

How-to guide for analyzing data and conducting statistical tests in R/ R Studio

PSYC2111/3111

Last updated: 15 July 2019

Table of Contents

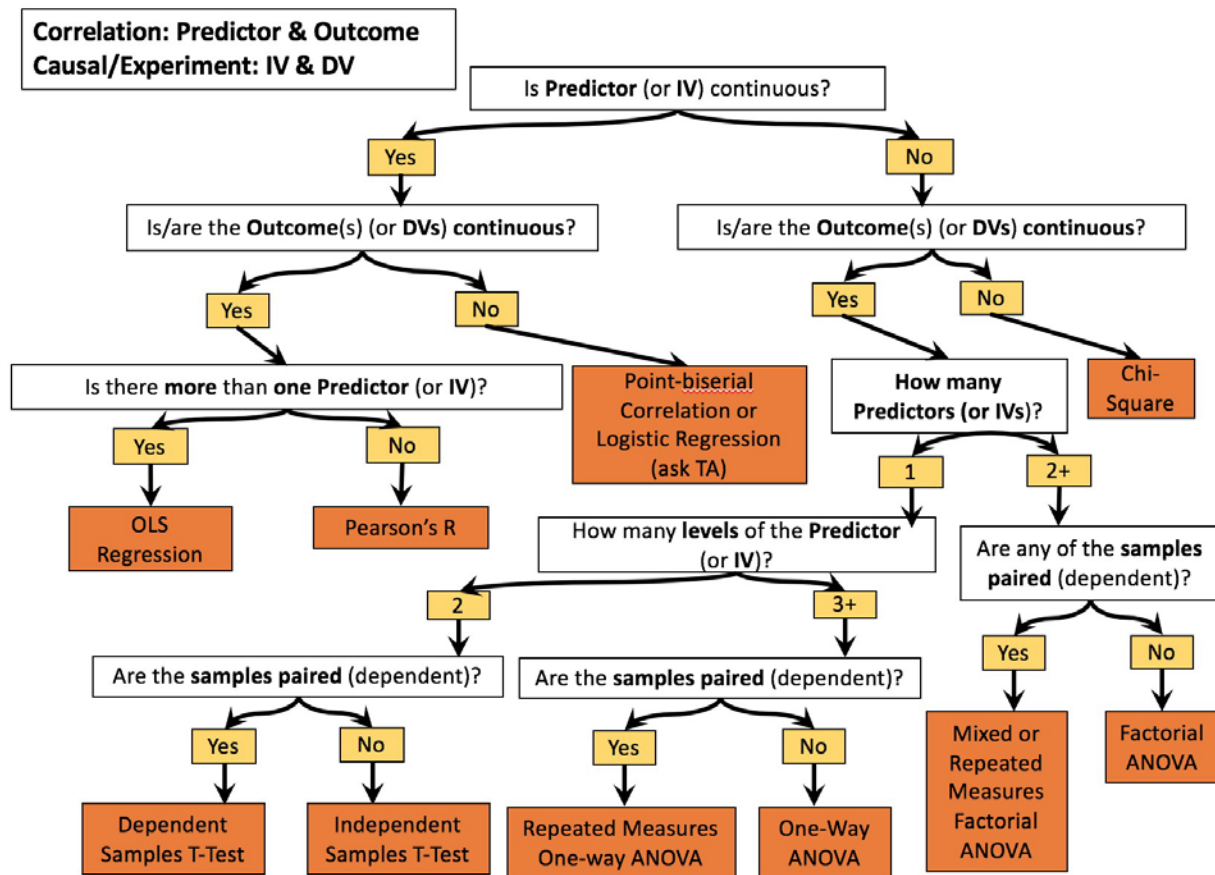
Acknowledgements	2
Flowchart: Deciding which statistical test you need to run	3
Introduction	3
Section 1: Some basics.....	4
1.1 Downloading and installing R and R Studio	4
1.2 The R/R Studio environment	7
1.3 Entering data in Excel.....	8
1.4 Importing data into R.....	9
1.5 Saving your data and R Script to use later	10
1.6 Loading a previously-saved RData file	10
1.7 Loading necessary packages	10
Section 2: 'Best Practices' in R Studio and Basic troubleshooting.....	11
2.2 Basic Troubleshooting in R Studio.....	11
Section 3: Descriptive Statistics.....	12
3.1 Descriptive Statistics for continuous variables.....	12
3.2 Descriptive statistics for categorical or grouping variables.....	13
3.3 Cronbach's alpha for the internal reliability of survey scales	15
Section 4: Inferential Statistics: Chi-Squared, t-tests, ANOVAs.....	17
4.1 Overall Data Organization for Chi-Squared, t-tests, ANOVAs	17
4.2 Categorical predictor(s) (IVs), continuous outcome(s) (DVs)	18
4.2.1 One predictor (IV) with 2 levels, unpaired (independent) samples: Independent Samples T-test.....	18
4.2.2 One predictor (IV) with 2 levels, paired (dependent) samples: Paired T-test.....	19
4.2.3 One predictor (IV) with 3+ levels, unpaired (independent) samples: One-way ANOVA	20

4.2.4 One predictor (IV) with 3+ levels, paired (dependent) samples: Repeated Measures ANOVA.....	22
4.2.5 Two or more predictors (IV), unpaired (independent) samples: Factorial ANOVA between subjects.....	24
4.2.6 Two or more predictors (IV), paired (dependent) samples: Repeated Measures Factorial ANOVA.....	26
4.3 Categorical predictor(s) (IVs), categorical outcome(s) (DVs).....	28
4.3.1 Chi-square Test of Independence.....	28
Section 5: Inferential Statistics: Correlation and Regression	30
5.1 Overall Data Organization for Correlation and Regression.....	30
5.2 Continuous predictor(s) (IVs), continuous outcome(s) (DVs).....	30
5.2.1 One predictor (IV) and one outcome (DV): Simple Correlation.....	30
5.2.2 One predictor (IV) and one outcome (DV): Simple Regression.....	31
5.2.3 More than one predictor (IV): Multiple Regression.....	32
5.3 Continuous predictor(s) (IVs), categorical outcome(s) (DVs)	34
5.3.1 Logistic Regression.....	34
Section 6: Graphing in R/R Studio and Alternatives to Graphing in R/R Studio.....	35
6.1 Creating Graphs in R/R Studio	35
6.2 Creating Graphs in Excel.....	39

Acknowledgements

This R book was created by Emma Johnson largely using materials from Arielle Gillman, as well as the generous help of Angela Bryan, Josh Correll, Steff Guillermo, Tiffany Ito, Matthew Keller, Diane Sasnett-Martichuski, Jennifer Stratford, and Katie Wolsiefer.

Flowchart: Deciding which statistical test you need to run



Follow the arrows! Then flip to the section of this book that describes the R code you need to run that test :)

Introduction

This "cookbook" is designed primarily for students in PSYC classes 2111 and 3111, statistics and research methods. It is not an R handbook per se, but is intended to help bridge the gap between what you learn in the stats and methods classes and the tests that you need to run in R to complete assignments (and maybe analyze your own data!). For example, if you are given a dataset that you need to analyze, you can consult the flowchart at the beginning of the book, decide which test you need to perform, and then flip to the section describing the R code needed to run that particular test.

The **first section** covers some R/R Studio basics, such as entering data into Excel and then importing that data into R Studio. The **second section** gives you tips for how to successfully navigate R/R Studio, including 'best practices' for using R/R Studio and how to troubleshoot in R/R Studio. The **third section** briefly covers descriptive statistics, such as how to get the mean and standard deviation for your variables of interest. The **fourth and fifth sections** introduces inferential statistics - once you've decided which test you need to

run, you can flip to the section that covers the R code and APA guidelines for reporting that statistical test. When in doubt, consult your TA or professor! 😊. Finally, the **sixth section** describes how to create graphs in R/R Studio and in Excel.

Good luck and have fun!

Section 1: Some basics

1.1 Downloading and installing R and R Studio

Using R version 3.6.1 (or whatever is the latest release)

R is command-line-driven statistical software that you can download for free at <http://cran.us.r-project.org/>. R works on both Mac and PC. R studio is an add-on program that is a friendlier way to use R. We will use excel to enter data, and then import the data into R studio to analyze data or run descriptive stats. Some students find it helpful to download the programs to their personal computers to practice more at home. Steps for Installing **R** and **R Studio** on your personal computer:

1) Download the base distribution of **R** for your operating system from: <http://cran.us.r-project.org/>. Under the **Download and Install R** section, click on your operating system:



CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)
[The R Journal](#)

Software
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages. **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-07-05, Action of the Toes) [R-3.6.1.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

For MacOS X: Click on R-3.6.1.pkg if you have El Capitan OS (X 10.11) or higher (Sierra). Click on R-3.3.3.pkg if you have Mavericks. Click on R-3.2.1-snowleopard.pkg if you have Snowleopard up through Mountain Lion. (Note that this number (3.2.1 or 3.2.2) will change depending on whatever the latest R version is called - download the latest version!)

R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) [here](#). Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the [old](#) directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

As of 2016/03/01 package binaries for R versions older than 2.12.0 are only available from the [CRAN archive](#) so users of such versions should adjust the CRAN mirror setting accordingly.

R 3.6.1 "Action of the Toes" released on 2019/07/05

Important: since R 3.4.0 release we are now providing binaries for OS X 10.11 (El Capitan) and higher using non-Apple toolkit to provide support for OpenMP and C++17 standard features. To compile packages you may have to download tools from the [tools](#) directory and read the corresponding note below.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type

```
md5 R-3.6.1.pkg
in the Terminal application to print the MD5 checksum for the R-3.6.1.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
pgutill --check-signature R-3.6.1.pkg
```

Latest release:

[R-3.6.1.pkg](#)
MD5=sha1: 27966621103666621944714926992
SHA1=sha1: 46521264011310326717954646027194161760
(ca. 761MB)

R 3.6.1 binary for OS X 10.11 (El Capitan) and higher, signed package. Contains R 3.6.1 framework, R.app GUI 1.70 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 5.2. The latter two components are optional and can be omitted when choosing "custom install", they are only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires [XQuartz](#) to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version.

Important: this release uses Clang 7.0.0 and GNU Fortran 6.1, neither of which is supplied by Apple. If you wish to compile R packages from sources, you will need to download and install those tools - see the [tools](#) directory.

News features and changes in the R.app Mac GUI

[NEWS](#) (for Mac GUI)

[Mac-GUI-1.70 tar.gz](#)
MD5=sha1: 1452223254400223693883005

Sources for the R.app GUI 1.70 for Mac OS X. This file is only needed if you want to join the development of the GUI, it is not intended for regular users. Read the `INSTALL` file for further instructions.

Note: Previous R versions for El Capitan can be found in the [el-capitan/base](#) directory.

Binaries for legacy OS X systems:

[R-3.3.3.pkg](#)
MD5=sha1: 8938e102f023e666a1994e4802f0f0f
SHA1=sha1: 3ae71900761591958505c03e459725511e434027
(ca. 711MB)

R 3.3.3 binary for Mac OS X 10.9 (Mavericks) and higher, signed package. Contains R 3.3.3 framework, R.app GUI 1.69 in 64-bit for Intel Macs, Tcl/Tk 8.6.0 X11 libraries and Texinfo 5.2. The latter two components are optional and can be omitted when choosing "custom install", it is only needed if you want to use the `tcltk` R package or build package documentation from sources.

Note: the use of X11 (including `tcltk`) requires [XQuartz](#) to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your OS X to a new major version.

[R-3.2.1-snowleopard.pkg](#)
MD5=sha1: 126680111440e971691c09145665
SHA1=sha1: 6e691d0128c225240531c142929493e4e0d
(ca. 681MB)

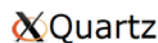
R 3.2.1 legacy binary for Mac OS X 10.6 (Snow Leopard) - 10.8 (Mountain Lion), signed package. Contains R 3.2.1 framework, R.app GUI 1.66 in 64-bit for Intel Macs.

This package contains the R framework, 64-bit GUI (R.app), Tcl/Tk 8.6.0 X11 libraries and Texinfo 5.2. GNU Fortran is **NOT** included (needed if you want to compile packages from sources that contain FORTRAN code) please see [the tools directory](#).

NOTE: the binary support for OS X before Mavericks is being phased out, we do not expect further releases!

o Under MacOS X, double-click the R-3.6.1.pkg (or R-3.3.3.pkg or R-3.2.1.pkg) contained in the disk image.

NOTE FOR MACS: You probably need **XQuartz** installed on your Mac. The latest version is a different number from this picture. You can do that through <https://www.xquartz.org/>



Home	The XQuartz project is an open-source effort to develop a version of the X.Org X Window System that runs on OS X. Together with supporting libraries and applications, it forms the X11.app that Apple shipped with OS X versions 10.5 through 10.7.		
Releases	Quick Download		
Support	Download	Version	Released
Contributing	XQuartz-2.7.11.dmg	2.7.11	2016-10-29
Bug Reporting			Info
GitHub			For OS X 10.6.3 or later
	License Info		
	An XQuartz installation consists of many individual pieces of software which have various licenses. The X.Org software components' licenses are discussed on the X.Org Foundation Licenses page . The quartzwm window manager included with the XQuartz distribution uses the Apple Public Source License Version 2 .		
	Web site based on a design by Kula J. Hickey for the XQuartz project. Web site content distribution services provided by CloudBees		
	Distributed by Frog Bintray		

For Windows: Under Subdirectories, click on the Install R for the first time and then on Download R 3.6.1 for Windows. Once downloaded, simply double-click to install R.



CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

[About R](#)
[R Homepage](#)
[The R Journal](#)

[Software](#)
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

[Documentation](#)
[Manuals](#)
[FAQs](#)
[Contributed](#)

R for Windows

Subdirectories:

[base](#)

Binaries for base distribution. This is what you want to [install R for the first time](#).

[contrib](#)

Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on [third party software](#) available for CRAN Windows services and corresponding environment and make variables.

[old contrib](#)

Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).

[Rtools](#)

Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

Once downloaded, double-click the executable *Download R 3.6.1 for Windows* (you should run R with administrator privileges to install packages: Right click the R icon and select "Run as administrator."). There are also additional instructions if you have problems under the *Download* link.

EVERYONE:

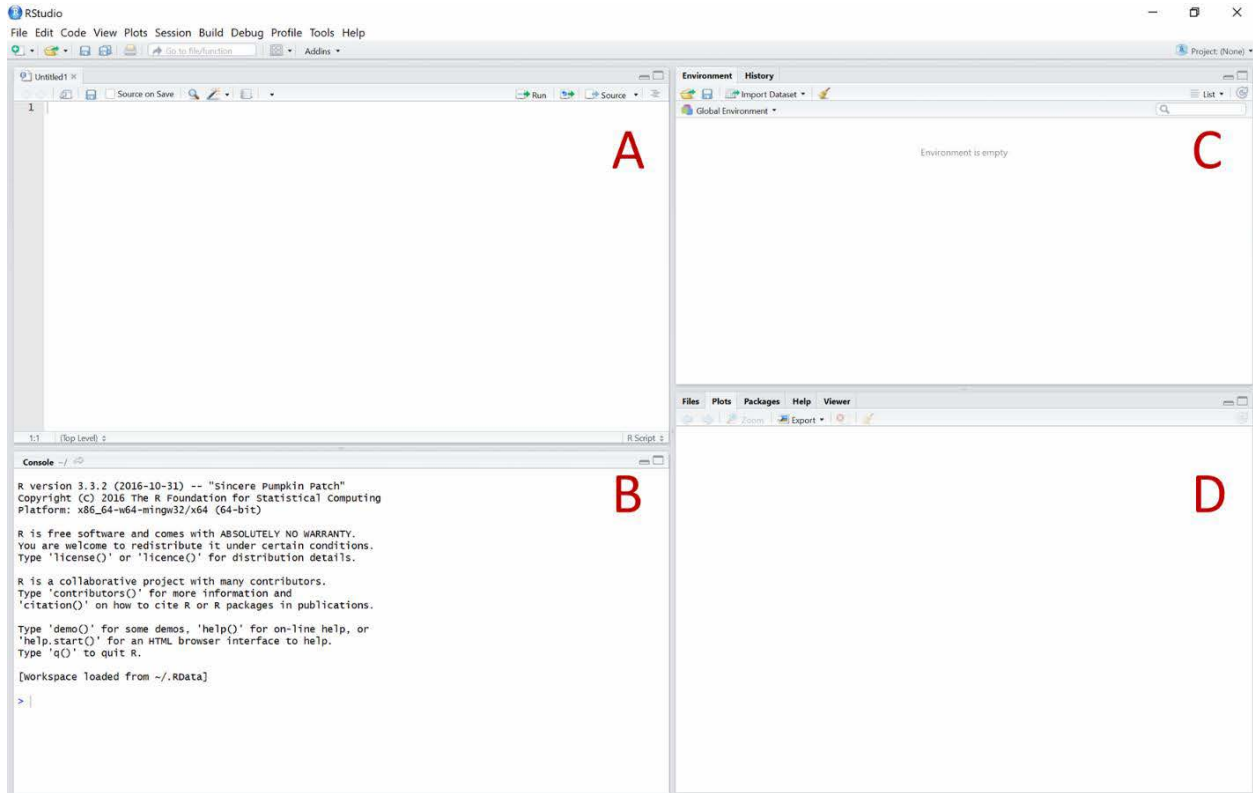
3) Download R studio at <https://www.rstudio.com/products/rstudio/download/>. Scroll to the bottom of the page. Under *Installers*, choose the right one for your system.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.2.1335 - Windows 7+ (64-bit)	126.9 MB	2019-04-08	d0e2470f1f8ef4cd35a669aa323a2136
RStudio 1.2.1335 - Mac OS X 10.12+ (64-bit)	121.1 MB	2019-04-08	6c570b0e2144583f7c48c284ce299eef
RStudio 1.2.1335 - Ubuntu 14/Debian 8 (64-bit)	92.2 MB	2019-04-08	c1b07d0511469abfe582919b183eee83
RStudio 1.2.1335 - Ubuntu 16 (64-bit)	99.3 MB	2019-04-08	c142d69c210257fb10d18c045fff13c7
RStudio 1.2.1335 - Ubuntu 18/Debian 10 (64-bit)	100.4 MB	2019-04-08	71a8d1990c0d97939804b46cfb0aea75
RStudio 1.2.1335 - Fedora 19/RedHat 7 (64-bit)	114.1 MB	2019-04-08	296b6ef88969a91297fab6545f256a7a
RStudio 1.2.1335 - Debian 9 (64-bit)	100.6 MB	2019-04-08	1e32d4d6f6e216f086a81ca82ef65a91
RStudio 1.2.1335 - OpenSUSE 15 (64-bit)	101.6 MB	2019-04-08	2795a63c7efd8e2aa2dae86ba09a81e5
RStudio 1.2.1335 - SLES/OpenSUSE 12 (64-bit)	94.4 MB	2019-04-08	c65424b06ef6737279d982db9eefcae1

Once downloaded, **you can just open R studio** (i.e., you don't need to open both R and R studio).

Here is what R studio should look like when you open it.



The R Studio layout; in this example, the console is in the bottom left and the source window (where we can open a R script) is in the top left.

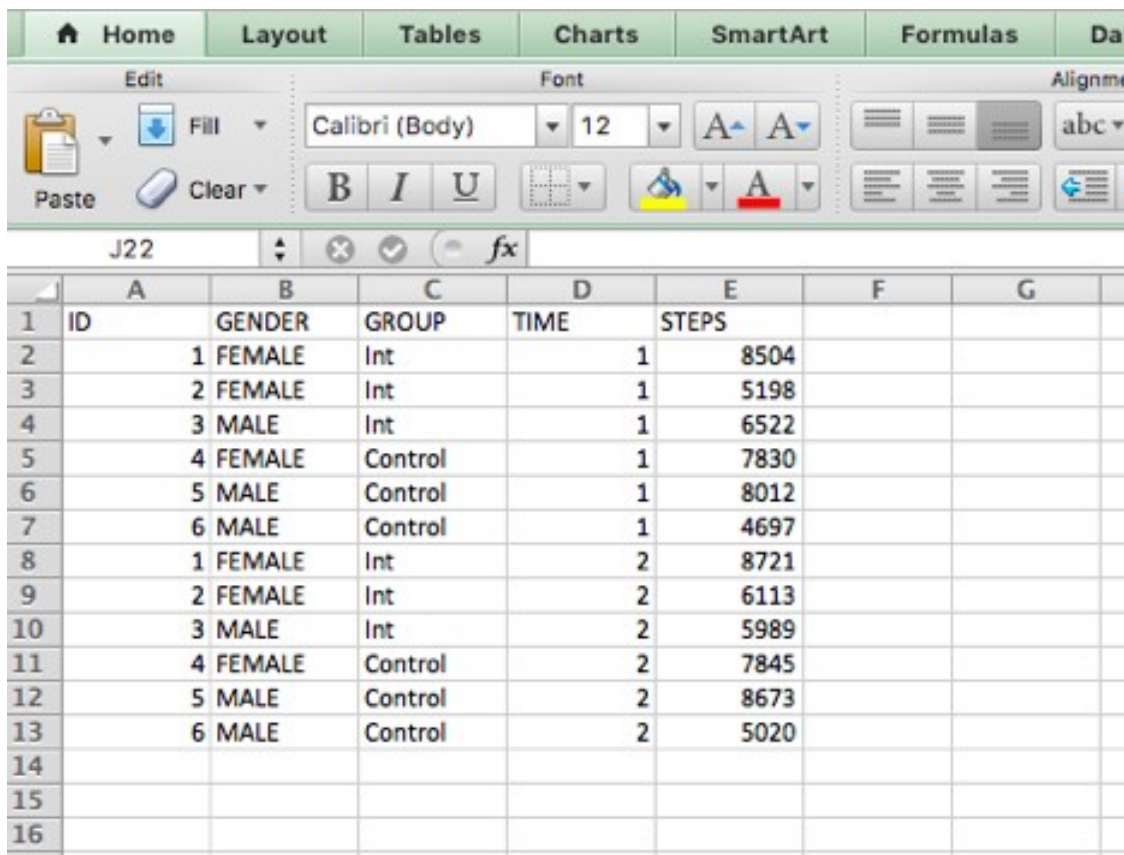
1.2 The R/R Studio environment

When you first open R Studio, you will see several different panels - a **source window**, where you can create an R script and run it (A); a **console**, where you can type/run code, calculations, etc. (B); a window with **environment** and **history** tabs, where you can see the R objects in your working environment or the history of your previous R commands (C); and a window with **Files**, **Plots**, **Packages**, and the **Help** tab (D).

We **highly** recommend saving all of your commands in an R Script *every time you open R Studio*. Do not just type your commands in the console! If you type them into an R Script, you can type Command + Return (Enter) and the commands will automatically run in the console. This way, you can save your commands in the R Script and access it later to easily re-run your analyses if you need to.

1.3 Entering data in Excel

- The first line in your data file should be the variable name. Be sure not to use spaces in the names of variables, files, etc. Instead, you can use an underscore_to_separate_words. We recommend using short abbreviations with no spaces for variable and dataset names.
- Each inferential statistical test (see pages 16 and 25 below) requires that the data files be set up in a specific way. For each of these tests, a sample of how each data file should be set up is included with the instructions for how to run each statistical test.
- Enter your data into the corresponding columns. IF YOU HAVE ANY MISSING DATA, LEAVE THOSE CELLS BLANK. Below is an example of a dataset created in Excel:

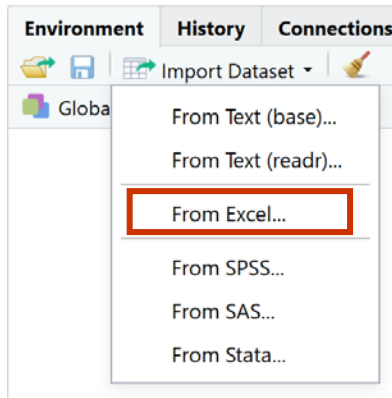


The image shows a screenshot of the Microsoft Excel interface. The ribbon at the top includes 'Home', 'Layout', 'Tables', 'Charts', 'SmartArt', 'Formulas', and 'Data'. The 'Home' ribbon is active, showing options for 'Edit', 'Font', and 'Alignment'. The font is set to 'Calibri (Body)' size 12. Below the ribbon, the formula bar shows 'J22'. The spreadsheet grid displays a dataset with the following columns: ID, GENDER, GROUP, TIME, and STEPS. The data is as follows:

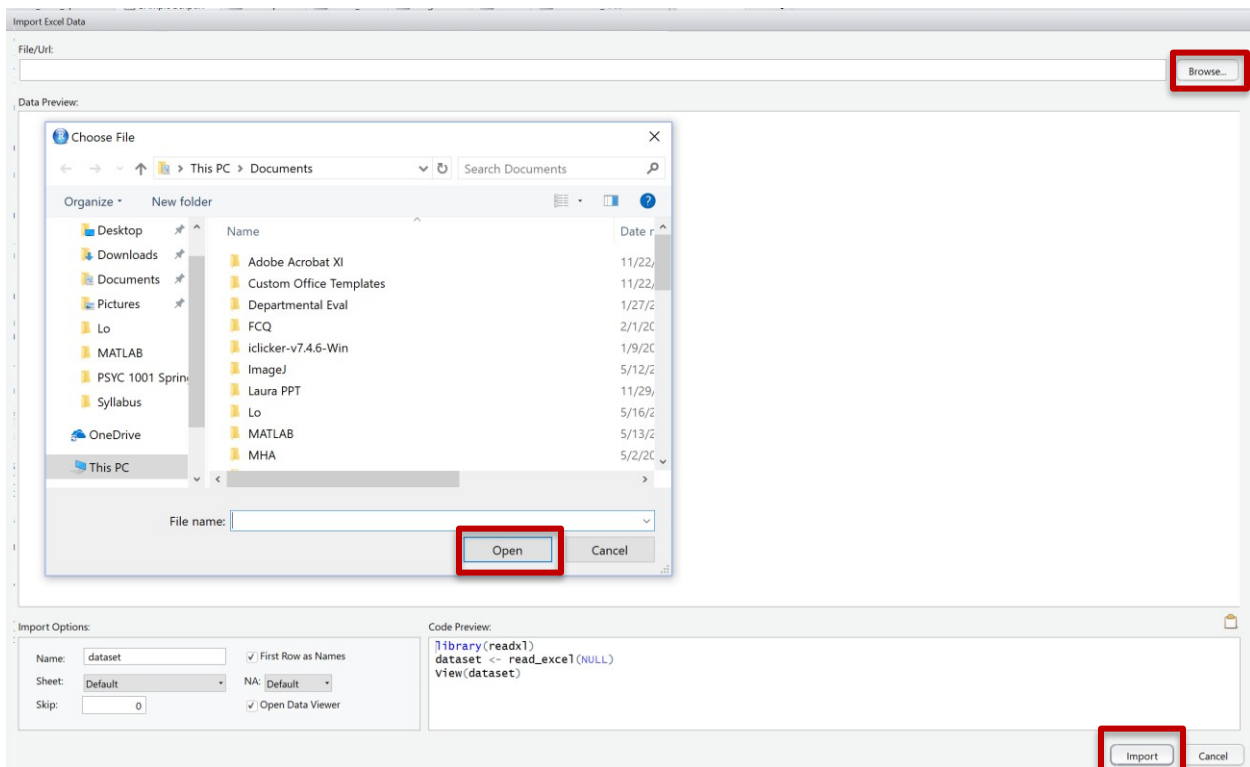
	A	B	C	D	E	F	G
1	ID	GENDER	GROUP	TIME	STEPS		
2	1	FEMALE	Int	1	8504		
3	2	FEMALE	Int	1	5198		
4	3	MALE	Int	1	6522		
5	4	FEMALE	Control	1	7830		
6	5	MALE	Control	1	8012		
7	6	MALE	Control	1	4697		
8	1	FEMALE	Int	2	8721		
9	2	FEMALE	Int	2	6113		
10	3	MALE	Int	2	5989		
11	4	FEMALE	Control	2	7845		
12	5	MALE	Control	2	8673		
13	6	MALE	Control	2	5020		
14							
15							
16							

- Once you have created your dataset in Excel, go to “Save As...” Under Format, select “Excel Workbook (.xlsx)” and click ‘Save.’ Be sure to save the file in a location that you will remember later.

1.4 Importing data into R



Click 'Browse' and find your data file, click 'open' and then click 'Import':



1.5 Saving your data and R Script to use later

- There are two things we can do to save our work for later.
- First, save the R script. To do this, either click File -> Save As... and save the R script under some name you'll remember (example: "FirstScript.R"). Or, click the little floppy disk icon - and that should direct you to name and save the R script.
- Second, save an .RData file. This will allow you to later access all of the R objects that you've created, without having to run the entire script again. To do this, type something like the following (but change the path name!)

```
save.image(file="/Users/yourname/Documents/FirstDataFile.RData")`
```

1.6 Loading a previously-saved RData file

- To open up an R script that you saved earlier, open R Studio, go to File -> Open File..., and choose your .R file of choice!
- To load a previously-saved .RData file, type something like the following (again, change the path name to match wherever your .RData file is saved!)

```
load("/Users/yourname/Documents/nameofyourfile.RData")
```

1.7 Loading necessary packages

Although there is a lot that we can do with the base (no-frills, as-is) R software, we can also load different packages to help us perform specific tasks or use datasets that aren't included in the basic R setup. For example, to use some of the built-in datasets included in R in the following sections, we'll only need one additional package, the "MASS" package. To load it, run the following:

```
install.packages("MASS")
```

```
library("MASS")
```

Section 2: 'Best Practices' in R Studio and Basic troubleshooting

2.1: General suggestions when using R Studio

Often, you will have multiple data files imported into R Studio at any one time. Thus, you must 'tell' R Studio which data file you want to currently use. There are several ways to do this, but the easiest is to type `attach(data)` where data is the name of your dataset each time you want to work with a different data file.

For Example:

```
attach(mtcars)
```

If you have **any missing data for the variable of interest**, you may receive an error message when trying to calculate any statistical analyses (i.e., both descriptive and inferential statistics). If this occurs, add the following `na.rm = TRUE` option (same for if you're calculating the variance, range, etc.):

For Example:

```
mean(mpg, na.rm=TRUE)  
# here we are telling R to ignore missing values in the mpg variable when calculating the mean
```

2.2: Basic Troubleshooting in R Studio

If you encounter a warning (see an example of this below), you likely will be able to continue to run your analyses, but you should consult the R Studio help tab located in the bottom right window of the R Studio environment (see Figure above) or 'google' the warning for details/more information.

```
Warning message:  
package 'Rserve' was built under R version 3.2.5
```

If you encounter an error (see an example of each below), you should consult the R Studio help tab located in the bottom right window of the R Studio environment (see Figure above) or 'google' the warning for details/more information.

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

When **in doubt, RE-TYPE!** It is very easy to forget a parenthesis, or capitalize something that should have been lowercase, or misspell a variable name. Triple-check everything!

Capitalization matters! It can be helpful to set some naming "rules" for yourself – for example, always name your objects with all lowercase letters (i.e., call your data frame "health_info" instead of "Health_Info"). If your dataset is called "data", R will not recognize it if you type in "Data". This goes for commands too.

If you're curious about **what a command means**, you can use R's built-in help function! For example, if you can't remember what the "seq" function does, type "?seq" in R.

If you **still can't figure something out... GOOGLE IT!** This seems obvious, but there are tons of helpful resources and threads out there – try googling "R how to write a function" for an example.

Anything you type after a "#" sign will be ignored by R. Many times in this R guide, we will use this to give you more information about something.

If you get this warning below, it means that you already have a dataset with that variable name (score). We recommend having different variable names in each dataset so that R does not mix them up.

The following object is masked from Workbook2:

score

Section 3: Descriptive Statistics

3.1: Descriptive Statistics for continuous variables

For the following examples, I'm going to use the `mtcars` dataset. This is a built-in dataset in R, so all you need to do is type the following:

```
mtcars
```

This dataset contains information about different car models, with info on each car's miles per gallon (`mpg`), the number of cylinders (`cyl`), whether the car is automatic or manual transmission (`am`), etc.

To get a quick summary of a variable, use the `summary()` function:

```
attach(mtcars)
```

```
summary(mpg)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  10.40  15.42   19.20   20.09  22.80   33.90
```

recall: 'attach' enables us to specify which data file we want to analyze

then, all you must do is type the name of the variable you want to analyze in the dataset (such as mpg)

Typically, in a research paper, the descriptive statistics that are commonly reported are the mean and standard deviation of a continuous variable, though you may also want to report the variance, median, range, or other descriptive statistics as necessary. Let's find the

mean, median, standard deviation, variance, range, and sample size of the mpg (miles per gallon) variable in the mtcars dataset:

```
mean(mpg)
## [1] 20.09062

median(mpg)
## [1] 19.2

sd(mpg)
## [1] 6.026948

var(mpg)
## [1] 36.3241

range(mpg)
## [1] 10.4 33.9

length (mpg) #this is sample size or n
## [1] 32
```

In an APA journal, we might say:

The mean for miles per gallon is 20.091 ($SD = 6.027$). Or, we would say: In our study, we measured miles per gallon ($M = 20.091$, $SD = 6.027$). (Note the capitalization of M and SD and italicizing when reporting mean and standard deviation.)

3.2: Descriptive statistics for categorical or grouping variables

Important: Converting numerical variables to factors. Often, the grouping variables for a project (i.e., political affiliation or case/control status) are not entered in a format that is compatible with the way R reads categorical variables. In our example mtcars dataset, we have a few numerical variables that are meant to represent categories (e.g., for the am variable, 1 = manual transmission, 0 = automatic transmission), and this could pose a problem because we want R to interpret “1” and “0” as manual and automatic but instead, R will read these values numerically instead of as category labels.

Therefore, if we want to use a numerical variable as a grouping variable (for example, “Manual” and “Auto” instead of 1’s and 0’s), we can **either** specify that R treat our numerical variable as a “factor” with the following code:

```
attach(mtcars)

table(as.factor(am))
```

OR we can create a new variable that R will recognize as a “factor” with something like the following code:

```
attach(mtcars)
am2 <- factor(am, labels=c("Auto", "Manual"))
```

am2 is the name of the new variable we have created. Note above that when you run this code, nothing will “happen” - that is, R won’t give you any output yet. However, the variable was still created, if you entered the code correctly! You could just check this by “looking” at mtcars:

```
head(mtcars)
##           mpg cyl  disp  hp drat   wt  qsec vs am gear carb
## Mazda RX4      21.0  6  160 110 3.90 2.620 16.46  0  1   4   4
## Mazda RX4 Wag  21.0  6  160 110 3.90 2.875 17.02  0  1   4   4
## Datsun 710     22.8  4  108  93 3.85 2.320 18.61  1  1   4   1
## Hornet 4 Drive 21.4  6  258 110 3.08 3.215 19.44  1  0   3   1
## Hornet Sportabout 18.7  8  360 175 3.15 3.440 17.02  0  0   3   2
## Valiant       18.1  6  225 105 2.76 3.460 20.22  1  0   3   1
##           am2
## Mazda RX4      Manual
## Mazda RX4 Wag  Manual
## Datsun 710     Manual
## Hornet 4 Drive  Auto
## Hornet Sportabout Auto
## Valiant       Auto
```

(By the way, that head() command is very useful: it will display the first five lines of the mtcars dataset (or whatever dataset you want).)

Now we can compute the descriptives for the grouping variable. All we have to do is type:

```
table(am2)
##
##   Auto Manual
##    19    13
```

So it looks like there are 19 cars with automatic transmissions, and 13 cars with manual transmissions.

To calculate the mean and SD of a variable (such as mpg) for two groups (e.g. Auto/Manual), we can use the following code:

```
aggregate(mpg, by=list(am), FUN=mean)
  Group.1      x
1      0 17.14737
2      1 24.39231
aggregate(mpg, by=list(am), FUN=sd)
  Group.1      x
1      0 3.833966
2      1 6.166504
```

3.3: Cronbach's alpha for the internal reliability of survey scales

Cronbach's alpha is a measure of internal consistency, that is, how closely related a set of items are as a group. A "high" value of alpha is often used as evidence that the items measure an underlying construct. Cronbach's alpha can range from 0 to 1. An alpha closer to 1 indicates better internal consistency.

This needs to be calculated and reported for every *scale* used in your paper.

First, we are going to need to install the psych package within R, which contains the alpha function for calculating Cronbach's alpha. To do this, first type:

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

Wait for it to install.

Once it installs, enter:

```
library(psych)
```

*After you do this once, you won't need to do the install.packages step anymore, but you will have to type library(psych) in order to use this function in a new R session.

Next, we are going to create a new dataset with just the items in the scale we are interested in. In this example, we will do this with the Attitudes items only.

We need to figure out which numbered columns in our current dataset correspond to the seven attitudes items.

You'll notice in the Excel file, ATT 1 is at column U (the 21st letter of the alphabet), and ATT7 is at column AA (the 1st repeated after the last letter of the alphabet) We could also figure this out within R by typing:

```
names(mydata)
```

```
> names(mydata)
 [1] "age"          "gender"      "year"        "int_1"       "int_2"       "int_3"       "norm_1"
 [8] "norm_2"      "norm_3"      "norm_4"      "norm_5"      "norm_6"      "pbc_1"       "pbc_2"
[15] "pbc_3"       "pbc_4"       "pbc_5"       "pbc_6"       "pbc_7"       "pbc_8"       "att_1"
[22] "att_2"       "att_3"       "att_4"       "att_5"       "att_6"       "att_7"       "vig_mins"
[29] "vig_sess"    "mod_mins"    "mod_sess"    "mild_mins"   "mild_sess"   "TOT_sess"    "TOT_mins"
[36] "intSCALE"    "normsSCALE" "pbcSCALE"
```

You'll notice that the numbers on the right-hand side indicate the column number of the first item on each line. Thus, we could figure out here that att_1 – att_7 correspond to items 21-27 in this dataset. As another example, pbc_1 – pbc_8 correspond to items 13-20.

Now we are going to create a new dataset with JUST the items from the attitudes scale with the following code:

```
attitudes_items = mydata[,21:27]
```

*This is what I am calling this new subset of my larger dataset.

**Here I am saying that I want all the rows from mydata, but only columns 21-27.

We can check to make sure this was done correctly by typing:

```
attitudes_items
> attitudes_items = mydata[,21:27]
> attitudes_items
# A tibble: 105 x 7
  att_1 att_2 att_3 att_4 att_5 att_6 att_7
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1     4     4     1     4     4     1     4
2     6     5     5     5     4     4     7
3     7     7     1     7     7     1     7
4     6     4     6     5     1     3     4
5     6     6     3     5     5     2     5
6     7     1     5     5     5     5     5
7     7     7     7     6     5     4     5
8     5     4     4     5     7     4     7
9     5     5     3     3     4     3     4
10    5     4     2     3     4     3     4
# ... with 95 more rows
```

Now I can find Cronbach's alpha for the attitudes scale by entering:

```
alpha(attitudes_items)
```

```
> alpha(attitudes_items)

Reliability analysis
Call: alpha(x = attitudes_items)

  raw_alpha std.alpha G6(smc) average_r S/N ase mean sd
    0.91      0.92      0.94      0.61  11 0.014  5.8 1.2

  lower alpha upper      95% confidence boundaries
0.88 0.91 0.94

Reliability if an item is dropped:
  raw_alpha std.alpha G6(smc) average_r S/N alpha se
att_1     0.90     0.90     0.92     0.61  9.4  0.015
att_2     0.90     0.91     0.93     0.63 10.1  0.014
att_3     0.90     0.91     0.91     0.62  9.7  0.015
att_4     0.89     0.89     0.92     0.58  8.4  0.017
att_5     0.89     0.90     0.92     0.60  9.0  0.016
att_6     0.90     0.91     0.91     0.62  9.6  0.016
att_7     0.89     0.90     0.93     0.59  8.8  0.017

Item statistics
  n raw.r std.r r.cor r.drop mean sd
att_1 105 0.77 0.80 0.77 0.71 6.5 1.0
att_2 105 0.74 0.76 0.72 0.64 6.2 1.4
att_3 105 0.81 0.79 0.78 0.72 4.9 1.7
att_4 105 0.87 0.88 0.85 0.82 6.1 1.3
att_5 105 0.83 0.83 0.81 0.76 6.0 1.5
att_6 105 0.82 0.79 0.79 0.74 4.8 1.7
att_7 105 0.85 0.85 0.81 0.79 6.1 1.3

Non missing response frequency for each item
  1 2 3 4 5 6 7 miss
att_1 0.01 0.01 0.00 0.02 0.11 0.15 0.70 0
att_2 0.04 0.00 0.01 0.06 0.10 0.15 0.65 0
att_3 0.05 0.05 0.14 0.11 0.26 0.16 0.23 0
att_4 0.02 0.01 0.02 0.08 0.12 0.17 0.58 0
att_5 0.03 0.00 0.03 0.10 0.12 0.10 0.61 0
att_6 0.05 0.08 0.10 0.16 0.23 0.19 0.19 0
att_7 0.02 0.00 0.03 0.08 0.13 0.16 0.58 0
```

We get a lot of info here, but for these purposes we are primarily interested in the number under **raw_alpha** (.91). That's it!

To report results from a Cronbach's alpha in an APA journal, we might say:

The attitudes scale from the Theory of Planned Behavior consisted of 7 items ($\alpha = .91$).

Section 4: Inferential Statistics: Chi-Squared, t-tests, ANOVAs

4.1: Overall Data Organization for Chi-Squared, t-tests, ANOVAs

Chi-squared:

vs	am
0	1
0	1
1	1
1	0
0	0
1	0
0	0
1	0
1	0
1	0

or with categories labeled:

vs	am
v_engine	manual
v_engine	manual
straight_engine	manual
straight_engine	automatic
v_engine	automatic
straight_engine	automatic
v_engine	automatic
straight_engine	automatic
straight_engine	automatic
straight_engine	automatic

ALL t-tests (paired AND unpaired):

qsec	am
16.46	1
17.02	1
18.61	1
19.44	0
17.02	0
20.22	0
15.84	0
20	0
22.9	0
18.3	0

or with categories labeled:

qsec	am
16.46	manual
17.02	manual
18.61	manual
19.44	automatic
17.02	automatic
20.22	automatic
15.84	automatic
20	automatic
22.9	automatic
18.3	automatic

Where 'qsec' is your DV, and 'am' is your IV

Between Subjects ANOVAs where "weight" is your DV, and "group" is your IV. **NOTE: You must use words for categories unless you convert your numerical variable to a factor (see page 13):**

weight	group
4.17	ctrl
5.58	ctrl
5.18	ctrl
6.11	trt1
4.5	trt1
4.61	trt1
4.53	trt2
5.33	trt2
5.14	trt2

Within Subjects (repeated measures) ANOVAs:

participant	condition	items
a01	norm	6
a01	bt	6
a01	phone	3
a02	norm	7
a02	bt	5
a02	phone	5

Where “items” is the DV, “condition” is the within-subjects IV, and “participant” is simply each subject’s study identifier. Note that each participant has multiple rows in the excel sheet – they have a different DV value (“items”) for each type of “condition”.

4.2: Categorical predictor(s) (IVs), continuous outcome(s) (DVs)

4.2.1 One predictor (IV) with 2 levels, unpaired (independent) samples: Independent Samples T-test

The independent-samples t-test is used when you are comparing two groups of data (such as democrats vs. republicans, or treatment vs. control), and there is no dependency between the two samples.

In this example, we’ll use the built-in R dataset `mtcars` again. We’ll compare the `qsec` variable (quarter-mile time in seconds) between the manual (`am = 1`) and automatic (`am = 0`) groups. The null hypothesis being tested is that the mean difference between the groups is zero. To run the independent samples t-test, run the following code (note the `var.equal=TRUE` option: this specifies that we want to perform a Student’s t-test, which assumes that both groups are sampled from populations with approximately equal variances):

```
attach(mtcars)

t.test(qsec ~ am, var.equal = TRUE)

# the outcome DV (qsec) goes on the left side of the equation
# the IV grouping variable (am) goes on the right side

##
## Two Sample t-test
##
## data: qsec by am
## t = 1.2936, df = 30, p-value = 0.2057
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
```

```
## -0.4763645  2.1226803
## sample estimates:
## mean in group 0 mean in group 1
##          18.18316          17.36000
```

The estimated group means are $qsec = 18.18$ for cars with an automatic transmission ($am = 0$), and $qsec = 17.36$ for cars with manual transmissions ($am = 1$). The test statistic is $t(30) = 1.294$, $p = 0.21$, with a 95% CI = $(-0.476, 2.123)$. This means we cannot reject the null hypothesis.

To report results from an independent samples t-test in an APA-formatted journal, we might say:

There is no difference in average quarter-mile time between cars with a manual transmission and those with an automatic transmission, $t(30) = 1.294$, $p = 0.21$.

4.2.2 One predictor (IV) with 2 levels, paired (dependent) samples: Paired T-test

The paired-samples t-test should be used when you wish to compare two groups of data and the groups are **NOT** independent - for example, if people were measured on an outcome at two different time points or if you are comparing non-independent groups (e.g., data from spouses in a married couple).

We'll use another built-in R dataset this time - the sleep dataset, where the outcome measurement is extra (indicating how much extra sleep patients got with Drug 1 vs. Drug 2), the grouping variable is group (Drug 1 vs. Drug 2), and ID is the unique ID for each student. In this example, the observations for the two groups are not independent, because the same people (check the IDs) are included in both groups. Here's what the dataset should look like:

```
head(sleep)
##   extra group ID
## 1  0.7     1  1
## 2 -1.6     1  2
## 3 -0.2     1  3
## 4 -1.2     1  4
## 5 -0.1     1  5
## 6  3.4     1  6
```

If you had entered this data into excel, it would look like the following:

extra	group	ID
0.7	1	1
-1.6	1	2
-0.2	1	3
-1.2	1	4
-0.1	1	5
3.4	1	6

NOTE: You will have two groups in your data set. Use words for categories.

To run a paired t-test, type the following (don't forget the paired = TRUE option!):

```
attach(sleep)

t.test(extra ~ group, paired=TRUE)

# the outcome/DV (extra, in this example) goes on the left side of the "~"
# the grouping IV (group) goes on the right side

##
## Paired t-test
##
## data: extra by group
## t = -4.0621, df = 9, p-value = 0.002833
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -2.4598858 -0.7001142
## sample estimates:
## mean of the differences
## -1.58
```

The estimated difference for sleep between drugs is -1.58 hours (it is negative because the mean for Drug 1 was lower than the mean for Drug 2). The test statistic is $t(9) = -4.06$, $p < .01$, with a 95% CI = (-2.46, -0.70). This means we can reject the null hypothesis.

To report results from a paired t-test in an APA-style report, we might say:

There is a significant difference in extra sleep with the two drugs, $t(9) = -4.06$, $p < .01$, where Drug 1 elicited 1.58 fewer hours of sleep than Drug 2.

4.2.3 One predictor (IV) with 3+ levels, unpaired (independent) samples: One-way ANOVA

One-way ANOVA is an extension of the two-sample t-test, but used when you want to compare outcome measures between more than two groups, or levels, of a single independent variable (e.g., comparing yearly income between people who completed high school, college, or graduate school.)

For this example, I'll use the built-in R dataset PlantGrowth - it compares the yield (weight variable) of different trees grouped into three different levels of treatment (group variable - ctrl, trt1, and trt2).

Let's check out the dataset:

```
head(PlantGrowth)

## weight group
## 1 4.17 ctrl
## 2 5.58 ctrl
## 3 5.18 ctrl
```

```
## 4 6.11 ctrl
## 5 4.50 ctrl
## 6 4.61 ctrl
```

Again, if you had entered this data into excel, the setup would look like this for the first 6 rows:

weight	group
4.17	ctrl
5.58	ctrl
5.18	ctrl
6.11	ctrl
4.5	ctrl
4.61	ctrl

Where “weight” is your outcome (DV), and “group” is your IV. For an anova, you will have at least three groups.

Now, to perform a one-way ANOVA and **test whether there are any differences in yield between the three different groups**, run this code:

```
attach(PlantGrowth)

myanova <- aov(weight ~ group)
# the outcome DV (e.g. weight) goes on the left side of the equation
# the IV grouping variable goes on the right side of the "~"
summary(myanova)

##           Df Sum Sq Mean Sq F value Pr(>F)
## group      2  3.766   1.8832   4.846 0.0159 *
## Residuals 27 10.492   0.3886
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results for this example would be reported as such: $F(2, 27) = 4.846$, $p < 0.05$. This means that the average weights of our three groups (ctrl, trt1, and trt2) are significantly different from each other. However, we can't tell from this test how the groups differ from each other - are the two treatment groups significantly heavier than the control group? Do all three groups have significantly different mean weights? To examine this more closely, we can look at the group means and standard deviations, and conduct a Bonferroni's posthoc pairwise comparison test. Run the following code (remembering that weight is our outcome measure of interest, while group is the grouping variable):

4.2.3b ANOVA: means table and Bonferroni post-hoc analyses

```
attach(PlantGrowth)

tapply(weight, group, mean)

## ctrl trt1 trt2
## 5.032 4.661 5.526
```

```

tapply(weight, group, sd)

##      ctrl      trt1      trt2
## 0.5830914 0.7936757 0.4425733

pairwise.t.test(weight, group, p.adjust.method = "bonferroni")

## Pairwise comparisons using t tests with pooled SD

##
## data: weight and group
##
##      ctrl trt1
## trt1 0.583 -
## trt2 0.263 0.013
##
## P value adjustment method: bonferroni

```

You can see the adjusted group means and standard deviations given from the `tapply` commands (for example, the control group weighs an average of 5.032, with a SD = 0.58), and the Bonferroni comparisons test gives you the p-value for each pairwise comparison. It appears that only the two treatment groups ('trt1' and 'trt2') are significantly different in weight ($p < 0.05$).

4.2.4 One predictor (IV) with 3+ levels, paired (dependent) samples: Repeated Measures ANOVA

Repeated measures ANOVA is an extension of the paired-sample t-test, of a sort - like the paired samples t-test, it is also used with dependent samples (i.e., each subject is measured two or more times) but unlike the t-test, you use it when you want to compare outcome measures between three or more levels of a single independent variable (e.g., comparing how well each person performs a task under easy, neutral, and stressful conditions.)

In this case, I'm going to import a dataset on cell phone use - it looks like this in my excel sheet:

participant	condition	items
a01	norm	6
a01	bt	6
a01	phone	3
a02	norm	7
a02	bt	5
a02	phone	5

Where "items" is the DV, "condition" is the within-subjects IV, and "participant" is simply each subject's ID.

This dataset is measuring the number of items correctly recalled (`items`) during a driving simulation performed under three different conditions (`conditions`): holding a cell phone, using a Bluetooth/hands-free phone, and normal (phone-free) driving conditions. You'll

notice the way the data is set up right now - each participant has 3 rows, one for each condition.

Let's perform a repeated measures ANOVA on this dataset, testing whether participants recall different numbers of items across the three driving conditions: To do this, we will use the `aov` function again, just as we did above for the one-way ANOVA, but this time we have to specify the fact that we need to control for the between-person variation across our within-person variable (condition) - in other words, some people might naturally be better at recalling items on the test than other people, regardless of condition, and we account for that natural variation in the `Error(participant/condition)` option in the code below.

```
attach(cell)

my.rm.anova <- aov(items ~ condition + Error(participant/condition))
# the outcome DV (items) goes on the left side of the aov() command
# the grouping IV (condition) goes on the right,
# and the Error term consists of the subjects' id variable (participant, in t
his example),
# a forward slash, and the within-person variable (condition),
# all in parentheses
summary(my.rm.anova)

## Error: participant
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 11  18.97   1.725
##
## Error: participant:condition
##           Df Sum Sq Mean Sq F value    Pr(>F)
## condition  2  84.39   42.19  17.32 3.04e-05 ***
## Residuals 22  53.61    2.44
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The results for this example would be reported as such: $F(2,22) = 17.32, p < 0.001$.

To report results from a repeated measures ANOVA in an APA journal, we might say:

There was a statistically significant effect of driving condition on the number of items recalled ($F(2,22) = 17.32, p < 0.001$).

4.2.4RM Anova: Means table and Bonferroni post-hoc analyses

We can also do a means table and post hoc test here to see where the differences are.

```
tapply(items, condition, mean)

##           bt           norm           phone
## 5.500000 7.416667 3.666667

tapply(items, condition, sd)
```

```
##          bt      norm      phone
## 1.167748 1.781640 1.435481
```

For the post hoc test, be sure to include the command showing that this is a repeated measures (or paired samples) with **paired=T**.

```
pairwise.t.test(items, condition, p.adj="bonferroni", paired=T)
```

```
## Pairwise comparisons using paired t tests
```

```
##
## data: items and condition
```

```
##          norm      bt
## bt      0.072      -
## phone 9e-05  0.037
```

```
##
## P value adjustment method: bonferroni
```

You can see from the table that the mean for phone is different from both the mean for Bluetooth and the mean for norm (but bt and norm are not different from each other). The decision rule is that if $p < .05$, the means are different.

4.2.5 Two or more predictors (IV), unpaired (independent) samples: Factorial ANOVA between subjects

When you have two or more independent variables that are grouping variables (i.e., categorical), you want to use a factorial ANOVA design.

For example, I'll use the built-in R dataset `crabs` to see if I can come up with a model that will **predict the crab's body depth (variable BD) from the crab's species and sex (variables sp and sex)**. This is one dataset that we'll need to install a package to access - if you haven't already, here are the commands to install and load the "MASS" package so we can use the "crabs" dataset.

```
install.packages("MASS")
```

```
library("MASS")
```

Here's a peek at what the crabs dataset looks like:

```
head(crabs)
```

```
##   sp sex index  FL  RW  CL  CW  BD
## 1  B  M     1  8.1 6.7 16.1 19.0 7.0
## 2  B  M     2  8.8 7.7 18.1 20.8 7.4
## 3  B  M     3  9.2 7.8 19.0 22.4 7.7
## 4  B  M     4  9.6 7.9 20.1 23.1 8.2
## 5  B  M     5  9.8 8.0 20.3 23.0 8.2
## 6  B  M     6 10.8 9.0 23.0 26.5 9.8
```


If I were to put only the relevant information in an excel spreadsheet that I could later import into R, this is what it would look like:

sp	sex	BD
B	M	7
B	F	7.4
B	M	10.1
B	F	9.2
O	M	8.2
O	M	9
O	F	8.3
O	M	11.2
O	F	10.1

Where “sp” and “sex” are the two IVs, and “BD” is my DV of interest.

Now to run the factorial ANOVA, run the following lines of code:

*Notice that in this code, I specify not only the main effects of the sex and species variables, but I also specified the interaction effect with the sp*sex argument*

```
attach(crabs)

myanova2 <- aov(BD ~ sex+sp+sex*sp)
# the outcome DV (BD) is on the left side of the "~"
# and the two IVs (sex and sp) are on the right side
summary(myanova2)

##           Df Sum Sq Mean Sq F value    Pr(>F)
## sex         1   18.8    18.8    1.986  0.1603
## sp          1  419.1   419.1  44.305 2.75e-10 ***
## sex:sp      1   42.4    42.4    4.484  0.0355 *
## Residuals 196 1853.8     9.5
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can see from the output that the main effect of species, $F(1,196) = 44.305, p < 0.001$, and the sex*species interaction, $F(1,196) = 4.484, p < 0.05$ are significant. The main effect for sex is not significant, $F(1, 196) = 1.99, p = .16$.

To check the means, standard deviations, and counts in each of the sex-by-species groups, run the following code (where the crabs\$BD argument is your outcome measure, and the crabs\$sex and crabs\$sp arguments inside the list are your grouping variables). (Looking at these group means, standard deviations, and counts will help determine the direction of the main effect and the interaction between sex and species.)

```
tapply(BD, list(sex, sp), mean, na.rm=TRUE)
```

```
##           B           O
## F 11.816 15.632
## M 13.350 15.324

tapply(BD, list(sex, sp), sd, na.rm=TRUE)

##           B           O
## F 2.752465 2.752620
## M 3.199888 3.527187

tapply(BD, list(sex, sp), function(x) sum(!is.na(x)))

##    B  O
## F 50 50
## M 50 50
```

From this output, we can see that the “O” species tends to have larger body depth than the “B” species, and this difference is larger for the Female crabs than for the Males.

NOTE about post hoc tests:

You can do a post hoc test for any significant effects (main effects or interaction). We do not need it here for either main effect because each main effect has only two groups. Recall that we didn’t need a post hoc test for t-tests because we knew which two means were different from each other (there were only two means to begin with!). If you wanted to run a post hoc on the ME for sp, you would use this command:

```
pairwise.t.test (BD, sp, p.adj="bonferroni")
```

or for the interaction:

```
pairwise.t.test (BD, interaction(sp,sex), p.adj="bonferroni")
```

4.2.6 Two or more predictors (IV), paired (dependent) samples: Repeated Measures Factorial ANOVA

Very similar to repeated measures ANOVA, the factorial repeated measures design is used when you want to compare outcome measures between three or more levels of **two or more** independent variables (e.g., comparing how well each person performs a task depending on their age (old vs. young) and whether they’re in an easy, neutral, or stressful conditions.)

In this case, I’m going to import a dataset on the amplitude in millivolts of an event related potential (ERP) that is associated with working memory load, measured on each participant under two systems (a and b) and three levels of difficulty (easy, medium, and hard) - it looks like this:

subID	system	difficulty	amp
1	a	easy	1.36
1	a	med	1.46

1	a	hard	1.89
1	b	easy	2.02
1	b	med	2.48
1	b	hard	1.22
2	a	easy	1.23
2	a	med	1.37
2	a	hard	1.75

Importantly, you'll notice the way the data is set up right now, again in a "long" format - each participant has 6 rows, one for each combination of the two grouping variables (system and difficulty). `amp` is our outcome measure (amplitude in millivolts), `system` and `difficulty` are our two within-subject independent variables, and `subID` is just the participant's unique study ID.

Let's perform a factorial repeated measures ANOVA on this dataset, testing whether subjects show different ERP amplitudes under the different combinations of system and difficulty: To do this, we will use the `aov` function again and include the additional `Error()` option, just as we did above for the one-way repeated measures ANOVA, but this time we have to specify the fact that we need to control for the between-person variation across *both* our within-person variables (`system` and `difficulty`). Let's also test for any potential interaction between `system` and `difficulty` with the asterisk: `system*difficulty`:

```
attach(erp)

my.frm.anova <- aov(amp ~ system*difficulty + Error(subID/(system*difficulty)
))
# the outcome DV (amp) goes on the left side of the aov() command
# the within-person IVs (system and difficulty) go on the right,
# and the Error term consists of the subjects' id variable (subID, in this ex
ample),
# a forward slash, and the within-person variables (system*difficulty),
# all in parentheses

summary(my.frm.anova)

##
## Error: subID
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  1  0.211    0.211
##
## Error: subID:system
##           Df Sum Sq Mean Sq
## system    1  3.506    3.506
##
## Error: subID:difficulty
##           Df Sum Sq Mean Sq
## difficulty 2  0.5897   0.2949
```

```
##
## Error: subID:system:difficulty
##           Df    Sum Sq Mean Sq
## system:difficulty  2 0.002308 0.001154
##
## Error: Within
##           Df Sum Sq Mean Sq F value   Pr(>F)
## system          1 1.0096  1.0096  32.970 6.2e-07 ***
## difficulty      2 0.5206  0.2603   8.501 0.000691 ***
## system:difficulty  2 0.0005  0.0003   0.009 0.991233
## Residuals      48 1.4698  0.0306
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We see significant main effects of both system ($F(1,48) = 32.970, p < .001$) and difficulty ($F(1,48) = 8.501, p < 0.001$), but no significant interaction ($F(2,48) = 0.009, p = .99$).

To report results from a repeated measures factorial ANOVA in an APA-style writeup, we might say:

There was a statistically significant main effect of both system ($F(1,48) = 32.970, p < 0.001$) and difficulty ($F(1,48) = 8.501, p < 0.001$) on the ERP amplitude in millivolts. The interaction was not significant ($F(2,48) = 0.009, p = .99$).

4.3: Categorical predictor(s) (IVs), categorical outcome(s) (DVs)

4.3.1 Chi-square Test of Independence

When should a chi-square test be used? When you have two categorical variables (such as political affiliation or education level) and you want to compare the count or proportion of the sample that occurs in each cell of the different categories. The chi-square test works by comparing the observed frequencies with the expected frequencies under the null hypothesis that there is no true association between the different categories. This test of independence is similar to a two-way interaction – it answers the question: Are cars with manual transmission more likely to have a V-engine or straight engine than cars with an automatic transmission?

In this example, we'll use the `mtcars` dataset again. We'll examine the number of cars that are manual (`am = 1`) or automatic (`am = 0`) and either a V-engine (`vs = 0`) or a straight engine (`vs = 1`). First, we'll make a table with the `table()` function, and then we'll run the chi-squared test on that table with the `chisq.test()` command:

```
attach(mtcars)
table(am, vs)

##
##      0  1
## 0 12  7
## 1  6  7
```

```
chisq.test(table(am, vs))  
  
##  
## Pearson's Chi-squared test with Yates' continuity correction  
##  
## data:  table(am, vs)  
## X-squared = 0.34754, df = 1, p-value = 0.5555  
  
# for this code, it doesn't matter what order you  
# specify the variables in the chisq.test(table()) command
```

To calculate the number of observations in a χ^2 test

```
sum(complete.cases(mtcars))  
[1] 32
```

The test statistic from our chi-squared test equals $\chi^2(1, N = 32) = 0.348$, $p = 0.56$, meaning that we cannot reject the null hypothesis that there is no significant association between a car's transmission type and what kind of engine it has. OR we accept the null, which means that there is no significant association between a car's transmission type and the engine type.

To report results from a Chi-Square in an APA-style writeup, we might say:

Cars with manual transmissions are just as likely as cars with automatic transmissions to have a V-engine or straight engine, $\chi^2(1, N = 32) = 0.348$, $p = 0.56$

5: Inferential Statistics: Correlation and Regression

5.1: Overall Data Organization for Correlation and Regression

When running a correlation or regression, you should set up your data so that each column contains the data for one of your variables:

mpg	cyl	am
21	6	1
21	6	1
22.8	4	1
21.4	6	0
18.7	8	0
18.1	6	0
14.3	8	0
24.4	4	0
22.8	4	0
19.2	6	0
17.8	6	0

5.2: Continuous predictor(s) (IVs), continuous outcome(s) (DVs)

5.2.1: One predictor (IV) and one outcome (DV): Simple Correlation

When you want to measure the strength of the association between two continuous variables, a single correlation, such as a Pearson's correlation, is the most effective.

Let's use the mtcars dataset as an example. In your R Studio session, type the following:

```
head(mtcars)
```

```
      mpg  cyl  disp  hp  drat    wt  qsec vs  am  gear  carb
Mazda RX4      21.0    6  160  110  3.90  2.620 16.46  0  1    4    4
Mazda RX4 Wag  21.0    6  160  110  3.90  2.875 17.02  0  1    4    4
Datsun 710     22.8    4  108   93  3.85  2.320 18.61  1  1    4    1
Hornet 4 Drive  21.4    6  258  110  3.08  3.215 19.44  1  0    3    1
Hornet Sportabout 18.7    8  360  175  3.15  3.440 17.02  0  0    3    2
Valiant       18.1    6  225  105  2.76  3.460 20.22  1  0    3    1
```

```
attach(mtcars)
```

```
cor.test(qsec, mpg)
```

```
      Pearson's product-moment correlation
data:  qsec and mpg
t = 2.5252, df = 30, p-value = 0.01708
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.08195487 0.66961864
```

```
sample estimates:
      cor
0.418684
```

The test statistic from our correlation test equals $r(30) = 0.42, p < 0.05$, meaning that we can reject the null hypothesis that there is no significant association between a car's qsec and mpg. OR we have a positive correlation between how long it takes to go ¼ mile and mpg.

To report results from a correlation in an APA-style writeup, we might say:

A car's mpg is positively correlated with how long it takes that car to go ¼ mile, $r(30) = 0.42, p < 0.05$.

5.2.2 One predictor (IV) and one outcome (DV): Simple Regression

When you want to measure the *precise* relationship between two continuous variables, and you want to predict some continuous outcome from one other independent continuous variable, simple regression is the test you want to use.

Let's use the mtcars dataset again as an example.

Let's see if we can predict a car's miles per gallon from the qsec (time to go a ¼ mile) variable. To do this, type the following:

```
attach(mtcars)

modell1 <- lm(mpg ~ qsec)
# my outcome variable or DV (mpg) goes on the left side of the equation
# the predictor or IV (qsec) goes on the right side
summary(modell1)

##
## Call:
## lm(formula = mpg ~ qsec)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -9.8760  -3.4539  -0.7203   2.2774  11.6491
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.1140     10.0295  -0.510   0.6139
## qsec          1.4121      0.5592   2.525   0.0171 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.564 on 30 degrees of freedom
## Multiple R-squared:  0.1753, Adjusted R-squared:  0.1478
## F-statistic: 6.377 on 1 and 30 DF,  p-value: 0.01708
```

Now what exactly does this output mean? First, the estimate for the intercept is the value of mpg that we would predict for a car with a value of zero for qsec. Next, we have the test of the individual predictor (qsec, in this case.) The estimate for the slope of qsec on mpg is 1.41 with a standard error of 0.559, a t-value = 2.525, and a significant p-value = 0.017. This significant p-value means that qsec predicts mpg

better than we would expect by chance, and the *slope* estimate means that **for each one-unit change** in *qsec*, we predict about a 1.41 increase in miles per gallon (i.e., for one unit of change in the predictor, this is the unit of change in the outcome variable).

Below the individual coefficient tests, the test of the overall model fit is given by the multiple R-squared (or, what proportion of the variance in our outcome is accounted for by our predictor), the F-statistic, and the p-value for the overall test. Here the multiple R-squared = 0.175, meaning that our model explains almost 18% of the variance in miles per gallon, and the F-statistic on 1 and 30 degrees of freedom (reported as such: $F(1,30) = 6.377$ and is significant ($p < 0.05$)).

To report results from a simple regression in an APA journal, we might say:

The time to go a ¼ mile (the *qsec* variable) is a significant predictor of miles per gallon ($\beta = 1.41$, $p = 0.05$), explaining almost 18% of the variance in miles per gallon ($R^2 = 0.175$, $F(1, 30) = 6.377$, $p < 0.05$.)

5.2.3 More than one predictor (IV): Multiple Regression

When you want to predict some continuous outcome from more than one independent continuous variable, multiple regression is the test you want to use.

Let's use the *mtcars* dataset as an example again. In your R Studio session, type the following:

```
head(mtcars)
##           mpg cyl  disp  hp  drat   wt  qsec vs am gear carb
## Mazda RX4      21.0   6  160  110  3.90  2.620  16.46  0  1   4    4
## Mazda RX4 Wag  21.0   6  160  110  3.90  2.875  17.02  0  1   4    4
## Datsun 710     22.8   4  108   93  3.85  2.320  18.61  1  1   4    1
## Hornet 4 Drive  21.4   6  258  110  3.08  3.215  19.44  1  0   3    1
## Hornet Sportabout 18.7   8  360  175  3.15  3.440  17.02  0  0   3    2
## Valiant        18.1   6  225  105  2.76  3.460  20.22  1  0   3    1
##           am2
## Mazda RX4      Manual
## Mazda RX4 Wag  Manual
## Datsun 710     Manual
## Hornet 4 Drive  Auto
## Hornet Sportabout Auto
## Valiant        Auto
```

The "head()" function lets us look at just the first few lines of the dataset. You should see the same dataset about cars, with information about miles per gallon (*mpg*), the number of cylinders (*cyl*), whether the car is automatic or manual transmission (*am*), etc. **Let's now see if we can predict a car's miles per gallon from the "qsec" (1/4 mile time) variable and the "wt" (weight) variable.** To do this, type the following:


```

attach(mtcars)

model2 <- lm(mpg ~ qsec + wt)
# the DV or outcome variable (mpg) goes on the left side of the "~"
# the two IVs or predictors go on the right side (ex: qsec and wt)
summary(model2)

##
## Call:
## lm(formula = mpg ~ qsec + wt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.3962 -2.1431 -0.2129  1.4915  5.7486
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.7462     5.2521   3.760 0.000765 ***
## qsec         0.9292     0.2650   3.506 0.001500 **
## wt          -5.0480     0.4840 -10.430 2.52e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.596 on 29 degrees of freedom
## Multiple R-squared:  0.8264, Adjusted R-squared:  0.8144
## F-statistic: 69.03 on 2 and 29 DF,  p-value: 9.395e-12

```

Let's interpret this output! First, we have the estimate of the intercept, 19.7462 - this is the predicted value of mpg when both wt and qsec = zero. The estimate of the first predictor, qsec, equals approximately 0.93 and is significant at a p-value < 0.05. But why is this slope different than the one in the previous example for simple regression? Now we're controlling for the effect of wt - this estimate means that holding the weight of the car constant, we expect that a one-unit increase on a car's ¼ mile time will result in an increase of 0.929 miles per gallon. Similarly, the estimate of the relationship between weight and miles per gallon is accounting for the effect of qsec. Holding a car's quarter mile time constant, we predict a one-unit increase in weight to result in a decrease in miles per gallon of around 5.048. The slope estimate for wt on mpg (controlling for qsec) is also highly significant, with a p-value = 2.52e-11 (i.e., p < 0.001).

To report results from a multiple regression in an APA journal, we might say:

Controlling for weight, the time to go a ¼ mile was a significant predictor of miles per gallon, $\beta = 0.93$, $p = 0.0015$. Controlling for the time to go a ¼ mile, weight was also a significant predictor of miles per gallon, $\beta = -5.05$, $p < .001$.

Below the individual coefficient tests, the test of the overall model fit is given by the multiple R-squared (or, what proportion of the variance in our outcome is accounted for by our predictors), the F-statistic, and the p-value for the overall test. Here the multiple R-squared = 0.826, meaning that our model explains about 83% of the variance in miles per gallon, and the F-statistic on 2 and 29 degrees of freedom (since we now have two predictors in the model) = 69.03 and is highly significant ($p = 9.395e-12$, or $p < 0.001$).

Notice that this model is a much better fit for the data - when we added weight to our regression model, we are able to explain much more of the variance in miles per gallon (around 60% more!).

5.3: Continuous predictor(s) (IVs), categorical outcome(s) (DVs)

5.3.1 Logistic Regression

When your outcome measure is binary (i.e., categorical), rather than continuous, such as when you are predicting whether or not someone has a disease, logistic regression is the test you want to use. **Let's use the mtcars dataset as an example again. In this case, our outcome measure is am - whether a car is an automatic transmission (am = 0) or manual (am = 1).**

This is the same dataset we used in section 3.1, with information about cars such as miles per gallon (mpg), the number of cylinders (cyl), whether the car is automatic or manual transmission (am), etc. (Remember, to use this dataset, just type "mtcars" in your R Studio console - it's a built-in R dataset.) Let's now see if we can predict whether or not a car is an automatic transmission or a manual engine from the mpg and hp (horsepower) variables. To run a logistic regression, type the following (make sure to include the family = binomial part):

```
attach(mtcars)

model3 <- glm(am ~ mpg + hp ,family='binomial')
# the categorical DV (am) goes on the left side of the "~"
# the IVs go on the right side (ex: mpg and hp)
summary(model3)

##
## Call:
## glm(formula = am ~ mpg + hp, family = "binomial", data = mtcars)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -1.41460  -0.42809  -0.07021   0.16041   1.66500
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -33.60517   15.07672  -2.229   0.0258 *
## mpg          1.25961    0.56747   2.220   0.0264 *
## hp           0.05504    0.02692   2.045   0.0409 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 43.230  on 31  degrees of freedom
## Residual deviance: 19.233  on 29  degrees of freedom
## AIC: 25.233
##
## Number of Fisher Scoring iterations: 7
```

Interpreting logistic regression output can be tricky - ask your TA if in doubt!

Very briefly: The estimate of the first predictor, mpg, = 1.26 and is significant at a p-value < 0.05. One way to interpret this coefficient is in terms of odds ratios: for each 1-unit increase in miles per gallon, the odds of the car being a manual (am = 1) are multiplied by $e^{1.26} = 3.5$, controlling for horsepower. We can interpret the coefficient for horsepower in the same way - for each 1-unit increase in horsepower, we predict that the odds of the car being a manual transmission are multiplied by $e^{0.055} = 1.06$, controlling for mpg.

Section 6: Graphing in R/R Studio and Alternatives to Graphing in R/R Studio

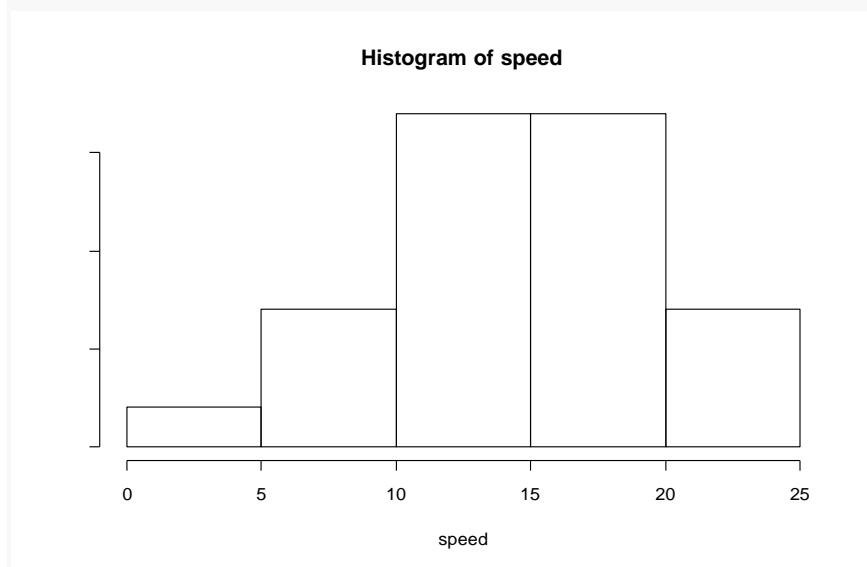
6.1 Creating Graphs in R/R Studio

Histogram (one continuous variable)

The code to make a histogram is `hist(variable)`. Let's make a histogram of the speed data from the cars dataset:

```
attach(cars)
```

```
hist(speed)
```



Scatterplot (two continuous variables)

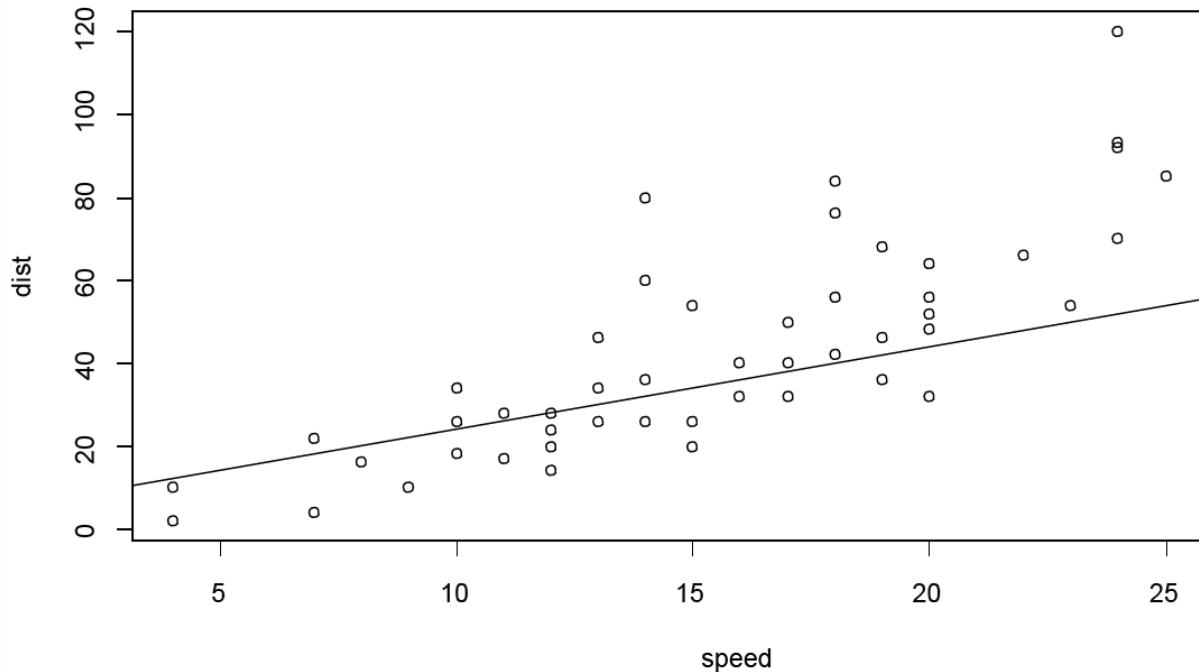
The code to make a scatterplot is `plot(variable1~ variable2)`. Let's make a scatterplot of the speed and distance (dist) data from the cars dataset:

```
attach(cars)
```

```
plot(dist~speed)
```

```
abline(lm(dist~speed))
```

* the "abline" command will draw a best fit line between our DV (dist) and our IV (speed)



Line graph (one continuous and one discrete variable)

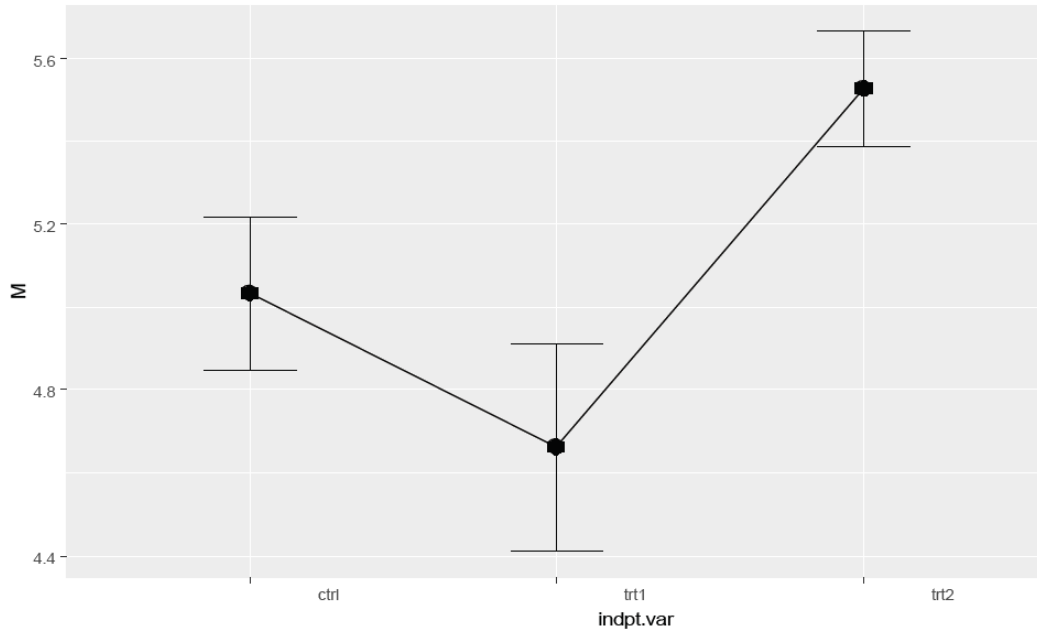
There are lots of ways to make bar and line graphs in R Studio, but the easiest is to use the `Line_Graph_Function` originally written by Sam Paskewitz and updated by Max Cantor. First, we need to **install the ggplot2 package**. To do this, click the 'Packages' tab (in the same area with 'Files' 'Plots' 'Help' and 'Viewer'), 'Install' and type 'ggplot2'. Then, **download the file called line_graph_function** and save to the desktop, which can be found here:

https://drive.google.com/file/d/0B_6EqJy05y9BbTZUNFVfcDFvSEU/view?usp=sharing

Next, we need to **open and run the line_graph_function in R studio**. To do this, use pull-down menus: File - Open File - `line_graph_function`. The function will open in a new window. Then, select all the text in the file (command + A or Ctrl + A) and click the Run button at the top of the script editor box, or type Ctrl (or command) enter.

Finally, we need to **run the line graph function for our data in the R console**. The code for this is `psych2111plotL (DV, IV, iv2="none")`. Let's create a line graph for the weight (DV) and group (IV) data in the PlantGrowth data file:

```
attach(PlantGrowth)
psych2111plotL(weight, group)
##Loading required package: ggplot2
```



* (Optional) **Bar Graph** (one continuous and one discrete variable)

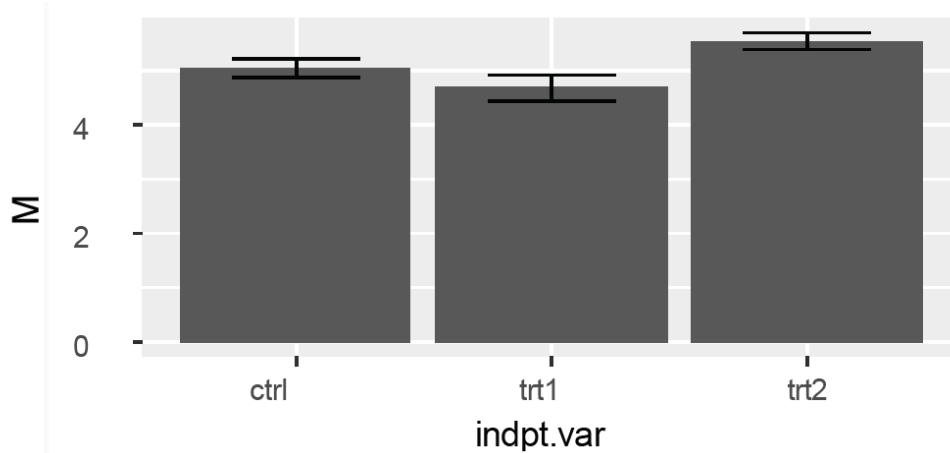
To make a bar graph, follow the same steps as a line graph above, but download the file called `bar_graph_function` and save to the desktop instead, which can be found here:

https://drive.google.com/file/d/0B_6EqJy05y9Bb1djQ0dFYURMLWM/view?usp=sharing

The code for this is `psych2111ploB` (DV, IV, `iv2="none"`). Let's create a bar graph for the weight (DV) and group (IV) data in the `PlantGrowth` data file:

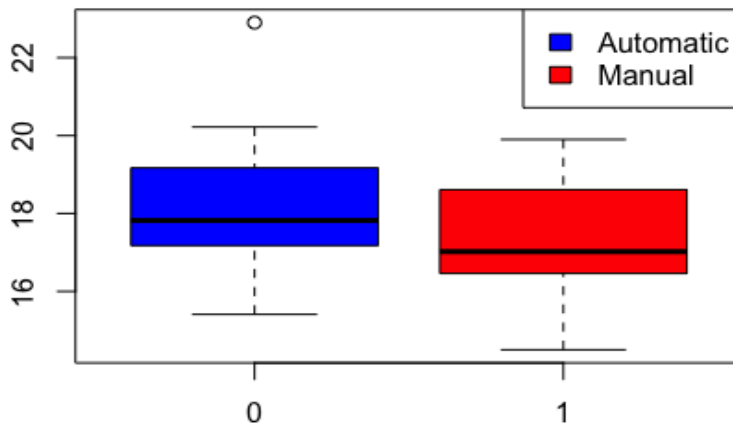
```
attach(PlantGrowth)
psych2111plotB(weight, group)
##Loading required package: ggplot2
```

Notice for the bar graph, that the default is to start the Y-axis at the 0 point. Notice how the line graph, which does not start the Y-axis on the 0 point shows the differences in a different way (and separates the three means with more detail).



Boxplot graph (one continuous and one discrete variable, with a non-normal, skewed distribution)

```
attach(mtcars)
boxplot(qsec~am, col=c('blue','red'))
legend("topright",c("Automatic","Manual"), fill= c("blue","red"))
```



What does this boxplot tell us? The middle line in each box is the median, the top and bottom of each box are the upper and lower quartiles, and the dotted lines extending from each box are the maximum and minimum values, excluding outliers (points that are more than 1.5 times the upper or lower quartile). Boxplots are useful for comparing the spread of different samples - in this case, we can see that the spread of both groups are similar, though the automatic group is a bit higher (and has an outlier that is much higher on the qsec variable than the rest of the sample).

6.2 Creating Graphs in Excel

Bar and Line Graphs (one categorical and one continuous variable)

1. Calculate mean (average) and standard deviation (SD)
2. Place the mean (average) and standard deviation (S.D.) for all groups
3. Select the cells that contain the words 'Group 1' and 'Group 2' as well as the average numbers you calculated
4. Under 'Insert' tab, click the first option under '2-D Column chart' for a bar graph and a '2-D Line chart' for a line graph
5. Click '+' sign on the figure, select 'Axes Titles', select 'Primary Vertical' only (unselect 'Primary Horizontal')
6. Double click on the vertical axes title and rename the axes with a new label
7. Add a title to the figure by clicking on the 'Chart Title' and typing in a new title
8. Click '+' sign on the figure. 'Error Bars', 'More Options', 'Custom' 'Specify Value'
9. Select the S.D numbers from Excel sheet
10. Hit 'enter'

Scatterplot (both continuous variables)

1. Select the cells that contain your data
2. On the 'Insert' tab, in the Charts group, click 'Scatter'
3. Click '+' sign on the figure, select 'Axes Titles', select 'Primary Vertical' and 'Primary Horizontal'
4. Double click on the vertical axes title and rename the axes with a new label
5. Double click on the horizontal axes and rename the axes with a new label

* A graphing template that you can download to make a scatterplot, bar graph, or line graph in Excel can be found at:

https://drive.google.com/file/d/0B_6EqJy05y9BYXltZTJ4ZURhblk/view?usp=sharing