the *Handbook* for information on these topics.

How to do the tests

R has built in methods to adjust a series of p-values either to control the family-wise error rate or to control the false discovery rate.

The methods Holm, Hochberg, Hommel, and Bonferroni control the family-wise error rate. These methods attempt to limit the probability of even one false discovery (a type I error, incorrectly rejecting the null hypothesis when there is no real effect), and so are all relatively strong (conservative).

The methods BH (Benjamini–Hochberg, which is the same as FDR in R) and BY control the false discovery rate. These methods attempt to control the expected proportion of false discoveries.

For more information on these methods, see `?p.adjust` or other resources.

Note that these methods require only the p-values to adjust and the number of p-values that are being compared. This is different from methods such as Tukey or Dunnett that require also the variability of the underlying data. Tukey and Dunnett are considered familywise error rate methods.

To get some sense of how conservative these different adjustments are, see the two plots below in this chapter.

There is no definitive advice on which p-value adjustment measure to use. In general, you should choose a method which will be familiar to your audience or in your field of study. In addition, there may be some logic which allows you to choose how you balance the probability of making a type I error relative to a type II error. For example, in a preliminary study, you might want to

keep as many significant values as possible to not exclude potentially significant factors from future studies. On the other hand, in a medical study where people's lives are at stake and very expensive treatments are being considered, you would want to have a very high level of certainty before concluding that one treatment is better than another.

Multiple comparisons example with 25 p-values

```
#### -----
### Multiple comparisons example, p. 262–263
### -----

Input = ("
  Food           Raw.p
  Blue_fish      .34
  Bread          .594
  Butter         .212
  Carbohydrates  .384
  Cereals_and_pasta .074
  Dairy_products .94
  Eggs           .275
  Fats           .696
  Fruit          .269
  Legumes        .341
  Nuts           .06
  Olive_oil      .008
  Potatoes       .569
  Processed_meat .986
  Proteins       .042
  Red_meat       .251
  Semi-skimmed_milk .942
  Skimmed_milk   .222
  Sweets         .762
  Total_calories .001
  Total_meat     .975
  Vegetables     .216
  White_fish     .205
  White_meat     .041
  Whole_milk     .039
")

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

### Order data by p-value

Data = Data[order(Data$Raw.p),]

### Check if data is ordered the way we intended

library(FSA)

headtail(Data)
```

```

      Food Raw.p
20 Total_calories 0.001
12 olive_oil 0.008
25 whole_milk 0.039
17 Semi-skimmed_milk 0.942
21 Total_meat 0.975
14 Processed_meat 0.986

```

Perform p-value adjustments and add to data frame

```
Data$Bonferroni =
  p.adjust(Data$Raw.p,
    method = "bonferroni")
```

```
Data$BH =
  p.adjust(Data$Raw.p,
    method = "BH")
```

```
Data$Holm =
  p.adjust(Data$ Raw.p,
    method = "holm")
```

```
Data$Hochberg =
  p.adjust(Data$ Raw.p,
    method = "hochberg")
```

```
Data$Hommel =
  p.adjust(Data$ Raw.p,
    method = "hommel")
```

```
Data$BY =
  p.adjust(Data$ Raw.p,
    method = "BY")
```

Data

	Food	Raw.p	Bonferroni	BH	Holm	Hochberg	Hommel	BY
20	Total_calories	0.001	0.025	0.0250000	0.025	0.025	0.025	0.09539895
12	olive_oil	0.008	0.200	0.1000000	0.192	0.192	0.192	0.38159582
25	whole_milk	0.039	0.975	0.2100000	0.897	0.882	0.682	0.80135122
24	white_meat	0.041	1.000	0.2100000	0.902	0.882	0.697	0.80135122
15	Proteins	0.042	1.000	0.2100000	0.902	0.882	0.714	0.80135122
11	Nuts	0.060	1.000	0.2500000	1.000	0.986	0.840	0.95398954
5	Cereals_and_pasta	0.074	1.000	0.2642857	1.000	0.986	0.962	1.00000000
23	white_fish	0.205	1.000	0.4910714	1.000	0.986	0.986	1.00000000
3	Butter	0.212	1.000	0.4910714	1.000	0.986	0.986	1.00000000
22	Vegetables	0.216	1.000	0.4910714	1.000	0.986	0.986	1.00000000
18	Skimmed_milk	0.222	1.000	0.4910714	1.000	0.986	0.986	1.00000000
16	Red_meat	0.251	1.000	0.4910714	1.000	0.986	0.986	1.00000000
9	Fruit	0.269	1.000	0.4910714	1.000	0.986	0.986	1.00000000
7	Eggs	0.275	1.000	0.4910714	1.000	0.986	0.986	1.00000000
1	Blue_fish	0.340	1.000	0.5328125	1.000	0.986	0.986	1.00000000
10	Legumes	0.341	1.000	0.5328125	1.000	0.986	0.986	1.00000000
4	Carbohydrates	0.384	1.000	0.5647059	1.000	0.986	0.986	1.00000000
13	Potatoes	0.569	1.000	0.7815789	1.000	0.986	0.986	1.00000000
2	Bread	0.594	1.000	0.7815789	1.000	0.986	0.986	1.00000000
8	Fats	0.696	1.000	0.8700000	1.000	0.986	0.986	1.00000000

19	Sweets	0.762	1.000	0.9071429	1.000	0.986	0.986	1.00000000
6	Dairy_products	0.940	1.000	0.9860000	1.000	0.986	0.986	1.00000000
17	Semi-skimmed_milk	0.942	1.000	0.9860000	1.000	0.986	0.986	1.00000000
21	Total_meat	0.975	1.000	0.9860000	1.000	0.986	0.986	1.00000000
14	Processed_meat	0.986	1.000	0.9860000	1.000	0.986	0.986	1.00000000

Plot

```

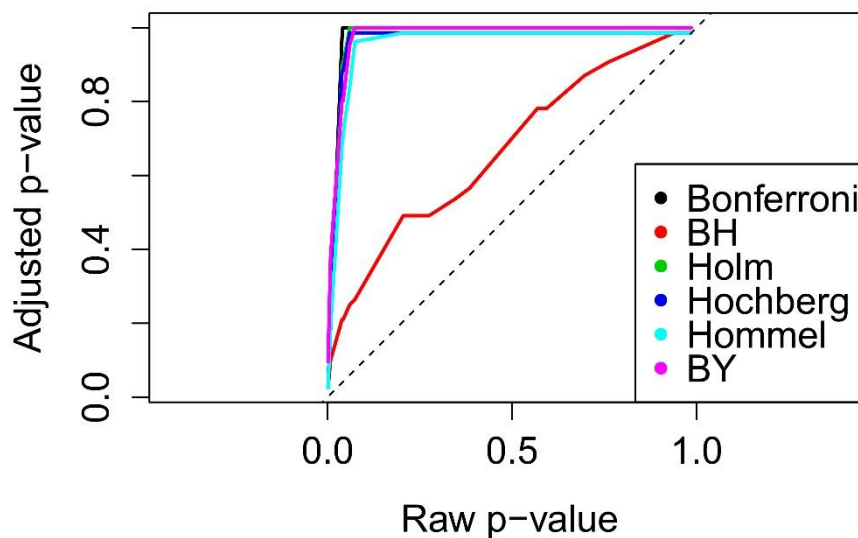
X = Data$Raw.p
Y = cbind(Data$Bonferroni,
           Data$BH,
           Data$Holm,
           Data$Hochberg,
           Data$Hommel,
           Data$BY)

matplot(X, Y,
        xlab="Raw p-value",
        ylab="Adjusted p-value",
        type="l",
        asp=1,
        col=1:6,
        lty=1,
        lwd=2)

legend('bottomright',
       legend = c("Bonferroni", "BH", "Holm", "Hochberg", "Hommel", "BY"),
       col = 1:6,
       cex = 1,
       pch = 16)

abline(0, 1,
       col=1,
       lty=2,
       lwd=1)

```



Plot of adjusted p-values vs. raw p-values for a series of 25 p-values. The dashed line represents a one-to-one line.

#

Multiple comparisons example with five p-values

```
### -----
### Multiple comparisons example, hypothetical example
### -----
```

```
Input = ("
Factor   Raw.p
A        .001
B        .01
C        .025
D        .05
E        .1
")
```

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

```
### Perform p-value adjustments and add to data frame
```

```
Data$Bonferroni =
  p.adjust(Data$Raw.p,
           method = "bonferroni")
```

```
Data$BH =
  signif(p.adjust(Data$Raw.p,
                 method = "BH"),
        4)
```

```
Data$Holm =
  p.adjust(Data$Raw.p,
           method = "holm")
```

```
Data$Hochberg =
  p.adjust(Data$Raw.p,
           method = "hochberg")
```

```
Data$Hommel =
  p.adjust(Data$Raw.p,
           method = "hommel")
```

```
Data$BY =
  signif(p.adjust(Data$Raw.p,
                 method = "BY"),
        4)
```

```
Data
```


	Factor	Raw.p	Bonferroni	BH	Holm	Hochberg	Hommel	BY
1	A	0.001	0.005	0.00500	0.005	0.005	0.005	0.01142
2	B	0.010	0.050	0.02500	0.040	0.040	0.040	0.05708
3	C	0.025	0.125	0.04167	0.075	0.075	0.075	0.09514
4	D	0.050	0.250	0.06250	0.100	0.100	0.100	0.14270
5	E	0.100	0.500	0.10000	0.100	0.100	0.100	0.22830

Plot

```

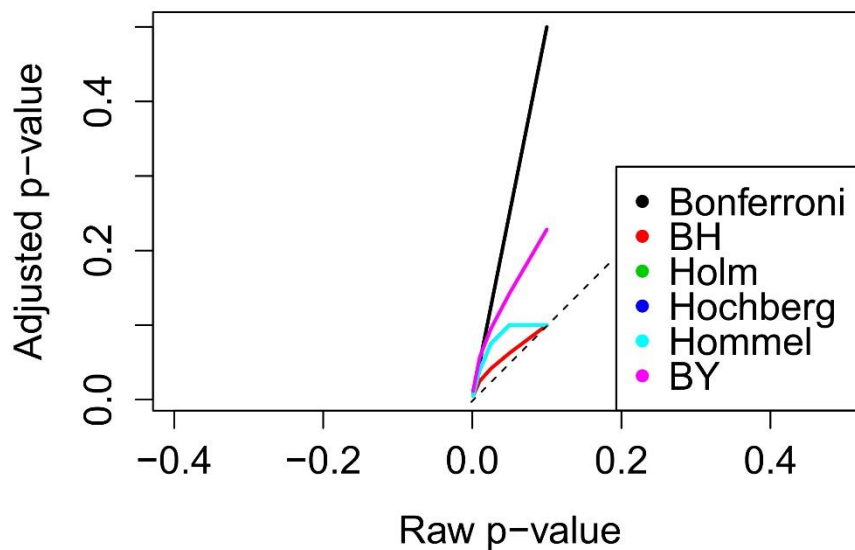
X = Data$Raw.p
Y = cbind(Data$Bonferroni,
          Data$BH,
          Data$Holm,
          Data$Hochberg,
          Data$Hommel,
          Data$BY)

matplot(X, Y,
        xlab="Raw p-value",
        ylab="Adjusted p-value",
        type="l",
        asp=1,
        col=1:6,
        lty=1,
        lwd=2)

legend('bottomright',
      legend = c("Bonferroni", "BH", "Holm", "Hochberg", "Hommel", "BY"),
      col = 1:6,
      cex = 1,
      pch = 16)

abline(0, 1,
       col=1,
       lty=2,
       lwd=1)

```



Plot of adjusted p-values vs. raw p-values for a series of five p-values between 0 and 0.1. Note that Holm and Hochberg have the same values as Hommel, and so are hidden by Hommel. The dashed line represents a one-to-one line.

#

Miscellany

Chapters Not Covered in this Book

Meta-analysis

Using spreadsheets for statistics

Guide to fairly good graphs

Presenting data in tables

Getting started with SAS

Choosing a statistical test

See the *Handbook* for information on these topics.

Contrasts in Linear Models

Contrasts can be used to make specific comparisons of treatments within a linear model.

One common use is when a factorial design is used, but control or check treatments are used in addition to the factorial design. In the first example below, there are two treatments (*D* and *C*) each at two levels (1 and 2), and then there is a *Control* treatment. The approach used here is to analyze the experiment as a one-way analysis of variance, and then use contrasts to test various hypotheses.

Another common use is when there are several treatments that could be thought of as members of a group. In the second example below, there are measurements for six wines, some of which are red (*Merlot*, *Cabernet*, *Syrah*) and some of which are white (*Chardonnay*, *Riesling*, *Gewürztraminer*). We can compare the treatments *within* the red wine group by setting up contrasts and conducting an F-test. This is analogous to testing the main effect of *Red Wine*.

The packages *lsmeans* and *multcomp* allow for unlimited tests of single-degree contrasts, with a p-value correction for multiple tests. They also allow for an F-test for multi-line contrasts, for example when testing *within* groups. The *aov* function in the native *stats* package has more limited functionality.

See the chapters on *One-way Anova* and *Two-way Anova* for general considerations on conducting analysis of variance

Packages used in this chapter

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

```
if(!require(car)){install.packages("car")}
if(!require(lsmeans)){install.packages("lsmeans")}
if(!require(multcomp)){install.packages("multcomp")}
```

Example for single degree-of-freedom contrasts

This hypothetical example could represent an experiment with a factorial design two treatments (*D* and *C*) each at two levels (1 and 2), and a control treatment. The 2-by-2 factorial plus control is treated as a one-way anova with five treatments.

```
Input = ("
Treatment Response
'D1:C1'      1.0
'D1:C1'      1.2
'D1:C1'      1.3
'D1:C2'      2.1
'D1:C2'      2.2
'D1:C2'      2.3
'D2:C1'      1.4
'D2:C1'      1.6
'D2:C1'      1.7
```

```
'D2:C2'      2.5
'D2:C2'      2.6
'D2:C2'      2.8
'Control'    1.0
'Control'    0.9
'Control'    0.8
")
```

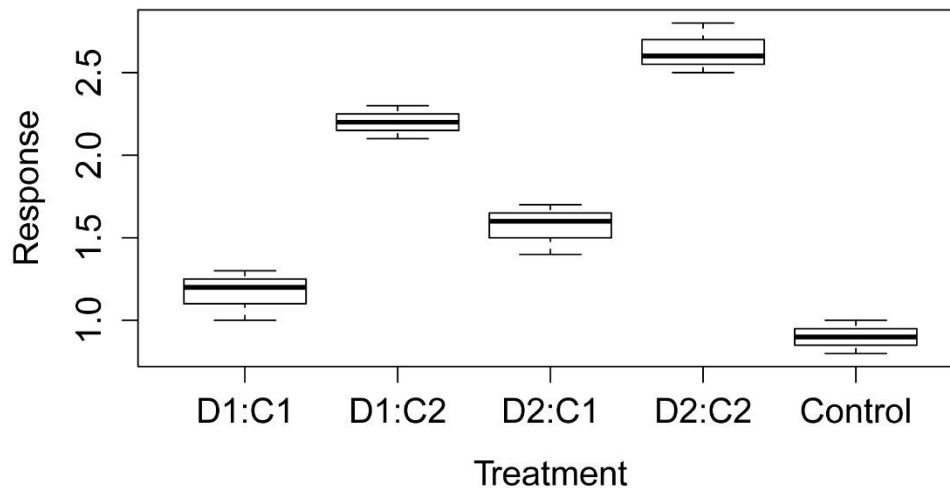
```
Data = read.table(textConnection(Input),header=TRUE)
```

```
### Specify the order of factor levels. Otherwise R will alphabetize them.
```

```
Data$Treatment = factor(Data$Treatment,
                        levels=unique(Data$Treatment))
```

```
Data
```

```
boxplot(Response ~ Treatment,
        data = Data,
        ylab="Response",
        xlab="Treatment")
```



```
### Define linear model
```

```
model = lm(Response ~ Treatment,
            data = Data)
```

```
library(car)
```

```
Anova(model, type="II")
```

```
summary(model)
```

Example with lsmeans

```
### You need to look at order of factor levels to determine the contrasts
```

```
levels(Data$Treatment)
```

```
[1] "D1:C1" "D1:C2" "D2:C1" "D2:C2" "Control"
```

```
library(lsmmeans)
```

```
leastsquare = lsmmeans(model, "Treatment")
```

```
Contrasts = list(D1vsD2      = c(1, 1, -1, -1, 0),
                  C1vsC2      = c(1, -1, 1, -1, 0),
                  InteractionDC = c(1, -1, -1, 1, 0),
                  C1vsC2forD1only = c(1, -1, 0, 0, 0),
                  C1vsC2forD2only = c(0, 0, 1, -1, 0),
                  TreatsvsControl = c(1, 1, 1, 1, -4),
                  T1vsC        = c(1, 0, 0, 0, -1),
                  T2vsC        = c(0, 1, 0, 0, -1),
                  T3vsC        = c(0, 0, 1, 0, -1),
                  T4vsC        = c(0, 0, 0, 1, -1))
```

```
### The column names match the order of levels of the treatment variable
```

```
### The coefficients of each row sum to 0
```

```
contrast(leastsquare, Contrasts, adjust="sidak")
```

contrast	estimate	SE	df	t.ratio	p.value
D1vsD2	-0.83333333	0.1549193	10	-5.379	0.0031
C1vsC2	-2.10000000	0.1549193	10	-13.555	<.0001
InteractionDC	0.03333333	0.1549193	10	0.215	1.0000
C1vsC2forD1only	-1.03333333	0.1095445	10	-9.433	<.0001
C1vsC2forD2only	-1.06666667	0.1095445	10	-9.737	<.0001
TreatsvsControl	3.96666667	0.3464102	10	11.451	<.0001
T1vsC	0.26666667	0.1095445	10	2.434	0.3011
T2vsC	1.30000000	0.1095445	10	11.867	<.0001
T3vsC	0.66666667	0.1095445	10	6.086	0.0012
T4vsC	1.73333333	0.1095445	10	15.823	<.0001

```
### Note that p-values are slightly different than those from multcomp
```

```
### due to different adjustment methods. If "none" is chosen as
```

```
### the adjustment method for both procedures,
```

```
### p-values and other statistics will be the same.
```

```
### With adjust="none", results will be the same as
```

```
### the aov method.
```

Example with multcomp

```
### You need to look at order of factor levels to determine the contrasts
```

```
levels(Data$Treatment)
```

```
[1] "D1:C1" "D1:C2" "D2:C1" "D2:C2" "Control"
```

```

Input = ("
Contrast.Name      D1C2  D1C2  D2C1  D2C2  Control
D1vsD2             1     1    -1    -1     0
C1vsC2             1    -1     1    -1     0
InteractionDC       1    -1    -1     1     0
C1vsC2forD1only    1    -1     0     0     0
C1vsC2forD2only    0     0     1    -1     0
TreatsvsControl    1     1     1     1    -4
T1vsC              1     0     0     0    -1
T2vsC              0     1     0     0    -1
T3vsC              0     0     1     0    -1
T4vsC              0     0     0     1    -1
")

### The column names match the order of levels of the treatment variable
### The coefficients of each row sum to 0

Matriz = as.matrix(read.table(textConnection(Input),
                             header=TRUE,
                             row.names=1))

Matriz

library(multcomp)

G = glht(model,
         linfct = mcp(Treatment = Matriz))

G$linfct

summary(G,
        test=adjusted("single-step"))

### Adjustment options: "none", "single-step", "Shaffer",
###                    "Westfall", "free", "holm", "hochberg",
###                    "hommel", "bonferroni", "BH", "BY", "fdr"

              Estimate Std. Error t value Pr(>|t|)
D1vsD2 == 0      -0.83333    0.15492   -5.379  0.00218 **
C1vsC2 == 0      -2.10000    0.15492  -13.555 < 0.001 ***
InteractionDC == 0  0.03333    0.15492   0.215  0.99938
C1vsC2forD1only == 0 -1.03333    0.10954  -9.433 < 0.001 ***
C1vsC2forD2only == 0 -1.06667    0.10954  -9.737 < 0.001 ***
TreatsvsControl == 0  3.96667    0.34641  11.451 < 0.001 ***
T1vsC == 0        0.26667    0.10954   2.434  0.17428
T2vsC == 0        1.30000    0.10954  11.867 < 0.001 ***
T3vsC == 0        0.66667    0.10954   6.086 < 0.001 ***
T4vsC == 0        1.73333    0.10954  15.823 < 0.001 ***

### With test=adjusted("none"), results will be the same as aov method
below.

```

#

Example for global F-test within a group of treatments

This example has treatments consisting of three red wines and three white wines. We will want to know if there is an effect of the treatments in the red wine group on the response variable, while keeping the individual identities of the wines in the *Treatment* variable. This approach is advantageous because post-hoc comparisons could still be made within the red wines, for example comparing Merlot to Cabernet.

```
Input = ("
Treatment      Response
Merlot         5
Merlot         6
Merlot         7
Cabernet       8
Cabernet       9
Cabernet      10
Syrah         11
Syrah         12
Syrah         13
Chardonnay     1
Chardonnay     2
Chardonnay     3
Riesling       1
Riesling       2
Riesling       2
Gewürtztraminer 1
Gewürtztraminer 2
Gewürtztraminer 4
")
```

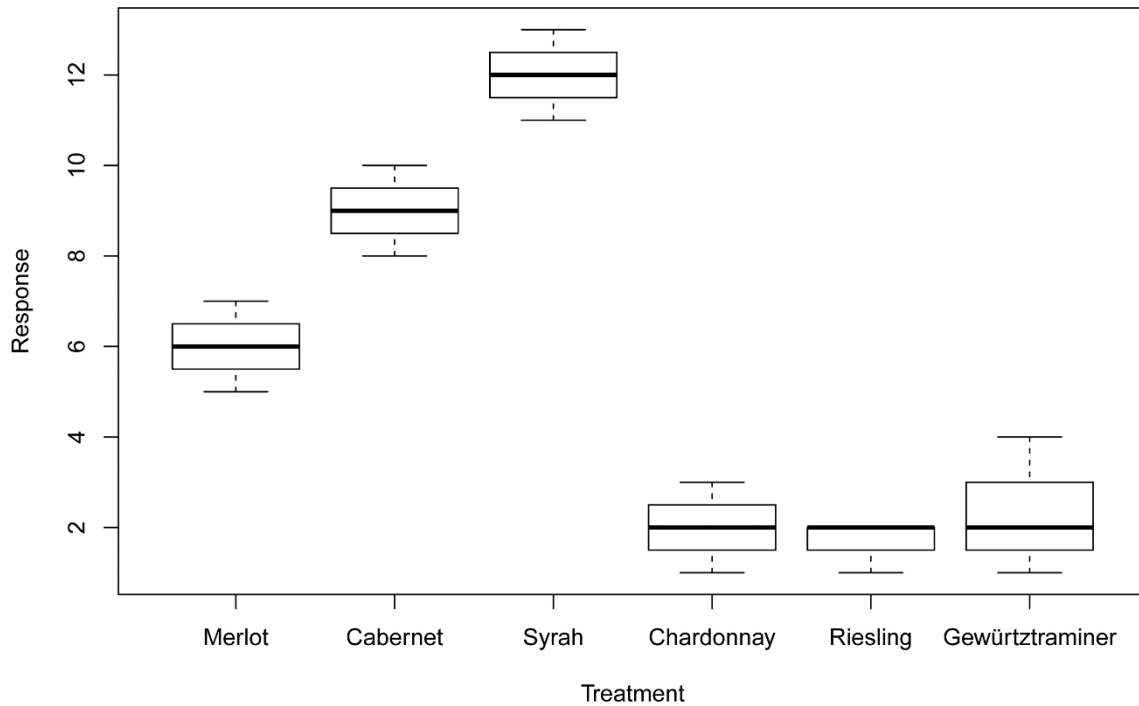
```
Data = read.table(textConnection(Input),header=TRUE)
```

```
### Specify the order of factor levels. Otherwise R will alphabetize them.
```

```
Data$Treatment = factor(Data$Treatment,
                        levels=unique(Data$Treatment))
```

```
Data
```

```
boxplot(Response ~ Treatment,
        data = Data,
        ylab="Response",
        xlab="Treatment")
```

You need to look at order of factor levels to determine the contrasts

```
levels(Data$Treatment)
```

```
[1] "Merlot" "Cabernet" "Syrah" "Chardonnay" "Riesling" "Gewürtztraminer"
```

Define linear model

```
model = lm(Response ~ Treatment,
            data = Data)
```

```
library(car)
```

```
Anova(model, type="II")
```

```
summary(model)
```

Tests of contrasts with lsmeans

Question: Is there an effect within red wine ?

```
library(lsmeans)
```

```
leastsquare = lsmeans(model, "Treatment")
```

```
Contrasts = list(Red_line1 = c(1, -1, 0, 0, 0, 0),
                  Red_line2 = c(0, 1, -1, 0, 0, 0))
```

```
### The column names match the order of levels of the treatment variable
### The coefficients of each row sum to 0
```

```
Test = contrast(leastsquare, Contrasts)
```

```
test(Test, joint=TRUE)
```

```
df1 df2    F p.value
  2   12 24.3  0.0001
```

```
### Note that two lines of contrasts resulted in one hypothesis test
### using 2 degrees of freedom. This investigated the effect within
### a group of 3 treatments.
```

```
### Results are essentially the same as those from multcomp
```

Question: Is there an effect within white wine ?

```
library(lsmmeans)
```

```
leastsquare = lsmmeans(model, "Treatment")
```

```
Contrasts = list(white_line1 = c(0, 0, 0, 1, -1, 0),
                 white_line2 = c(0, 0, 0, 0, 1, -1))
```

```
### The column names match the order of levels of the treatment variable
### The coefficients of each row sum to 0
```

```
Test = contrast(leastsquare, Contrasts)
```

```
test(Test, joint=TRUE)
```

```
df1 df2    F p.value
  2   12  0.3  0.7462
```

```
### Note that two lines of contrasts resulted in one hypothesis test
### using 2 degrees of freedom. This investigated the effect within
### a group of 3 treatments
```

```
### Results are the same as those from multcomp
```

Question: Is there a difference between red and white wines? And, mean separation for red wine

```
library(lsmmeans)
```

```
leastsquare = lsmmeans(model, "Treatment")
```

```
Contrasts = list(Red_vs_white   = c( 1,  1,  1, -1, -1, -1),
                 Merlot_vs_Cab  = c( 1, -1,  0,  0,  0,  0),
                 Cab_vs_Syrah   = c( 0,  1, -1,  0,  0,  0),
                 Syrah_vs_Merlot = c(-1,  0,  1,  0,  0,  0))
```

```
### The column names match the order of levels of the treatment variable
```

```
### The coefficients of each row sum to 0
```

```
contrast(leastsquare, Contrasts, adjust="sidak")
```

contrast	estimate	SE	df	t.ratio	p.value
Red_vs_white	21	1.490712	12	14.087	<.0001
Merlot_vs_Cab	-3	0.860663	12	-3.486	0.0179
Cab_vs_Syrah	-3	0.860663	12	-3.486	0.0179
Syrah_vs_Merlot	6	0.860663	12	6.971	0.0001

```
### Note that p-values are slightly different than those from multcomp
### due to different adjustment methods. If "none" is chosen as
### the adjustment method for both procedures,
### p-values and other statistics will be the same.
```

Tests of contrasts with multcomp

Question: Is there an effect within red wine ?

```
Input = "
Contrast  Merlot  Cabernet  Syrah  Chardonnay  Riesling  Gewürtztraminer
Red_line1  1      -1        0      0             0         0
Red_line2  0       1       -1      0             0         0
"
```

```
### Note: there are two lines of contrasts for a group of three treatments
### The column names match the order of levels of the treatment variable
### The coefficients of each row sum to 0
```

```
Matriz = as.matrix(read.table(textConnection(Input),
                                header=TRUE,
                                row.names=1))
```

```
Matriz
```

```
library(multcomp)
```

```
G = glht(model, linfct = mcp(Treatment = Matriz))
```

```
G$linfct
```

```
summary(G,
        test = Ftest())
```

```
Global Test:
      F  DF1  DF2    Pr(>F)
1  24.3    2   12  6.029e-05
```

```
### Note that two lines of contrasts resulted in one hypothesis test
### using 2 degrees of freedom. This investigated the effect within
### a group of 3 treatments.
```

Question: Is there an effect within white wine ?

```
Input = "
Contrast    Merlot  Cabernet  Syrah  Chardonnay  Riesling  Gewürtztraminer
white_line1  0       0         0       1          -1         0
white_line2  0       0         0       0          1        -1
"

### Note: there are two lines of contrasts for a group of three treatments

### The column names match the order of levels of the treatment variable
### The coefficients of each row sum to 0

Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))

Matriz

library(multcomp)

G = glht(model, linfct = mcp(Treatment = Matriz))

G$linfct

summary(G,
        test = Ftest())

Global Test:
      F DF1 DF2 Pr(>F)
1 0.3   2  12 0.7462

### Note that two lines of contrasts resulted in one hypothesis test
### using 2 degrees of freedom. This investigated the effect within
### a group of 3 treatments.

#      #      #
```

Question: Is there a difference between red and white wines? And, mean separation for red wine

```
Input = "
Contrast      Merlot  Cabernet  Syrah  Chardonnay  Riesling  Gewürtztraminer
Red_vs_white   1       1         1      -1          -1        -1
Merlot_vs_Cab   1      -1         0       0          0         0
Cab_vs_Syrah    0       1        -1       0          0         0
Syrah_vs_Merlot -1       0         1       0          0         0
"
```

```
names match the order of levels of the treatment variable
### The coefficients of each row sum to 0
```

```
Matriz = as.matrix(read.table(textConnection(Input),
                              header=TRUE,
                              row.names=1))
```

```
Matriz
```

```
library(multcomp)
```

```
G = glht(model,
         linfct = mcp(Treatment = Matriz))
```

```
G$linfct
```

```
summary(G,
        test=adjusted("single-step"))
```

```
### Adjustment options: "none", "single-step", "Shaffer",
###                      "westfall", "free", "holm", "hochberg",
###                      "hommel", "bonferroni", "BH", "BY", "fdr"
```

```
Linear Hypotheses:
```

	Estimate	Std. Error	t value	Pr(> t)	
Red_vs_white == 0	21.0000	1.4907	14.087	<0.001	***
Merlot_vs_Cab == 0	-3.0000	0.8607	-3.486	0.0157	*
Cab_vs_Syrah == 0	-3.0000	0.8607	-3.486	0.0156	*
Syrah_vs_Merlot == 0	6.0000	0.8607	6.971	<0.001	***

```
(Adjusted p values reported -- single-step method)
```

```
### with test=adjusted("none"), results will be the same as aov method
below.
```

Tests of contrasts within *aov*

Another method to use single-degree-of-freedom contrasts within an anova is to use the *split* option within the *summary* function for an *aov* analysis. The number of degrees of freedom that a factor can be split into for contrast tests is limited.

```
Input =("
Treatment  Response
'D1:C1'    1.0
'D1:C1'    1.2
'D1:C1'    1.3
'D1:C2'    2.1
'D1:C2'    2.2
'D1:C2'    2.3
'D2:C1'    1.4
'D2:C1'    1.6
'D2:C1'    1.7
```

```
'D2:C2'      2.5
'D2:C2'      2.6
'D2:C2'      2.8
'Control'    1.0
'Control'    0.9
'Control'    0.8
")
```

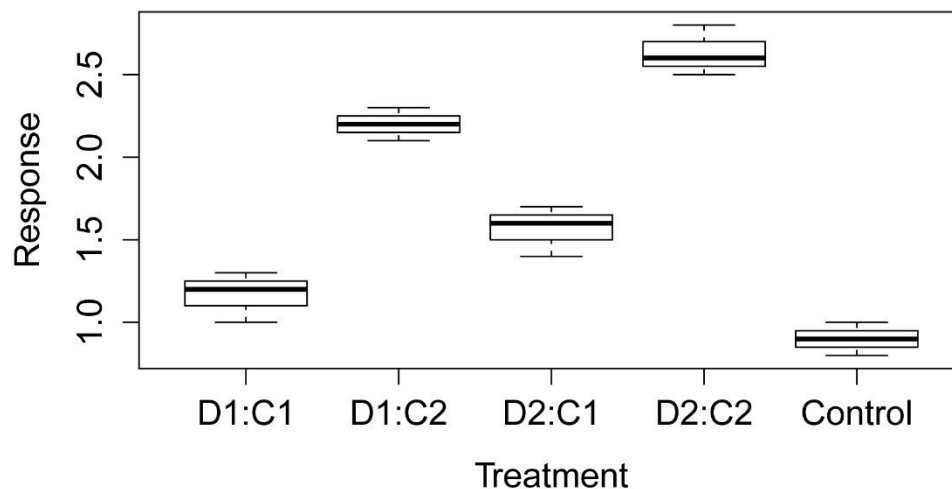
```
Data = read.table(textConnection(Input),header=TRUE)
```

```
Data$Treatment = factor(Data$Treatment, levels=unique(Data$Treatment))
```

```
### Specify the order of factor levels. Otherwise R will alphabetize them.
```

```
Data
```

```
boxplot(Response ~ Treatment,
         data = Data,
         ylab="Response",
         xlab="Treatment")
```



```
levels(Data$Treatment)
```

```
### You need to look at order of factor levels to determine the contrasts
```

```
[1] "D1:C1" "D1:C2" "D2:C1" "D2:C2" "Control"
```

```
### Define contrasts
```

```
D1vsD2 = c(1, 1, -1, -1, 0)
C1vsC2 = c(1, -1, 1, -1, 0)
InteractionDC = c(1, -1, -1, 1, 0)
TreatsvsControl = c(1, 1, 1, 1, -4)
```

```
Matriz = cbind(D1vsD2, C1vsC2,
               InteractionDC, TreatsvsControl)
```

```

contrasts(Data$Treatment) = Matriz

CList = list("D1vsD2" = 1,
            "C1vsC2" = 2,
            "InteractionDC" = 3,
            "TreatsvsControl" = 4)

### Define model and display summary

model = aov(Response ~ Treatment, data = Data)

summary(model,
        split=list(Treatment=CList))

```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
Treatment	4	6.189	1.547	85.963	1.06e-07	***
Treatment: D1vsD2	1	0.521	0.521	28.935	0.00031	***
Treatment: C1vsC2	1	3.307	3.307	183.750	9.21e-08	***
Treatment: InteractionDC	1	0.001	0.001	0.046	0.83396	
Treatment: TreatsvsControl	1	2.360	2.360	131.120	4.53e-07	***
Residuals	10	0.180	0.018			

#

Cate–Nelson Analysis

Cate–Nelson analysis is used to divide bivariate data into two groups: one where a change in the x variable is likely to correspond to a change in the y variable, and the other group where a change in x is unlikely to correspond to a change y. Traditionally this method was used for soil test calibration. For example to determine if a certain level of soil test phosphorus would indicate that adding phosphorus to the soil would likely cause an increase in crop yield or not.

The method can be used for any case in which bivariate data can be separated into two groups, one with a large x variable is associated with a large y, and a small x associated with a small y. Or vice-versa.

For a fuller description of Cate–Nelson analysis and examples in soil-test and other applications, see [Mangiafico](#) (2013) and the references there.

Custom function to develop Cate–Nelson models

My *cateNelson* function follows the method of [Cate and Nelson](#) (1971). A critical x value is determined by iteratively breaking the data into two groups and comparing the explained sum of squares of the iterations. A critical y value is determined by using an iterative process which minimizes the number of data point which fall into Quadrant I and III for data with a positive trend.

Options in the *cateNelson* function:

- *plotit=TRUE* (the default) produces a plot of the data, a plot of the sum of squares of the iterations, a plot of the data points in error quadrants, and a final plot with critical x and critical y drawn as lines on the plot.
- *hollow=TRUE* (the default) for the final plot, points in the error quadrants as open circles
- *trend="negative"* (not the default) needs to be used if the trend of the data is negative.
- *xthreshold* and *ythreshold* determine how many options the function will return for critical x and critical y. A value of 1 would return all possibilities. A value of 0.10 returns values in the top 10% of the range of maximum sum of squares.
- *clx* and *cly* determine which of the listed critical x and critical y the function should use to build the final model. A value of 1 selects the first displayed value, and a value of 2 selects the second. This is useful when you have more than one critical x that maximizes or nearly maximizes the sum of squares, or if you want to force the critical y value to be close to some value such as 90% of maximum yield. Note that changing the *clx* value will also change the list of critical y values that is displayed. In the second example I set *clx=2* to select a critical x that more evenly divides the errors across the quadrants.

Example of Cate–Nelson analysis

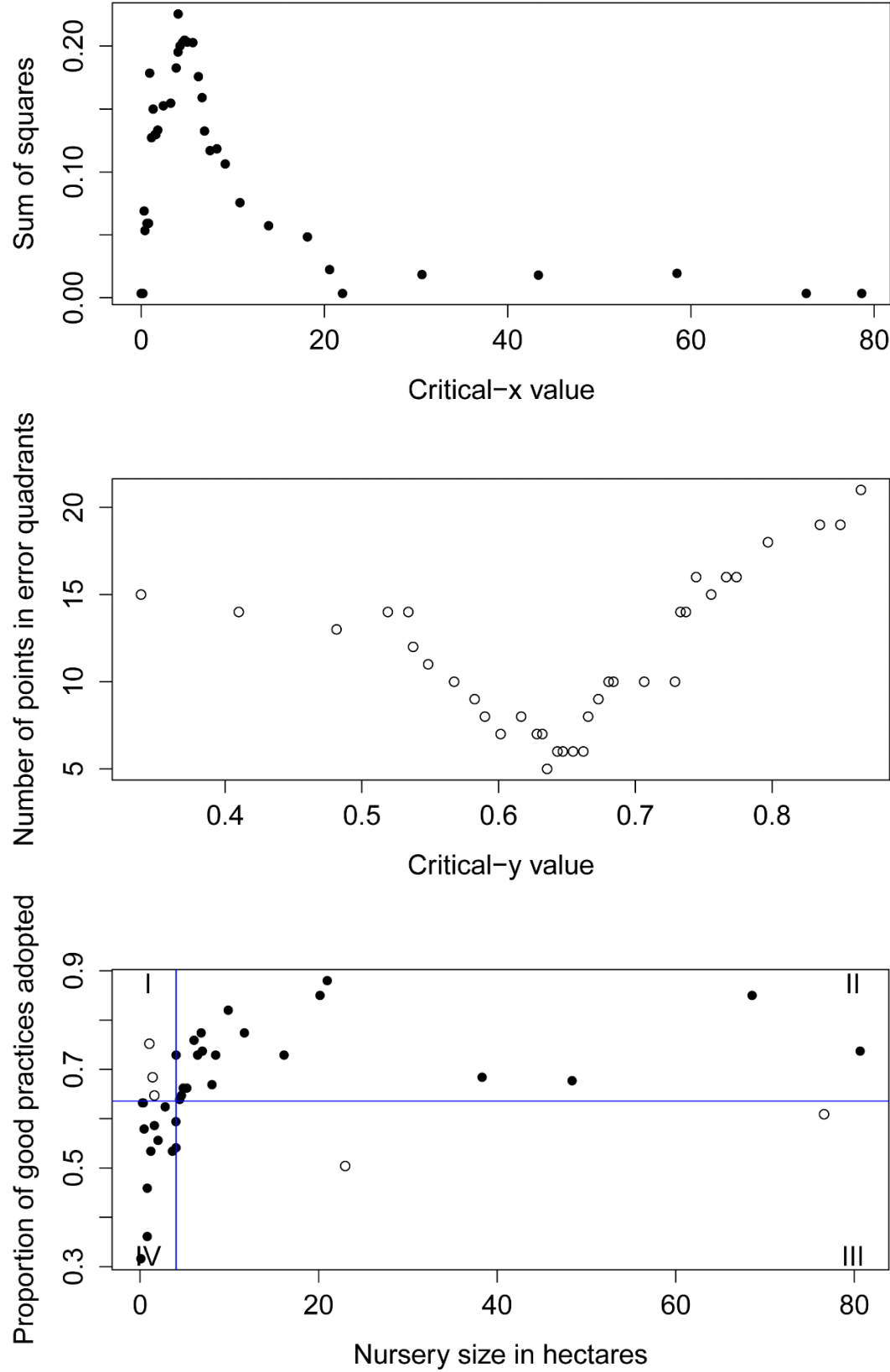
```
##-----
## Cate-Nelson analysis
## Data from Mangiafico, S.S., Newman, J.P., Mochizuki, M.J.,
##   & Zurawski, D. (2008). Adoption of sustainable practices
##   to protect and conserve water resources in container nurseries
##   with greenhouse facilities. Acta horticulturae 797, 367–372.
##-----

size = c(68.55,6.45,6.98,1.05,4.44,0.46,4.02,1.21,4.03,
        6.05,48.39,9.88,3.63,38.31,22.98,5.24,2.82,1.61,
        76.61,4.64,0.28,0.37,0.81,1.41,0.81,2.02,20.16,
        4.04,8.47,8.06,20.97,11.69,16.13,6.85,4.84,80.65,1.61,0.10)

proportion = c(0.850,0.729,0.737,0.752,0.639,0.579,0.594,0.534,
              0.541,0.759,0.677,0.820,0.534,0.684,0.504,0.662,
              0.624,0.647,0.609,0.647,0.632,0.632,0.459,0.684,
              0.361,0.556,0.850,0.729,0.729,0.669,0.880,0.774,
              0.729,0.774,0.662,0.737,0.586,0.316)

library(rcompanion)

cateNelson(x = size,
           y = proportion,
           plotit=TRUE,
           hollow=TRUE,
           xlab="Nursery size in hectares",
           ylab="Proportion of good practices adopted",
           trend="positive",
```

Plots showing the results of Cate–Nelson analysis. In the final plot, the critical x value is indicated with a vertical blue line, and the critical y value is indicated with a

horizontal blue line. Points agreeing with the model are solid, while hollow points indicate data not agreeing with model. (Data from Mangiafico, S.S., Newman, J.P., Mochizuki, M.J., & Zurawski, D. (2008). Adoption of sustainable practices to protect and conserve water resources in container nurseries with greenhouse facilities. *Acta horticulturae* 797, 367–372.)

#

Example of Cate–Nelson analysis with negative trend data

```
##-----
## Cate-Nelson analysis
## Hypothetical data
##-----

Input =("
  x      y
  5      55
  7     110
  6     120
  5     130
  7     120
 10      55
 12      60
 11     110
 15      50
 21      55
 22      60
 20      70
 24      55
")

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

library(rcompanion)

cateNelson(x = Data$x,
            y = Data$y,
            plotit=TRUE,
            hollow=TRUE,
            xlab="x",
            ylab="y",
            trend="negative",
            clx=2,      # Normally leave as 1 unless you wish to
            cly=1,      # select a specific critical x value
            xthreshold=0.10,
            ythreshold=0.15)
```

Critical x that maximize sum of squares:

	Critical.x.value	Sum.of.squares
1	11.5	5608.974
2	8.5	5590.433

Critical y that minimize errors:

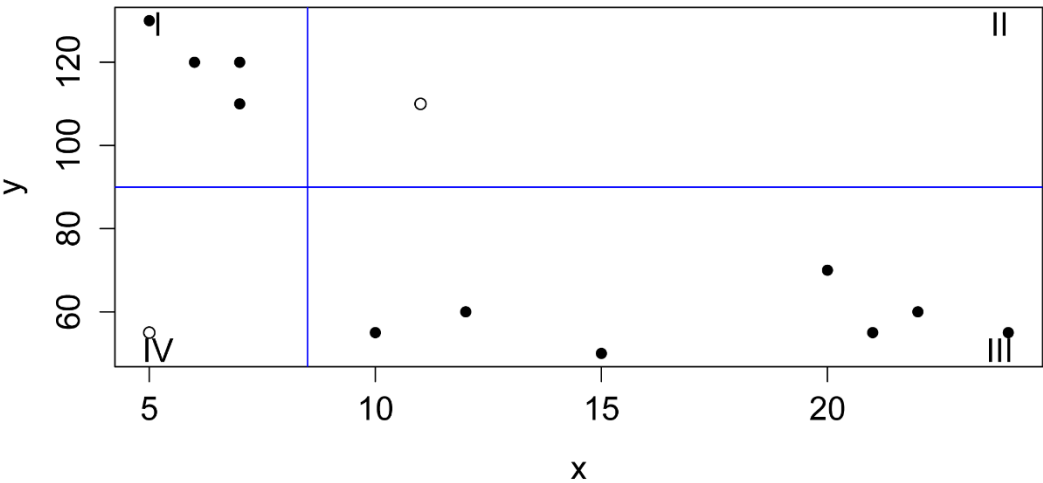
	Critical.y.value	Q.i	Q.ii	Q.iii	Q.iv	Q.model	Q.err
1	90	4	1	7	1	11	2
2	110	4	1	7	1	11	2
3	115	3	0	8	2	11	2
4	120	3	0	8	2	11	2

n = Number of observations
CLx = Critical value of x
SS = Sum of squares for that critical value of x
CLy = Critical value of y
Q = Number of observations which fall into quadrants I, II, III, IV
Q.Model = Total observations which fall into the quadrants predicted by the model
p.Model = Percent observations which fall into the quadrants predicted by the model
Q.Error = Observations which do not fall into the quadrants predicted by the model
p.Error = Percent observations which do not fall into the quadrants predicted by the model
Fisher = p-value from Fisher exact test dividing data into these quadrants

Final model:

	n	CLx	SS	CLy	Q.I	Q.II	Q.III	Q.IV	Q.Model	p.Model	Q.Error
1	13	8.5	5608.974	90	4	1	7	1	11	0.8461538	2

p.Error Fisher.p.value
0.1538462 0.03185703



Plot showing the final result of Cate–Nelson analysis, for data with a negative trend.

#

References

Mangiafico, S.S. 2013. Cate-Nelson Analysis for Bivariate Data Using R-project. J.of Extension 51:5, 5TOT1. <http://www.joe.org/joe/2013october/tt1.php>.

Cate, R. B., & Nelson, L.A. (1971). A simple statistical procedure for partitioning soil test correlation data into two classes. *Soil Science Society of America Proceedings* 35, 658–660.

Additional Helpful Tips

Reading SAS Datalines in R

Reading SAS datalines with *DescTools*

The *ParseSASDatalines* function in the *DescTools* package will read in data with simple SAS DATALINES code. More complex INPUT schemes may not work.

```
### -----
### Reading SAS datalines, DescTools::ParseSASDatalines example
### -----
```

```
Input = ("
DATA survey;
INPUT id sex $ age inc r1 r2 r3 @@;
DATALINES;
1   F  35 17  7 2 2  17  M  50 14  5 5 3  33  F  45  6  7 2 7
49  M  24 14  7 5 7  65  F  52  9  4 7 7  81  M  44 11  7 7 7
2   F  34 17  6 5 3  18  M  40 14  7 5 2  34  F  47  6  6 5 6
50  M  35 17  5 7 5
;
")
```

```
library(DescTools)
```

```
Data = ParseSASDatalines(Input)
```

```
### You can omit the DATA statement, the @@, and the final semi-colon.
### The $ is required for factor variables.
```

```
Data
```

```
      id sex age inc r1 r2 r3
1     1  F  35 17  7  2  2
2    17  M  50 14  5  5  3
3    33  F  45  6  7  2  7
4    49  M  24 14  7  5  7
5    65  F  52  9  4  7  7
6    81  M  44 11  7  7  7
7     2  F  34 17  6  5  3
8    18  M  40 14  7  5  2
9    34  F  47  6  6  5  6
10   50  M  35 17  5  7  5
```

```
#      #      #
```