Yet Another Guide to Statistical Computing with R

Version 1.1

Michael P. Morris
Research Fellow
Australian National University

Wednesday 30 October 2024

Abstract

This document is a very short guide to using R for data analysis. It outlines some of R's main conventions, data management and methods for obtaining descriptive and inferential statistics. It is aimed at people who are new to R, or are returning to it, and mainly focuses on Windows users. It uses the latest version of R available at the time of writing: R v4.4.0 'Puppy Cup'.

Contents

List of Figures

List of Tables

1 Introduction

R is a language and environment for statistical computing and graphics. It was written by Robert Gentleman and Ross Ihaka of the University of Auckland, and is currently managed by the R Core Team. It is available as a free download from The Comprehensive R Archive Network website at http://cran.r-project.org. It is available for a wide variety of operating systems including Linux-alike, Windows and MacOS. R is widely used and is under constant development. Apart from R itself, there are a large number of add-on packages that extend R's capabilities. These are also freely available from R's website.

This document is a short introduction to R. It grew from some scripts I developed as part of a program of professional development. The annotations included with the scripts became a separate file of their own, and those files became this document.

This document is no substitute for a course in statistics. It assumes a working knowledge of statistics and gives some guidance on how to manage data and perform an analysis with R. There are a large number of very good reference texts available for R and the best of these is the material by the R Core Team and the package documentation. These are available through the Help menu on the R Console and more recent versions are available from http://cran.r-project.org/manuals.html. There is a short list of useful texts in the references.

Most analysis involving R needs to be coded by hand. For those who are used to a GUI interface, well, they say that you can get used to almost anything.

There are a large number of add-on packages for R, and some of these have overlapping functionality. The package choices in this document reflect my own personal preferences. Anyone else's mileage may vary.

The examples included in this text are fairly bare-bones. They don't reflect all of R's statistical techniques, and don't take advantage of all of R's capabilities. When performing an analysis, it worth checking the documentation for additional features and limitations.

The version of R referred to in this document is the most recent version available at the time of writing: R version 4.4.0 'Puppy Cup' running under Windows 10. This document focuses on Windows users so some sections, such as installation, may be irrelevant to those using other operating systems. For these users, please consult your documentation.

All the data used in the examples in this document are fictional and refers in no way to any actual study and are there for illustrative purposes only. The data is contained in the appendix.

The author is interested in receiving error reports, especially if it relates to the R code, and suggestions for improvements and additions. They may be sent to the author at mmorris1k@gmail.com. More recent versions of this document may be obtained from

the same email address.

Early versions of this document were named 'A Very Short Guide to Statistical Computing with R'. The name was changed to better reflect it's contents.

Acknowledgements Thanks to RAM Gregson and those anonymous people who have provided helpful comments and suggestions.

2 Installing and Running R

2.1 Installation

The latest version of R can be freely downloaded from http://cran.r-project.org. After installation, R's binary files should be added to the Windows 10 search path. These are the files that launch R and they have an `.exe` suffix. They are located in

`C:\Program Files\R-4.4.0\ bin\x64}`

To add them to Window's search path go to:

`Control Panel -> System and Security -> System -> Advanced System Settings -> Environmental Variables -> Path in the User Variables box -> Edit -> New -> Browse...`

Then navigate to R and click on the folder containing the R binaries. Click `OK` and a new line will be added to the `Edit environmental variable` box. Click `OK` again in this box, `OK` in the `Environmental Variables` box, `OK` in the `System Properties` box and then close the `Settings` and `System and Security` boxes. Finally, Windows will need to be restarted so that it can include R in it's search path.

When R is installed and in the search path, the console preferences can be adjusted. To do this, launch R, then go to the console menu and select:

`Edit -> GUI preferences`

This brings up the `Rgui Configuration Editor`. This has options for colours, fonts, number of rows and columns and `Single or Multiple`. This is best set to `Single`. This will allow multiple independant windows with their own drop-down menus. This is helpful when running an analysis because scripts and graphs can be in seperate windows and output in the console window. Other options may be adjusted to suit your individual requirements.

2.1.1 Packages

R relies on packages to provide functionality. These are software add-ons with additional commands. For instance, the `jmv` package provides a `describe()` command that provides comprehensive descriptive statistics. These packages need to be downloaded, installed, and loaded into R when required.

R's base installation comes with seven packages. A large number of additional packages are available from CRAN's website at http://cran.r-project.org. They can be installed through `Packages` in the R Console's menu. Select

There are multiple locations around the world that host copies, or mirrors, of `cran` and the first step is to select your preferred location from the pop up list of `Secure cran mirrors`:

`Packages -> Install Package(s) -> Select your preferred location from the list -> click OK`

When the download location is selected, a pop up list titled `Packages` will appear. Select the required package(s) from the list and click OK to download them.

Packages often require that additional packages be present on your system and these also need to be installed. R downloads and installs these dependencies automatically at the same time as the required package.

Packages need to be loaded into R before users can access their functionality. The `library (PackageName)` command loads the package and any dependencies. Loading `asbio` will also load it's dependency `tcltk`:

```
>
> library(asbio)
Loading required package: tcltk
>
```

Packages can also be loaded from the R Console menu. Click

`Packages -> Load Package -> Select the required package from the list and click OK`

To get a list of loaded packages, use `search()`. This returns:

```
>
> search()
[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
>
```

These are the packages that R loads at the start of a session. To unload a package, use `detach(package:"PackageName")`.

Table 1 lists the packages used for the examples in this document. All of them, and their dependencies are freely available from http://cran.r-project.org.

2.1.2 Searching R

R has a search function that helps users find functions and features. This can be accessed either from the R Console menu via

`Help -> Html help}`

or by the `?CommandName` function. The `?CommandName` option only works if its package is loaded. If the package is not loaded, use `?PackageName::CommandName`. Searching via HTML allows keyword searches and has a list of installed packages for browsing.

Package Name
asbio
car
correlation
DescTools
effectsize
epitools
ez
ggpubr
jmv
psych
rockchalk
rstatix

Table 1: R packages used in this document

2.1.3 Additional Software: Ghostscript, Graphics Editors and Viewers

One of R's strengths is its ability to generate graphs and will produce them in a wide range of popular raster and vector formats. To view and/or edit these, a graphics editor is required. Among the many packages that are available are Gnu Image Manipulation Program, (GIMP), and Inkscape. GIMP is an image manipulation program that will work with a wide range of vector and raster formats. Inkscape works with vector formats only. They available from http://www.gimp.org and http://inkscape.org respectively.

To take full advantage of R's vector graphics capabilities, Ghostscript will need to be installed. This is an interpreter for Postscript (ps), Encapsulated Postscript (eps) and Portable Document Format (pdf) files. It is freely available from http://www.ghostscript.com. Once the executable file is downloaded and installed, it should be placed in the Windows search path so that R can find it. This is done in the same way that R is. To do this, go to:

```
Control Panel -> System and Security -> Security and Maintenance -> System
-> Advanced System Settings and a box named System Properties will appear.

Click on the Advanced tab -> Environmental Variables box -> Path in
the User Variables section -> Edit and the Edit environmental variable
box will appear -> New and a cursor will appear on a new line at the
bottom of the list of software that has been added to the search path

-> Browse and then browse to the location on your system where Ghostscript's
*.exe files are located. Ghostscript's default location is:

C:\Program Files\gs\gsVersion\ bin
```

The `bin` folder contains ghostscript's executable files. When this is selected, click the OK box then the next OK box, OK in System Properties and finally close the Control Panel. The computer now needs to be restarted so that Ghostscript's location is included in Windows search path. It will then become available when working with Postscript and PDF files.

2.2 Running R

R can be initiated in Windows 10 by clicking it's icon on the desktop, the taskbar or it's entry in the Start menu. If it has been placed in the Windows path, it can be run in the command prompt window by typing `R` in the command line. R scripts can be run from the R console or from the command line. The output can be directed to either the screen, which is the default, or to a text file in the working directory.

When running R from the console it is convenient to write the commands in a separate script window. This helps with editing and saving scripts. It can be invoked from the console's `File` menu by clicking on

```
File -> New script
```

This creates a new window with drop down menus. R commands can be entered directly into this window or cut and pasted into it. Existing scripts are loaded with

```
File -> Open script
```

then by navigating to the directory that contains the script, selecting it and clicking `open`.

Commands can be run from the `New Script` box a single line at a time, as a selected section or the entire script at once. To run a single line of code, go to the relevant line and press `ctrl-R` or go to the editor window:

```
Edit -> Run line or selection
```

```
or
```

```
Edit -> Run all
```

To run a selection, select the required lines with the mouse and select the same options. To run an entire script, select `Run all` in the same menu.

To run an existing script from the R Console, place a text file containing the script into the working directory, then use `source()` to run it. This command requires the name of the file. In the following example, `source()` runs a script contained in a file named `RBatch.txt`. The `echo = TRUE` option sends the output to the screen.

```
source('RBatch.txt', echo = TRUE)
```

R will also run batch files from the command prompt with `Rscript`. First, navigate to the working directory, in this case, a folder called R_Project on User Michael's desktop:

```
cd C:/Users/Michael/Desktop/R_Project
```

Next, run the script with `Rscript`. In this example, the file is called `RBatch.txt`:

```
Rscript RBatch.txt
```

R sends output to the screen by default. To send it to a file, the script should include the `sink()` command. It should be placed at the beginning of the file along with the name of the file to be written to. This example sends output to a file named `SinkTest.txt` in the current working directory. To append output to an existing file, specify `append = TRUE`. If append is set to `FALSE`, the named file will be over written if it already exists. The following code uses sink to direct output to a file named SinkTest.txt, executes a script then redirects out back to the screen:

```
sink(file = "SinkTest.txt", append = FALSE, type = "output")

# R script here.

# Close sink() and direct output back to the screen:
sink()
```

2.2.1 R Syntax

R is an Object-Oriented Programming language. This programming paradigm is based on objects that contain data, and code that performs procedures. R objects are data structures such as vectors, lists, matrices, arrays, factors, and data frames. R has six data types: character, numeric, integer, logical, complex numbers and raw. These combine to form data structures.

Data structures that contain only one type of data are called atomic vectors. The most common data type in R is the data frame which is a type of list where each element has the same length and forms a rectangle. It can contain any combination of R's data types. Data frames are created from external files with the `read.csv()` or `read.table()` commands. The type of data in an object can be found with `typeof()`.

Objects also have attributes that add information. These are names, dimnames, dims and classes, and can be used, for example, to name columns or rows. The command `attr()` can list an object's attributes or edit them.

R's default code for missing data is NA, which means 'Not Available'. The name of an R object must begin with a letter`(a-z or A-Z)`. Otherwise, it can contain letters, digits `(0-9)`, dots `(.)` or underscores `(_)`. R is case sensitive: `Lm()` is not the same as `lm()` and a data frame named `study` is not the same as `Study`.

R objects may be created manually by means of the assign operator. This is an arrow created with a bracket and minus sign: `<-`. The following example assigns the value 7 to object `x` and 5 to object `y`. Object content can be printed by typing in their names, and R's calculator can perform operations with them:

```
>
> x <- 7
> y <- 5
>
> x
[1] 7
> y
[1] 5
>
> x*y
[1] 35
>
```

R's calculator has a range of functions, some of which are statistical. The following commands use R's calculator to calculate means for two variables from the Skydiving data file, named Atlantis and Lemuria respectively with `mean()`:

```
>
> mean(Atlantis)
[1] 14.6
> mean(Lemuria)
[1] 10.13333
>
```

R functions must be typed with parentheses: `function()`. This indicates to R that it is a command. If the command is typed without them, R will treat it as an object and print its contents.

Many of R's commands have options that may be switched on or off. This may be specified by the user with `option = TRUE` or `option = FALSE`. For the default settings, see the command's documentation. This may be accessed with `?CommandName` if their package is loaded, or `??CommandName`, with two question marks, if it is not. For instance, `?chisq.test()` indicates that the default settings for `chisq.test()` has Yates Continuity Correction turned on. This may be used if there are small expected frequencies in 2×2 contingency tables. To turn this off, the `correct` option should be set to `FALSE` with `correct = FALSE`.

Individual variables can be specified by either of two ways: by name only, or by both data frame name and variable name with the two separated by a `$`: `X1` and `Study$X1` both refer to variable `X1` in the `Study` data frame. The latter is useful if you are working with multiple versions of the same data frame as it will distinguish between them.

It is always a good idea to place comments in your code. Annotations can be inserted into R scripts by placing the # sign at the beginning of each line that is a comment. This prevents R from attempting to run these lines and returning an error.

One of R's common functions is `c()`. This means 'combine' and returns a vector, or one-dimensional array. For example, the following code creates two vectors named `Var.1` and `Var.2`. The contents of each are contained in the `c()` command:

```
>
> Var.1 <- c(1, 2, 3, 4, 5)
> Var.2 <- c(6, 7, 8, 9, 10)
>
> Var.1
[1] 1 2 3 4 5
> Var.2
[1]  6  7  8  9 10
>
```

To get a list of R objects, use `ls()`:

```
>
> ls()
[1] "Var.1" "Var.2"
>
```

To remove an R object from the workspace, use the `rm(ObjectName)` command. To remove all objects, use `rm(list = ls())`.

2.2.2 R's Formula Interface

R uses a formula interface to denote the various cells of experimental designs. This may be individual variables for data in wide format, or the data in individual cells of factor levels in long format data frames. It uses a scheme similar to Wilkinson and Rogers (1973). This requires the name of the column containing the dependant variable, or responses, the names of the column(s) containing the independent variable(s) or factor(s) and the name of the data frame containing the data. The simplest formula is either a one factor model, with either a single independent variable, or predictor, or a single factor with multiple levels indicating the cells of an experimental design.

```
Command(Dependant Variable ~ Independent Variable, data = DataFrameName)
```

or

```
Command(Dependant Variable ~ Factor, data = DataFrameName)
```

Operator	Function
`+`	Additive
`*`	Crossing
`:`	Interaction

Table 2: Common functions of R's formula interface.

The ∼ character is a tilde, sometimes called a 'twiddle', and denotes a weak approximation. It is located on the top left of the keyboard's typing keys below the escape key.

Formulas can become quite complex when there are multiple factors with multiple levels or when there are multiple predictors. R uses several functions to separate the cells in an experimental design and some common functions are listed in Table 2.

In models that have two or more factors, such as for a multi-way Analysis of Variance, where we want all the possible cells and interactions, the `*`, or crossing operator is used:

```
Dependant Variable ∼ Factor 1 * Factor 2
```

To restrict it to the factors, or variables, with no crossing, the `+`, or additive operator is used. This is often used when performing Multiple Regression with the data in wide format:

```
Dependant Variable ∼ Predictor 1 + Predictor 2 + Predictor 3 ...
```

If the interaction alone is required, the `:` operator is used:

```
Response ∼ Factor 1 :  Factor 2 :  Factor 3
```

For more information on this, see the R Documentation for Model Formula in the `stats` package. This may be accessed with `?formula`.

3 Project and Data Management

3.1 Project Management

The first step in any analysis using R is to nominate the working directory. This tells R where to find input files and where to write output. A unique directory should be created for each project as this will keep data files, graphs, scripts and output in the same place. The current working directory can be found with the command `getwd()`. The default working directory in Windows 10 is the current user's Documents directory:

```
>
> getwd()
[1] "C:/Users/Michael/Documents"
>
```

This can be changed with `setwd()` and entering the entire path to the directory or by going to the `File` menu on the R Console:

```
File -> Change dir
```

and then navigating to the directory.
To change it to a directory called R_Project on the desktop:

```
>
> setwd("C:/Users/Michael/Desktop/R_Project")
> getwd()
[1] "C:/Users/Michael/Desktop/R_Project"
>
```

3.2 Importing and Saving Data Files

R is capable of importing data in a wide range of formats. These are outlined in the R Data Import / Export Manual (2024). This manual also comments that the easiest file format for R to import is a text file. This section outlines how to input data into R, by either loading a saved R file, importing a text file or entering it directly into R.

3.2.1 Inputting Data Directly Into R

Data may be entered directly into R with the c() function and assigning it to an R object:

```
RObject <- c()
```

The example above creates an R object with a single column of data. Multiple R objects are required to create a data frame. When these are created, they may be combined to form a data frame. Ranges, such as for participant ID numbers, can be generated by R with a colon `(:)`. `1:100` generates a list of consecutive numbers between 1–100.

The following code creates four R objects named $X1$, $X2$ and $X3$ and a column of ID numbers. Each column has five data points, and the ID column has a range of 1–5 for data from five participants:

```
>
> X1 <- c(5, 3, 7, 6, 4)
> X2 <- c(8, 3, 4, 3, 5)
> X3 <- c(6, 4, 6, 1, 6)
> ID <- c(1:5)
>
```

These individual R objects will form the data columns. They are combined into a single data frame with `data.frame()`. In this example, the data frame is named `Study`. Which columns are included and their order is specified by the list in `data.frame()`. In this case the first column contains `ID` followed by `X1`, `X2` and `X3`. Details on the data frame are available with `ls.str()`. It forms a 1×3 experimental design:

```
>
> Study <- data.frame(ID, X1, X2, X3)
>
> ls.str(Study)
ID :  int [1:5] 1 2 3 4 5
X1 :  num [1:5] 5 3 7 6 4
X2 :  num [1:5] 8 3 4 3 5
X3 :  num [1:5] 6 4 6 1 6
>
```

R also accepts character data. When this is entered through the console, each character string should be enclosed between single quotation marks. The following code creates an R object named Birds that contains a list of five birds:

```
>
> Birds <- c('Magpie', 'Emu', 'Rosella', 'Lyre Bird', 'Rainbow Lorrikeet')
> Birds
[1] "Magpie"   "Emu"    "Rosella"   "Lyre Bird"   "Rainbow Lorrikeet"
>
```

3.3 Wide and Long Data Formats

R accepts data in two formats: wide and long formats. The `Study` example above is in wide format. Wide format data columns contain data for one variable per column. Long format data frames have a column of participant ID numbers, a single column of participant responses and one or more grouping columns indicating which experimental group a participant belongs to. These grouping columns are referred to as factors.

Factors may be defined by either numbers (e.g.: 1 for Male / 2 for female) or with text labels (e.g.: Male / Female). It is convenient to define factors with text labels as R uses these to label graphs.

Columns may be specified as factors with `factor()`. Factors may be ordered or unordered. If the factors have a natural order, such as time, dosage level, or are a continuous variable divided into sections, specify them as ordered. This is done with

`factor()`'s `ordered` option: `ordered = TRUE`. R defaults to unordered factors that are sorted alphabetically.

The following example creates a long format data frame with three columns: Participant ID numbers, a response column, and a factor column with three levels. This reflects a 1×3 experimental design. In this example, there are five participants. Each contributed a response to each of the three experimental conditions for a repeated measures design. Because the participant numbering repeats 1–5, it is possible to use the range command rather than type in each number. If each participant had contributed a single data point to the entire study rather than three, the ID numbering would be 1–15.

```
>
> ID <- c(1:5, 1:5, 1:5)
> Response <- c(5, 3, 7, 6, 4, 8, 3, 4, 3, 5, 6, 4, 6, 1, 6)
> Group <- c('X1', 'X1', 'X1', 'X1', 'X1', 'X2', 'X2', 'X2', 'X2', 'X2',
'X3', 'X3', 'X3', 'X3', 'X3')
>
```

These R objects are then combined into a single data frame named Study with the `data.frame()` command. This command requires the names of the R objects to be included in the data frame:

```
>
> Study <- data.frame(ID, Response, Group)
>
> ls.str(Study)
Group :  chr [1:15] "X1" "X1" "X1" "X1" "X1" "X2" "X2" "X2" "X2" "X2" ...
ID :  int [1:15] 1 2 3 4 5 1 2 3 4 5 ...
Response :  num [1:15] 5 3 7 6 4 8 3 4 3 5 ...
>
```

Finally, the levels of the `Group` factor need to be specified with the `factors()` command:

```
>
> factor(Group, levels = c('X1', 'X2', 'X3'), ordered = TRUE)
[1] X1 X1 X1 X1 X1 X2 X2 X2 X2 X2 X3 X3 X3 X3 X3
Levels: X1 < X2 < X3
>
> Study
   ID Response Group
1   1        5    X1
2   2        3    X1
3   3        7    X1
4   4        6    X1
5   5        4    X1
6   1        8    X2
```

```
7   2        3    X2
8   3        4    X2
9   4        3    X2
10  5        5    X2
11  1        6    X3
12  2        4    X3
13  3        6    X3
14  4        1    X3
15  5        6    X3
>
```

The presence of the directional signs `<` in the `Levels` part of the output indicates ordered factors. `X1` comes first, followed by `X2`, then `X3`.

3.3.1 Inputting Data Directly Into R: Contingency Tables

The procedure for creating data frames for contingency tables is somewhat different. These are created with `matrix()`. The raw data may be entered into R with the `c()` function. These data structures are similar to vectors, except they contain dimensions in rows and columns. The dimensions are specified with `nrow` and `ncol`. The `byrow` option instructs R to fill the table row-wise if set to `TRUE`, or column-wise if set to `FALSE`.

The following code generates a 2×3 contingency table named `Cont.Table` from the six data points contained in `c()` with two rows and three columns. This design is specified with `nrow=2` and `ncol=3`. `byrow` is set to `TRUE` so the data contained in `c()` fills row 1 first followed by row 2. The contents can be printed by typing in the name of the data frame.

```
>
> Cont.Table <- matrix(c(1, 2, 3, 4, 5, 6), nrow=2, ncol=3, byrow=TRUE)
>
> Cont.Table
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
>
```

Labels can be added to these tables with `colnames` and `rownames`. They require the name of the data frame and a list of labels:

```
>
> colnames(Cont.Table) <- c("Column 1","Column 2", "Column 3")
> rownames(Cont.Table) <- c("Row 1","Row 2")
>
> Cont.Table
      Column 1 Column 2 Column 3
Row 1        1        2        3
Row 2        4        5        6
>
```

3.3.2 Loading Data From an .rds file

R's native file format is `.rds`. This format preserves information such as headers, factors, etc. Files in this format are loaded with `readRDS()`. The following loads a file in `.rds` format called `Study.rds` and generates a data frame named `Study`:

```
Study <- readRDS("Study.rds")
```

To see information about a data frame, including factors, use `ls.str()`, and to list the contents, type in its name.

3.3.3 Loading Data from Text Files

R's Data Import/Export manual comments that the easiest files that aren't in `.rds` format to import are text files with delimiter separated values. The most common of these delimiters are commas, tabs, and colons. Comma delimited files are referred to as CSV files. This is the format that the data files in the following examples will use. A text file is also the preferred format for archival purposes.

The first step in loading an external file into R is to set the working directory that contains the data file. This folder is where R will look for data files and where it will write output. The current working directory can be obtained with `getwd()` and can be changed with `setwd()`.

This example changes the working directory from the default, which in Windows 10 is the user's 'Documents' directory, to a folder on the User 'Michael' desktop called 'R_Project':

```
>
> getwd()
[1] "C:/Users/Michael/Documents"
> setwd("C:/Users/Michael/Desktop/R_Project")
> getwd()
[1] "C:/Users/Michael/Desktop/R_Project"
>
```

Comma delimited text files can be loaded with `read.csv()`. The following imports a comma delimited data file in plain text format with column headings and creates a data frame called `Experiment`. This loads the Experiment data set in the Appendix:

```
>
> Experiment <- read.csv("Experiment.txt", header = TRUE, sep = ",")
> attach(Experiment)
> ls()
[1] "Experiment"
>
```

The `header = TRUE` option informs R that the file contains variable names, and `sep = ","` indicates that the data points and headers are separated by commas. The `attach()` command places the object into R's search path so that it can find it. A list of the data for each column may be obtained with `DataFrameName$ColumnName`:

```
>
> Experiment$Response
[1] 65 68 62 63 64 55 66 69 63 70 66 63 58 64 65 73 68 75 68 70 74 72 75
[24] 79 73 76 70 69 74 72 86 78 81 82 72 79 87 77 78 83 82 83 77 85 73 74
[47] 77 70 77 79 79 72 70 75 74 71 73 74 72 73 92 90 87 90 88 85 93 90 92
[70] 86 98 96 94 90 87 76 72 76 82 80 75 77 78 79 74 75 76 81 84 73
>
```

R may not preserve the names of the columns in imported data. To change them, the `colnames` attribute can be edited. It requires the name of the data frame and a list of names:

```
colnames(DataFrameName) <- c("NewName1", "NewName2" . . .)
```

3.3.4 Saving Data in `.rds` Format

R's native format for data files is `*.rds`. This format preserves data attributes, such as factors, column and row names. To save a data frame called to the current working directory in this format, use `saveRDS()`. This command requires the name of the data frame to be saved and the filename. The name of the data frame is `Study`, and is to be saved as `Study.rds`:

```
saveRDS(Study, file = "Study.rds")
```

3.3.5 Saving Data to a Text File

Data frames can be saved as comma delimited text files with `write.csv()`. This requires the name of the data frame to be saved and the name of the file. In the case of the Study data:

```
write.csv(Study, file = "Study.txt")
```

3.3.6 Adding Columns to existing Data Frames

Data columns can be added to existing data frames if required. This is done in a similar way to creating new vectors, except it requires the name of the existing data frame and the name of the new variable separated by a `$`. The data points are listed inside `c()`:

```
DataFrameName$NewColumnName <- c(New Data)
```

The following code adds a new variable named $X4$ to the Study data frame:

```
>
> Study$X4 <- c(6, 5, 6, 4, 5)
>
> ls.str(Study)
ID :  int [1:5] 1 2 3 4 5
X1 :  num [1:5] 5 3 7 6 4
X2 :  num [1:5] 8 3 4 3 5
X3 :  num [1:5] 6 4 6 1 6
X4 :  num [1:5] 6 5 6 4 5
>
```

This function is especially helpful when R generates lists of output relating to individual responses. These may include z-Scores or fitted values and residuals in multiple regression, etc. The following example generates the square root of each data column in the `Study` data frame:

```
>
> sqrt(Study$X1)
[1] 2.236068 1.732051 2.645751 2.449490 2.000000
> sqrt(Study$X2)
[1] 2.828427 1.732051 2.000000 1.732051 2.236068
> sqrt(Study$X3)
[1] 2.44949 2.00000 2.44949 1.00000 2.44949
>
```

R prints the output in the format above. It is easier to read if it is added as a new column. This requires the name of the data frame, the name of the new column and the function that will perform the calculations.

```
DataFrameName$NewColumnName <- Function()
```

The following code adds three new columns named `X1_SQRT`, `X2_SQRT` and `X3_SQRT` to '`Study`':

```
>
> Study$X1_SQRT <- sqrt(Study$X1)
> Study$X2_SQRT <- sqrt(Study$X2)
> Study$X3_SQRT <- sqrt(Study$X3)
>
> Study
  ID X1 X2 X3  X1_SQRT  X2_SQRT X3_SQRT
1  1  5  8  6 2.236068 2.828427 2.44949
2  2  3  3  4 1.732051 1.732051 2.00000
3  3  7  4  6 2.645751 2.000000 2.44949
4  4  6  3  1 2.449490 1.732051 1.00000
5  5  4  5  6 2.000000 2.236068 2.44949
>
```

3.3.7 Creating new data frames from existing ones

New data frames may be created that contain sub-sections of larger ones. This is useful when a command operates on an entire data frame and some columns are irrelevant. The `correlation()` command in the `correlation` package operates on entire data frames so columns containing Participant ID numbers, for instance, should not be present.

The following code creates a new data frame which consists of observations 1–20 and column numbers 2, 3 and 4 from a dataframe with six columns:

```
NewDataFrame <- OldDataFrame[c(1:20), c(2, 3, 4)]
```

When using this function to create a new data frame from sections of a larger one, square brackets are used and then the rows and columns that form the new data frame are specified with `[c(rows), c(columns)]`. The numbers of the rows and columns that are to be included in the new data frame are placed in the list formed by the `c()` commands. If the new data frame was to contain all the rows in columns 2, 3 and 4, a comma is inserted after the first square bracket to indicate all the columns are to be included. If a subsection of rows, say 10–20, but all the columns, are to make up the new data frame, the square brackets would contain [c(10:20),] with no column selection. The comma tells R that columns or rows are referred to:

```
NewDataFrame <- OldDataFrame[, c(2, 3, 4)]
```

3.3.8 Converting Data Frames Between Wide and Long Formats: Stacking and Unstacking

Data frames can be converted to or from wide and long format with `stack()` and `unstack()`. The `stack()` command converts a data frame in wide format to long format. It requires the name of the data frame and a list of columns to be included with `c()`. The following code creates a data frame in long format called Study.Long from the Study example above. It includes the data in columns `X1`, `X2` and `X3`.

```
>
> Study.Long <- stack(Study, c(X1, X2, X3))
>
> Study.Long
   values ind
1       5  X1
2       3  X1
3       7  X1
4       6  X1
5       4  X1
6       8  X2
7       3  X2
8       4  X2
```

```
9       3  X2
10      5  X2
11      6  X3
12      4  X3
13      6  X3
14      1  X3
15      6  X3
>
```

The `values` column contains responses and the `ind` column contains the factor levels, $X1$–$X3$. The columns may need to be renamed and the factors may need to be specified as factors following this procedure.

`unstack()` converts data frames from long to wide format. The following converts `Study.Long` from long to wide format. It doesn't contain any extra data columns so `unstack()` requires only the name of the data frame:

```
>
> Study.Wide <- unstack(Study.Long)
>
> Study.Wide
  X1 X2 X3
1  5  8  6
2  3  3  4
3  7  4  6
4  6  3  1
5  4  5  6
>
```

As with `stack()`, the columns may need to be renamed and new Participant ID numbers may need to be added.

4 Descriptive Statistics

Descriptive statistics are summaries of data sets and are common to most analyses. This section describes how to produce a range of commonly used descriptive and diagnostic statistics. This section is restricted to numerical summaries. For graphical methods of exploring data, see the section on graphs.

An extensive list of descriptive statistics is available through the `descriptives()` command in the `jmv` package. It includes a wide range of both numerical and graphical output. These are selected with `TRUE` or `FALSE` options. Each one is switched to `FALSE` by default, so each required statistic must be set to `TRUE` for R to produce them. This command can handle data frames in both wide and long format. A list of numerical statistics is in Table 3 and a list of graphs is in Table 4.

Statistic	Command
Confidence intervals for the mean	`ci`
Width of confidence intervals	`ciWidth (50 - 99.9, default: 95)`
Extreme values	`extreme`
Interquartile range	`iqr`
Kurtosis	`kurt`
Max	`maximum`
Mean	`mean`
Median	`median`
Min	`minimum`
Mode	`mode`
Sample size	`n`
Number of missing values	`missing`
Percentiles	`pc`
Percentile Values	`a comma-sepated list (default: 25,50,75) specifying the percentiles`
Quantiles	`pcEqGr`
Range	`range`
Skewness	`skew`
Standard deviation	`sd`
Standard error	`se`
Sum	`sum`
Shapiro-Wilk normality test	`sw`
variance	`variance`

Table 3: Statistics available in `jmv (descriptives)` command

Graph Type	Command
Frequency table	freq
Histogram	his
Density plot	dens
Bar plot	bar
Box plot	box
Violin plot	violin
Dot plot	dot
Q-Q plot	qq

Table 4: Graphs available in jmv (descriptives) command

The following example uses data from the `Skydiving` dataset in Appendix. It contains two variables in wide format: Atlantis and Lemuria. These are ratings of skydiving performance of their respective citizens. The first step sets the working directory, imports a comma delimited text file named Skydiving.txt, creates a data frame named `Skydiving`, and attaches it to R's search path:

```
> # Set working directory:
> getwd()
[1] "C:/Users/Michael/Documents"
> setwd("C:/Users/Michael/Desktop/R_Project")
> getwd()
[1] "C:/Users/Michael/Desktop/R_Project"
>
> # Import data and create a data frame called Skydiving_Wide:
Skydiving <- read.csv("Skydiving.txt", header = TRUE, sep = ",")
attach(Skydiving)
>
```

In the second step, package `jmv` is loaded and the `descriptives()` command produces descriptive statistics. The first part of this command nominates the data frame. For data frames in wide format, the `vars = vars()` option contains the list of variable columns to be included. Following this, the required statistics are nominated by switching them to `TRUE` or `FALSE` as required.

```
>
> library(jmv)
>
> descriptives(Skydiving,
+ vars = vars(Atlantis, Lemuria),
+ freq = TRUE, desc = "columns", n = TRUE,
+ missing = TRUE, mean = TRUE, median = TRUE, mode = TRUE,
+ sum = TRUE, sd = TRUE, variance = TRUE, range = TRUE,
+ min = TRUE, max = TRUE, se = TRUE, ci = TRUE,
+ ciWidth = 95, iqr = TRUE, skew = TRUE, kurt = TRUE,
```

```
+ sw = TRUE, pcEqGr = TRUE, pcNEqGr = 4, pc = TRUE,
+ pcValues = "25,50,75")

DESCRIPTIVES

Descriptives
-----------------------------------------------------
                           Atlantis      Lemuria
-----------------------------------------------------
N                                15            15
Missing                           0             0
Mean                        14.60000      10.13333
Std. error mean             0.6078847     0.6161839
95% CI mean lower bound      13.29622      8.811750
95% CI mean upper bound      15.90378      11.45492
Median                           15            10
Mode                        12.00000      9.000000
Sum                              219           152
Standard deviation          2.354327      2.386470
Variance                    5.542857      5.695238
IQR                         4.000000      3.000000
Range                             7             9
Minimum                          11             6
Maximum                          18            15
Skewness                   0.02576793     0.2551150
Std. error skewness         0.5801194     0.5801194
Kurtosis                   -1.409820     -0.1066973
Std. error kurtosis         1.120897      1.120897
Shapiro-Wilk W             0.9286255     0.9725677
Shapiro-Wilk p             0.2602435     0.8940912
25th percentile             12.50000      9.000000
50th percentile             15.00000      10.00000
75th percentile             16.50000      12.00000
-----------------------------------------------------
Note. The CI of the mean assumes sample means
follow a t-distribution with N - 1 degrees of
freedom

>
```

This command also handles data frames in long format through the formula interface. The formula option replaces `vars = vars()` with `formula = Response ~ Factor1 + Factor2 ...`, `data = DataFrameName`. This is placed at the end of the command options. The following example uses the data from `Skydiving.txt` in long format.

The first step is to import the data in long format and format the `Continent` column into a two level factor.

```
> # Set working directory:
> getwd()
[1] "C:/Users/Michael/Desktop/R_Project"
> setwd("C:/Users/Michael/Desktop/R_Project")
>
> # Import data and create a data frame called Skydiving_Long:
> Skydiving <- read.csv("Skydiving_Long.txt", header = TRUE, sep = ",")
> attach(Skydiving)
>
> factor (Continent, levels = c('Atlantis', 'Lemuria'), ordered = FALSE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[14] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[27] <NA> <NA> <NA> <NA>
Levels: Atlantis Lemuria
>
```

This dataframe has one column of responses named `Rating` and one column of factors named `Continent`. The two factors are `Altantis` and `Lemuria`. The formula in this instance is `Rating` $\sim$ `Continent`. It generates output in a slightly different format:

```
>
> # Descriptive statistics for Atlantis and Lemuria:
> library(jmv)
>
> descriptives(Skydiving,
+ freq = TRUE, desc = "columns", n = TRUE,
+ missing = TRUE, mean = TRUE, median = TRUE, mode = TRUE,
+ sum = TRUE, sd = TRUE, variance = TRUE, range = TRUE,
+ min = TRUE, max = TRUE, se = TRUE, ci = TRUE,
+ ciWidth = 95, iqr = TRUE, skew = TRUE, kurt = TRUE,
+ sw = TRUE, pcEqGr = TRUE, pcNEqGr = 4, pc = TRUE,
+ pcValues = "25,50,75",
+ formula = Rating ~ Continent)

DESCRIPTIVES

Descriptives
-----------------------------------------------------
                            Continent      Rating
-----------------------------------------------------
N                           Atlantis            15
                            Lemuria             15
Missing                     Atlantis             0
                            Lemuria              0
Mean                        Atlantis      14.60000
                            Lemuria       10.13333
Std. error mean             Atlantis     0.6078847
                            Lemuria      0.6161839
```

95% CI mean lower bound	Atlantis	13.29622
	Lemuria	8.811750
95% CI mean upper bound	Atlantis	15.90378
	Lemuria	11.45492
Median	Atlantis	15
	Lemuria	10
Mode	Atlantis	12.00000
	Lemuria	9.000000
Sum	Atlantis	219
	Lemuria	152
Standard deviation	Atlantis	2.354327
	Lemuria	2.386470
Variance	Atlantis	5.542857
	Lemuria	5.695238
IQR	Atlantis	4.000000
	Lemuria	3.000000
Range	Atlantis	7
	Lemuria	9
Minimum	Atlantis	11
	Lemuria	6
Maximum	Atlantis	18
	Lemuria	15
Skewness	Atlantis	0.02576793
	Lemuria	0.2551150
Std. error skewness	Atlantis	0.5801194
	Lemuria	0.5801194
Kurtosis	Atlantis	-1.409820
	Lemuria	-0.1066973
Std. error kurtosis	Atlantis	1.120897
	Lemuria	1.120897
Shapiro-Wilk W	Atlantis	0.9286255
	Lemuria	0.9725677
Shapiro-Wilk p	Atlantis	0.2602435
	Lemuria	0.8940912
25th percentile	Atlantis	12.50000
	Lemuria	9.000000
50th percentile	Atlantis	15.00000
	Lemuria	10.00000
75th percentile	Atlantis	16.50000
	Lemuria	12.00000

Note. The CI of the mean assumes sample means follow a t-distribution with N - 1 degrees of freedom

>

Outliers can have a significant effect on any analysis and a good way to check for them is to standardize the scores to z-Scores and check for values greater than 3.0. Values greater than this are more than three standard deviations from the mean and may qualify as outliers.

R's calculator function provides functions for means and standard deviations so z-scores can be calculated manually. This method may be used for data frames in wide format. The formula for z-scores is:

$$z = \frac{X - \mu}{\sigma}$$

This translates into a format R's calculator can use as:

```
(ColumnName - mean(ColumnName)) / sd(ColumnName)
```

R prints the results automatically:

```
>
> (Atlantis - mean(Atlantis)) / sd(Atlantis)
[1] -1.1043494  1.4441492 -0.6795996  0.1698999 -1.5290992  0.5946497
[7] -1.1043494 -0.2548499  1.0193995 -1.1043494 -0.6795996  1.0193995
[13]  0.1698999  0.5946497  1.4441492
>
>
> (Lemuria - mean(Lemuria)) / sd(Lemuria)
[1] -0.47489951 -1.31295746 -1.73198644  0.78218743  0.36315845 -0.05587053
[7] -0.47489951 -0.47489951  0.78218743  2.03927436 -0.47489951  1.20121640
[13] -0.89392849  0.78218743 -0.05587053
>
```

This output isn't particularly readable. It can be added to the Skydiving data frame by creating two new columns: `Atlantis_Z` and `Lemuria_Z`. This can be done manually and the results can be seen by typing in the name of the data frame and the new data column:

```
>
> Skydiving$Atlantis_Z <- (Atlantis - mean(Atlantis)) / sd(Atlantis)
>
> Skydiving$Lemuria_Z <- (Lemuria - mean(Lemuria)) / sd(Lemuria)
>
> Skydiving
   ID Atlantis Lemuria Atlantis_Z   Lemuria_Z
1   1       12       9 -1.1043494 -0.47489951
2   2       18       7  1.4441492 -1.31295746
3   3       13       6 -0.6795996 -1.73198644
4   4       15      12  0.1698999  0.78218743
5   5       11      11 -1.5290992  0.36315845
```

```
6   6       16      10  0.5946497 -0.05587053
7   7       12       9 -1.1043494 -0.47489951
8   8       14       9 -0.2548499 -0.47489951
9   9       17      12  1.0193995  0.78218743
10 10       12      15 -1.1043494  2.03927436
11 11       13       9 -0.6795996 -0.47489951
12 12       17      13  1.0193995  1.20121640
13 13       15       8  0.1698999 -0.89392849
14 14       16      12  0.5946497  0.78218743
15 15       18      10  1.4441492 -0.05587053
>
```

R's `ave()` command also computes z-Scores with the `FUN = scale` option. This command handles data frames in both wide and long format. For data in wide format, only the column name is specified:

```
>
> ave(Atlantis, FUN = scale)
[1] -1.1043494  1.4441492 -0.6795996  0.1698999 -1.5290992  0.5946497
[7] -1.1043494 -0.2548499  1.0193995 -1.1043494 -0.6795996  1.0193995
[13]  0.1698999  0.5946497  1.4441492
>
>
> ave(Lemuria, FUN = scale)
[1] -0.47489951 -1.31295746 -1.73198644  0.78218743  0.36315845 -0.05587053
[7] -0.47489951 -0.47489951  0.78218743  2.03927436 -0.47489951  1.20121640
[13] -0.89392849  0.78218743 -0.05587053
>
```

The `ave()` command doesn't require a formula to handle data in long format, only a list containing the response column followed by the factor columns. For a data frame with three factors:

```
ave(Response, Factor 1, Factor 2, Factor3, FUN = scale)
```

The `Skydiving` data frame in long format has a response column, `Rating`, and a single factor, `Continent`, with two levels: `Atlantis`, and `Lemuria`. The results can be added as an additional column to the data frame:

```
>
> Skydiving$ZScore <- ave(Rating, Continent, FUN = scale)
>
> Skydiving
   ID Rating Continent      ZScore
1   1     12  Atlantis -1.10434941
2   2     18  Atlantis  1.44414923
3   3     13  Atlantis -0.67959964
```

```
4   4     15  Atlantis  0.16989991
5   5     11  Atlantis -1.52909919
6   6     16  Atlantis  0.59464968
7   7     12  Atlantis -1.10434941
8   8     14  Atlantis -0.25484986
9   9     17  Atlantis  1.01939946
10 10     12  Atlantis -1.10434941
11 11     13  Atlantis -0.67959964
12 12     17  Atlantis  1.01939946
13 13     15  Atlantis  0.16989991
14 14     16  Atlantis  0.59464968
15 15     18  Atlantis  1.44414923
16 16      9   Lemuria -0.47489951
17 17      7   Lemuria -1.31295746
18 18      6   Lemuria -1.73198644
19 19     12   Lemuria  0.78218743
20 20     11   Lemuria  0.36315845
21 21     10   Lemuria -0.05587053
22 22      9   Lemuria -0.47489951
23 23      9   Lemuria -0.47489951
24 24     12   Lemuria  0.78218743
25 25     15   Lemuria  2.03927436
26 26      9   Lemuria -0.47489951
27 27     13   Lemuria  1.20121640
28 28      8   Lemuria -0.89392849
29 29     12   Lemuria  0.78218743
30 30     10   Lemuria -0.05587053
>
```

5 Tests for differences between two groups

5.1 *t*-test

The *t*-test is performed by `t.test()` in the `stats` package. This loads when R loads, so there is no need to load it seperately. The `t.test()` command provides options for single sample, Students and Welch's *t*-tests with independant and paired samples. The data for these examples is from the Placebo data set in Appendix. This is data set contains two groups: a placebo group and a treatment group.

To perform a single sample *t*-test, specify the variable of interest only.

```
>
> t.test(Placebo$Treatment)

One Sample t-test

data:  Placebo$Treatment
t = 19.974, df = 24, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
5.595217 6.884783
sample estimates:
mean of x
     6.24

>
```

The output provides the t statistic, degrees of freedom, p value, mean and 95% confidence interval of the mean.

To perform a test for differences between two means, both variables are specified, and the `paired` and `var.equal` options need to be set. Student's *t*-test assumes that the variances are equal, so to specify this test, set `var.equal = TRUE`. Welch's *t*-test doesn't assume this, so set this option to `FALSE`. If the sample is paired, and each participant gave data to both experimental conditions, set `paired = TRUE`. The following conducts Student's *t*-test with different participants in each group:

```
>
> t.test(Placebo$Placebo, Placebo$Treatment,
+ paired = FALSE, var.equal = TRUE)

Two Sample t-test

data:  Placebo$Placebo and Placebo$Treatment
t = -4.802, df = 48, p-value = 1.577e-05
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
-3.177913 -1.302087
sample estimates:
mean of x mean of y
     4.00      6.24

>
```

5.2 Wilcoxon Test

The nonparametric alternatives to the *t*-test are the Wilcoxon Rank Sum Test, and the Mann-Whitney Test. The Mann-Whitney test is for samples that are independant, and the Wilcoxon Signed Rank Test for samples that are paired. Both are available through `wilcox.test()` in the `stats` package. The same command performs both tests. The paired option specifies which test is to be performed: `paired = FALSE` for the Rank Sum Test, and `paired = TRUE` for the Signed Rank Test.

This command has options for a continuity correction and an exact p value turned on by default. The `exact` option calculates an exact p value if the sample size is less than 50 and there are no ties in the data. The continuity correction takes into account the use of a continuous distribution to approximate a discrete distribution. These may be turned off with `exact = FALSE` and `correct = FALSE`. Both of these options are turned off in the following examples.

This command accepts data for two groups in either wide or long format. If the data is in long format, a formula must be specified with the data column and the factor column: `Data` $\sim$ `Factor`. For data frames in wide format, only the variable names are specified.

If the experimental design has more than two cells and multiple comparisons are to be conducted, such as following a nonparametric ANOVA, `wilcox_test()` in the `rstatix` package is recommended. This will conduct all of them simultaneously.

The following example conducts a Wilcoxon Rank Sum Test / Mann-Whitney Test, for data in two groups from the Acheulean data set. They are fictitous preference ratings from *Homo Sapiens* for Neolithic stone tools made of either basalt or flint materials in Acheulean toolkits. The rating scale was a nominal scale and was scored as 5 = I love it, 4 = A nice piece of kit, 3 = Good enough, 2 = Myeah and 1 = I hate it.

The first step is to read into R a comma delimited text file in wide format with headers called `Acheulean.txt` from the working directory, create an R object called `StoneTools`, and place it in the search path so that R can find it:

```
>
> StoneTools <- read.csv("Acheulean.txt", header = TRUE, sep = ",")
> attach(StoneTools)
>
```

The `wilcox.test()` command requires the name of the R object that contains the data, the variable names, and if the data is paired or not. The name of the R object is StoneTools, the variable names are Basalt and Flint and for this example, the same persons rated each tool materials, so the paired option is set to `paired = TRUE`. Both `exact` and `correct` options are turned off. Because the data is paired, R performs a Wilcoxon Signed Rank test.

```
>
> wilcox.test(data = StoneTools, Basalt, Flint, exact = FALSE, correct = FALSE,
+ paired = TRUE)

        Wilcoxon signed rank test

data:  Basalt and Flint
V = 9, p-value = 0.000429
alternative hypothesis: true location shift is not equal to 0

>
```

If the test were performed on data with independant groups, the Mann-Whitney, or Wilcoxon Signed Rank Sum Test, is more appropriate. This is specified by setting the `paired` option to `paired = FALSE`:

```
>
> wilcox.test(data = StoneTools, Basalt, Flint, exact = FALSE, correct = FALSE,
+  paired = FALSE)

        Wilcoxon rank sum test

data:  Basalt and Flint
W = 126, p-value = 0.0001848
alternative hypothesis: true location shift is not equal to 0

>
```

5.3 *F* Test

The F test tests two samples for equal variances. It is performed with the `var.test()` command in the stats package. It only requires the names of the variables. The data for the following example is from the `Placebo` data set. This compares two groups, one that was given a treatment and one that was given a placebo. The results give an F test with degrees of freedom and a p value.

```
>
> var.test(Placebo$Placebo, Placebo$Treatment)

F test to compare two variances
```

```
data:  Placebo$Placebo and Placebo$Treatment
F = 1.2295, num df = 24, denom df = 24, p-value = 0.6168
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.5418061 2.7900950
sample estimates:
ratio of variances
          1.229508

>
```

6 Correlation

R's `correlation` package provides a wide variety of correlation methods and comprehensive output for correlations between single and multiple pairs of variables. It provides Pearson's r, Kendall's τ, Spearman's ρ, biserial, polychoric, tetrachoric, biweight, distance, Percentage Bend and Shepherd's π correlation.

The `correlation` package accepts data in wide format. If a correlation matrix is to be generated, the data frame should not contain any extranious variables, such as Participant ID numbers. This is because the `correlation` command generates correlations for all the columns in the data frame automatically.

R has extensive graphics capabilities and this includes generating scatterplots and scatterplot matrices. How to generate these is covered in the Graphs section.

6.1 Single Correlations

`cor_test()` provides correlations between pairs of variables. This command works with data in wide format and requires the name of the data frame, the names of the two variables and the required method with the `method` option. Common options are `method = "pearson"` for Pearson's r and `method = "spearman"` for Spearman's Rank correlation. This example uses the data file `MoneyLove.txt`. This data is listed in the appendix. It is in wide format, has a column of ID numbers, a column labelled `Salary`, and two columns labelled `Love` and `Happiness`.

The name of the data frame in this example is `MoneyLove`, the two variables are `Salary` and `Happiness`, and the method is `pearson` for Pearson's r. The results give both the variables, the correlation coefficients, a t-test, confidence intervals, method, and number of observations.

```
>
>
> MoneyLove <- read.csv("MoneyLove.txt",
+ header = TRUE, sep = ",")
> attach(MoneyLove)
>
> ls.str(MoneyLove)
Happiness :  int [1:20] 9 4 4 5 2 2 4 1 3 4 ...
ID :  int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
Love :  int [1:20] 47 27 38 44 8 17 30 21 35 37 ...
Salary :  int [1:20] 50 125 100 27 175 35 82 107 117 190 ...
>
> library(correlation)
>
>
> cor_test(MoneyLove, "Salary", "Happiness", method = "pearson")
Parameter1 | Parameter2 |     r |        95% CI | t(18) |     p
-------------------------------------------------------------
```

```
Salary     |  Happiness | -0.16 | [-0.56, 0.31] | -0.68 | 0.507

Observations: 20
>
```

A common nonparametric alternative to Pearson's r is Spearman's Rank correlation. This can be calculated with the same commands as the example for Pearson's r but specifying `method = "spearman"`. The following calculates Spearman's Rank correlation for two variables, `Tea Drinker` and `Gardiner` from a data frame named `Tea`. It contains three variables named `Tea Drinker`, `Gardiner`, and `Horse Rider`. The data is listed in the Appendix. The ratings are on a 1–5 scale where 1 = Never, 2 = Almost Never, 3 = Sometimes, 4 = Often and 5 = Always. These are ordinal scales so a non-parametric method is appropriate.

```
>
> getwd()
[1] "C:/Users/Michael/Documents"
> setwd("C:/Users/Michael/Desktop/R_Project")
>
> Tea <- read.csv("Tea.txt",
+ header = TRUE, sep = ",")
> attach(Tea)
>
> ls.str(Tea)
Gardiner :  int [1:15] 4 5 4 4 3 1 4 5 4 1 ...
Horse.Rider :  num [1:15] 2 2 3 1 2 1 2 2 2 4 ...
ID :  int [1:15] 1 2 3 4 5 6 7 8 9 10 ...
Tea.Drinker :  int [1:15] 5 4 5 3 2 1 3 4 5 2 ...
>
> library(correlation)
> cor_test(Tea, "Tea.Drinker", "Gardiner", method = "Spearman")
Parameter1  | Parameter2 |  rho |       95% CI |      S |      p
-----------------------------------------------------------------
Tea.Drinker |   Gardiner | 0.57 | [0.07, 0.84] | 238.84 | 0.025*

Observations: 15
>
```

The output gives the names of both variables as Parameters, Spearmans ρ, the S statistic, a p value, and the number of observations.

6.2 Correlation Matrices

`correlation()` produces correlation matrices. This command requires a data frame in wide format. It operates on entire data frames so it should contain only the variables of

interest.

The first example conducts multiple Pearson's r correlations with a data frame named `MoneyLove.2`. This was derived from the `MoneyLove.txt` data file and contains only the variables `Salary`, `Love`, and `Happiness`.

```
>
>
> MoneyLove <- read.csv("MoneyLove.txt",
+ header = TRUE, sep = ",")
> attach(MoneyLove)
>
> ls.str(MoneyLove)
Happiness :  int [1:20] 9 4 4 5 2 2 4 1 3 4 ...
ID :  int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
Love :  int [1:20] 47 27 38 44 8 17 30 21 35 37 ...
Salary :  int [1:20] 50 125 100 27 175 35 82 107 117 190 ...
>
> MoneyLove.2 <- (MoneyLove[, c(2,3,4)])
>
> ls.str(MoneyLove.2)
Happiness :  int [1:20] 9 4 4 5 2 2 4 1 3 4 ...
Love :  int [1:20] 47 27 38 44 8 17 30 21 35 37 ...
Salary :  int [1:20] 50 125 100 27 175 35 82 107 117 190 ...
>
```

`correlation()` creates an R object, in this case named `MoneyLove.2.cor`. Pearson's r coefficients are generated by specifying `method = "pearson"`. By default, this command adjusts the p values for the t-tests to take into account the number of correlations performed. The `p_adjust = "none"` option turns this off.

Listing the contents of MoneyLove.2.cor produces a list of correlations with r, a 95% Confidence Interval, a t-test and p values. `summary()` produces a correlation matrix.

```
>
> MoneyLove.2.cor <- correlation(MoneyLove.2, method = "pearson",
+ p_adjust = "none")
>
> MoneyLove.2.cor
# Correlation Matrix (pearson-method)

Parameter1 | Parameter2 |     r |        95% CI | t(18) |       p
-----------------------------------------------------------------
Salary     |       Love | -0.24 | [-0.62, 0.22] | -1.07 | 0.299
Salary     |  Happiness | -0.16 | [-0.56, 0.31] | -0.68 | 0.507
Love       |  Happiness |  0.65 | [ 0.29, 0.85] |  3.64 | 0.002**
```

```
p-value adjustment method: none
Observations: 20
>
>
> summary(MoneyLove.2.cor)
# Correlation Matrix (pearson-method)

Parameter | Happiness |  Love
-----------------------------
Salary    |     -0.16 | -0.24
Love      |    0.65** |

p-value adjustment method: none
>
```

Output with Spearman's Rank Sum correlations can be generated by changing the `method` command to `method = "spearman"`. This example uses the three data only columns of the Tea data. Firstly an R object containing this data is generated. It contains no `ID` column:

```
>
Tea.2 <- (Tea[, c(2,3,4)])
>

> ls.str(Tea.2)
Gardiner :  int [1:15] 4 5 4 4 3 1 4 5 4 1 ...
Horse.Rider :  num [1:15] 2 2 3 1 2 1 2 2 2 4 ...
Tea.Drinker :  int [1:15] 5 4 5 3 2 1 3 4 5 2 ...
>
```

Following this Spearman correlations are generated in an R object named `Tea.2.cor` with the `correlation()` command and a correlation matrix with `summary()`:

```
> Tea.2.cor <- correlation(Tea.2, method = "spearman", p_adjust = "none")
> Tea.2.cor
# Correlation Matrix (spearman-method)

Parameter1  |  Parameter2 |   rho |        95% CI |      S |      p
-------------------------------------------------------------------
Tea.Drinker |    Gardiner |  0.57 | [ 0.07, 0.84] | 238.84 | 0.025*
Tea.Drinker | Horse.Rider |  0.42 | [-0.13, 0.78] | 322.78 | 0.116
Gardiner    | Horse.Rider | -0.08 | [-0.58, 0.46] | 604.32 | 0.779

p-value adjustment method: none
Observations: 15
```

```
>
>summary(Tea.2.cor)
# Correlation Matrix (spearman-method)

Parameter   | Horse.Rider | Gardiner
-----------------------------------
Tea.Drinker |        0.42 |    0.57*
Gardiner    |       -0.08 |

p-value adjustment method: none
>
```

7 Comparing Multiple Groups: Analysis of Variance

Analysis of variance (ANOVA) can be performed with `aov()`. This is part of the `stats` package and is loaded automatically when R loads. `aov()` performs a Type I ANOVA. This assumes an equal number of respondants in each cell of the experimental design and no missing cells. R offers Type II and Type III ANOVA for designs that violate these assumptions. These may performed with `Anova()` in the `car` package. See the `car` documentation for more details.

To perform an ANOVA with `aov()`, the data frame should be formatted in long format with a column of Participant ID numbers, a column containing the responses, and column(s) for groups. The participant ID numbers are necessary for repeated measures designs as they are used to compute the error term.

In Between Groups designs, each participant contributes a single data point to the study. In within groups designs, each participant contributes a data point to each cell in the experimental design.

Table 5 illustrates data frame formats for both of these experimental designs. Both of these experimental designs have a single factor with three levels and six data points in each cell. In the between groups design, each participant contributed a single data point so the Participant ID's are numbered 1–18. In the Within Groups data frame, six participants contributed a data point to each of the three cells and so the ID numbering repeats 1–6 for each level of the factor.

R's `aov()` command uses a formula interface. One-way Between Groups designs use the model `y` $\sim$ `a` with `y` specifying the response variable and `a` specifying the factor. Repeated measures designs have an added error term. These models are specified as `y` $\sim$ `a + error(ID / a)`.

Setting R's contrast scheme for the analysis can help with more accurate results. This is set with the `contrast()` command which is also from the `stats` package. Table 6 gives a summary of contrasts available in R. Additional information on these is available through the `stats` package documentation. The default contrasts are `Treatment` for ordered factors and `Orthogonal Polynomials` for unordered factors. The current settings can be obtained with `options("contrasts")`.

```
>
> options("contrasts")
$contrasts
        unordered           ordered
"contr.treatment"      "contr.poly"

>
```

The contrasts can be set with `options`:

Between Groups			Within Groups		
ID,	Response,	Factor a	ID,	Response,	Factor a
1,	5,	X1	1,	5,	X1
2,	3,	X1	2,	3,	X1
3,	7,	X1	3,	7,	X1
4,	6,	X1	4,	6,	X1
5,	4,	X1	5,	4,	X1
6,	5,	X1	6,	5,	X1
7,	8,	X2	1,	8,	X2
8,	3,	X2	2,	3,	X2
9,	4,	X2	3,	4,	X2
10,	3,	X2	4,	3,	X2
11,	5,	X2	5,	5,	X2
12,	4,	X2	6,	4,	X2
13,	6,	X3	1,	6,	X3
14,	4,	X3	2,	4,	X3
15,	6,	X3	3,	6,	X3
16,	1,	X3	4,	1,	X3
17,	6,	X3	5,	6,	X3
18,	6,	X3	6,	6,	X3

Table 5: Sample data frames for One-Way ANOVA: Between Group and Within Group experimental designs

```
options(contrasts = c("UnorderedFactor", "OrderedFactor"))
```

7.1 One Way Analysis of Variance

The following scripts give examples of performing a one way ANOVA in R using `aov()`. The data frame should be in long format and the Participant ID's numbered depending on whether the experimental design is Between Groups or Within Groups.

The first step is to create an R object that contains a model of the data. This requires the name of the R object that contains the model, a formula and the name of the data frame:

```
ANOVAModel <- aov(Formula, data = DataFrameName)
```

The second step is to print the ANOVA Summary Table with `summary()`:

```
summary(ANOVAModel)
```

7.1.1 One Way ANOVA: Between Groups

The following performs a between groups ANOVA. Firstly, import a comma delimited text file containing the data and format it. This example uses the ID, Responses and Treatment columns of the `Experiment` text file:

Type	Command	Description
Simple	`contr.treatment`	Compares each mean to a reference mean. The reference mean is the first factor level.
Deviation	`contr.sum`	Compares each mean to the grand mean.
Orthoganal Polynomials	`contr.poly`	Orthoganal polynomials.
Helmert	`contr.helmert`	Contrasts factor level 2 with factor level 1, factor level 3 with the mean of the first two, etc.
User Defined		User defined contrasts.

Table 6: Contrast options in R.

```
>
> Experiment <- read.csv("Experiment.txt", header = TRUE, sep = ",")
> attach(Experiment)
>
> ls.str(Experiment)
Diet :  chr [1:90] " Apples" " Apples" " Apples" " Apples" " Apples" ...
ID :  int [1:90] 1 2 3 4 5 6 7 8 9 10 ...
Response :  int [1:90] 65 68 62 63 64 55 66 69 63 70 ...
Treatment :  chr [1:90] " Control" " Control" " Control" " Control" " Control" ...
>
```

R reads variable columns containing labels as unordered factors and places them in alphabetical order automatically. If the factor has a natural order, it may be specified with `factor()`. In this case, the `Treatment` factor has a natural order with `Control` being the first factor, followed by `Treatment 1` and `Treatment 2`. `factor()` requires the name of the data frame and then a list of levels contained in `c()`. If `ordered = TRUE`, R will put the factors in the listed order. The `levels` line of the output indicates the order: Control, then Treatment 1, then Treatment 2.

```
>
> factor(Treatment, levels = c('Control', 'Treatment 1', 'Treatment 2'),
+ ordered = TRUE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[15] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[29] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[43] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[57] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[71] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[85] <NA> <NA> <NA> <NA> <NA> <NA>
Levels: Control < Treatment 1 < Treatment 2
>
```

Contrasts are defined with `options()`. In this case, they are defined as deviation contrasts with `contr.sum` for both ordered and unordered factors. The contrast matrix is available with `ContrastType (NumberOfFactors)`. In this example, there are three factors:

```
>
> options(contrasts = c('contr.sum', 'contr.sum'))
>
> options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"

>
> contr.sum(3)
  [,1] [,2]
1    1    0
2    0    1
3   -1   -1
>
```

Next, an R object called `Experiment.aov` is created with the formula `aov(Response ~ Treatment)` from the `Experiment` data frame. The ANOVA summary table is obtained with `summary()`:

```
>
> Experiment.aov <- aov(Response ~ Treatment, data = Experiment)
>
> summary(Experiment.aov)
            Df Sum Sq Mean Sq F value   Pr(>F)
Treatment    2   3655  1827.7   48.77 6.22e-15 ***
Residuals   87   3260    37.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

7.1.2 One Way ANOVA: Repeated Measures

A similar procedure conducts a repeated measures ANOVA. The difference between the procedures is that repeated measures requires an error term. This involves the participant ID numbers. These are converted into factors with `factor()`. There was no particular order for the participants so they were converted to unordered factors. There are 30 Participants in each group. They are consecutively numbered so they may be specified with `1:30`:

```
> getwd()
[1] "C:/Users/Michael/Documents"
> setwd("C:/Users/Michael/Desktop/R_Project")
```

```
>
> Experiment <- read.csv("Experiment.txt", header = TRUE, sep = ",")
> attach(Experiment)
>
> ls.str(Experiment)
Diet :  chr [1:90] " Apples" " Apples" " Apples" " Apples" " Apples" ...
ID :  int [1:90] 1 2 3 4 5 6 7 8 9 10 ...
Response :  int [1:90] 65 68 62 63 64 55 66 69 63 70 ...
Treatment :  chr [1:90] " Control" " Control" " Control" " Control" ...
>
> factor(Treatment, levels = c('Control', 'Treatment 1', 'Treatment 2'),
+ ordered = TRUE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[15] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[29] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[43] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[57] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[71] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[85] <NA> <NA> <NA> <NA> <NA> <NA>
Levels: Control < Treatment 1 < Treatment 2
>
>
> factor(ID, levels = 1:30, ordered = FALSE)
[1] 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
[25] 25 26 27 28 29 30 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18
[49] 19 20 21 22 23 24 25 26 27 28 29 30 1  2  3  4  5  6  7  8  9  10 11 12
[73] 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
30 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 ... 30
>
```

Repeated measures ANOVA has an assumption of sphericity. This means that the variances of differences between all combinations of cells in the experimental design are equal. If this assumption is violated, it may lead to an increase in Type I error.

Maunchly's Test of Sphericity can be calculated with the `ezANOVA()` command in the `ez` package. This command creates an R object containing the results of Maunchly's Test. It requires the name of the data frame, the name of the dependant variable (`dv`), the within groups factor column (`Treatment`) and the ID numbers (`wid`). The results are obtained by entering the name of the R object, in this case `maunch`:

```
>
> library(ez)
>
> maunch <- ezANOVA(data = Experiment,
+ dv = (Response), within = (Treatment), wid = (ID))
Warning: Converting "ID" to factor for ANOVA.
Warning: Converting "Treatment" to factor for ANOVA.
```

```
>
> maunch
$ANOVA
     Effect DFn DFd        F            p p<.05       ges
2 Treatment   2  58 41.71655 5.935134e-12     * 0.5285588

$‘Mauchly’s Test for Sphericity‘
     Effect         W            p p<.05
2 Treatment 0.4429654 1.119895e-05     *

$‘Sphericity Corrections‘
     Effect       GGe        p[GG] p[GG]<.05       HFe        p[HF]
2 Treatment 0.6422465 1.833916e-08         * 0.6591038 1.254157e-08
  p[HF]<.05
2         *

>
```

The first line of the output is an ANOVA result with a Generalized η^2 (`ges`). This is an estimate of the amount of variability in the within groups factor.

The ‘Mauchly’s Test for Sphericity‘ section of the output gives Maunchly’s W along with a p value for each factor with repeated measures, in this example, `Treatment`. If the p value for Maunchly’s W is significant, as it is in this example, and the sphericity assumption is violated, the F statistic for the ANOVA may be positively biased.

The ‘Sphericity Corrections’ section gives a measure of by how much sphericity is violated. This is measured on a scale of 0–1 with lower numbers indicating a greater violation. R provides two measures of this: Greenhouse-Geisser ϵ (`GGe`), and Huynh-Feldt ϵ (`HFe`). If Huynh-Feldt ϵ is greater than 1, the value is reset to 1.

These same ϵ values may be applied as a correction to the degrees of freedom relating to the ANOVA’s F statistic. This will result in lower values for the degrees of freedom, and so a more conservative p value. The new degrees of freedom are calculated with:

$$df_{condition} = \epsilon(k-1)$$

$$df_{error} = \epsilon((k-1)(n-1))$$

`k` is the number of repeated measures and `n` is the number of subjects. These formulas are converted into a form R can use with:

ϵ `(Main Effect Degrees of Freedom - 1)`

Adjusted Degrees of Freedom for Error:

ϵ `((Main Effect Degrees of Freedom - 1)(Number of Participants - 1))`

R can calculate an adjusted p value relating to the F-statistic's adjusted degrees of freedom using the `pf()` function. This command requires the F statistic, both corrected degrees of freedom, and has an option to return the probability relating to the lower tail of the F distribution. This is set to `TRUE` by default. In this example, it is set to `FALSE`.

```
pf(F Statistic, Adjusted degrees of freedom for Condition,
Adjusted degrees of freedom for error, lower.tail = FALSE)
```

The F statistic becomes available following generation of the ANOVA Summary Table, so this procedure an example of this will be illustrated following that.

The ANOVA is conducted following Maunchly's Test:

```
>>
options(contrasts = c('contr.sum', 'contr.sum'))
>>
options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"
>>
contr.sum(3)
[,1] [,2]
1 1 0
2 0 1
3 -1 -1
>
>
> Experiment.rep <- aov(Response ~ Treatment + Error(ID/Treatment),
+ data = Experiment)
>
> summary(Experiment.rep)

Error: ID
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals  1  225.8   225.8

Error: ID:Treatment
          Df Sum Sq Mean Sq
Treatment  2   1249   624.7

Error: Within
          Df Sum Sq Mean Sq F value Pr(>F)
Treatment  2   3765  1882.7   94.39 <2e-16 ***
Residuals 84   1675    19.9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
```

If the sphericity assumption is violated, as it was in this example, R can calculate an adjusted p value relating to the F-test with `pf()`. Both the Greenhouse-Geisser and Huynh-Feldt ϵ values are similar. In this example, the Greenhouse-Geisser ϵ will be used to calculate the new F statistic.

```
>
> # Degrees of Freedom: Treatment
>
> .6422465 * (2 - 1)
[1] 0.6422465
>
>
> # Degrees of Freedom: Error
>
> .6422465 * ((2 - 1) * (90 - 1))
[1] 57.15994
>
>
```

These values are used to calculate an adjusted p value for the F value with `pf()`:

```
>
> pf(94.39, 0.6422465, 57.15994, lower.tail = FALSE)
[1] 5.99832e-11
>
>
```

This gives $p = 5.99832e - 11$ which is more conservative than that given by the ANOVA summary table.

7.1.3 Effect Size and Post Hoc Tests

Effect sizes are available through `eta_squared()` and `omega_squared()` in the `effectsize` package. It requires the name of the R object created by the `aov()` command and whether or not partials are required. For a one way ANOVA, this may be set to false. This example uses the repeated measures ANOVA object created above. For η^2:

```
>
> library(effectsize)
>
> eta_squared(Experiment.rep, partial = FALSE)
# Effect Size for ANOVA (Type I)

Group        | Parameter | Eta2 |       95% CI
-----------------------------------------------
ID:Treatment | Treatment | 0.18 | [    , 1.00]
Within       | Treatment | 0.54 | [0.42, 1.00]
```

```
- One-sided CIs: upper bound fixed at [1.00].Warning messages:
1: Argument ‘effectsize_type‘ is deprecated. Use ‘es_type‘ instead.
2: Some CIs could not be estimated due to non-finite F, df, or df_error
values.
>
>
```

For ω^2:

```
>
> omega_squared(Experiment.rep, partial = FALSE)
# Effect Size for ANOVA (Type I)

Group        | Parameter | Omega2 |       95% CI
-----------------------------------------------
ID:Treatment | Treatment |   0.17 | [    , 1.00]
Within       | Treatment |   0.52 | [0.40, 1.00]

- One-sided CIs: upper bound fixed at [1.00].Warning messages:
1: Argument ‘effectsize_type‘ is deprecated. Use ‘es_type‘ instead.
2: Some CIs could not be estimated due to non-finite F, df, or df_error
values.
>
```

For this repeated measures one-way ANOVA, the (`Within, Treatment`) stratum is the result of interest. This gives $\eta^2 = 0.54$ and a $\omega^2 = 0.52$.

If there is a significant main effect, the next step is to perform *post hoc* comparisons between the cell means.

post hoc tests, such as Tukey's HSD and Sheffe's Test, are provided by `PostHocTest()` in the `DescTools` package. This command operates on `aov()` objects and has a formula interface which is used to specify the required comparisons. It uses the format

```
PostHocTest(aov(formula, dataframe), method, confidence level)
```

`PostHocTest()` offers a range of tests. These are listed in Table 7. These are specified with the `method` option. The confidence level can be specified with `conf.level`.

The following code performs Tukey's HSD on the repeated measures ANOVA conducted above. The output gives the value, upper and lower confidence intervals and a p value for each comparison:

```
>
> library(DescTools)
>
> PostHocTest(aov(Response ~ Treatment, data = Experiment),
```

Comparison Type	Command
Fisher's lsd	`hsd`
Dunns t-test	`bonf`
Newman-Keuls	`newmankeuls`
Tukey's hsd	`hsd`
Scheffe's Test	`scheffe`

Table 7: *Post Hoc* comparisons available in DescTools `PostHocTest()`.

```
+ method = "hsd", conf.level = 0.95)

  Posthoc multiple comparisons of means : Tukey HSD
    95% family-wise confidence level

$Treatment
                              diff    lwr.ci   upr.ci    pval
Treatment 1- Control      8.800000  5.030982 12.56902 8.4e-07 ***
Treatment 2- Control     15.566667 11.797648 19.33568 3.1e-10 ***
Treatment 2- Treatment 1  6.766667  2.997648 10.53568 0.00014 ***

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

>
```

7.2 Multi-Way Analysis of Variance

Multi-way ANOVA models contain multiple factors. These models may use formulae that have crossings (*), additive (+), and interactive (:) components. A two-way ANOVA that is fully crossed may use the formula `y ~ a * b`. This produces an ANOVA summary table with main effects for factors `a` and `b` along with the interaction. Additional terms may be added to the model: `y ~ a * b * c` produces a three-way model. `y ~ a + b` will produce a summary table with main effects for factors `a` and `b` only, and `y ~ a : b` produces only the interaction.

As with one way ANOVA, multi-way designs that contain repeated measures require an error term for each within groups factor. This is done in the same way as one-way designs but with the addition of multiple factors: `y ~ a * b + error(ID / a * b)` is a design with responses from all participants appearing in all cells. The error term for mixed within/between designs require only the factor(s) that are within groups. For a design with two factors, `a` and `b`, but only factor `a` is repeated, only factor `a` is included in the error term: `y ~ a * b + error(ID / a)`. Some common formulae for multi-way ANOVA are in Table 8. `y` is the responses and `a` and `b` are factors. `ID` in the error term is participant ID number.

Experimental Design	Formula
Fully crossed between groups	`y ~ a * b`
Fully crossed repeated measures	`y ~ a * b + error(ID / a * b)`
Mixed within / between design	`y ~ a * b + error(ID / a)`
Main effects only	`y ~ a + b`
Interaction only	`y ~ a : b`

Table 8: Some common formulae for multi-way ANOVA

Table 9 illustrates data frame formatting for multi way ANOVA. The two examples illustrate a 2×3 experimental design for between groups and mixed within / between groups. The between groups data frame has 18 participants with each contributing a single data point, and each cell contains three data points. This data frame has ID numbering from 1–18.

The second example is a mixed within / between design with participants 1–3 contributing data to each cell in Factor a and participants 4–6 contributing data to each cell of Factor b.

Between Groups				Mixed Within / Between Groups			
ID,	Response,	Factor a,	Factor b,	ID,	Response,	Factor a,	Factor b
1,	5,	X1,	X1,	1,	5,	X1,	X1
2,	3,	X1,	X1,	2,	3,	X1,	X1
3,	7,	X1,	X1,	3,	7,	X1,	X1
4,	8,	X1,	X2,	1,	8,	X1,	X2
5,	3,	X1,	X2,	2,	3,	X1,	X2
6,	4,	X1,	X2,	3,	4,	X1,	X2
7,	6,	X1,	X3,	1,	6,	X1,	X3
8,	4,	X1,	X3,	2,	4,	X1,	X3
9,	6,	X1,	X3,	3,	6,	X1,	X3
10,	6,	X2,	X1,	4,	6,	X2,	X1
11,	4,	X2,	X1,	5,	4,	X2,	X1
12,	5,	X2,	X1,	6,	5,	X2,	X1
13,	3,	X2,	X2,	4,	3,	X2,	X2
14,	5,	X2,	X2,	5,	5,	X2,	X2
15,	4,	X2,	X2,	6,	4,	X2,	X2
16,	1,	X2,	X3,	4,	1,	X2,	X3
17,	6,	X2,	X3,	5,	6,	X2,	X3
18,	6,	X2,	X3,	6,	6,	X2,	X3

Table 9: Data frame formats for between groups and mixed within / between groups ANOVA

7.2.1 Multi-Way ANOVA: Between Groups

The data set for this example, `Experiment`, includes the factors `Treatment` and `Diet`. For this example, the participants experienced a single treatment and diet, and contributed a single data point to the entire study. For this study, the data points would be numbered 1–90 because there are 90 individuals contributing to the study. As with the one-way examples above, the first step in a multi-way ANOVA is to load the data, attach it to R's search path and format the data frame. This example has two factors: `Treatment` and `Diet`. Treatment is an ordered factor with three levels and Diet is an unordered factor with two levels.

```
>
> Experiment <- read.csv("Experiment.txt", header = TRUE, sep = ",")
> attach(Experiment)
>
> factor(Treatment, levels = c("Control", "Treatment 1", "Treatment 2"),
+ ordered = TRUE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[15] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[29] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[43] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[57] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[71] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[85] <NA> <NA> <NA> <NA> <NA> <NA>
Levels: Control < Treatment 1 < Treatment 2
>
> factor(Diet, levels = c("Apples", "Oranges"), ordered = FALSE)
[1] Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples
[10] Apples  Apples  Apples  Apples  Apples  Apples  Oranges Oranges Oranges
[19] Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges
[28] Oranges Oranges Oranges Apples  Apples  Apples  Apples  Apples  Apples
[37] Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples
[46] Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges
[55] Oranges Oranges Oranges Oranges Oranges Oranges Apples  Apples  Apples
[64] Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples
[73] Apples  Apples  Apples  Oranges Oranges Oranges Oranges Oranges Oranges
[82] Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges
Levels: Apples Oranges
>
```

The absence of < signs in the Levels section of the Diet factor indicates R has formatted these as unordered factors. The next step is to set the contrasts.

```
>
> options(contrasts = c('contr.sum', 'contr.sum'))
> options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"
```

```
>
```

The next commands create a model called `Experiment.2way` for a two way between groups ANOVA and gives the ANOVA summary table. This is a fully crossed design and gives significance tests for both main effects (`Treatment` and `Diet`) and their interaction:

```
>
> Experiment.2way <- aov(Response ~ Treatment * Diet, data = Experiment)
> summary(Experiment.2way)
               Df Sum Sq Mean Sq F value Pr(>F)
Treatment       2   3655  1827.7  139.42 <2e-16 ***
Diet            1    306   306.2   23.36  6e-06 ***
Treatment:Diet  2   1853   926.5   70.68 <2e-16 ***
Residuals      84   1101    13.1
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Effect sizes, including η^2 and ω^2 are provided by `eta_squared()` and `omega_squared()` in the `effects` package. It requires the name of the `aov()` model from the ANOVA. The partial option should be set to `TRUE` due to the presence of multiple main effects. This example calculates η^2. For ω^2, please use the `omega_squared()` command:

```
>
> library(effectsize)
>
> eta_squared(Experiment.2way, partial = TRUE)
# Effect Size for ANOVA (Type I)

Parameter      | Eta2 (partial) |       95% CI
-------------------------------------------------
Treatment      |           0.77 | [0.70, 1.00]
Diet           |           0.22 | [0.10, 1.00]
Treatment:Diet |           0.63 | [0.52, 1.00]

- One-sided CIs: upper bound fixed at [1.00].Warning message:
Argument ‘effectsize_type‘ is deprecated. Use ‘es_type‘ instead.
>
```

Post Hoc tests are provided by `PostHocTest()` in the `DescTools` package. This command operates on `aov()` objects. These may be individual main effects and interactions, or the entire experimental design. A list of available tests is in Table 7. These are specified with the `method` option.

This command uses the formula interface to specify which main effects and interactions are to be the subject of additional analysis. The following example tests all main effects and interactions. It generates tables with comparisons between all the levels of `Treatment`, under the `Treatment` heading, both levels of `Diet` under the `Diet` heading, and all cells in the experimental design under `Treatment:Diet`.

```
>
> library(DescTools)

Attaching package: 'DescTools'

The following object is masked from 'package:car':

Recode

> PostHocTest(aov(Response ~ Treatment * Diet, data = Experiment),
+ method = "hsd", conf.level = 0.95)

  Posthoc multiple comparisons of means : Tukey HSD
     95% family-wise confidence level

$Treatment
                              diff    lwr.ci   upr.ci    pval
 Treatment 1- Control      8.800000  6.569457 11.03054 1.6e-10 ***
 Treatment 2- Control     15.566667 13.336124 17.79721 1.6e-10 ***
 Treatment 2- Treatment 1  6.766667  4.536124  8.99721 7.5e-10 ***

$Diet
                      diff    lwr.ci    upr.ci  pval
 Oranges- Apples -3.688889 -5.206819 -2.170959 6e-06 ***

$`Treatment:Diet`
                                                 diff      lwr.ci
 Treatment 1: Apples- Control: Apples        16.133333  12.2773830
 Treatment 2: Apples- Control: Apples        26.466667  22.6107163
 Control: Oranges- Control: Apples            8.466667   4.6107163
 Treatment 1: Oranges- Control: Apples        9.933333   6.0773830
 Treatment 2: Oranges- Control: Apples       13.133333   9.2773830
 Treatment 2: Apples- Treatment 1: Apples    10.333333   6.4773830
 Control: Oranges- Treatment 1: Apples       -7.666667 -11.5226170
 Treatment 1: Oranges- Treatment 1: Apples   -6.200000 -10.0559504
 Treatment 2: Oranges- Treatment 1: Apples   -3.000000  -6.8559504
 Control: Oranges- Treatment 2: Apples      -18.000000 -21.8559504
 Treatment 1: Oranges- Treatment 2: Apples  -16.533333 -20.3892837
 Treatment 2: Oranges- Treatment 2: Apples  -13.333333 -17.1892837
 Treatment 1: Oranges- Control: Oranges       1.466667  -2.3892837
 Treatment 2: Oranges- Control: Oranges       4.666667   0.8107163
 Treatment 2: Oranges- Treatment 1: Oranges   3.200000  -0.6559504
                                                 upr.ci    pval
 Treatment 1: Apples- Control: Apples        19.9892837 1.6e-10 ***
 Treatment 2: Apples- Control: Apples        30.3226170 1.6e-10 ***
 Control: Oranges- Control: Apples           12.3226170 1.2e-07 ***
 Treatment 1: Oranges- Control: Apples       13.7892837 1.0e-09 ***
```

```
 Treatment 2: Oranges- Control: Apples       16.9892837 1.6e-10 ***
 Treatment 2: Apples- Treatment 1: Apples    14.1892837 3.7e-10 ***
 Control: Oranges- Treatment 1: Apples       -3.8107163 1.7e-06 ***
 Treatment 1: Oranges- Treatment 1: Apples   -2.3440496 0.00015 ***
 Treatment 2: Oranges- Treatment 1: Apples    0.8559504 0.21840
 Control: Oranges- Treatment 2: Apples      -14.1440496 1.6e-10 ***
 Treatment 1: Oranges- Treatment 2: Apples  -12.6773830 1.6e-10 ***
 Treatment 2: Oranges- Treatment 2: Apples   -9.4773830 1.6e-10 ***
 Treatment 1: Oranges- Control: Oranges       5.3226170 0.87622
 Treatment 2: Oranges- Control: Oranges       8.5226170 0.00860 **
 Treatment 2: Oranges- Treatment 1: Oranges   7.0559504 0.16110

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

>
```

7.2.2 Multi Way ANOVA: Mixed Within-Between

This example again uses the `Experiment` data set. In this example the participants contributed data to each treatment condition, but only to one of the diet conditions. This gives a mixed within / between design. There are 90 data points from 30 participants with 15 participants in each diet condition. They each contributed a data point to each of the three treatment conditions. The data is imported and formatted in the same way as the Between Groups ANOVA above.

```
>
> Experiment <- read.csv("Experiment.txt", header = TRUE, sep = ",")
> attach(Experiment)
>
>
> factor(Treatment, levels = c("Control", "Treatment 1", "Treatment 2"),
+ ordered = TRUE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[15] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[29] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[43] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[57] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[71] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[85] <NA> <NA> <NA> <NA> <NA> <NA>
Levels: Control < Treatment 1 < Treatment 2
>
> factor(Diet, levels = c("Apples", "Oranges"), ordered = FALSE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[15] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[29] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[43] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

```
[57] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[71] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[85] <NA> <NA> <NA> <NA> <NA> <NA>
Levels: Apples Oranges
>
> factor(ID, levels = c(1:30), ordered = FALSE)
[1] 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
[25] 25 26 27 28 29 30 1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18
[49] 19 20 21 22 23 24 25 26 27 28 29 30 1  2  3  4  5  6  7  8  9  10 11 12
[73] 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
30 Levels: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 ... 30
>
> options(contrasts = c('contr.sum', 'contr.sum'))
> options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"

>
```

`ezANOVA()` from the `ez` package checks the sphericity assumption. For a mixed within-between design, it is conducted only on the factor(s) that are repeated measures, in this case `Treatment`:

```
>
> library(ez)
>
> maunch.2wayMix <- ezANOVA(data = Experiment,
+ dv = (Response), within = (Treatment), wid = (ID))
Warning: Converting "ID" to factor for ANOVA.
Warning: Converting "Treatment" to factor for ANOVA.
>
> maunch.2wayMix
$ANOVA
     Effect DFn DFd        F            p p<.05       ges
2 Treatment   2  58 41.71655 5.935134e-12     * 0.5285588

$`Mauchly's Test for Sphericity`
     Effect         W            p p<.05
2 Treatment 0.4429654 1.119895e-05     *

$`Sphericity Corrections`
     Effect       GGe        p[GG] p[GG]<.05       HFe        p[HF]
2 Treatment 0.6422465 1.833916e-08         * 0.6591038 1.254157e-08
p[HF]<.05
2         *

>
```

Maunchly's Test for the Treatment factor is significant so the Degrees of Freedom and F values of the within groups `Treatment` factor may require adjustment following the ANOVA to correct the ANOVA summary table. The formulas, and an example of how to do this, is in the repeated measures example in the one way ANOVA section above. In essence, the Greenhouse-Geisser ϵ or Huynh-Feldt ϵ, `GGe` and `HFe` respectively, in the `Sphericity Corrections` section of the output, is used as a multiplier for the degrees of freedom for the within groups factor. A lower ϵ indicates a greater violation of the sphericity assumption, and this smaller multiplier will reduce the degrees of freedom. This correction will give a more conservative p value.

Next comes the ANOVA. This example is for a mixed within/between groups design with `Treatment` as a within groups factor and `Diet` as a between groups factor. `Treatment` is included in the error term:

```
>
> Experiment.2way <- aov(Response ~ Treatment * Diet + Error(ID / Treatment),
+  data = Experiment)
> summary(Experiment.2way)

Error: ID
     Df Sum Sq Mean Sq
Diet  1  225.8   225.8

Error: ID:Treatment
          Df Sum Sq Mean Sq
Treatment  2   1249   624.7

Error: Within
               Df Sum Sq Mean Sq F value   Pr(>F)
Treatment       2   3765  1882.7 142.841  < 2e-16 ***
Diet            1     80    80.4   6.101   0.0156 *
Treatment:Diet  2    527   263.7  20.011 8.67e-08 ***
Residuals      81   1068    13.2
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

Repeated measures ANOVA has a sphericity assumption. If Maunchly's W was significant, a correction may be applied to the within groups factor$s$, in this case `Treatment`, to obtain a corrected p value. As in the one way repeated measures ANOVA above, the Greenhouse-Geisser ϵ is the multiplier:

```
>
> # Degrees of Freedom: Treatment
>
.6422465 * (2 - 1)
[1] 0.6422465
```

```
>
>
>
> # Degrees of Freedom: Error
>
>
> .6422465 * ((2 - 1) * (90 - 1))
[1] 57.15994
>
> # Corrected p value for factor: Treatment
>
> pf(94.39, 0.6422465, 57.15994, lower.tail = FALSE)
[1] 5.99832e-11
>
```

Next is the amount of variance explained by each main effect and interaction. The `effectsize` package provides both η^2 with `eta_squared()` and ω^2 with `omega_squared()`. The main difference between multi way and one way ANOVA is that the `partial` option should be set to `TRUE` because there are multiple main effects. The following code provides Partial η^2 for each main effect and the interaction:

```
>
> library(effectsize)
>
> eta_squared(Experiment.2way, partial = TRUE)
# Effect Size for ANOVA (Type I)

Parameter      | Eta2 (partial) |       95% CI
-------------------------------------------------
Treatment      |           0.77 | [0.70, 1.00]
Diet           |           0.22 | [0.10, 1.00]
Treatment:Diet |           0.63 | [0.52, 1.00]

- One-sided CIs: upper bound fixed at [1.00].Warning message:
Argument ‘effectsize_type‘ is deprecated. Use ‘es_type‘ instead.
>
```

If there are significant main effects and/or interactions, the next step is *post hoc* comparisons between means. `DescTools` provides the `PosHocTest()` command which offers Tukey's HSD, Bonferroni, Fisher's LSD, and Sheffe contrasts. The available *posthoc* tests are listed in Table 7. Protected t-tests can be performed with `t.test()`. In the case of mixed within / between ANOVA, these may have to be conducted one at a time and are illustrated in the section on 'Tests for differences between two groups' above.

Post Hoc tests may be conducted with `PostHocTest()` in the `DescTools` package. The `Treatment : Diet` interaction was significant so *post hoc* tests were performed for the entire model. For Tukey's HSD:

```
>
>library(DescTools)
>
> PostHocTest(aov(Response ~ Treatment * Diet, data = Experiment),
+ method = "hsd", conf.level = 0.95)

   Posthoc multiple comparisons of means : Tukey HSD
     95% family-wise confidence level

$Treatment
                              diff    lwr.ci   upr.ci    pval
 Treatment 1- Control      8.800000  6.569457 11.03054 1.6e-10 ***
 Treatment 2- Control     15.566667 13.336124 17.79721 1.6e-10 ***
 Treatment 2- Treatment 1  6.766667  4.536124  8.99721 7.5e-10 ***

$Diet
                      diff    lwr.ci    upr.ci  pval
 Oranges- Apples -3.688889 -5.206819 -2.170959 6e-06 ***

$`Treatment:Diet`
                                                    diff       lwr.ci
 Treatment 1: Apples- Control: Apples         16.133333  12.2773830
 Treatment 2: Apples- Control: Apples         26.466667  22.6107163
 Control: Oranges- Control: Apples             8.466667   4.6107163
 Treatment 1: Oranges- Control: Apples         9.933333   6.0773830
 Treatment 2: Oranges- Control: Apples        13.133333   9.2773830
 Treatment 2: Apples- Treatment 1: Apples     10.333333   6.4773830
 Control: Oranges- Treatment 1: Apples        -7.666667 -11.5226170
 Treatment 1: Oranges- Treatment 1: Apples    -6.200000 -10.0559504
 Treatment 2: Oranges- Treatment 1: Apples    -3.000000  -6.8559504
 Control: Oranges- Treatment 2: Apples       -18.000000 -21.8559504
 Treatment 1: Oranges- Treatment 2: Apples   -16.533333 -20.3892837
 Treatment 2: Oranges- Treatment 2: Apples   -13.333333 -17.1892837
 Treatment 1: Oranges- Control: Oranges        1.466667  -2.3892837
 Treatment 2: Oranges- Control: Oranges        4.666667   0.8107163
 Treatment 2: Oranges- Treatment 1: Oranges    3.200000  -0.6559504
                                                   upr.ci    pval
 Treatment 1: Apples- Control: Apples         19.9892837 1.6e-10 ***
 Treatment 2: Apples- Control: Apples         30.3226170 1.6e-10 ***
 Control: Oranges- Control: Apples            12.3226170 1.2e-07 ***
 Treatment 1: Oranges- Control: Apples        13.7892837 1.0e-09 ***
 Treatment 2: Oranges- Control: Apples        16.9892837 1.6e-10 ***
 Treatment 2: Apples- Treatment 1: Apples     14.1892837 3.7e-10 ***
 Control: Oranges- Treatment 1: Apples        -3.8107163 1.7e-06 ***
 Treatment 1: Oranges- Treatment 1: Apples    -2.3440496 0.00015 ***
 Treatment 2: Oranges- Treatment 1: Apples     0.8559504 0.21840
 Control: Oranges- Treatment 2: Apples       -14.1440496 1.6e-10 ***
```

```
 Treatment 1: Oranges- Treatment 2: Apples  -12.6773830 1.6e-10 ***
 Treatment 2: Oranges- Treatment 2: Apples   -9.4773830 1.6e-10 ***
 Treatment 1: Oranges- Control: Oranges       5.3226170 0.87622
 Treatment 2: Oranges- Control: Oranges       8.5226170 0.00860 **
 Treatment 2: Oranges- Treatment 1: Oranges   7.0559504 0.16110

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

>
```

7.3 Non-Parametric Analysis of Variance

Nonparametric ANOVA is for experimental designs that don't meet the assumptions for the ANOVA's outlined above. The `rstatix` package provides the Kruskal-Wallis H test for between groups designs and Friedman's Test for repeated measures designs as well as measures of effect size and *post hoc* tests for differences between cells.

The following examples use the `rstatix` package. The data is provided by the `Pareidolia_BG and Pareidolia_WG` data files. These files contains fictitous estimates of whether or not respondants were able to see shapes or objects in clouds. There were three types of clouds ordered in height: High, Medium and Low, and each was measured on a five point scale: 1 = I can see true to life depictions, 2 = I often recognise shapes in them, 3 = I have to squint to see anything, 4 = No shapes spring to mind, 5 = The cloud cover almost formed a ganzfeld. This is ordinal data so a non-parametric method is appropriate.

7.3.1 Between Groups ANOVA: Kruskal-Wallis H Test

The following code creates a data frame named Pareidolia from a comma delimited text file with headers called `Pareidolia_BG.txt` in the current working directory and places it in the R search path with `attach()`. This is a Between Groups design with 15 participants who each contributed a single data point to the entire study, so the Participant ID numbering is from 1—15:

```
>
> Pareidolia_BG <- read.csv("Pareidolia_BG.txt", header = TRUE, sep = ",")
> attach(Pareidolia_BG)
>
> ls.str(Pareidolia_BG)
CloudHeight :  chr [1:15] "  High" "  High" "  High" "  High" "  High" ...
ID :  int [1:15] 1 2 3 4 5 6 7 8 9 10 ...
Rating :  int [1:15] 4 5 4 3 5 2 3 2 3 4 ...
>
```

This data frame contains a column of Participant ID numbers, a column of responses named `Rating` and a factor column named `CloudHeight`. This contains three levels:

High, Medium and Low. The next stage is to transform the levels of the `CloudHeight` column into a factor column with three levels. There is no natural order to these, so they are specified as unordered factors with `ordered = FALSE`:

```
>
> factor(CloudHeight, levels = c("High", "Medium", "Low"),
+ ordered = FALSE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[15] <NA>
Levels: High Medium Low
>
```

The `library()` command loads the `rstatix` package and the Kruskal-Wallis H test is performed with `kruskal_test()`. This uses a formula interface with the scores in the `Ratings` column on the left-hand side and the factor column `CloudHeight` on the right-hand side from the data frame `Pareidolia_BG`:

```
>
> library(rstatix)

Attaching package: ‘rstatix’

The following object is masked from ‘package:stats’:

filter

>
> kruskal_test(Rating ~ CloudHeight, data = Pareidolia_BG)
# A tibble: 1 × 6
  .y.        n statistic    df      p method
* <chr>  <int>     <dbl> <int>  <dbl> <chr>
1 Rating    15      7.70     2 0.0213 Kruskal-Wallis
>
```

Effect size is provided by `kruskal_effsize()`. It uses the same formula as `kruskal_test()` and calculates η^2.

```
>
> kruskal_effsize(Rating ~ CloudHeight, data = Pareidolia_BG)
# A tibble: 1 × 5
  .y.        n effsize method  magnitude
* <chr>  <int>   <dbl> <chr>   <ord>
1 Rating    15   0.475 eta2[H] large
>
```

Finally, the cells are compared with Wilcoxon's Test. This is a non-parametric t-test performed with `wilcox_test()`. It uses the same formula as `kruskal_test()`. It has a

paired option which is set to TRUE for repeated measures, or FALSE for between groups designs, as it is here.

```
>
> wilcox_test (data = Pareidolia_BG, Rating ~ CloudHeight, paired = FALSE)
# A tibble: 3 × 9
  .y.    group1   group2        n1    n2 statistic     p p.adj p.adj.signif
* <chr>  <chr>    <chr>      <int> <int>     <dbl> <dbl> <dbl> <chr>
1 Rating "  High" "  Low"        5     5        24 0.019 0.057 ns
2 Rating "  High" "  Medium"     5     5        22 0.052 0.105 ns
3 Rating "  Low"  "  Medium"     5     5         8 0.373 0.373 ns
>
```

7.3.2 Within Groups ANOVA: Friedman's Test

Friedman's Test is designed for experimental designs with repeated measures. This is available in the rstatix package with the command friedman_test(). It uses the same formula as the Kruskal-Wallis Test with the addition of an error term. This is supplied by the ID column. rstatix uses the logical operator | to indicate the column for the error term. This character is found on the keyboard just below the backspace key.

This example uses the same data as that for Kruskall-Wallis H test with the assumption that each of five participants contributed a rating for each of the three cloud levels. Both the ID numbers and the CloudHeight levels are converted to factors. The ID factor has five levels, and the CloudHeight factor three.

```
>
> Pareidolia_WG <- read.csv("Pareidolia_WG.txt", header = TRUE, sep = ",")
> attach(Pareidolia_WG)
>
> ls.str(Pareidolia_WG)
CloudHeight :  chr [1:15] "  High" "  High" "  High" "  High" "  High"  ...
ID :  int [1:15] 1 2 3 4 5 1 2 3 4 5 ...
Rating :  int [1:15] 4 5 4 3 5 2 3 2 3 4 ...
>
> factor(CloudHeight, levels = c("High", "Medium", "Low"),
+ ordered = FALSE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[15] <NA>
Levels: High Medium Low
>
> factor(ID, levels = c(1, 2, 3, 4, 5), ordered = FALSE)
[1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
Levels: 1 2 3 4 5
>
> library(rstatix)

Attaching package: ‘rstatix’
```

```
The following object is masked from 'package:stats':

filter

>
> friedman_test(Rating ~ CloudHeight | ID, data = Pareidolia_WG)
# A tibble: 1 × 6
  .y.        n statistic    df      p method
* <chr>  <int>     <dbl> <dbl>  <dbl> <chr>
1 Rating     5      7.43     2 0.0244 Friedman test
>
```

There is no measure of effect size available for Friedman's Test, but `rstatix` provides Kendall's W, the Coefficient of Concordance with `friedman_effsize()`. This command requires the same formula as `friedman_test()`:

```
>
> friedman_effsize(Rating ~ CloudHeight | ID, data = Pareidolia_WG)
# A tibble: 1 × 5
  .y.        n effsize method    magnitude
* <chr>  <int>   <dbl> <chr>     <ord>
1 Rating     5   0.743 Kendall W large
>
```

Finally, the Wilcoxon Signed Rank Test is available to find differences between groups. This is performed with the same command as for the Wilcoxon Rank Sum Test, or Mann-Whitney Test, but the `paired` option is set to `TRUE`:

```
>
> wilcox_test(Rating ~ CloudHeight, paired = TRUE, data = Pareidolia_WG)
# A tibble: 3 × 9
  .y.    group1   group2        n1    n2 statistic     p p.adj p.adj.signif
* <chr>  <chr>    <chr>      <int> <int>     <dbl> <dbl> <dbl> <chr>
1 Rating "  High" "  Low"        5     5        10 0.095 0.267 ns
2 Rating "  High" "  Medium"     5     5        10 0.089 0.267 ns
3 Rating "  Low"  "  Medium"     5     5         0 0.371 0.371 ns
>
```

8 Multiple Regression

Multiple regression may be performed in R with `lm()` from the `stats` package. This package is loaded when R loads so there is no need to load it manually. `lm()` requires a data frame in wide format. This command uses a formula interface with the dependant variable y on the left hand side and the predictors on the right. The model is specified as additive:

$$y \sim a + b + c \ldots$$

This section illustrates two type of regression, simultaneous and hierarchical. The examples in this section use the packages `correlation`, `car`, `psych`, `rockchalk` and their dependant packages. The examples in this section use the `Curse` data set.

This is a fictitous data set refering to the Curse of Knowledge. This is a cognitive bias that occurs when highly literate people assume that everyone has the same amount of knowledge on a given topic. This term was first described by Camerer, Loewenstein and Weber (1989). This example uses fictitous data from 20 participants. They were assessed on a 1–20 point scale for their level of the Curse of Knowledge, the number of undergraduate and postgraduate courses on their topic they had completed (1–10), a 1–5 point scale on level of expertise and a 1–12 point scale on the personality factor Conscientiousness.

There are three steps to performing a multiple regression in R.

1. Preliminary Data Screening

   Data screening for regression requires several more steps than most other statistical techniques. In addition to normality, missing data, outliers, etc, the data should also be screened for correlations between predictors, correlations between the dependant variable and the predictors, multicollinearity and singularity.

2. Generate the `lm()` model

   After the data is screened, an R object containing the regression model is created with `lm()`. After the model is generated, multiple and adjusted R^2 with an F test and β weights with significance tests can be obtained with `summary()`. An ANOVA summary table can be obtained with `anova()`.

3. Assess the model

   After the model is generated, the model's residuals, the difference between measured values and those predicted by the model, can be examined for normality, linearity, homoscedasticity, and independence. The model should also be screened for influential observations with Cook's Distance (D_i), and multivariate outliers with

Mahalanobis Distance (D^2). If issues relating to these are present, they should be resolved and the procedure repeated.

R can produce six diagnostic plots for lm() models. These are listed in Table 10:

Plot Type	Number
Residuals vs Fitted	1
Normal Q - Q plot	2
Scale - Location	3
Cooks Distance	4
Residuals vs Leverage	5
Cooks Distance vs Leverage	6

Table 10: Diagnostic plots for multiple regression

8.1 Simultaneous Regression

Simultaneous regression is a type of regression where all the terms of the model are entered into the equation at the same time.

As with all R projects, the first step is to nominate the working directory and load the data. This example uses the `Curse` dataset. Next, the data is loaded and a data frame named `SimReg`, referring to a simultaneous regression, is created. Several versions of this data frame will be created during the course of this analysis. Because they are part of a simultanious regression, they will all have the same `SimReg` prefix with an identifying suffix. `SimReg.Z` is an R object that contains z-scores:

```
>
> getwd()
[1] "C:/Users/Michael/Documents"
> setwd("C:/Users/Michael/Desktop/R_Project")
>
> SimReg <- read.csv("Curse.txt", header = TRUE, sep = ",")
> attach(SimReg)
>
> ls.str(SimReg)
Consc :  int [1:20] 6 9 7 6 6 6 9 10 10 6 ...
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
Expertise :  int [1:20] 2 3 3 3 4 3 4 2 2 2 ...
ID :  int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
>
```

The next step is to load the `jmv` package and obtain descriptive statistics with the `descriptives()` command:

```
>
> library(jmv)
>
> descriptives(SimReg,
+ vars = vars(Curse, Courses, Expertise, Consc),
+ freq = TRUE, desc = "columns", n = TRUE,
+ missing = TRUE, mean = TRUE, median = TRUE, mode = TRUE,
+ sum = TRUE, sd = TRUE, variance = TRUE, range = TRUE,
+ min = TRUE, max = TRUE, se = TRUE, ci = TRUE,
+ ciWidth = 95, iqr = TRUE, skew = TRUE, kurt = TRUE,
+ sw = TRUE, pcEqGr = TRUE, pcNEqGr = 4, pc = TRUE,
+ pcValues = "25, 50, 75")

DESCRIPTIVES

Descriptives
---------------------------------------------------------------------------------
                                Curse         Courses       Expertise     Consc
---------------------------------------------------------------------------------
N                                  20              20              20        20
Missing                             0               0               0         0
Mean                         9.800000        5.350000        2.900000  7.050000
Std. error mean              1.050313       0.4604060       0.1432701 0.3802700
95% CI mean lower bound      7.601669        4.386359        2.600132  6.254086
95% CI mean upper bound      11.99833        6.313641        3.199868  7.845914
Median                       9.500000        5.500000        3.000000  6.500000
Mode                         5.000000        4.000000        3.000000  6.000000
Sum                               196             107              58       141
Standard deviation           4.697144        2.058998       0.6407233 1.700619
Variance                     22.06316        4.239474       0.4105263 2.892105
IQR                          6.500000        3.000000       0.2500000 2.250000
Range                              17               8               2         5
Minimum                             2               1               2         5
Maximum                            19               9               4        10
Skewness                     0.3801509      -0.2842891      0.08003775 0.6985984
Std. error skewness          0.5121033       0.5121033       0.5121033 0.5121033
Kurtosis                    -0.5856797      -0.2156383      -0.2498356 -0.7890889
Std. error kurtosis          0.9923836       0.9923836       0.9923836 0.9923836
Shapiro-Wilk W               0.9697020       0.9740802       0.7875505 0.8616526
Shapiro-Wilk p               0.7485670       0.8375718       0.0005668 0.0084091
25th percentile              6.000000        4.000000        2.750000  6.000000
50th percentile              9.500000        5.500000        3.000000  6.500000
75th percentile              12.50000        7.000000        3.000000  8.250000
---------------------------------------------------------------------------------
Note. The CI of the mean assumes sample means follow a t-distribution with N
```

```
- 1 degrees of freedom

>
```

Next, z-Scores can be obtained to check for outliers. Because regressions generate a lot of diagnostics, a seperate data frame named `SimReg.Z` for the z-scores may be created, and the scores added:

```
>
> SimReg.Z <- (SimReg[, c(1,2,3,4,5)])
>
> SimReg.Z$Curse.Z <- (Curse - mean(Curse)) / sd(Curse)
> SimReg.Z$Courses.Z <- (Courses - mean(Courses)) / sd(Courses)
> SimReg.Z$Expertise.Z <- (Expertise - mean(Expertise)) / sd(Expertise)
> SimReg.Z$Consc.Z <- (Consc - mean(Consc)) / sd(Consc)
>
> SimReg.Z
   ID Curse Courses Expertise Consc     Curse.Z  Courses.Z Expertise.Z
1   1     5       3         2     6 -1.02189765 -1.1413317  -1.4046626
2   2     2       4         3     9 -1.66058368 -0.6556587   0.1560736
3   3    15       8         3     7  1.10705579  1.2870337   0.1560736
4   4    18       7         3     6  1.74574182  0.8013606   0.1560736
5   5    10       4         4     6  0.04257907 -0.6556587   1.7168098
6   6     8       6         3     6 -0.38321162  0.3156875   0.1560736
7   7     7       5         4     9 -0.59610696 -0.1699856   1.7168098
8   8    12       9         2    10  0.46836976  1.7727067  -1.4046626
9   9    11       6         2    10  0.25547441  0.3156875  -1.4046626
10 10     6       2         2     6 -0.80900231 -1.6270048  -1.4046626
11 11    15       5         3     5  1.10705579 -0.1699856   0.1560736
12 12     9       8         3     6 -0.17031628  1.2870337   0.1560736
13 13     4       6         4     8 -1.23479299  0.3156875   1.7168098
14 14    10       7         3     7  0.04257907  0.8013606   0.1560736
15 15    12       4         3     7  0.46836976 -0.6556587   0.1560736
16 16     8       4         3     7 -0.38321162 -0.6556587   0.1560736
17 17     6       1         3     5 -0.80900231 -2.1126779   0.1560736
18 18    14       5         3     6  0.89416044 -0.1699856   0.1560736
19 19     5       6         3     5 -1.02189765  0.3156875   0.1560736
20 20    19       7         2    10  1.95863716  0.8013606  -1.4046626
       Consc.Z
1  -0.61742221
2   1.14664126
3  -0.02940106
4  -0.61742221
5  -0.61742221
6  -0.61742221
7   1.14664126
8   1.73466241
9   1.73466241
```

```
10 -0.61742221
11 -1.20544337
12 -0.61742221
13  0.55862010
14 -0.02940106
15 -0.02940106
16 -0.02940106
17 -1.20544337
18 -0.61742221
19 -1.20544337
20  1.73466241
>
```

Next comes correlations between predictors and between the dependent variable and predictors, and squared multiple correlations.

A new data frame should be created that contains only the data columns of interest. This is because the `correlation()` command uses all the columns in the data frame. The new data frame will be named `SimReg.2`:

```
>
> SimReg.2 <- (SimReg[, c(2,3,4,5)])
>
> ls.str(SimReg.2)
Consc :  int [1:20] 6 9 7 6 6 6 9 10 10 6 ...
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
Expertise :  int [1:20] 2 3 3 3 4 3 4 2 2 2 ...
>
```

Next, the correlations are computed with the `correlation` command in the `correlation` package:

```
>
> library(correlation)
>
> SimReg.2.cor <- correlation(SimReg.2, method = "pearson", p_adjust = "none")
>
> SimReg.2.cor
# Correlation Matrix (pearson-method)

Parameter1 | Parameter2 |     r |        95% CI | t(18) |      p
-------------------------------------------------------------------
Curse      |    Courses |  0.49 | [ 0.06, 0.76] |  2.36 | 0.030*
Curse      |  Expertise | -0.22 | [-0.60, 0.25] | -0.94 | 0.358
Curse      |      Consc |  0.11 | [-0.35, 0.52] |  0.46 | 0.654
Courses    |  Expertise | -0.05 | [-0.48, 0.40] | -0.22 | 0.828
Courses    |      Consc |  0.39 | [-0.07, 0.71] |  1.77 | 0.093
```

```
Expertise  |      Consc | -0.24 | [-0.61, 0.23] | -1.03 | 0.315

p-value adjustment method: none
Observations: 20
>
>
> summary(SimReg.2.cor)
# Correlation Matrix (pearson-method)

Parameter | Consc | Expertise | Courses
---------------------------------------
Curse     |  0.11 |     -0.22 |   0.49*
Courses   |  0.39 |     -0.05 |
Expertise | -0.24 |           |

p-value adjustment method: none
>
```

Finally, squared multiple correlations are computed with `smc()` in the `psych` package:

```
>
> library(psych)

Attaching package: 'psych'

The following objects are masked from 'package:jmv':

    pca, reliability

>
> smc(SimReg.2)
    Curse   Courses Expertise     Consc
0.2921593 0.3666963 0.1174277 0.2163658
>
```

When data screening is complete, an R object containing the regression model is generated with `lm()`. A summary table is obtained with `summary()`. These commands are contained in the `stats` package. This is loaded automatically when R loads so there is no need to load this package separately. `Curse` is the response column and `Course`, `Expertise` and `Consc` are predictors. `summary()` prints the summary table:

```
>
> SimReg.lm <- lm(Curse ~ Courses + Expertise + Consc, data = SimReg.2)
>
> summary(SimReg.lm)

Call:
lm(formula = Curse ~ Courses + Expertise + Consc, data = SimReg.2)
```

```
Residuals:
    Min      1Q  Median      3Q     Max
-6.2882 -3.1175  0.0754  3.4101  6.9464

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.0414     7.0530   1.565   0.1370
Courses       1.2171     0.5205   2.338   0.0327 *
Expertise    -1.6511     1.5886  -1.039   0.3141
Consc        -0.4206     0.6478  -0.649   0.5254
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.306 on 16 degrees of freedom
Multiple R-squared:  0.2922,    Adjusted R-squared:  0.1594
F-statistic: 2.201 on 3 and 16 DF,  p-value: 0.1276


>
```

The `Call` section of the output contains the model formula and data frame name. The `Residuals` section is a summary of the difference between observed responses and the scores predicted by the model. Ideally the median should be 0 with a symmetrical distribution around it. The sign of the median indicates the direction of skew: + being to the right and − to the left. `1Q` and `3Q` gives the interquartile range and `min` and `max` give the minimum and maximum values.

The Coefficients section gives the intercept as well as the coefficients and t-tests for each predictor. These are measures of the relationships between the dependant variable and the predictors. The first line of the `Estimate` column contains the intercept. This is the y score of where the regression line crosses the x axis, or the mean of y when all the predictor scores are 0. This is followed by the β weights for each variable. The β weights give the slope which indicates the change in y for each unit increase in x.

The `Std.Error` column gives estimates of the standard deviation of the coefficient. If a predictor's standard error is relatively large compared to its β weight, it's contribution to the model may not differ from 0. The next columns contain a t statistic and a p value. The t value is the coefficient divided by its standard error and the p value gives its significance.

The final part of the summary table gives multiple R^2, adjusted R^2 and an F test. Multiple R^2 is the amount of variance in y explained by the predictors. R^2 increases as more predictors are added to the model. Adjusted R^2 takes this into account and increases only if predictors add more than they would by chance, and decreases if the predictor improves the model only by chance.

An ANOVA Summary Table is available with `anova(ModelName)`. The contrasts were changed to deviation with `cont.sum`:

```
>
> options(contrasts = c("contr.sum", "contr.sum"))
> options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"

>
> anova(SimReg.lm)
Analysis of Variance Table

Response: Curse
          Df  Sum Sq Mean Sq F value  Pr(>F)
Courses    1  99.222  99.222  5.3502 0.03435 *
Expertise  1  15.434  15.434  0.8322 0.37517
Consc      1   7.817   7.817  0.4215 0.52540
Residuals 16 296.727  18.545
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

R can create several diagnostic plots. These are listed in Table 10. For a Normal – Quantile Quantile plot, specify plot 2:

```
plot(SimReg.lm, which = 2)
```

This produces the plot in Figure 1.

When the model is created, it can be assessed for multicollinearity, the contribution of each variable to the model, outliers, and multivariate outliers.

Multicollinearity is the degree of correlation between the predictors. It can be assessed by means of the variance inflation factor (VIF). A VIF score of 1 is absence of multicollinearity and a score above 5 indicates it's presence. It can be computed with `vif()` in the `car` package. It requires the name of the regression model created with `lm()`:

```
>
> library(car)
Loading required package: carData

Attaching package: ‘car’

The following object is masked from ‘package:psych’:

    logit
```

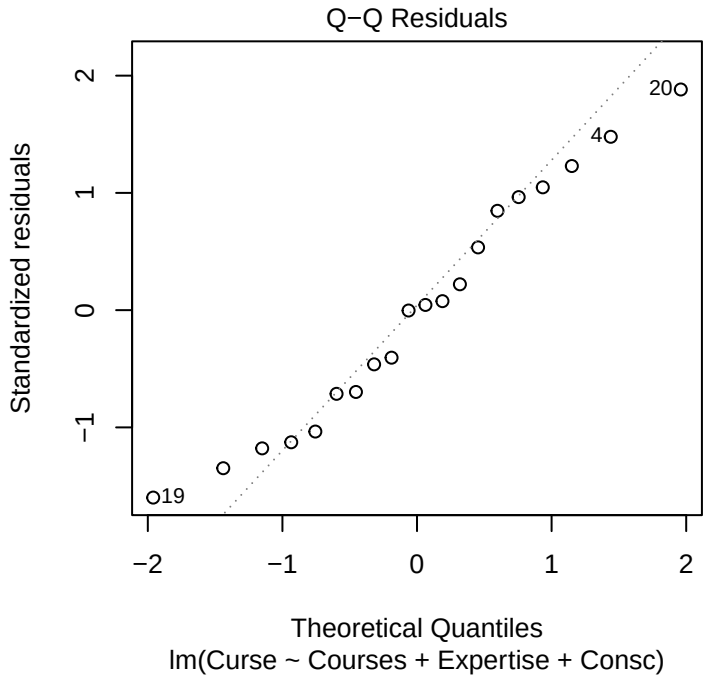

Figure 1: A Normal – QQ plot.

```
>
> vif(SimReg.lm)
 Courses Expertise     Consc
1.176866  1.061391  1.243350
>
```

The next step is to obtain the Squared Semi Partial Correlations. This is a measure of association between single predictors and the dependant variable. These are available through `getDeltaRsquare(ModelName)` in the `rockchalk` package:

```
>
> library(rockchalk)
>
> getDeltaRsquare(SimReg.lm)
The deltaR-square values: the change in the R-square
observed when a single term is removed.
Same as the square of the 'semi-partial correlation coefficient'
           deltaRsquare
Courses     0.24188156
Expertise   0.04779013
Consc       0.01864757
>
```

R calculates predicted y values for each participant from the regression model with `fitted.values()` in the `car` package. The difference between these values and the observed values are residuals. These are obtained with `residuals()`, which is also part of the `car` package. Both of them require the name of the regression model created with `lm()`.

These scores may be added to the data frame in additional columns. The following commands create a dataframe from the `SimReg` data frame that contains two new columns named `fittedlm` with the values predicted by the model, and `residualslm` with the residuals. The results can be seen by listing the data frame:

```
>
>
> SimReg.Res$fittedlm <- fitted.values(SimReg.lm)
> SimReg.Res$residualslm <- residuals(SimReg.lm)
>
>
> SimReg.Res
   ID Curse Courses Expertise Consc  fittedlm residualslm
1   1     5       3         2     6  8.867281 -3.86728067
2   2     2       4         3     9  7.171639 -5.17163941
3   3    15       8         3     7 12.881357  2.11864339
4   4    18       7         3     6 12.084777  5.91522299
5   5    10       4         4     6  6.782251  3.21774911
6   6     8       6         3     6 10.867631 -2.86763094
7   7     7       5         4     9  6.737698  0.26230245
8   8    12       9         2    10 14.487891 -2.48789119
9   9    11       6         2    10 10.836453  0.16354701
10 10     6       2         2     6  7.650135 -1.65013460
11 11    15       5         3     5 10.071051  4.92894866
12 12     9       8         3     6 13.301923 -4.30192307
13 13     4       6         4     8  8.375410 -4.37541008
14 14    10       7         3     7 11.664211 -1.66421054
15 15    12       4         3     7  8.012772  3.98722766
16 16     8       4         3     7  8.012772 -0.01277234
17 17     6       1         3     5  5.202467  0.79753292
18 18    14       5         3     6  9.650485  4.34951512
19 19     5       6         3     5 11.288197 -6.28819741
20 20    19       7         2    10 12.053599  6.94640094
>
```

Outliers are an important issue in any analysis. The `Car` package's `outlierTest()` identifies the model's most outlying response. This doesn't necessarily mean that this score qualifies as an outlier:

```
>
> outlierTest(SimReg.lm)
No Studentized residuals with Bonferroni p < 0.05
```

```
Largest |rstudent|:
   rstudent unadjusted p-value Bonferroni p
20 2.064975           0.056659           NA
>
```

Cook's Distance (D_i) detects influential observations. It is calculated with `influence.measures()` from the `car` package. It only requires the name of the R object that contains the regression model.

```
>
>
> influence.measures(SimReg.lm)
Influence measures of
         lm(formula = Curse ~ Courses + Expertise + Consc, data = SimReg.2) :

      dfb.1_  dfb.Crss  dfb.Expr  dfb.Cnsc     dffit cov.r   cook.d    hat inf
1  -5.12e-01  0.249735  0.426775  0.151657 -0.595550 1.304 8.82e-02 0.2477
2   2.64e-01  0.432115 -0.175926 -0.567726 -0.707206 1.007 1.18e-01 0.2065
3  -1.77e-02  0.182867  0.011968 -0.069415  0.223659 1.425 1.31e-02 0.1547
4   1.34e-01  0.428837 -0.014952 -0.374793  0.613691 0.833 8.67e-02 0.1369
5  -1.60e-01 -0.114424  0.357604  0.000240  0.448427 1.385 5.12e-02 0.2221
6  -6.81e-02 -0.098982  0.002663  0.129017 -0.214472 1.256 1.19e-02 0.0890
7  -4.44e-02 -0.016183  0.044590  0.037892  0.056871 2.050 8.62e-04 0.3695   *
8   4.13e-02 -0.247314  0.214315 -0.170001 -0.509385 1.736 6.70e-02 0.3447
9  -2.57e-04 -0.003904 -0.011633  0.016886  0.026132 1.771 1.82e-04 0.2695   *
10 -2.40e-01  0.179129  0.190960  0.043611 -0.303964 1.785 2.43e-02 0.3128   *
11  2.78e-01  0.100339 -0.045356 -0.380682  0.489832 1.004 5.79e-02 0.1330
12 -8.53e-02 -0.486689  0.018375  0.349501 -0.593409 1.183 8.65e-02 0.2141
13  5.42e-01 -0.007990 -0.604063 -0.288361 -0.702589 1.212 1.20e-01 0.2571
14  7.04e-03 -0.083101 -0.010913  0.031288 -0.125287 1.367 4.14e-03 0.0916
15  2.55e-02 -0.162117  0.042344  0.064011  0.278748 1.105 1.95e-02 0.0775
16 -7.92e-05  0.000504 -0.000132 -0.000199 -0.000867 1.403 2.00e-07 0.0775
17  6.70e-02 -0.104062 -0.003188 -0.024799  0.138150 1.813 5.07e-03 0.2945   *
18  1.19e-01  0.018349  0.001750 -0.144960  0.289106 1.048 2.08e-02 0.0704
19 -3.56e-01 -0.361536  0.072239  0.609876 -0.752394 0.774 1.27e-01 0.1657
20 -4.64e-02  0.104363 -0.569555  0.696140  1.241024 0.648 3.20e-01 0.2653
>
```

Mahalanobis Distance (D^2) detects multivariate outliers. It can be computed with `mahalanobis()` in the `stats` package. This command includes *all* the columns in the data frame so a new data frame with no extranious data columns is required.

The following code creates a new data frame with no `ID` column named `SimReg.Mahal` and adds two new columns with the Mahalanobis distance and a p value with the `mahalanobis()` and `pchisq()` commands.

It requires the name of the data frame, the means of each column with `colMeans()` and its variance-covariance matrix `cov()`. The results are added to the data frame `SimReg.Mahal` as a new column named `Mahalanobis`.

The next step is to calculate the significance of the (D^2) values. This is done with `pchisq()` which will calculate a p value for each observation. It requires the name of the column with the calculated (D^2) values, the degrees of freedom and has a lower tail option. The degrees of freedom is equal to the number of predictors. In this case there are three. The `lower.tail` option is set to `TRUE` by default. In this example, it is set to `FALSE`.

```
pchisq(MahalanobisDistanceScores, DegreesofFreedom, LowerTail)
```

The output from this command can be added as an additional column to the data frame. In this case, it was added to the SimReg.Mahal data frame with the title `p`. Using a conservative criterion of $p < .001$, there are no multivariate outliers in this sample.

```
>
> SimReg.Mahal <- (SimReg[, c(2,3,4,5)])
>
> SimReg.Mahal$Mahalanobis <- mahalanobis(SimReg.Mahal,
+ colMeans(SimReg.Mahal), cov(SimReg.Mahal))
>
> SimReg.Mahal$p <- pchisq(SimReg.Mahal$Mahalanobis, df = 3, lower.tail = FALSE)
>
> SimReg.Mahal
   Curse Courses Expertise Consc Mahalanobis          p
1      5       3         2     6   4.7141455 0.19396604
2      2       4         3     9   4.6855527 0.19632463
3     15       8         3     7   2.2767836 0.51698313
4     18       7         3     6   3.8922049 0.27334178
5     10       4         4     6   3.9323407 0.26886340
6      8       6         3     6   1.2681748 0.73670379
7      7       5         4     9   6.0751909 0.10800878
8     12       9         2    10   5.9962454 0.11179304
9     11       6         2    10   4.1719251 0.24348779
10     6       2         2     6   5.1670677 0.15996450
11    15       5         3     5   3.1324397 0.37165288
12     9       8         3     6   4.3020550 0.23064080
13     4       6         4     8   5.1603090 0.16042792
14    10       7         3     7   0.9671961 0.80918883
15    12       4         3     7   1.5404582 0.67296456
16     8       4         3     7   0.5224896 0.91392348
17     6       1         3     5   4.6866154 0.19623649
18    14       5         3     6   1.5980741 0.65982659
19     5       6         3     5   4.7294451 0.19271486
20    19       7         2    10   7.1812862 0.06633863
>
```

8.2 Hierarchical Regression

Hierarchical regression is aimed at estimating the contribution of individual variables to a regression model. This is done by creating a regression model with one predictor, then a second model with two predictors, a third with three predictors, etc and observing the change in R^2 for each one.

The code and output presented in this section is somewhat bare bones. Please refer to the section on simultaneous regression for more details if required.

The data for this example is the same 'Curse of Knowledge' data that was used in the simultaneous regression example above. Because this example is for hierarchical regression, the data frames will have the prefix `HierReg`.

The data file for this example, `Curse`, has three predictors, so three regression models are required. To distinguish between them, those belonging to the first regression will be named `HierReg.1.xx`, the second `HierReg.2.xx` and the third `HierReg.3.xx` The first of these will list z-Scores for the first regression. It is named `Reg.1.Z`.

The first step is to import the data, obtain descriptive statistics and generate z-Scores for the model with the first predictor which is 'Courses'.

```
>
>
> getwd()
[1] "C:/Users/Michael/Documents"
> setwd("C:/Users/Michael/Desktop/R_Project")
>
> HierReg.1 <- read.csv("Curse.txt", header = TRUE, sep = ",")
> attach(HierReg.1)
>
> ls.str(HierReg.1)
Consc :  int [1:20] 6 9 7 6 6 6 9 10 10 6 ...
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
Expertise :  int [1:20] 2 3 3 3 4 3 4 2 2 2 ...
ID :  int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
>
> library(jmv)
>
> descriptives(HierReg.1,
+ vars = vars(Curse, Courses),
+ freq = TRUE, desc = "columns", n = TRUE,
+ missing = TRUE, mean = TRUE, median = TRUE, mode = TRUE,
+ sum = TRUE, sd = TRUE, variance = TRUE, range = TRUE,
+ min = TRUE, max = TRUE, se = TRUE, ci = TRUE,
+ ciWidth = 95, iqr = TRUE, skew = TRUE, kurt = TRUE,
+ sw = TRUE, pcEqGr = TRUE, pcNEqGr = 4, pc = TRUE,
+ pcValues = "25, 50, 75")
```

```
DESCRIPTIVES

Descriptives
-------------------------------------------------------
                              Curse         Courses
-------------------------------------------------------
N                                 20            20
Missing                            0             0
Mean                        9.800000      5.350000
Std. error mean             1.050313     0.4604060
95% CI mean lower bound     7.601669      4.386359
95% CI mean upper bound     11.99833      6.313641
Median                      9.500000      5.500000
Mode                        5.000000      4.000000
Sum                              196           107
Standard deviation          4.697144      2.058998
Variance                    22.06316      4.239474
IQR                         6.500000      3.000000
Range                             17             8
Minimum                            2             1
Maximum                           19             9
Skewness                   0.3801509    -0.2842891
Std. error skewness        0.5121033     0.5121033
Kurtosis                  -0.5856797    -0.2156383
Std. error kurtosis        0.9923836     0.9923836
Shapiro-Wilk W             0.9697020     0.9740802
Shapiro-Wilk p             0.7485670     0.8375718
25th percentile             6.000000      4.000000
50th percentile             9.500000      5.500000
75th percentile             12.50000      7.000000
-------------------------------------------------------
Note. The CI of the mean assumes sample means
follow a t-distribution with N - 1 degrees of
freedom

>
>
> HierReg.1.Z <- (HierReg.1[, c(1,2,3)])
>
> HierReg.1.Z$Curse.Z <- (Curse - mean(Curse)) / sd(Curse)
> HierReg.1.Z$Courses.Z <- (Courses - mean(Courses)) / sd(Courses)
>
>
> HierReg.1.Z
   ID Curse Courses     Curse.Z  Courses.Z
1   1     5       3 -1.02189765 -1.1413317
```

```
2   2     2       4 -1.66058368 -0.6556587
3   3    15       8  1.10705579  1.2870337
4   4    18       7  1.74574182  0.8013606
5   5    10       4  0.04257907 -0.6556587
6   6     8       6 -0.38321162  0.3156875
7   7     7       5 -0.59610696 -0.1699856
8   8    12       9  0.46836976  1.7727067
9   9    11       6  0.25547441  0.3156875
10 10     6       2 -0.80900231 -1.6270048
11 11    15       5  1.10705579 -0.1699856
12 12     9       8 -0.17031628  1.2870337
13 13     4       6 -1.23479299  0.3156875
14 14    10       7  0.04257907  0.8013606
15 15    12       4  0.46836976 -0.6556587
16 16     8       4 -0.38321162 -0.6556587
17 17     6       1 -0.80900231 -2.1126779
18 18    14       5  0.89416044 -0.1699856
19 19     5       6 -1.02189765  0.3156875
20 20    19       7  1.95863716  0.8013606
>
>
```

The next step is to obtain correlations and squared multiple correlations between predictor and response variables. Correlations may be generated by the `correlation()` command from the `correlation` package. Squared multiple correlations are obtained through the `smc()` command in the `psych` package:

```
>
> HierReg.1.2 <- (HierReg.1[, c(2,3)])
>
> ls.str(HierReg.1.2)
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
>
>
> library(correlation)
>
> HierReg.1.cor <- correlation(HierReg.1.2, method = "pearson", p_adjust = "none")
>
> HierReg.1.cor
# Correlation Matrix (pearson-method)

Parameter1 | Parameter2 |    r |       95% CI | t(18) |      p
-------------------------------------------------------------
Curse      |    Courses | 0.49 | [0.06, 0.76] |  2.36 | 0.030*

p-value adjustment method: none
```

```
Observations: 20
>
> summary(HierReg.1.cor)
# Correlation Matrix (pearson-method)

Parameter | Courses
-------------------
Curse     |   0.49*

p-value adjustment method: none
>
>
> library(psych)

Attaching package: 'psych'

The following objects are masked from 'package:jmv':

pca, reliability

>
> smc(HierReg.1.2)
    Curse   Courses
0.2366945 0.2366945
>
>
```

Next the first regression model is generated with `lm()` and `anova()` produces an ANOVA Summary Table. This model contains only the first predictor, `X1`:

```
>
> HierReg.1_lm <- lm(Curse ~ Courses, data = HierReg.1)
>
> summary(HierReg.1_lm)

Call:
lm(formula = Curse ~ Courses, data = HierReg.1)

Residuals:
    Min      1Q  Median      3Q     Max
-6.5214 -2.4390 -0.1918  2.6187  7.3687

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.8622     2.6843   1.439   0.1674
Courses       1.1099     0.4698   2.363   0.0296 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 4.216 on 18 degrees of freedom
Multiple R-squared:  0.2367,    Adjusted R-squared:  0.1943
F-statistic: 5.582 on 1 and 18 DF,  p-value: 0.02961

>
>
> options(contrasts = c("contr.sum", "contr.sum"))
> options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"

>
> anova(HierReg.1_lm)
Analysis of Variance Table

Response: Curse
          Df Sum Sq Mean Sq F value  Pr(>F)
Courses    1  99.22  99.222  5.5816 0.02961 *
Residuals 18 319.98  17.777
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
>
```

The first section of output for the regression summary contains the model formula and data frame name followed by residuals. The residuals section is a summary of the difference between observed responses and the scores predicted by the model. Ideally the median should be 0 with a symmetrical distribution around it. The sign of the median indicates the direction of skew: + being to the right and − to the left.

The next section gives the coefficients. The first line gives the intercept. This is the y score of where the regression line crosses the x axis, or the mean of y when all the predictor scores are 0. This is followed by the β weights for each variable. The β weights give the slope which indicates the change in y for each unit increase in x. Next comes the standard error. This is an estimate of the standard deviation of the coefficient. If a predictor's standard error is relatively large compared to its β weight, it's contribution to the model may not differ from 0. The final columns contain a t statistic and a p value. The t value is the coefficient divided by its standard error and the p value gives its significance.

Finally comes multiple R^2, adjusted R^2 and an F test. Multiple R^2 is the amount of variance in y explained by the predictors. This value increases as more predictors are added. Adjusted R^2 takes this into account and increases only if predictors add more than they would by chance, and decreases if the predictor improves the model only by chance.

An ANOVA Summary Table is available with `anova(ModelName)`. This produces an F test for each predictor. Contrasts may be set with `options(Contrasts)`

Multicollinearity is assessed by means of Variance Inflation Factor. It is not calculated for this regression as there is only a single variable. Next are squared semi-partial correlations. These are calculated with `getDeltaRsquare()` from the `rockchalk` package. Fitted values and residuals may be calculated with `fitted.values()` and `residuals()`. A new data frame named Reg.1.Res is created for these values and they are added as additional columns:

```
>
> library(rockchalk)
>
> getDeltaRsquare(HierReg.1_lm)
The deltaR-square values: the change in the R-square
observed when a single term is removed.
Same as the square of the 'semi-partial correlation coefficient'
        deltaRsquare
Courses    0.2366945
>
>
> HierReg.1.Res <- (HierReg.1[, c(1,2,3)])
>
> HierReg.1.Res$fittedlm <- fitted.values(HierReg.1.lm)
> HierReg.1.Res$residualslm <- residuals(HierReg.1.lm)
>
>
> HierReg.1.Res
   ID Curse Courses  fittedlm residualslm
1   1     5       3  7.191806 -2.19180633
2   2     2       4  8.301676 -6.30167598
3   3    15       8 12.741155  2.25884544
4   4    18       7 11.631285  6.36871508
5   5    10       4  8.301676  1.69832402
6   6     8       6 10.521415 -2.52141527
7   7     7       5  9.411546 -2.41154562
8   8    12       9 13.851024 -1.85102421
9   9    11       6 10.521415  0.47858473
10 10     6       2  6.081937 -0.08193669
11 11    15       5  9.411546  5.58845438
12 12     9       8 12.741155 -3.74115456
13 13     4       6 10.521415 -6.52141527
14 14    10       7 11.631285 -1.63128492
15 15    12       4  8.301676  3.69832402
16 16     8       4  8.301676 -0.30167598
17 17     6       1  4.972067  1.02793296
18 18    14       5  9.411546  4.58845438
19 19     5       6 10.521415 -5.52141527
20 20    19       7 11.631285  7.36871508
```

```
>
>
```

The next step is to assess for outliers. These may be estimated with `outlierTest()`, and Cooks D. Cooks D may be calculated with `influence.measures()`.

```
>
> outlierTest(HierReg.1_lm)
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
rstudent unadjusted p-value Bonferroni p
20 1.965799           0.065873           NA
>
> influence.measures(HierReg.1_lm)
Influence measures of
lm(formula = Curse ~ Courses, data = HierReg.1) :

     dfb.1_ dfb.Crss   dffit cov.r   cook.d    hat inf
1  -0.18713  0.15137 -0.1991 1.229 2.06e-02 0.1186
2  -0.36918  0.25314 -0.4535 0.907 9.43e-02 0.0726
3  -0.12055  0.17984  0.2256 1.252 2.64e-02 0.1372
4  -0.16156  0.31733  0.4997 0.908 1.14e-01 0.0838
5   0.09305 -0.06380  0.1143 1.186 6.85e-03 0.0726
6  -0.00667 -0.04503 -0.1461 1.137 1.11e-02 0.0552
7  -0.06808  0.02308 -0.1343 1.137 9.37e-03 0.0515
8   0.16549 -0.22267 -0.2541 1.390 3.37e-02 0.2154   *
9   0.00125  0.00846  0.0275 1.185 3.99e-04 0.0552
10 -0.00997  0.00870 -0.0101 1.383 5.44e-05 0.1893   *
11  0.16496 -0.05592  0.3255 0.951 5.03e-02 0.0515
12  0.20302 -0.30287 -0.3799 1.171 7.25e-02 0.1372
13 -0.01841 -0.12431 -0.4034 0.876 7.40e-02 0.0552
14  0.03859 -0.07579 -0.1193 1.202 7.47e-03 0.0838
15  0.20646 -0.14157  0.2536 1.100 3.25e-02 0.0726
16 -0.01645  0.01128 -0.0202 1.208 2.16e-04 0.0726
17  0.17679 -0.16097  0.1773 1.553 1.66e-02 0.2849   *
18  0.13298 -0.04508  0.2624 1.024 3.39e-02 0.0515
19 -0.01524 -0.10289 -0.3339 0.959 5.31e-02 0.0552
20 -0.19223  0.37757  0.5945 0.812 1.52e-01 0.0838
>
>
```

Finally, Mahalanobis Distance and p values are computed and added to the data frame. Because the degrees of freedom in `pchisq()` is the number of predictors, it was set to `1`. In this example, none of the p values are less than a conservative criterion of 0.001, so there are no multivariate outliers. Finally, R produces a quantile-quantile plot with `plot()`. This is produced by selecting plot 2 with the `which` option.

```
>
>
> HierReg.1.Mahal <- (HierReg.1[, c(2,3)])
>
> HierReg.1.Mahal$Mahalanobis <- mahalanobis(HierReg.1.Mahal,
+ colMeans(HierReg.1.Mahal), cov(HierReg.1.Mahal))
>
> HierReg.1.Mahal$p <- pchisq(HierReg.1.Mahal$Mahalanobis, df = 1,
+ lower.tail = FALSE)
>
> HierReg.1.Mahal
   Curse Courses Mahalanobis          p
1      5       3   1.5878964 0.20762693
2      2       4   2.7879007 0.09497858
3     15       8   1.9594307 0.16157421
4     18       7   3.0506285 0.08070590
5     10       4   0.6011558 0.43813737
6      8       6   0.4771636 0.48970991
7      7       5   0.3742176 0.54071423
8     12       9   3.3459394 0.06737102
9     11       6   0.1132590 0.73646389
10     6       2   2.6475433 0.10371006
11    15       5   1.8833547 0.16995336
12     9       8   2.4875403 0.11475094
13     4       6   2.6249858 0.10519344
14    10       7   0.8001921 0.37103596
15    12       4   1.2420525 0.26507585
16     8       4   0.4352923 0.50940347
17     6       1   4.5261506 0.03338062
18    14       5   1.2790585 0.25807416
19     5       6   1.9098928 0.16697516
20    19       7   3.8663454 0.04926368
>
>
```

R can produce six diagnostic plots with `plot()`. Table 10 lists the available plots. Figure 2 gives a Normal Q-Q plot. These were generated with the following commands:

```
plot(HReg_lm.1, which = 2)
```

The next stage in the analysis adds a second predictor to the regression. As with the first model, the first step is to load the data and screen it. Apart from descriptive statistics, z-Scores and correlations will be generated. R objects relating to this regression will have the prefix HierReg.2 :

```
>
> HierReg.2 <- read.csv("Curse.txt", header = TRUE, sep = ",")
> attach(HierReg.2)
```

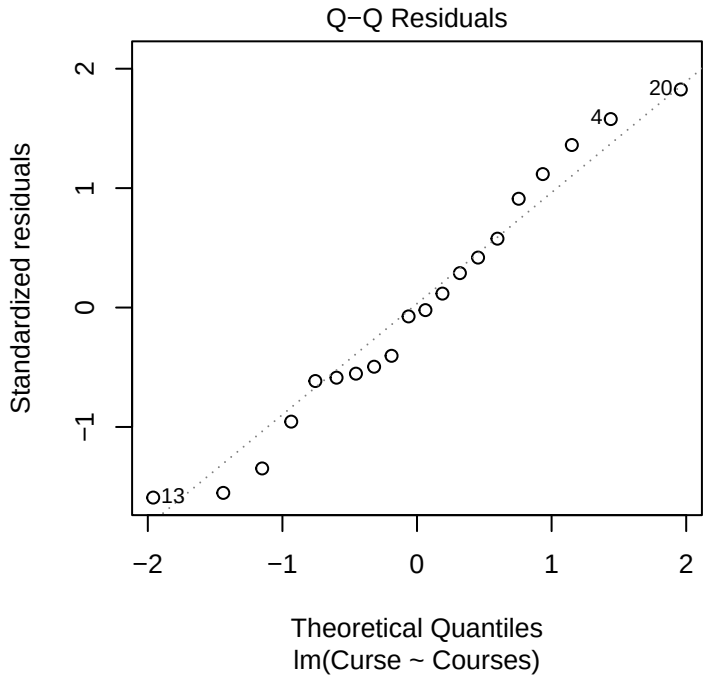

Figure 2: Hierarchical Regression 1: Normal Q-Q plot.

```
The following objects are masked from HierReg.1:

Consc, Courses, Curse, Expertise, ID

>
> ls.str(HierReg.2)
Consc :  int [1:20] 6 9 7 6 6 6 9 10 10 6 ...
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
Expertise :  int [1:20] 2 3 3 3 4 3 4 2 2 2 ...
ID :  int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
>
> library(jmv)
>
> descriptives(HierReg.2,
+ vars = vars(Curse, Courses, Expertise),
+ freq = TRUE, desc = "columns", n = TRUE,
+ missing = TRUE, mean = TRUE, median = TRUE, mode = TRUE,
+ sum = TRUE, sd = TRUE, variance = TRUE, range = TRUE,
+ min = TRUE, max = TRUE, se = TRUE, ci = TRUE,
+ min = TRUE, max = TRUE, se = TRUE, ci = TRUE,
+ ciWidth = 95, iqr = TRUE, skew = TRUE, kurt = TRUE,
```

```
+ sw = TRUE, pcEqGr = TRUE, pcNEqGr = 4, pc = TRUE,
+ pcValues = "25, 50, 75")

DESCRIPTIVES

Descriptives
---------------------------------------------------------------------
                            Curse         Courses       Expertise
---------------------------------------------------------------------
N                              20              20              20
Missing                         0               0               0
Mean                     9.800000        5.350000        2.900000
Std. error mean          1.050313       0.4604060       0.1432701
95% CI mean lower bound  7.601669        4.386359        2.600132
95% CI mean upper bound  11.99833        6.313641        3.199868
Median                   9.500000        5.500000        3.000000
Mode                     5.000000        4.000000        3.000000
Sum                           196             107              58
Standard deviation       4.697144        2.058998       0.6407233
Variance                 22.06316        4.239474       0.4105263
IQR                      6.500000        3.000000       0.2500000
Range                           17               8               2
Minimum                          2               1               2
Maximum                         19               9               4
Skewness                 0.3801509      -0.2842891      0.08003775
Std. error skewness      0.5121033       0.5121033       0.5121033
Kurtosis                -0.5856797      -0.2156383      -0.2498356
Std. error kurtosis      0.9923836       0.9923836       0.9923836
Shapiro-Wilk W           0.9697020       0.9740802       0.7875505
Shapiro-Wilk p           0.7485670       0.8375718      0.0005668
25th percentile          6.000000        4.000000        2.750000
50th percentile          9.500000        5.500000        3.000000
75th percentile          12.50000        7.000000        3.000000
---------------------------------------------------------------------
Note. The CI of the mean assumes sample means follow a
t-distribution with N - 1 degrees of freedom

>
>
> HierReg.2.Z <- (HierReg.2[, c(1,2,3,4)])
>
> HierReg.2.Z <- (HierReg.2[, c(1,2,3,4)])
>
> HierReg.2.Z$Curse.Z <- (Curse - mean(Curse)) / sd(Curse)
> HierReg.2.Z$Courses.Z <- (Courses - mean(Courses)) / sd(Courses)
> HierReg.2.Z$Expertise.Z <- (Expertise - mean(Expertise)) / sd(Expertise)
>
```

```
> HierReg.2.Z
   ID Curse Courses Expertise     Curse.Z  Courses.Z Expertise.Z
1   1     5       3         2 -1.02189765 -1.1413317  -1.4046626
2   2     2       4         3 -1.66058368 -0.6556587   0.1560736
3   3    15       8         3  1.10705579  1.2870337   0.1560736
4   4    18       7         3  1.74574182  0.8013606   0.1560736
5   5    10       4         4  0.04257907 -0.6556587   1.7168098
6   6     8       6         3 -0.38321162  0.3156875   0.1560736
7   7     7       5         4 -0.59610696 -0.1699856   1.7168098
8   8    12       9         2  0.46836976  1.7727067  -1.4046626
9   9    11       6         2  0.25547441  0.3156875  -1.4046626
10 10     6       2         2 -0.80900231 -1.6270048  -1.4046626
11 11    15       5         3  1.10705579 -0.1699856   0.1560736
12 12     9       8         3 -0.17031628  1.2870337   0.1560736
13 13     4       6         4 -1.23479299  0.3156875   1.7168098
14 14    10       7         3  0.04257907  0.8013606   0.1560736
15 15    12       4         3  0.46836976 -0.6556587   0.1560736
16 16     8       4         3 -0.38321162 -0.6556587   0.1560736
17 17     6       1         3 -0.80900231 -2.1126779   0.1560736
18 18    14       5         3  0.89416044 -0.1699856   0.1560736
19 19     5       6         3 -1.02189765  0.3156875   0.1560736
20 20    19       7         2  1.95863716  0.8013606  -1.4046626
>
>
> HierReg.2.2 <- (HierReg.2[, c(2,3,4)])
>
> ls.str(HierReg.2.2)
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
Expertise :  int [1:20] 2 3 3 3 4 3 4 2 2 2 ...
>
>
>
> library(correlation)
>
> HierReg.2.cor <- correlation(HierReg.2.2, method = "pearson",
+ p_adjust = "none")
>
> HierReg.2.cor
# Correlation Matrix (pearson-method)

Parameter1 | Parameter2 |     r |        95% CI | t(18) |      p
-----------------------------------------------------------------
Curse      |    Courses |  0.49 | [ 0.06, 0.76] |  2.36 | 0.030*
Curse      |  Expertise | -0.22 | [-0.60, 0.25] | -0.94 | 0.358
Courses    |  Expertise | -0.05 | [-0.48, 0.40] | -0.22 | 0.828
```

```
p-value adjustment method: none
Observations: 20
>
> summary(HierReg.2.cor)
# Correlation Matrix (pearson-method)

Parameter | Expertise | Courses
------------------------------
Curse     |     -0.22 |   0.49*
Courses   |     -0.05 |

p-value adjustment method: none
>
>
> library(psych)
>
> smc(HierReg.2.2)
     Curse    Courses  Expertise
0.27351172 0.23971349 0.05079399
>
>
```

The next section generates the regression model with `lm()` and ANOVA Summary Table with `anova()`. Both of these indicate that both the predictors make significant contributions to the model:

```
>
> HierReg.2_lm <- lm(Curse ~ Courses + Expertise, data = HierReg.2)
>
> summary(HierReg.2_lm)

Call:
lm(formula = Curse ~ Courses + Expertise, data = HierReg.2)

Residuals:
    Min      1Q  Median      3Q     Max
-6.1915 -3.1550 -0.8222  3.3649  6.5471

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   8.0686     5.2725   1.530   0.1443
Courses       1.0871     0.4722   2.302   0.0342 *
Expertise    -1.4086     1.5175  -0.928   0.3663
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.233 on 17 degrees of freedom
Multiple R-squared:  0.2735,    Adjusted R-squared:  0.188
```

```
F-statistic:   3.2 on 2 and 17 DF,  p-value: 0.06614

>
>
> options(contrasts = c("contr.sum", "contr.sum"))
> options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"

>
> anova(HierReg.2_lm)
Analysis of Variance Table

Response: Curse
          Df  Sum Sq Mean Sq F value Pr(>F)
Courses    1  99.222  99.222  5.5387 0.0309 *
Expertise  1  15.434  15.434  0.8615 0.3663
Residuals 17 304.544  17.914
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
>
```

Next comes Variance Inflation Factor. This is calculated with `vif()` in the `car` package. This estimates how much multicollinearity there is between predictors. A value of around 1.0 indicates none. Squared semi-partial correlations are calculated with `getDeltaSquare()` in the `Rockchalk` package. Fitted values and residuals are added to the `reg.2` data frame.

```
>
> library(car)
>
> vif(HierReg.2_lm)
 Courses Expertise
1.002697  1.002697
>
> library(rockchalk)
>
> getDeltaRsquare(HierReg.2_lm)
The deltaR-square values: the change in the R-square
observed when a single term is removed.
Same as the square of the 'semi-partial correlation coefficient'
deltaRsquare
Courses     0.22648686
Expertise   0.03681719
>
> HierReg.2.Res <- (HierReg.2[, c(1,2,3,4)])
```

```
>
> HierReg.2.Res$fittedlm <- fitted.values(HierReg.2_lm)
> HierReg.2.Res$residualslm <- residuals(HierReg.2_lm)
>
>
>
> HierReg.2.Res
   ID Curse Courses Expertise  fittedlm residualslm
1   1     5       3         2  8.512927  -3.5129269
2   2     2       4         3  8.191510  -6.1915097
3   3    15       8         3 12.540057   2.4599425
4   4    18       7         3 11.452921   6.5470795
5   5    10       4         4  6.782956   3.2170444
6   6     8       6         3 10.365784  -2.3657836
7   7     7       5         4  7.870093  -0.8700926
8   8    12       9         2 15.035748  -3.0357485
9   9    11       6         2 11.774338  -0.7743377
10 10     6       2         2  7.425790  -1.4257900
11 11    15       5         3  9.278647   5.7213533
12 12     9       8         3 12.540057  -3.5400575
13 13     4       6         4  8.957229  -4.9572295
14 14    10       7         3 11.452921  -1.4529205
15 15    12       4         3  8.191510   3.8084903
16 16     8       4         3  8.191510  -0.1915097
17 17     6       1         3  4.930099   1.0699011
18 18    14       5         3  9.278647   4.7213533
19 19     5       6         3 10.365784  -5.3657836
20 20    19       7         2 12.861475   6.1385254
>
```

Next come assessment for outliers. `outliertest()` finds the most outlying response, which may or may not qualify as an actual outlier. `influence.measures()` generates Cook's Distance.

```
>
> outlierTest(HierReg.2_lm)
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
  rstudent unadjusted p-value Bonferroni p
4 1.706409            0.10726           NA
>
> influence.measures(HierReg.2_lm)
Influence measures of
         lm(formula = Curse ~ Courses + Expertise, data = HierReg.2) :

     dfb.1_ dfb.Crss dfb.Expr   dffit cov.r   cook.d    hat inf
1  -0.48936  0.30033  0.36209 -0.5182 1.327 9.01e-02 0.2316
2  -0.14501  0.24510 -0.04620 -0.4464 0.835 6.10e-02 0.0734
```

```
3  -0.09626  0.19731  0.03395  0.2480 1.300 2.13e-02 0.1398
4  -0.15502  0.33188  0.08103  0.5230 0.795 8.19e-02 0.0859
5  -0.21340 -0.12617  0.37471  0.4568 1.349 7.07e-02 0.2221
6   0.01659 -0.04319 -0.02300 -0.1384 1.199 6.65e-03 0.0568
7   0.07000  0.00467 -0.09864 -0.1140 1.496 4.59e-03 0.2055
8  -0.10743 -0.40055  0.30933 -0.5677 1.513 1.09e-01 0.3063
9  -0.05893 -0.01176  0.06722 -0.0835 1.413 2.46e-03 0.1570
10 -0.24275  0.18487  0.16194 -0.2620 1.680 2.41e-02 0.3063   *
11  0.04370 -0.05468  0.04974  0.3374 0.882 3.57e-02 0.0527
12  0.14032 -0.28762 -0.04948 -0.3615 1.204 4.41e-02 0.1398
13  0.51374 -0.14184 -0.60743 -0.7054 1.102 1.58e-01 0.2138
14  0.03176 -0.06800 -0.01660 -0.1072 1.282 4.04e-03 0.0859
15  0.08514 -0.14390  0.02713  0.2621 1.105 2.31e-02 0.0734
16 -0.00417  0.00705 -0.00133 -0.0128 1.294 5.84e-05 0.0734
17  0.09041 -0.16627  0.00367  0.1836 1.651 1.19e-02 0.2850   *
18  0.03535 -0.04423  0.04023  0.2729 0.995 2.43e-02 0.0527
19  0.03927 -0.10225 -0.05446 -0.3277 0.926 3.42e-02 0.0568
20  0.41313  0.31240 -0.58453  0.7960 0.896 1.90e-01 0.1819
>
>
```

Next is Mahalanobis Distance which checks for multivariate outliers. This is calculated with `mahalanobis()` and the results are added to the `Reg.2` data frame along with p values.

```
>
>
> HierReg.2.Mahal <- (HierReg.2[, c(2,3,4)])
>
> HierReg.2.Mahal$Mahalanobis <- mahalanobis(HierReg.2.Mahal,
+ colMeans(HierReg.2.Mahal), cov(HierReg.2.Mahal))
>
> HierReg.2.Mahal$p <- pchisq(HierReg.2.Mahal$Mahalanobis, df = 2,
+ lower.tail = FALSE)
>
> HierReg.2.Mahal
   Curse Courses Expertise Mahalanobis          p
1      5       3         2   4.2212062 0.12116487
2      2       4         3   2.8364749 0.24214043
3     15       8         3   2.0837719 0.35278871
4     18       7         3   3.3555749 0.18678680
5     10       4         4   3.9150392 0.14120824
6      8       6         3   0.4786601 0.78715505
7      7       5         4   3.0012376 0.22299213
8     12       9         2   5.4453367 0.06569921
9     11       6         2   2.0696136 0.35529503
10     6       2         2   4.9972077 0.08219968
11    15       5         3   2.0928524 0.35119060
```

```
12     9       8         3   2.4880916 0.28821579
12     9       8         3   2.4880916 0.28821579
13     4       6         4   4.6448219 0.09803694
14    10       7         3   0.8130443 0.66596236
15    12       4         3   1.3497476 0.50922067
16     8       4         3   0.4471174 0.79966795
17     6       1         3   4.5369914 0.10346771
18    14       5         3   1.4413488 0.48642410
19     5       6         3   1.9257399 0.38179557
20    19       7         2   4.8561220 0.08820770
>
>
```

Here is a Normal Q-Q Plot:

```
>
> plot(HReg_lm.2, which = 4)
>
```

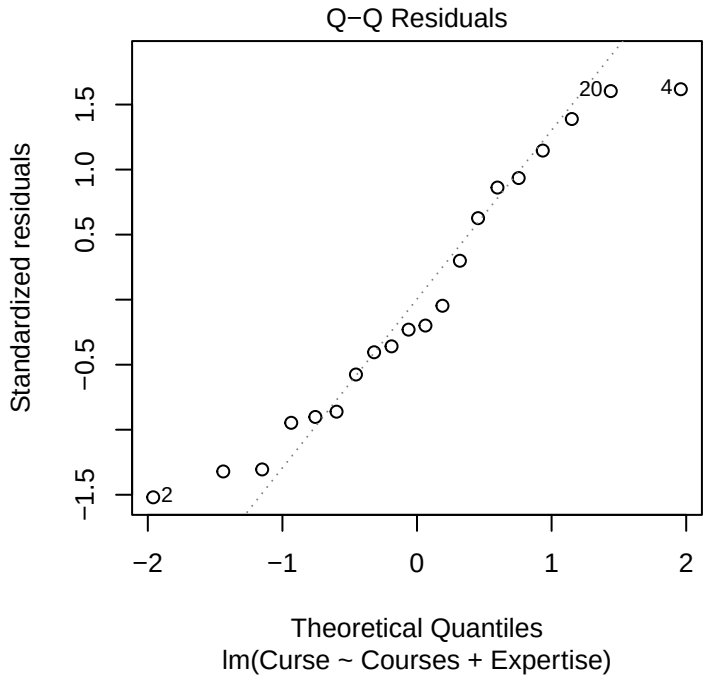

Figure 3: Hierarchical Regression 2: Normal Q-Q plot.

Finally, an ANOVA compares the two models. It is performed with `anova()` and requires the names of the two linear models, that is the R objects created by the `lm()` commands:

```
>
> anova(HierReg.1.lm, HierReg.2_lm)
Analysis of Variance Table

Model 1: Curse ~ Courses
Model 2: Curse ~ Courses + Expertise
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     18 319.98
2     17 304.54  1    15.434 0.8615 0.3663
>
```

The third regression adds the third and final predictor. Other than having three predictors, it is the same procedure as those described above. Only the output is presented here.

```
>
> getwd()
[1] "C:/Users/Michael/Documents"
> setwd("C:/Users/Michael/Desktop/R_Project")
>
> HierReg.3 <- read.csv("Curse.txt", header = TRUE, sep = ",")
> attach(HierReg.3)
The following objects are masked from HierReg.3 (pos = 3):

Consc, Courses, Curse, Expertise, ID

The following objects are masked from HierReg.2:

Consc, Courses, Curse, Expertise, ID

The following objects are masked from HierReg.1:

Consc, Courses, Curse, Expertise, ID

>
> ls.str(HierReg.3)
Consc :  int [1:20] 6 9 7 6 6 6 9 10 10 6 ...
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
Expertise :  int [1:20] 2 3 3 3 4 3 4 2 2 2 ...
ID :  int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
>
> library(jmv)
>
> descriptives(HierReg.3,
+ vars = vars(Curse, Courses, Expertise, Consc),
+ freq = TRUE, desc = "columns", n = TRUE,
+ missing = TRUE, mean = TRUE, median = TRUE, mode = TRUE,
```

```
+ sum = TRUE, sd = TRUE, variance = TRUE, range = TRUE,
+ min = TRUE, max = TRUE, se = TRUE, ci = TRUE,
+ ciWidth = 95, iqr = TRUE, skew = TRUE, kurt = TRUE,
+ sw = TRUE, pcEqGr = TRUE, pcNEqGr = 4, pc = TRUE,
+ pcValues = "25, 50, 75")

DESCRIPTIVES

Descriptives
```

	Curse	Courses	Expertise	Consc
N	20	20	20	20
Missing	0	0	0	0
Mean	9.800000	5.350000	2.900000	7.050000
Std. error mean	1.050313	0.4604060	0.1432701	0.3802700
95% CI mean lower bound	7.601669	4.386359	2.600132	6.254086
95% CI mean upper bound	11.99833	6.313641	3.199868	7.845914
Median	9.500000	5.500000	3.000000	6.500000
Mode	5.000000	4.000000	3.000000	6.000000
Sum	196	107	58	141
Standard deviation	4.697144	2.058998	0.6407233	1.700619
Variance	22.06316	4.239474	0.4105263	2.892105
IQR	6.500000	3.000000	0.2500000	2.250000
Range	17	8	2	5
Minimum	2	1	2	5
Maximum	19	9	4	10
Skewness	0.3801509	-0.2842891	0.08003775	0.6985984
Std. error skewness	0.5121033	0.5121033	0.5121033	0.5121033
Kurtosis	-0.5856797	-0.2156383	-0.2498356	-0.7890889
Std. error kurtosis	0.9923836	0.9923836	0.9923836	0.9923836
Shapiro-Wilk W	0.9697020	0.9740802	0.7875505	0.8616526
Shapiro-Wilk p	0.7485670	0.8375718	0.0005668	0.0084091
25th percentile	6.000000	4.000000	2.750000	6.000000
50th percentile	9.500000	5.500000	3.000000	6.500000
75th percentile	12.50000	7.000000	3.000000	8.250000

```
Note. The CI of the mean assumes sample means follow a t-distribution with N
- 1 degrees of freedom

>
> HierReg.3.Z <- (HierReg.3[, c(1,2,3,4,5)])
>
> HierReg.3.Z$Curse.Z <- (Curse - mean(Curse)) / sd(Curse)
> HierReg.3.Z$Courses.Z <- (Courses - mean(Courses)) / sd(Courses)
> HierReg.3.Z$Expertise.Z <- (Expertise - mean(Expertise)) / sd(Expertise)
> HierReg.3.Z$Consc.Z <- (Consc - mean(Consc)) / sd(Consc)
```

```
>
>
> HierReg.3.Z
   ID Curse Courses Expertise Consc     Curse.Z  Courses.Z Expertise.Z     Consc.Z
1   1     5       3         2     6 -1.02189765 -1.1413317  -1.4046626 -0.61742221
2   2     2       4         3     9 -1.66058368 -0.6556587   0.1560736  1.14664126
3   3    15       8         3     7  1.10705579  1.2870337   0.1560736 -0.02940106
4   4    18       7         3     6  1.74574182  0.8013606   0.1560736 -0.61742221
5   5    10       4         4     6  0.04257907 -0.6556587   1.7168098 -0.61742221
6   6     8       6         3     6 -0.38321162  0.3156875   0.1560736 -0.61742221
7   7     7       5         4     9 -0.59610696 -0.1699856   1.7168098  1.14664126
8   8    12       9         2    10  0.46836976  1.7727067  -1.4046626  1.73466241
9   9    11       6         2    10  0.25547441  0.3156875  -1.4046626  1.73466241
10 10     6       2         2     6 -0.80900231 -1.6270048  -1.4046626 -0.61742221
11 11    15       5         3     5  1.10705579 -0.1699856   0.1560736 -1.20544337
12 12     9       8         3     6 -0.17031628  1.2870337   0.1560736 -0.61742221
13 13     4       6         4     8 -1.23479299  0.3156875   1.7168098  0.55862010
14 14    10       7         3     7  0.04257907  0.8013606   0.1560736 -0.02940106
15 15    12       4         3     7  0.46836976 -0.6556587   0.1560736 -0.02940106
16 16     8       4         3     7 -0.38321162 -0.6556587   0.1560736 -0.02940106
17 17     6       1         3     5 -0.80900231 -2.1126779   0.1560736 -1.20544337
18 18    14       5         3     6  0.89416044 -0.1699856   0.1560736 -0.61742221
19 19     5       6         3     5 -1.02189765  0.3156875   0.1560736 -1.20544337
20 20    19       7         2    10  1.95863716  0.8013606  -1.4046626  1.73466241
>
>
> HierReg.3.2 <- (HierReg.3[, c(2,3,4,5)])
>
> ls.str(HierReg.3.2)
Consc :  int [1:20] 6 9 7 6 6 6 9 10 10 6 ...
Courses :  int [1:20] 3 4 8 7 4 6 5 9 6 2 ...
Curse :  int [1:20] 5 2 15 18 10 8 7 12 11 6 ...
Expertise :  int [1:20] 2 3 3 3 4 3 4 2 2 2 ...
>
>
>
> library(correlation)
>
> HierReg.3.2.cor <- correlation(HierReg.3.2, method = "pearson",
+ p_adjust = "none")
>
> HierReg.3.2.cor
# Correlation Matrix (pearson-method)

Parameter1 | Parameter2 |     r |        95% CI | t(18) |      p
-----------------------------------------------------------------
Curse      |    Courses |  0.49 | [ 0.06, 0.76] |  2.36 | 0.030*
```

```
Curse      |  Expertise | -0.22 | [-0.60, 0.25] | -0.94 | 0.358
Curse      |      Consc |  0.11 | [-0.35, 0.52] |  0.46 | 0.654
Courses    |  Expertise | -0.05 | [-0.48, 0.40] | -0.22 | 0.828
Courses    |      Consc |  0.39 | [-0.07, 0.71] |  1.77 | 0.093
Expertise  |      Consc | -0.24 | [-0.61, 0.23] | -1.03 | 0.315

p-value adjustment method: none
Observations: 20
>
> summary(HierReg.3.2.cor)
# Correlation Matrix (pearson-method)

Parameter | Consc | Expertise | Courses
---------------------------------------
Curse     |  0.11 |     -0.22 |   0.49*
Courses   |  0.39 |     -0.05 |
Expertise | -0.24 |           |

p-value adjustment method: none
>
>
> library(psych)
>
> smc(HierReg.3.2)
Curse   Courses Expertise     Consc
0.2921593 0.3666963 0.1174277 0.2163658
>
> HierReg.3_lm <- lm(Curse ~ Courses + Expertise + Consc, data = HierReg.3.2)
>
> summary(HierReg.3_lm)

Call:
lm(formula = Curse ~ Courses + Expertise + Consc, data = HierReg.3.2)

Residuals:
    Min      1Q  Median      3Q     Max
-6.2882 -3.1175  0.0754  3.4101  6.9464

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.0414     7.0530   1.565   0.1370
Courses       1.2171     0.5205   2.338   0.0327 *
Expertise    -1.6511     1.5886  -1.039   0.3141
Consc        -0.4206     0.6478  -0.649   0.5254
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
Residual standard error: 4.306 on 16 degrees of freedom
Multiple R-squared:  0.2922,    Adjusted R-squared:  0.1594
F-statistic: 2.201 on 3 and 16 DF,  p-value: 0.1276

>
>
> options(contrasts = c("contr.sum", "contr.sum"))
> options("contrasts")
$contrasts
[1] "contr.sum" "contr.sum"

>
> anova(HierReg.3_lm)
Analysis of Variance Table

Response: Curse
Df  Sum Sq Mean Sq F value  Pr(>F)
Courses    1  99.222  99.222  5.3502 0.03435 *
Expertise  1  15.434  15.434  0.8322 0.37517
Consc      1   7.817   7.817  0.4215 0.52540
Residuals 16 296.727  18.545
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
>
> library(car)
>
> vif(HierReg.3_lm)
Courses Expertise     Consc
1.176866  1.061391  1.243350
>
> library(rockchalk)
>
> getDeltaRsquare(HierReg.3_lm)
The deltaR-square values: the change in the R-square
observed when a single term is removed.
Same as the square of the ’semi-partial correlation coefficient’
          deltaRsquare
Courses     0.24188156
Expertise   0.04779013
Consc       0.01864757
>
> HierReg.3.Res <- (HierReg.3[, c(1,2,3,4,5)])
>
> HierReg.3.Res$fittedlm <- fitted.values(HierReg.3_lm)
> HierReg.3.Res$residualslm <- residuals(HierReg.3_lm)
>
```

```
>
> HierReg.3.Res
   ID Curse Courses Expertise Consc  fittedlm residualslm
1   1     5       3         2     6  8.867281 -3.86728067
2   2     2       4         3     9  7.171639 -5.17163941
3   3    15       8         3     7 12.881357  2.11864339
4   4    18       7         3     6 12.084777  5.91522299
5   5    10       4         4     6  6.782251  3.21774911
6   6     8       6         3     6 10.867631 -2.86763094
7   7     7       5         4     9  6.737698  0.26230245
8   8    12       9         2    10 14.487891 -2.48789119
9   9    11       6         2    10 10.836453  0.16354701
10 10     6       2         2     6  7.650135 -1.65013460
11 11    15       5         3     5 10.071051  4.92894866
12 12     9       8         3     6 13.301923 -4.30192307
13 13     4       6         4     8  8.375410 -4.37541008
14 14    10       7         3     7 11.664211 -1.66421054
15 15    12       4         3     7  8.012772  3.98722766
16 16     8       4         3     7  8.012772 -0.01277234
17 17     6       1         3     5  5.202467  0.79753292
18 18    14       5         3     6  9.650485  4.34951512
19 19     5       6         3     5 11.288197 -6.28819741
20 20    19       7         2    10 12.053599  6.94640094
>
> outlierTest(HierReg.3_lm)
No Studentized residuals with Bonferroni p < 0.05
Largest |rstudent|:
   rstudent unadjusted p-value Bonferroni p
20 2.064975           0.056659           NA
>
> influence.measures(HierReg.3_lm)
Influence measures of
lm(formula = Curse ~ Courses + Expertise + Consc, data = HierReg.3.2) :

      dfb.1_  dfb.Crss  dfb.Expr  dfb.Cnsc     dffit cov.r   cook.d    hat inf
1  -5.12e-01  0.249735  0.426775  0.151657 -0.595550 1.304 8.82e-02 0.2477
2   2.64e-01  0.432115 -0.175926 -0.567726 -0.707206 1.007 1.18e-01 0.2065
3  -1.77e-02  0.182867  0.011968 -0.069415  0.223659 1.425 1.31e-02 0.1547
4   1.34e-01  0.428837 -0.014952 -0.374793  0.613691 0.833 8.67e-02 0.1369
5  -1.60e-01 -0.114424  0.357604  0.000240  0.448427 1.385 5.12e-02 0.2221
6  -6.81e-02 -0.098982  0.002663  0.129017 -0.214472 1.256 1.19e-02 0.0890
7  -4.44e-02 -0.016183  0.044590  0.037892  0.056871 2.050 8.62e-04 0.3695   *
8   4.13e-02 -0.247314  0.214315 -0.170001 -0.509385 1.736 6.70e-02 0.3447
9  -2.57e-04 -0.003904 -0.011633  0.016886  0.026132 1.771 1.82e-04 0.2695   *
10 -2.40e-01  0.179129  0.190960  0.043611 -0.303964 1.785 2.43e-02 0.3128   *
11  2.78e-01  0.100339 -0.045356 -0.380682  0.489832 1.004 5.79e-02 0.1330
12 -8.53e-02 -0.486689  0.018375  0.349501 -0.593409 1.183 8.65e-02 0.2141
```

```
13  5.42e-01 -0.007990 -0.604063 -0.288361 -0.702589 1.212 1.20e-01 0.2571
14  7.04e-03 -0.083101 -0.010913  0.031288 -0.125287 1.367 4.14e-03 0.0916
15  2.55e-02 -0.162117  0.042344  0.064011  0.278748 1.105 1.95e-02 0.0775
16 -7.92e-05  0.000504 -0.000132 -0.000199 -0.000867 1.403 2.00e-07 0.0775
17  6.70e-02 -0.104062 -0.003188 -0.024799  0.138150 1.813 5.07e-03 0.2945   *
18  1.19e-01  0.018349  0.001750 -0.144960  0.289106 1.048 2.08e-02 0.0704
19 -3.56e-01 -0.361536  0.072239  0.609876 -0.752394 0.774 1.27e-01 0.1657
20 -4.64e-02  0.104363 -0.569555  0.696140  1.241024 0.648 3.20e-01 0.2653
>
> HierReg.3.Mahal <- (HierReg.3[, c(2,3,4,5)])
>
> HierReg.3.Mahal$Mahalanobis <- mahalanobis(HierReg.3.Mahal,
+ colMeans(HierReg.3.Mahal), cov(HierReg.3.Mahal))
>
> HierReg.3.Mahal$p <- pchisq(HierReg.3.Mahal$Mahalanobis, df = 3,
+ lower.tail = FALSE)
>
> HierReg.3.Mahal
   Curse Courses Expertise Consc Mahalanobis          p
1      5       3         2     6   4.7141455 0.19396604
2      2       4         3     9   4.6855527 0.19632463
3     15       8         3     7   2.2767836 0.51698313
4     18       7         3     6   3.8922049 0.27334178
5     10       4         4     6   3.9323407 0.26886340
6      8       6         3     6   1.2681748 0.73670379
7      7       5         4     9   6.0751909 0.10800878
8     12       9         2    10   5.9962454 0.11179304
9     11       6         2    10   4.1719251 0.24348779
10     6       2         2     6   5.1670677 0.15996450
11    15       5         3     5   3.1324397 0.37165288
12     9       8         3     6   4.3020550 0.23064080
13     4       6         4     8   5.1603090 0.16042792
14    10       7         3     7   0.9671961 0.80918883
15    12       4         3     7   1.5404582 0.67296456
16     8       4         3     7   0.5224896 0.91392348
17     6       1         3     5   4.6866154 0.19623649
18    14       5         3     6   1.5980741 0.65982659
19     5       6         3     5   4.7294451 0.19271486
20    19       7         2    10   7.1812862 0.06633863
>
>
> anova(HierReg.1_lm, HierReg.2_lm, HierReg.3_lm)
Analysis of Variance Table

Model 1: Curse ~ Courses
Model 2: Curse ~ Courses + Expertise
Model 3: Curse ~ Courses + Expertise + Consc
```

```
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     18 319.98
2     17 304.54  1   15.4338 0.8322 0.3752
3     16 296.73  1    7.8171 0.4215 0.5254
>
>
```

Finally, here is a Q-Q Plot:

```
>
> plot(HReg_lm.2, which = 4)
>
```

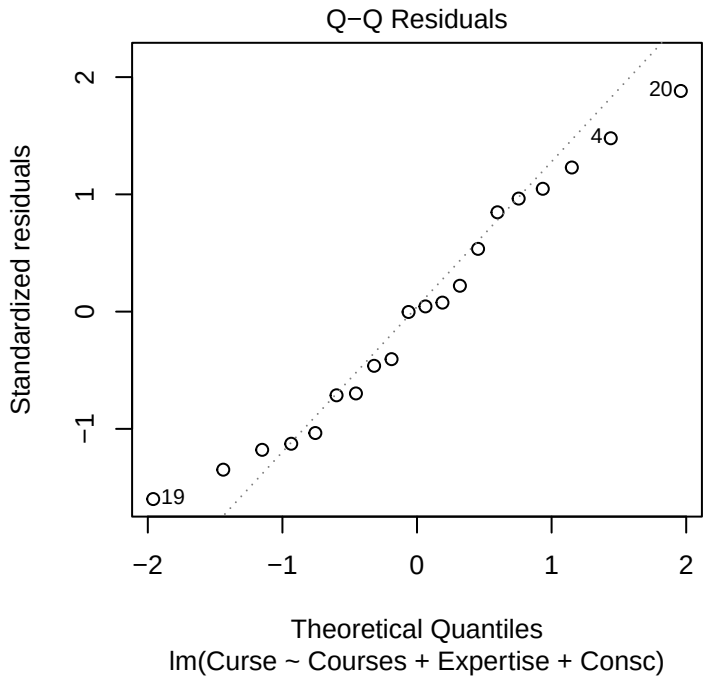

Figure 4: Hierarchical Regression 3: Normal Q-Q plot.

9 Categorical Data Analysis

The statistical procedures described above are appropriate for interval, ratio or ordinal data. They are not appropriate for data organised into categories. This can include political party membership, tea drinkers vs coffee drinkers, supporters vs opponents of the death penalty, age groups etc. In these studies, participants are classified as belonging to a category.

Two popular tests for this type of data are Pearson's χ^2 Test and Fisher's Exact Test. Both are available in the `stats` package and can be conducted with `chisq.test()` and `fisher.test()` respectively. The `stats` package is loaded by default when R starts so there is no need to load it with `library()`. The first step is to either import a data file or enter the data directly into R.

9.0.1 χ^2 and Fisher's Exact Test

The following example is an analysis of a 2×2 contingency table. The data is counts of the number of Bunyip sightings in Country and Metropolitan areas of New South Wales and the data is divided into two age categories: Over 65 and Under 65.

Contingency tables may be entered into R directly with `matrix()`. This command requires a list of the data, the number of rows, the number of columns and whether R should read the data by row or by column. The following code creates a data frame named `Bunyips`. It reads the data by row. Data may be read by column by setting the `byrow` option to `FALSE`:

```
>
> Bunyips <- matrix(c(49, 33, 40, 14), nrow=2, ncol=2, byrow=TRUE)
> Bunyips
     [,1] [,2]
[1,]   49   33
[2,]   40   14
>
```

Column and row names may be added with the `colnames()` and `rownames()` commands:

```
>
> colnames(Bunyips) <- c("Country","Metropolitan")
> rownames(Bunyips) <- c("Over 65's","Under 65's")
>
> Bunyips
           Country Metropolitan
Over 65's       49           33
Under 65's      40           14
>
```

The next stage of the analysis creates an R object named `Sightings`. This contains the results of a χ^2 test conducted with `chisq.test()`. This command has Yates Continuity Correction turned on by default. This may be appropriate if the sample size is small to avoid underestimating the p value. It can be turned off by setting the `correct = FALSE` option as it is here:

```
>
> Sightings <- chisq.test(Bunyips, correct = FALSE)
>
```

To see the results with χ^2, degrees of freedom and a p value:

```
>
> Sightings

        Pearson's Chi-squared test

data:  Bunyips
X-squared = 2.9514, df = 1, p-value = 0.08581

>
```

A table of expected frequencies can be obtained with `Sightings$expected`:

```
>
> Sightings$expected
           Country Metropolitan
Over 65s  53.66176     28.33824
Under 65s 35.33824     18.66176
>
```

Fisher's Exact Test is an alternative to χ^2 and is performed with `fisher.test()`. This is also available from the `stats` package. It requires only the name of the data frame:

```
>
> fisher.test(Bunyips)

        Fisher's Exact Test for Count Data

data:  Bunyips
p-value = 0.09933
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.2255137 1.1661891
sample estimates:
odds ratio
 0.5221697

>
```

9.0.2 Effect Sizes: ϕ, the Odds Ratio and Cramer's V

Three measures of effect size for contingency tables are ϕ, the Odds Ratio and Cramer's V. ϕ and Odds Ratio can only be used for 2×2 contingency tables. Cramer's V is suitable for tables of any size. These three measures are available from the `psych`, `epitools` and `DescTools` packages respectively. ϕ requires only the name of the data frame:

```
>
> library(psych)
> phi(Bunyips)
[1] -0.15
>
```

The Odds Ratio can be calculated with `oddsratio.wald()` in the `epitools` package:

```
>
> library(epitools)
> oddsratio.wald(Bunyips)
$data
          Country Metropolitan Total
Over 65s       49           33    82
Under 65s      40           14    54
Total          89           47   136

$measure
                        NA
odds ratio with 95% C.I. estimate     lower    upper
               Over 65s  1.000000        NA       NA
               Under 65s 0.519697 0.2450414 1.102201

$p.value
           NA
two-sided   midp.exact fisher.exact chi.square
  Over 65s          NA           NA         NA
  Under 65s 0.08919726   0.09932539 0.08580541

$correction
[1] FALSE

attr(,"method")
[1] "Unconditional MLE & normal approximation (Wald) CI"
>
```

Cramer's V can be calculated with `CramerV()` in the `DescTools` package:

```
>
> library(DescTools)
```

```
Attaching package: ‘DescTools’

The following objects are masked from ‘package:psych’:

    AUC, ICC, SD

> CramerV(Bunyips)
[1] 0.1473132
>
```

10 Graphs

Graphs present data in a format that is easily readable and they have been long used as an aid to understanding data. They also reveal patterns in the data that may otherwise be missed by researchers who rely on numerical summaries alone. Anscombe (1973) presents four graphs that have identical descriptive statistics but differing scatterplots. These are named 'Anscombe's Quartet'. Table 11 gives the Quartet's means and standard deviations and Figure 5 gives the scatterplots. Two of these graphs suggest the presense of outliers, one a curvelinear relationship between two variables and the other a fairly linear relationship. The data can be accessed in R through the `anscombe` data file in the `asbio` package.

Variable	Mean	sd
X1	9.00	7.50
Y1	9.00	7.50
X2	9.00	7.50
Y2	9.00	7.50
X3	9.00	7.50
Y3	9.00	7.50
X4	9.00	7.50
Y4	9.00	7.50

Table 11: Anscombes Quartet: Means and Standard Deviations.

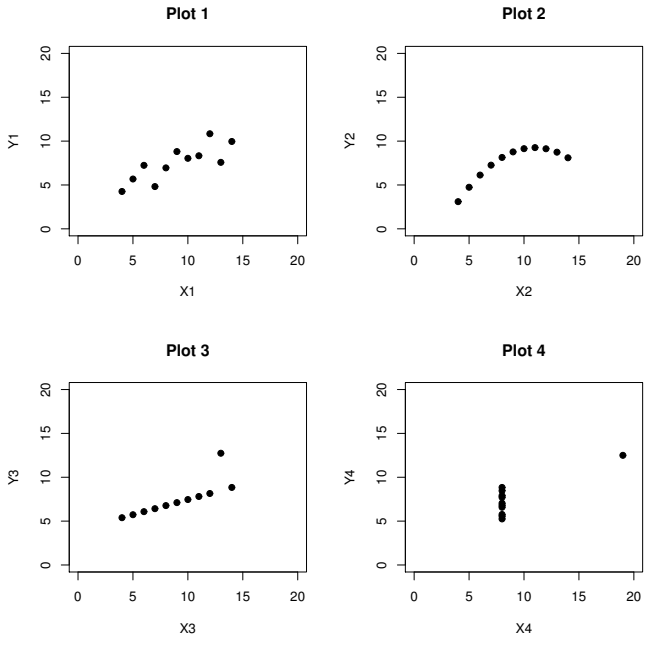

Figure 5: Anscombes Quartet.

R has comprehensive facilities for producing graphs. This section only touches on R's capabilities and additional plots may be found using R's search facility. Most of the graphs in this section are generated by R's `graphics` package. This comes with with R's base installation and is automatically loaded when R loads. Line plots are provided by `ggline()` in the `ggpubr` package. Both of these packages produce publication quality graphs in a range of popular formats such as postscript, encapsulated postscript, pdf and jpeg.

Most of R's graphics functions accept data frames in wide format and generate single graphs. Exceptions to this are `pairs()`, which generates a scatterplot matrix, and `boxplot()` which may contain multiple boxplots. R also supports placing multiple plots in the same graphic. Table 12 gives a list of common graphs in the `graphics` package and their commands.

Graph Type	Command
1-D Scatter Plot	`stripchart()`
Association Plot	`assocplot()`
Bar Chart	`barplot()`
Box Plot	`boxplot()`
Cleveland's Dot Plot	`dotchart()`
Conditional Density Plot	`cdplot()`
Fourfold Plot	`fourfoldplot()`
Histogram	`hist()`
Mosaic Plot	`mosaicplot()`
Perspective Plot	`persp()`
Pie Chart	`pie()`
Scatterplot Matrix	`pairs()`
Spine Plots	`spineplot()`
Stem-and-Leaf Plots	`stem()`
Sunflower Scatter Plot	`sunflowerplot()`

Table 12: Graph types available in the `graphics` package.

There are a number of pitfalls when producing graphs. Raster graphic formats may lose information if they are resized, so it is best to produce them in the size that they will be printed or displayed on-screen. The file format requires some thought as well because some word processors and typesetting programs don't accept all the graphic formats that R can produce. If graphics are to be printed professionally, the printer's requirements will also have to be taken into account. This can include format, colour depth, resolution and colour space. Colour depth and colour space are best edited with an external graphics program. Many packages with these facilities are freely downloadable. Two of them are GIMP and Inkscape.

10.1 Universal parameters of graphs: The `par()` command

The `par()` command is part of the `graphics` package and is used to set graphic parameters such as point type, line type, background and foreground colours. If the graphic is to have multiple plots, it will define the rows and columns for the layout. `par()` options are specified before the main plot commands. The Box Type command options are somewhat cryptic. The shapes of the glyphs are said to reflect the shape of the box surrounding the graph. See the documentation on the par() command for more details.

Table 13 gives a short list of options for this command. Additional options are listed in the package documentation. The 26 point types and 7 line types available in R are illustrated in Appendix A.

Plot Option	Command	Options
Line Type	`lty`	`0-6 or solid, dashed, dotted, dotdash, longdash, twodash`
Point Type	`pch`	`0-25`
Box Type	`bty`	`o, 1, 7, c, u, ] or n`
Multiple plots	`mfrow`	`c(Number of Rows, Number of Columns)`
Background Colour	`bg`	`default is "transparent"`
Plotting Colour	`col`	`default is "black"`
Foreground Colour	`fg`	`default is "black"`

Table 13: Some parameters controlled by `par()`

10.2 Some Common Plots

10.2.1 Scatterplots

Scatterplots are generated with `plot()`. A scatterplot for the `MoneyLove` data set in the appendix is illustrated in Figure 6. The simplest plot requires the names of the two variables to be plotted:

```
plot(MoneyLove$Salary, MoneyLove$Happiness)
```

`plot()` has options for scale limits, titles and labels. Participants income in $ was between $0 and $250,000 yearly, and their personal happiness was rated on a 1–10 scale. The ranges of the x and y axes were changed with the `xlim` and `ylim` options to reflect this. The plot was given a title with `main` which says something about what it represents and the axis labels were changed with `xlab` and `ylab`.

The box surrounding the plot was changed with the `par()` command `bty = "n"` and the point character was changed from the default hollow circle to a solid circle with `pch = 20`. This is number 20 of the 26 point options.

```
setwd("C:/Users/Michael/Desktop/R_Project")

MoneyLove <- read.csv("MoneyLove.txt", header = TRUE, sep = ",")
attach(MoneyLove)

par(bty = "n", pch = 20)
plot(MoneyLove$Salary, MoneyLove$Happiness,
xlim = c(0, 250),
ylim = c(0, 10),
main = "Income vs Happiness",
xlab = "Income ($000)",
ylab = "Personal Happiness"
)
```

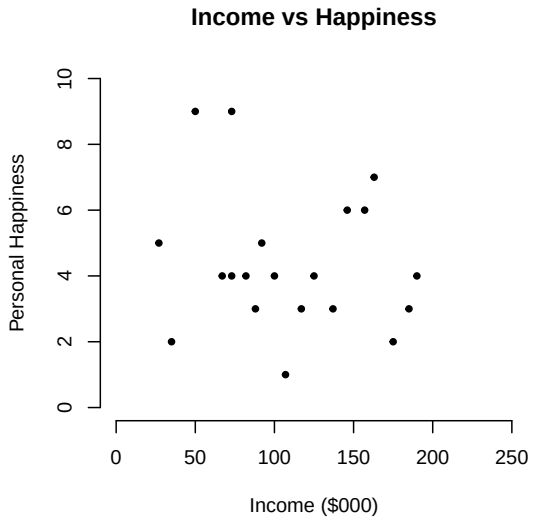

Figure 6: Scatterplot

10.2.2 Scatterplot Matrix

Scatterplot matrices can be generated with the `pairs()` command if there are more than two variables. This requires a list of variable columns to be included in the matrix and the name of the dataframe. It uses a one-sided additive version of the formula interface to generate the matrix: `pairs(~ a + b + c ...)`. This example uses data from the `MoneyLove` data set. It includes `par()`'s `pch` option to change the default point type. The `pairs()` options include a heading and labels for each variable. By default, the matrix is mirrored in the upper and lower quadrants. The upper half of the matrix may be turned off with `upper.panel = NULL`, and the lower panel with `lower.panel = null`:

```
setwd("C:/Users/Michael/Desktop/R_Project")
```

```
MoneyLove <- read.csv("MoneyLove.txt", header = TRUE, sep = ",")
attach(MoneyLove)

par(pch = 20)
pairs(~ Salary + Happiness + Love, data = MoneyLove,
main = "Scatterplot for Salary, Happiness and Love",
labels = c("Salary", "Happiness", "Love")
)
```

Figure 7: Scatterplot Matrix

10.2.3 Boxplots

Boxplots are produced with `boxplot()` and multiple boxplots may be contained in a single graphic. It accepts data frames in either wide or long format. If the data frame is in wide format, the name of the data frame should be specified and it will create boxplots for each column. If the dataframe contains columns not required for the plot, a new data frame containing only the columns of interest should be created. If the data frame is in long format, R accepts a formula interface.

This example uses the `Skydiving` data in Appendix. The following code loads the Skydiving data, converts the data to long format and generates boxplots. The boxplots are oriented vertically by default, but this can be changed to horizontal with `horizontal = TRUE`. Notches may be placed in the boxes with `notch = TRUE`. Axis labels were added with `xlab()` and `ylab()`. The default `boxplot()` command creates boxes with a grey fill. This can be changed with the `col` option. In this case it was changed to `transparent`.

```
setwd("C:/Users/Michael/Desktop/R_Project")

Skydiving <- read.csv("Skydiving.txt", header = TRUE, sep = ",")
attach(Skydiving)

Skydiving.Long <- stack(Skydiving, c(Atlantis, Lemuria))

colnames(Skydiving.Long) <- c("Rating","Continent")

factor(Skydiving.Long$Location, levels = c('Atlantis', 'Lemuria'),
ordered = FALSE)

boxplot(Rating ~ Continent, data = Skydiving.Long,
col = "transparent",
xlab = "Continent",
ylab = "Skydiving Rating")
```

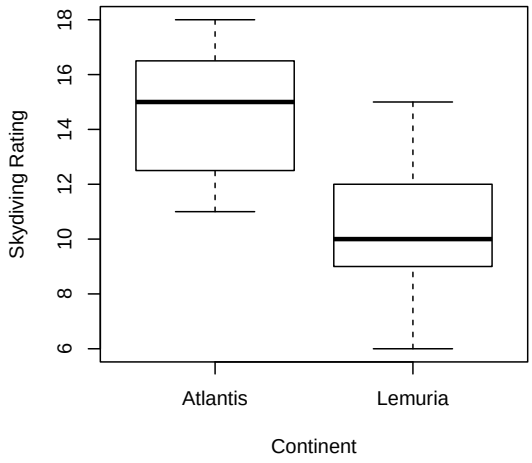

Figure 8: Boxplot

The thick line represents the median, the limits of the box represent the first and third quartiles, or 25th and 75th percentiles, and the whiskers 1.5 $\times$ the interquartile range. Outliers, if any, appear as small open circles beyond the limits of the whiskers.

10.2.4 Histograms and Density Plots

Both histogram and density plots are created with `hist()` and are illustrated in Figure 9. These commands accept data frames in wide format and require the name of the column to be plotted. To specify a histogram with frequency counts, set `freq = TRUE`. `xlim()` reflects the rating scale, and `ylim()` reflects the frequencies. By default, R produces these graphs with a grey fill. This can be changed to transparent with the `col = "transparent"` option as it has been in this example.

```
setwd("C:/Users/Michael/Desktop/R_Project")

Skydiving <- read.csv("Skydiving.txt", header = TRUE, sep = ",")
attach(Skydiving)

hist(Lemuria,
col = "transparent",
freq = TRUE,
xlim = c(0, 20),
ylim = c(0, 10),
main = "Histogram: Lemuria",
xlab = "Skydiving Rating",
ylab = "Frequency"
)
```

To generate a density plot, `freq` is set to `FALSE`. This converts the y axis to probabilities. Both `xlim()` and `ylim()` may be set normally. The y axis represents probabilities, so `ylim()` has a maximum value of 1.0. A density curve can be added with `lines()`. This requires the type of line and the variable name:

```
setwd("C:/Users/Michael/Desktop/R_Project")

Skydiving <- read.csv("Skydiving.txt", header = TRUE, sep = ",")
attach(Skydiving)

hist(Lemuria,
col = "transparent",
freq = FALSE,
xlim = c(0, 20),
ylim = c(0, 0.3),
main = "Density Plot: Lemuria",
xlab = "Skydiving Rating")
lines(density(Lemuria))
```

10.2.5 Mosaic Plots

Mosaic plots illustrate the frequencies of contingency tables. A basic mosaic plot can be produced with `mosaicplot(DataFrameName)` from the `graphics` package. The plot in Figure 10 was generated with

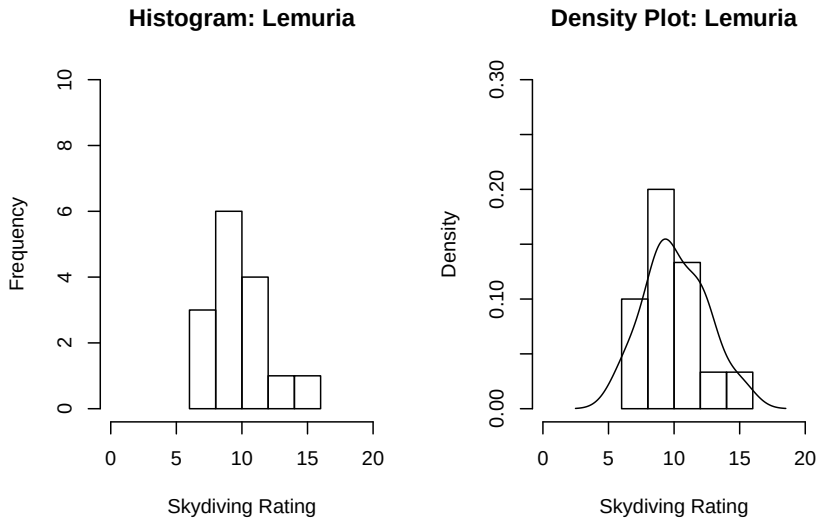

Figure 9: Histogram and density plots

```
> Bunyips <- matrix(c(49, 33, 40, 14), nrow=2, ncol=2, byrow=TRUE)
>
> colnames(Bunyips) <- c("Country","Metropolitan")
> rownames(Bunyips) <- c("Over 65's","Under 65's")
>
> Bunyips
Country Metropolitan
Over 65's       49           33
Under 65's      40           14
>
>
> mosaicplot(Bunyips,
+ main = "Bunyip Sightings",
+ col = FALSE)
```

This command only requires the name of the data frame, in this case `Bunyips`. Two additional options used here are `main` and `col`. The `main` option specifies a title and `col = FALSE` produces a plain plot with no fillings. By default, `mosaicplot()` produces a plot in shades of grey.

10.2.6 Line Plots with ggpubr

R's graphics package offers line plots with the `plot()` and `lines()` commands. These will generate plots with single or multiple lines. However they don't provide plots with error bars. Line plots with error bars are available with `ggline()` in the `ggpubr` package. `ggpubr` is an easy-to-use interface to ggplot2 which is based on Leland Wilkinson's 'The Grammar of Graphics' (1999). This package provides a range of publication quality plots.

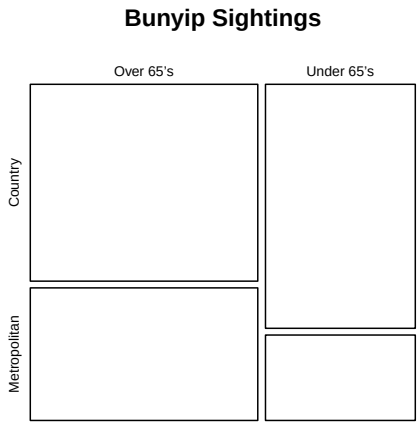

Figure 10: Mosaic Plot

For a full list, see the documentation.

These graphs have three steps.

1. The first step is to create an R object containing the basic graph.
2. This graph may then be modified with `ggpar()`. This command contains options to specify axis limits and scales, titles, plot orientation, ticks and labels and legends.
3. After reviewing the graph, chose a graphic device and save it.

`ggline()` accepts data in long format and may have up to two factors on a single graph. It produces graphs based on means or medians with error bars. These may indicate the standard error of the mean, standard deviations, confidence intervals for means and the inter-quartile range, median absolute deviation, and the range. Plots with a single factor and with two factors are given in Figure 12. They both show means with standard deviations for the `Experiment` data set.

The first stage in generating a graph with `ggpubr()` is to load the data in long format and format the factor columns as factors:

```
> Experiment <- read.csv("Experiment.txt", header = TRUE, sep = ",")
> attach(Experiment)
>
> factor(Treatment, levels = c("Control", "Treatment 1", "Treatment 2"),
+ ordered = TRUE)
[1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[16] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[31] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
```

```
[46] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[61] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[76] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
Levels: Control < Treatment 1 < Treatment 2
>
> factor(Diet, levels = c("Apples","Oranges"), ordered = FALSE)
[1] Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples
[10] Apples  Apples  Apples  Apples  Apples  Apples  Oranges Oranges Oranges
[19] Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges
[28] Oranges Oranges Oranges Apples  Apples  Apples  Apples  Apples  Apples
[37] Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples
[46] Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges
[55] Oranges Oranges Oranges Oranges Oranges Oranges Apples  Apples  Apples
[64] Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples  Apples
[73] Apples  Apples  Apples  Oranges Oranges Oranges Oranges Oranges Oranges
[82] Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges Oranges
Levels: Apples Oranges
>
```

The script below produces a line plot with one factor. This factor has three levels: `Control`, `Treatment 1` and `Treatment 2`. The first step is to generate the plot with `ggline()`. This requires the name of the data frame, the factor for the x axis, in this case `Treatment` which has three levels, and the name of the column that contains the responses for the y axis, in this case `Response`. The x and y axes can be renamed with `xlab` and `ylab` and the `add` option can add error bars. In this case, standard deviations are added with `"mean_sd"`. `ggline()` provides additional options, such as standard error of the mean and confidence intervals for means, and medians, interquartile ranges and median absolute deviations. For a complete list, see the package documentation. Firstly, load `ggpubr` and generate the basic plot:

```
library(ggpubr)

ggline(Experiment, x = "Treatment", y = "Response",
xlab = "Independant Variables", ylab = "Dependant Variable",
add = "mean_sd")
```

`ggline()` doesn't have an option for scale limits. These are edited with `ggpar()`. To do this, the line plot is created as an R object, then `ggpar()` adjusts the limits of the y axis with `ylim()`. The first part of the following code produces a line plot with a single factor and error bars showing standard deviations. This is an R object named `LinePlot`. The second part changes the y axis limits to 50–100 with `ylim()`. The values separated by a comma define the lower and upper limits of the scale respectively:

```
LinePlot <- ggline(Experiment, x = "Treatment", y = "Response",
xlab = "Independant Variables", ylab = "Dependant Variable",
add = "mean_sd")
```

```
ggpar(LinePlot,
ylim = c(50, 100))
```

Multiple groups may be added with the `group` option. In this example, it is the `Diet` factor from the Experiment dataset. This has two levels: Apples and Oranges. R will automatically generate differing line types and point shapes for each level of the factor. R will also generate a legend, and this can be placed at the `top` of the graph, which is the default, or at the `bottom`, `left` or `right` side with the `legend` option in `ggpar()`.

```
LinePlot <-
ggline(Experiment, x = "Treatment", y = "Response",
group = "Diet",
linetype = "Diet", shape = "Diet",
xlab = "Independant Variables", ylab = "Dependant Variable",
add = "mean_sd")

ggpar(LinePlot,
ylim = c(50, 100),
legend = "top")
```

10.3 Multiple plots in a single graphic

It is useful to compare graphs side by side and a helpful way of doing this is to place multiple plots in a single graph. R can do this with graphs produced by the `graphics` package, and with graphs produced by `ggpubr`. The procedures are somewhat different for each.

The first step in doing this with graphs generated by the `graphics` package is to divide the graphic device into rows and columns. This is done with `par(mfrow = c(Number of Rows, Number of Columns))`. If there were to be six graphs in three rows and two columns, the command is `par(mfrow = c(3, 2))`. If there are three graphs placed vertically on the page in one column, use `par(mfrow = c(1, 3))` for one column and three rows. The next step is to generate the plots.

In this example Density Plots and Quantile–Quantile Plots will be generated for the Atlantean and Lemurian respondents in the Skydiving data set. This will require two rows in two columns:

```
setwd("C:/Users/Michael/Desktop/R_Project")

Skydiving <- read.csv("Skydiving.txt", header = TRUE, sep = ",")
attach(Skydiving)

par(mfrow = c(2, 2))
```

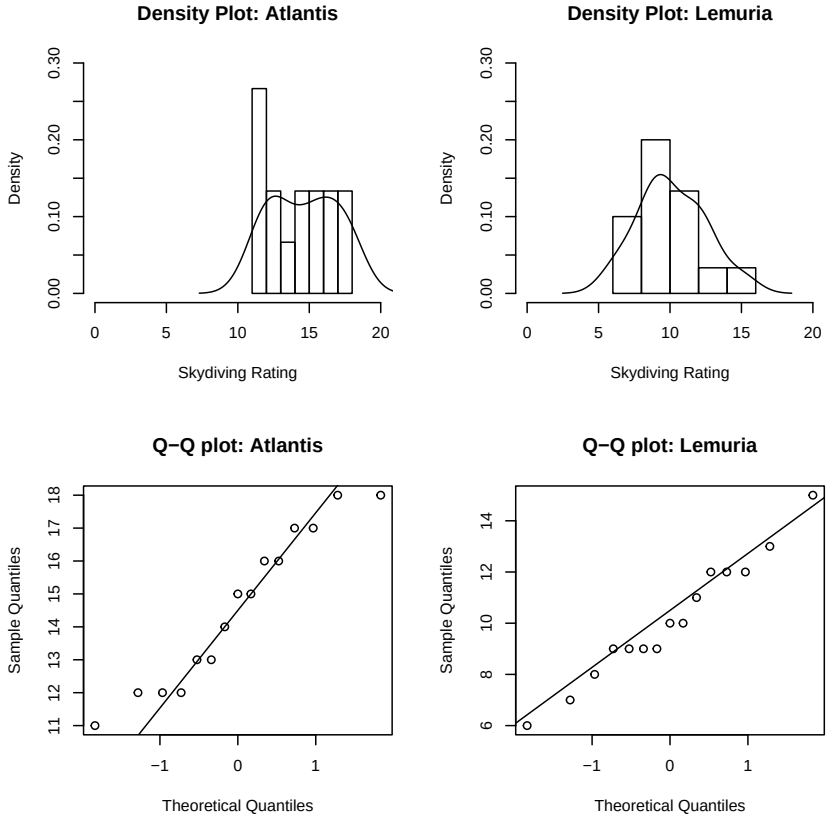

Figure 11: Multiple plots: Graphics

```
# Histograms
hist(Atlantis,
col = "transparent",
freq = FALSE,
xlim = c(0, 20),
ylim = c(0, 0.3),
main = "Density Plot: Atlantis",
xlab = "Skydiving Rating")
lines(density(Atlantis))

hist(Lemuria,
col = "transparent",
freq = FALSE,
xlim = c(0, 20),
ylim = c(0, 0.3),
main = "Density Plot: Lemuria",
xlab = "Skydiving Rating")
lines(density(Lemuria))
```

```
# Q-Q Normal Plots
qqnorm(Atlantis,
main = "Q-Q plot: Atlantis");qqline(Atlantis)

qqnorm(Lemuria,
main = "Q-Q plot: Lemuria");qqline(Lemuria)
```

Multiple plots are handled differently for graphs produced by `ggpubr()`. The first step is to generate the plots as R objects, then secondly place them on a single graphic with `ggarrange()`. This command requires the names of the plots, labels and legends for each one if required, and the number of columns and rows.

The following code generates the two line plots above, and places them on a single graphic. The first part of the code generates two line plots with `ggline()` named `LinePlot_1a` and `Lineplot_2a`:

```
LinePlot1 <- ggline(Experiment, x = "Treatment", y = "Response",
xlab = "Independant Variables", ylab = "Dependant Variable",
add = "mean_sd")

ggpar(LinePlot1,
ylim = c(50,100))


LinePlot2 <- ggline(Experiment, x = "Treatment", y = "Response",
group = "Diet",
linetype = "Diet", shape = "Diet",
xlab = "Independant Variables", ylab = "Dependant Variable",
add = "mean_sd")

ggpar(LinePlot2,
ylim = c(50, 100),
legend = "top")
```

These plots are then placed on a single graph with `ggarrange()`. The two R objects are `LinePlot1` and `LinePlot2`, labels are added for each plot, and they are arranged in a single row with `nrow = 1`. To place them in a single column, `ncol = 1` would be specified:

```
ggarrange(LinePlot1, LinePlot2,
labels = c("Single Factor Lineplot", "Two Factor Lineplot"),
ncol = 1)
```

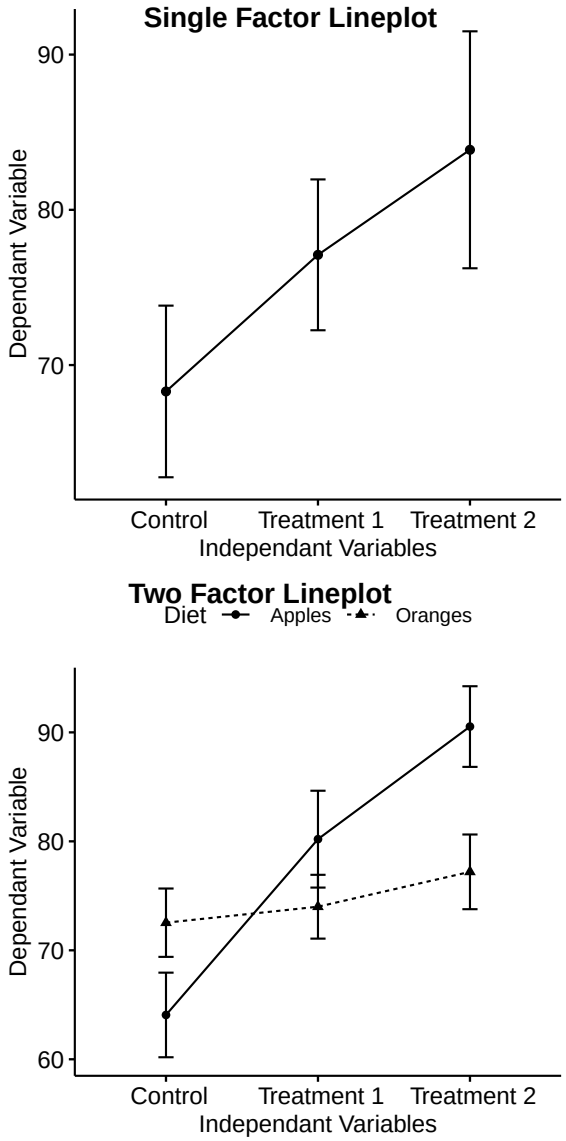

Figure 12: Multiple plots: ggplot2

10.4 R's Graphic Devices and Saving Graphs to File

R's graphic devices are where R sends the output of plotting commands. This may be to the screen or saved to a file. The default graphic device is the screen. If the graphic device is a file, it will be saved in the current working directory. A list of graphic devices is in Table 14.

Graphic Device	Command	File Extension
Screen (The default)	`windows()`	
Bitmap	`bmp()`	`.bmp`
Joint Photographic Experts Group	`jpeg()`	`.jpg`
Portable Document Format	`pdf()`	`.pdf`
Portable Network Graphics	`png()`	`.png`
Postscript / Encapsulated Postscript	`postscript()`	`.ps` or `.eps`
Scalable Vector Graphics	`svg()`	`.svg`
Tagged Image File Format	`tiff()`	`.tif`

Table 14: R's Graphic Devices

These graphic devices work as wrappers around the code that generates the graph. If the graph is to be saved to a file, the device is switched on prior to generating the graph and then switched off with `dev.off()`. This returns the device to the default output setting, which is the screen.

10.5 Choosing a File Format: Vector vs Raster Formats

10.5.1 Vector Formats: Postscript (ps), Encapsulated Postscript (eps) and Portable Document Format (pdf).

Vector formats are based on geometric concepts such as points, lines, curves and polygons. They have the advantages that they can easily be upsampled or downsampled (enlarged or shrunk) with no loss of quality, and that they have small file sizes. These formats are most often used for line art such as diagrams, graphs, 3D models, etc. R can produce graphs in Postscript, Encapsulated Postscript and Portable Document Formats.

Postscript and encapsulated postscript is a page formatting language and is the 'gold standard' for printed documents. A shortcoming with postscript is that it works with entire pages and so restricts the size of the output that can be produced. Encapsulated postscript format allows files of any dimensions and is the preferred format for graphs. They are both produced by the `postscript()` graphic device. For a postscript file, use the `ps` file extension, and for an encapsulated postscript file, use the `eps` file extension.

Although vector graphics may be upsampled or downsampled with no loss of quality, R's fonts are not vector graphics and they will degrade. R's default graph and font sizes are 7" square with 12 point fonts. If the graph is to be larger or smaller than this the font size should be increased or decreased accordingly.

Both postscript and pdf devices use a similar set of commands. At their most basic, they only require the name of the file to be saved into the current working directory. Both devices have options for changing height and width, and the size of the fonts for labels. It is normally best to keep the background as `transparent` if the graph it to be placed on a white background. If it is set to white and printed it will use a lot of white ink unnecessarily. Table 15 has a list of default settings for files saved as vector formats.

Graphic Parameter	Default
Size	7" × 7"
Font	Helvetica
Font Size	12pt
Background Colour	Transparent
Foreground Colour	Black
Measurement Unit	Inches

Table 15: R's Graphic Defaults: Vector Formats

The following script creates an encapsulated postscript file named `Filename.eps` in the current working directory. Size is set to 4" square with `width` and `height`. Because the size of the graph is reduced, the size of the font should also be reduced. In this case, it it reduced to 10 points with `pointsize`. R does not match font sizes to graph sizes and finding the appropriate size can be a bit hit and miss. It may take several attempts to find an aesthetically pleasing combination.

If an encapsulated postscript file (eps) is required of the postscript graphic device, the `onefile` option is set to `FALSE`. If postscript is required, `onefile` is set to `TRUE`. The width and height commands accept measurements in inches only. Both the background and foreground colours were left at their default settings. This example used the `MoneyLove` data set in the Appendix.

```
# Set the working directory and load the MoneyLove data

setwd("C:/Users/Michael/Desktop/R_Project")

MoneyLove <- read.csv("MoneyLove.txt", header = TRUE, sep = ",")
attach(MoneyLove)

# Turn on the Encapsulated Postscript graphics device, name the save file
and specify width and height
```

```
postscript(file = "Filename.eps", onefile = FALSE,
width = 4, height = 4, pointsize = 10)

# Create a scatterplot with 'plot'.

plot(MoneyLove$Salary, MoneyLove$Happiness,
xlim = c(0, 250000),
ylim = c(0, 10),
main = "Income vs Happiness",
xlab = "Income in ($)",
ylab = "Personal Happiness",
pch = 20)

# Turn the graphics device off

dev.off()
```

The resulting file can be found in the current working directory with an `.eps` suffix. It can be viewed or edited with GIMP or Inkscape.

pdf creates a file that may be viewed on a wide range of systems, and will preserve the document's original formatting and fonts. It can contain both vector and raster content and also other pdf files.

The following script generates a graph in pdf format. The graphic device and file extension are changed to `pdf` and the `onefile` option is omitted:

```
pdf(file = "Filename.pdf",
width = 4, height = 4, pointsize = 10)

plot(MoneyLove$Salary, MoneyLove$Happiness,
xlim = c(0, 250000),
ylim = c(0, 10),
main = "Income vs Happiness",
xlab = "Income in ($)",
ylab = "Personal Happiness",
pch = 20)

dev.off()

embedFonts("Filename.pdf", outfile = "Filename_Embedded.pdf")
```

An additional step for pdf files is to embed the fonts used in the graphic. This procedure includes a copy of the fonts used in the graph with the file. This is necessary because if the fonts are not embedded, and they are not present on other systems the file may be viewed on, they will be substituted with fonts present on the readers system, and the

graphic may not appear as the author intended. R doesn't embed fonts by default. They can be embedded with `grDevices embedFonts()` command. This package is loaded automatically when R loads so it doesn't need to be loaded seperately. `embedFonts()` is run after the graph is generated. It creates a new file with the required fonts and requires the name of the graphic file, the name of the new file and produces a file that is slightly larger.

svg format is similar to `ps` and `eps` formats. This means that plots in this format may be resized without losing quality. This format supports interactive graphs for web pages with Java. This may be done with the `plotly` package. This is somewhat outside the scope of this document, so if any additional information on this is required, please see the package documentation.

10.5.2 Raster Formats: Tagged Image File Format (tiff), Bitmap Image File (bmp), Joint Photographic Experts Group (jpeg) and Portable Network Graphic (png).

Raster format images are made up of grids of pixels. Each pixel has a value attached to it indicating it's colour, size and location. They have much larger file sizes than vector graphics. R can create raster graphics in four formats: `tiff`, `bmp`, `jpg` and `png`. `pdf` is also available in R. A `pdf` file can contain both vector and raster format images. These files are discussed further in the section on vector formats.

tiff format offers very high-quality images and has optional lossless file compression. It supports transparency. The high image quality comes with relatively large file sizes.

jpeg is a compressed file format and produces smaller file sizes at the cost of quality. The more these files are compressed, the lower the quality. If these files are opened and saved multiple times, quality will be degraded.

bmp stands for bitmap format. These files are uncompressed with large file sizes that are suitable for original and/or highly detailed images. They can be saved multiple times during editing without any loss of quality. It's drawback is that it was developed for Windows systems and may not display correctly on other operating systems.

png is a compressed format using lossless compression, meaning it can be compressed with no loss of quality and that it may be saved multiple times without image degradation. It supports background transparency, and it's files are somewhat larger than `jpeg`.

Table 16 lists some of R's default settings for raster formats. The default background for raster graphics is white. This may be overridden by `pars()`'s `bg` option and set to `transparent` if required.

The following code creates a file in `tiff` format called `Filename.tif` that is 4" square. Size is set with `width` and `height` options. Raster graphic devices accept measurement

Graphic Parameter	Default
Size	480px × 480px
Resolution	72ppi
Font	Arial
Font Size	12pt
Background Colour	White
Foreground Colour	Black

Table 16: R's Graphic Defaults: Raster Formats

units in pixels (`"px"`), inches (`"in"`), centimetres (`"cm"`) or millimetres (`"mm"`). This is specified by the `units` option. The font size for labels was set to 10 point with `pointsize`. R's default resolution is 72ppi (points per inch). This is designed for viewing graphics on-screen. If the image is to be printed, the resolution should be set to 300ppi. This gives enough resolution to appear crisp when printed. This is specified with `res`. After the plot is generated, the Graphic Device is reset to the default 'screen' setting with `dev.off()`:

```
tiff(filename = "Filename.tif", width = 4, height = 4, units = "in",
pointsize = 10, res = 300)

plot(MoneyLove$Salary, MoneyLove$Happiness,
xlim = c(0, 250000),
ylim = c(0, 10),
main = "Income vs Happiness",
xlab = "Income ($)",
ylab = "Personal Happiness",)

dev.off()
```

This produces a very high-quality file that is about 4.1mb in size. The file size can be reduced by creating it in `jpg` format. This is a compressed format and produces smaller file sizes. The `jpeg` graphic device has a `quality` option. This refers to the 'quality of the jpg' and is the amount of compression as a percentage. Smaller values produce more compressed, and smaller files, at the cost of image quality. A value of 50 produces a file 50% the size of the original, and a value of 25 produces a file 25% the size of the original. With quality set to 100%, the same image has a file size at about 60kb:

```
jpeg(filename = "Filename.jpg", width = 4, height = 4, units = "in",
quality = 100, pointsize = 10, res = 300)

plot(MoneyLove$Salary, MoneyLove$Happiness,
xlim = c(0, 250000),
ylim = c(0, 10),
main = "Income vs Happiness",
xlab = "Income ($)",
ylab = "Personal Happiness")
```

```
dev.off()
```

11 References

Anscombe, F. J. (1973). Graphs in statistical analysis. *The American Statistician, 27(1),* 17—21.

Camerer, C., Loewenstein, G., and Weber, M. (1989). The curse of knowledge in economic settings: An experimental analysis. *Journal of Political Economy, 97(5),* 1232–1254.

Fox, J. and Weisburg, S. (2011). *An R Companion to Applied Regression, 2nd Edition.* Thousand Oaks: Sage Publications.

Friendly, M. (2000). *Visualizing Categorical Data.* Cary, NC: SAS Institute.

Murrell, P. (2011). *R Graphics, 2nd Edition.* Boca Raton, Fl: Chapman and Hall / CRC.

The R Core Team. (2024a). *R Data Import/Export, v.4.4.0.* Retrieved from: http://www.cran.r-project.org

The R Core Team. (2024b). *R Installation and Administration, v.4.4.0.* Retrieved from: http://www.cran.r-project.org

The R Core Team. (2024c). *R: A language and Environment for Statistical Computing. Reference Index, v4.4.0.* Retrieved from: http://www.cran.r-project.org

Venables, W. N. and Ripley, B. D. (2000). *S programming.* New York: Springer-Verlag New York Inc.

Venables, W.N. Smith, D.M. and The R Core Team. (2024d). *An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics, v.4.4.0.* Retrieved from: http://www.cran.r-project.org

Wilkinson, G. N. and Rogers, C. E. (1973). Symbolic description of factorial models for analysis of variance. *Journal of the Royal Statistical Society. Series C (Applied Statistics), 22,* 392—399.

Wilkinson, L. (1999). *The Grammar of Graphics.* New York: Springer-Verlag, New York Inc.

12 Appendices

Appendix A Point shapes and line types available in R

A graphic showing the point shapes and line types available in R for graphs can be generated with the following code from the ggpubr package:

```
ggpubr::show_point_shapes()
ggpubr::show_line_types()
```

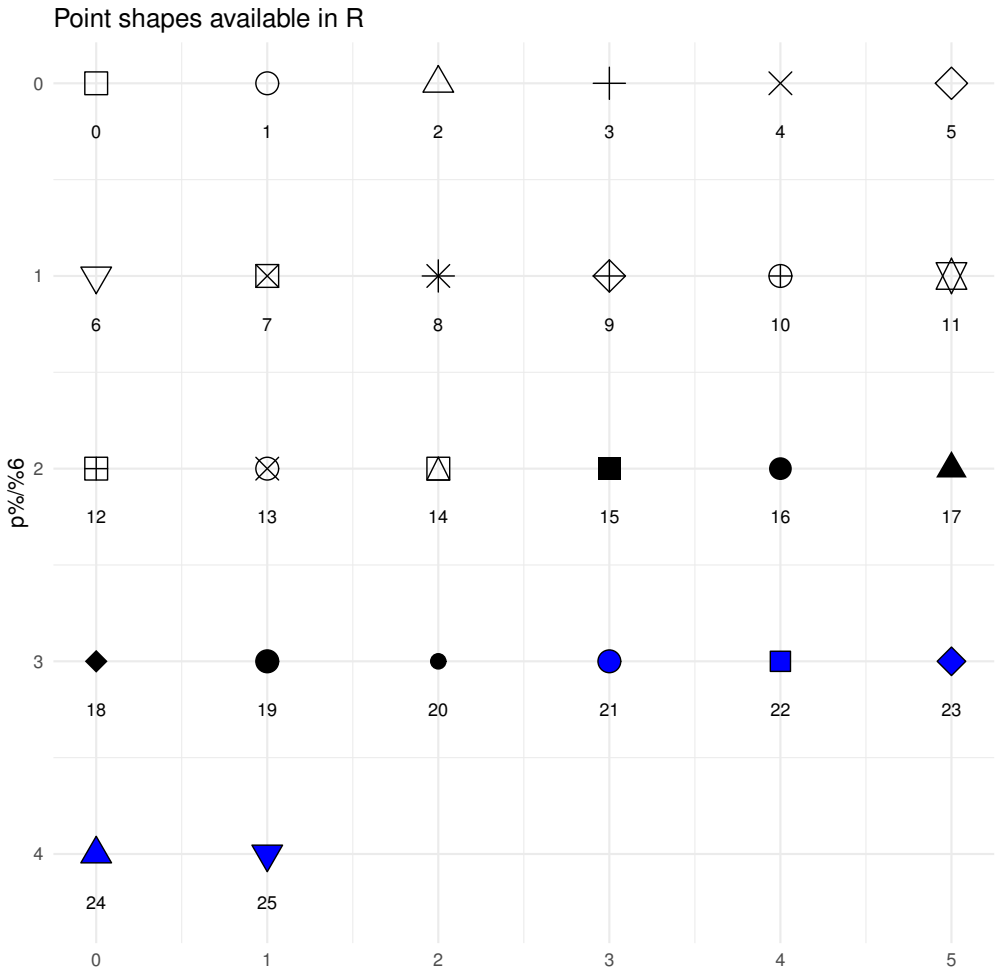

Figure 13: Point Shapes Available in R.

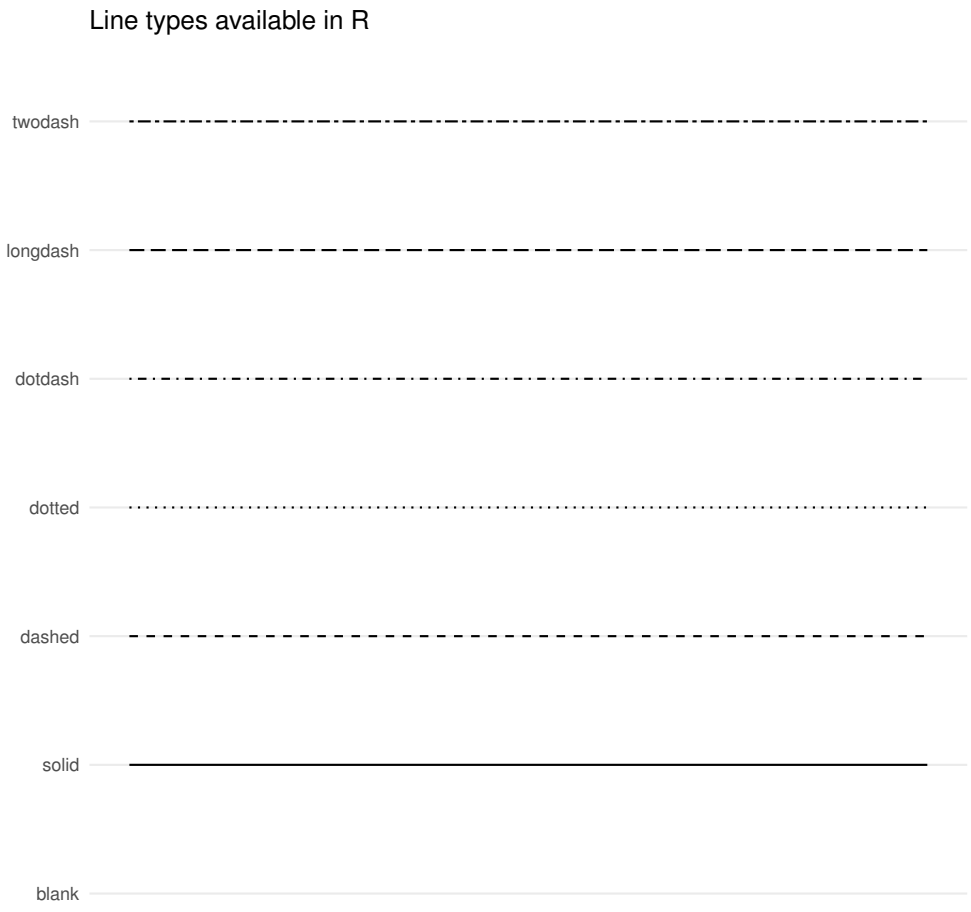

Figure 14: Line Types Available in R.

Appendix B Datasets

Appendix B.A Acheulean

Filename: Acheulean.txt

Notes: These are imagined user ratings on a 1-5 scale of materials for tools in Acheulean toolsets:
5 = I love it, 4 = A nice piece of kit, 3 = Good enough, 2 = Myeah and 1 = I hate it

ID	Basalt	Flint
1	2	3
2	3	5
3	1	4
4	2	5
5	2	4
6	3	3
7	4	3
8	2	4
9	3	4
10	4	5
11	3	5
12	3	4
13	1	3
14	2	4
15	1	5
16	3	3
17	3	4
18	2	2
19	2	2
20	1	3
21	4	4
22	3	2
23	3	4
24	2	2
25	2	4

Appendix B.B Bunyips

Filename: Bunyips.txt

Notes: Bunyip sightings in country and metropolitan areas by people aged under 65 and over 65.

	Country	Metropolitan
Over 65s	49	33
Under 65s	40	14

Appendix B.C Curse

Filename: Curse.txt

This data gives some fictive data relating to the ‘Curse of Knowledge’.

The Curse column contains a measure of the level of cursedness on a 0--20 scale,
Courses is the number of courses taken by the participants on a 0--10 scale, Expertis

ID	Curse	Courses	Expertise	Conscientiousness
1	5	3	2	6
2	2	4	3	9
3	15	8	3	7
4	18	7	3	6
5	10	4	4	6
6	8	6	3	6
7	7	5	4	9
8	12	9	2	10
9	11	6	2	10
10	6	2	2	6
11	15	5	3	5
12	9	8	3	6
13	4	6	4	8
14	10	7	3	7
15	12	4	3	7
16	8	4	3	7
17	6	1	3	5
18	14	5	3	6
19	5	6	3	5
20	19	7	2	10

Appendix B.D Experiment

Filename: Experiment.txt

Notes: Data from an experimental study with three treatments and two diets. Responses were rated on a 50---100 point scale.

ID	Response	Treatment	Diet
1	65	Control	Apples
2	68	Control	Apples
3	62	Control	Apples
4	63	Control	Apples
5	64	Control	Apples
6	55	Control	Apples
7	66	Control	Apples
8	69	Control	Apples
9	63	Control	Apples
10	70	Control	Apples
11	66	Control	Apples
12	63	Control	Apples
13	58	Control	Apples
14	64	Control	Apples
15	65	Control	Apples
16	73	Control	Oranges
17	68	Control	Oranges
18	75	Control	Oranges
19	68	Control	Oranges
20	70	Control	Oranges
21	74	Control	Oranges
22	72	Control	Oranges
23	75	Control	Oranges
24	79	Control	Oranges
25	73	Control	Oranges
26	76	Control	Oranges
27	70	Control	Oranges
28	69	Control	Oranges
29	74	Control	Oranges
30	72	Control	Oranges
1	86	Treatment 1	Apples
2	78	Treatment 1	Apples
3	81	Treatment 1	Apples
4	82	Treatment 1	Apples
5	72	Treatment 1	Apples
6	79	Treatment 1	Apples
7	87	Treatment 1	Apples

```
 8,       77, Treatment 1,  Apples
 9,       78, Treatment 1,  Apples
10,       83, Treatment 1,  Apples
11,       82, Treatment 1,  Apples
12,       83, Treatment 1,  Apples
13,       77, Treatment 1,  Apples
14,       85, Treatment 1,  Apples
15,       73, Treatment 1,  Apples
16,       74, Treatment 1, Oranges
17,       77, Treatment 1, Oranges
18,       70, Treatment 1, Oranges
19,       77, Treatment 1, Oranges
20,       79, Treatment 1, Oranges
21,       79, Treatment 1, Oranges
22,       72, Treatment 1, Oranges
23,       70, Treatment 1, Oranges
24,       75, Treatment 1, Oranges
25,       74, Treatment 1, Oranges
26,       71, Treatment 1, Oranges
27,       73, Treatment 1, Oranges
28,       74, Treatment 1, Oranges
29,       72, Treatment 1, Oranges
30,       73, Treatment 1, Oranges
 1,       92, Treatment 2,  Apples
 2,       90, Treatment 2,  Apples
 3,       87, Treatment 2,  Apples
 4,       90, Treatment 2,  Apples
 5,       88, Treatment 2,  Apples
 6,       85, Treatment 2,  Apples
 7,       93, Treatment 2,  Apples
 8,       90, Treatment 2,  Apples
 9,       92, Treatment 2,  Apples
10,       86, Treatment 2,  Apples
11,       98, Treatment 2,  Apples
12,       96, Treatment 2,  Apples
13,       94, Treatment 2,  Apples
14,       90, Treatment 2,  Apples
15,       87, Treatment 2,  Apples
16,       76, Treatment 2, Oranges
17,       72, Treatment 2, Oranges
18,       76, Treatment 2, Oranges
19,       82, Treatment 2, Oranges
20,       80, Treatment 2, Oranges
21,       75, Treatment 2, Oranges
```

```
22,       77, Treatment 2, Oranges
23,       78, Treatment 2, Oranges
24,       79, Treatment 2, Oranges
25,       74, Treatment 2, Oranges
26,       75, Treatment 2, Oranges
27,       76, Treatment 2, Oranges
28,       81, Treatment 2, Oranges
29,       84, Treatment 2, Oranges
30,       73, Treatment 2, Oranges
```

Appendix B.E MoneyLove

Filename: MoneyLove

Notes: The participants income in $, personal happiness rated on a 1---10 scale with 1 = Clinical Depression and 10 being enraptured and the amount of love in their lives rated on a 0---50 scale with 0 being none and 50 being bliss.

ID	Salary	Love	Happiness
1	50000	47	5
2	125000	27	2
3	100000	38	4
4	27000	44	5
5	175000	22	2
6	35000	17	2
7	82000	30	4
8	107000	21	1
9	117000	20	3
10	190000	37	4
11	67000	37	4
12	157000	43	3
13	185000	37	5
14	92000	44	4
15	137000	24	2
16	88000	20	3
17	73000	31	4
18	146000	20	3
19	163000	30	4
20	73000	47	5

Appendix B.F Pareidolia

Filename: Pareidolia_BG.txt and Pareidolia_BW.txt

Notes: This data file contains fictitious estimates of whether or not respondants were

The BG and WG versions of the data refer to Between Groups and Within Groups experimen

Pareidolia_BG

ID	CloudHeight	Rating
1	High	4
2	High	5
3	High	4
4	High	3
5	High	5
6	Medium	2
7	Medium	3
8	Medium	2
9	Medium	3
10	Medium	4
11	Low	1
12	Low	3
13	Low	2
14	Low	3
15	Low	2

Pareidolia_WG

ID	Cloud Height	Rating
1	High	4
2	High	5
3	High	4
4	High	3
5	High	5
1	Medium	2
2	Medium	3
3	Medium	2
4	Medium	3
5	Medium	4
1	Low	1
2	Low	3
3	Low	2

```
4,  Low,          3
5,  Low,          2
```

Appendix B.G Placebo

Filename: Placebo.txt

Note: Ratings of a placebo and a treatment. They rated suffering on a 10 point scale with 1 = Intense to 10 = None.

ID	Placebo	Treatment
1	3	5
2	2	4
3	4	8
4	7	5
5	5	4
6	6	6
7	5	8
8	3	6
9	2	4
10	7	9
11	6	6
12	5	5
13	6	8
14	3	8
15	4	7
16	5	5
17	4	9
18	2	6
19	2	5
20	4	7
21	1	5
22	1	6
23	4	8
24	4	7
25	5	5

Appendix B.H Skydiving

Filename: Skydiving.txt

Notes: Ratings of skydiving skills on a 0---20 scale with 0 being Poor and 20 being excellent.

ID,	Atlantis,	Lemuria
1,	12,	9
2,	18,	7
3,	13,	6
4,	15,	12
5,	11,	11
6,	16,	10
7,	12,	9
8,	14,	9
9,	17,	12
10,	12,	15
11,	13,	9
12,	17,	13
13,	15,	8
14,	16,	12
15,	18,	10

Appendix B.I Tea

Filename: Tea.txt

This data asked 15 respondants how often they drank tea, gardened and rode horses.
It contains three variables named Tea Drinker, Gardiner, and Horse Rider.

The ratings are on a 1--5 scale where 1 = Never, 2 = Almost Never, 3 = Sometimes, 4 =

ID	Tea Drinker	Gardiner	Horse Rider
1	5	4	2
2	4	5	2
3	5	4	3
4	3	4	1
5	2	3	2
6	1	1	1
7	3	4	2
8	4	5	2
9	5	4	2
10	2	1	4
11	5	5	5
12	5	4	5
13	4	2	5
14	5	3	4
15	2	2	3

13 GNU Free Documentation License

Version 1.2, November 2002

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for

automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. verbatim COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you

begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.