R for DNEP Day One

Andrew Stewart,
Division of Neuroscience and Experimental Psychology,
School of Biological Sciences,
University of Manchester.

By the end of today you will…

- Understand and be able to work with R and the RStudio environment.
- Be able to do basic data wrangling (aka data mangling when it goes wrong…)
- Be able to build AN(C)OVA models for different designs and be able to interpret interactions.
- Be able to build simple linear and multiple regression models in R (incl. diagnostic plots and model selection).
- Know how to produce reports using R Markdown.

Why R?

- R allows you to engage in reproducible research.
- Statistical packages in R reflect the latest advances in the fields of Statistics and Data Science.
- Linear Mixed Models (LMMs) are easy to build in R - many of the best journals now require LMMs to be used as they are more powerful and flexible than classical techniques such as ANOVA.
- Lots of amazing data visualisation packages available.
- In many institutions, the next generation of academics (PhD students, post-docs etc) are learning R skills as part of their training.
- Plays an important role in Open Science.

What role can R play in Open Science?

- R scripts are easy to share allowing for reproducibility and easy public sharing of data and code.
- R is free, open source software that is much more flexible and powerful than SPSS.
- There is an active R community continuously updating statistical tests and packages that run in R.
- As R is a programming language, it forces you to know your data.

R *vs.* SPSS

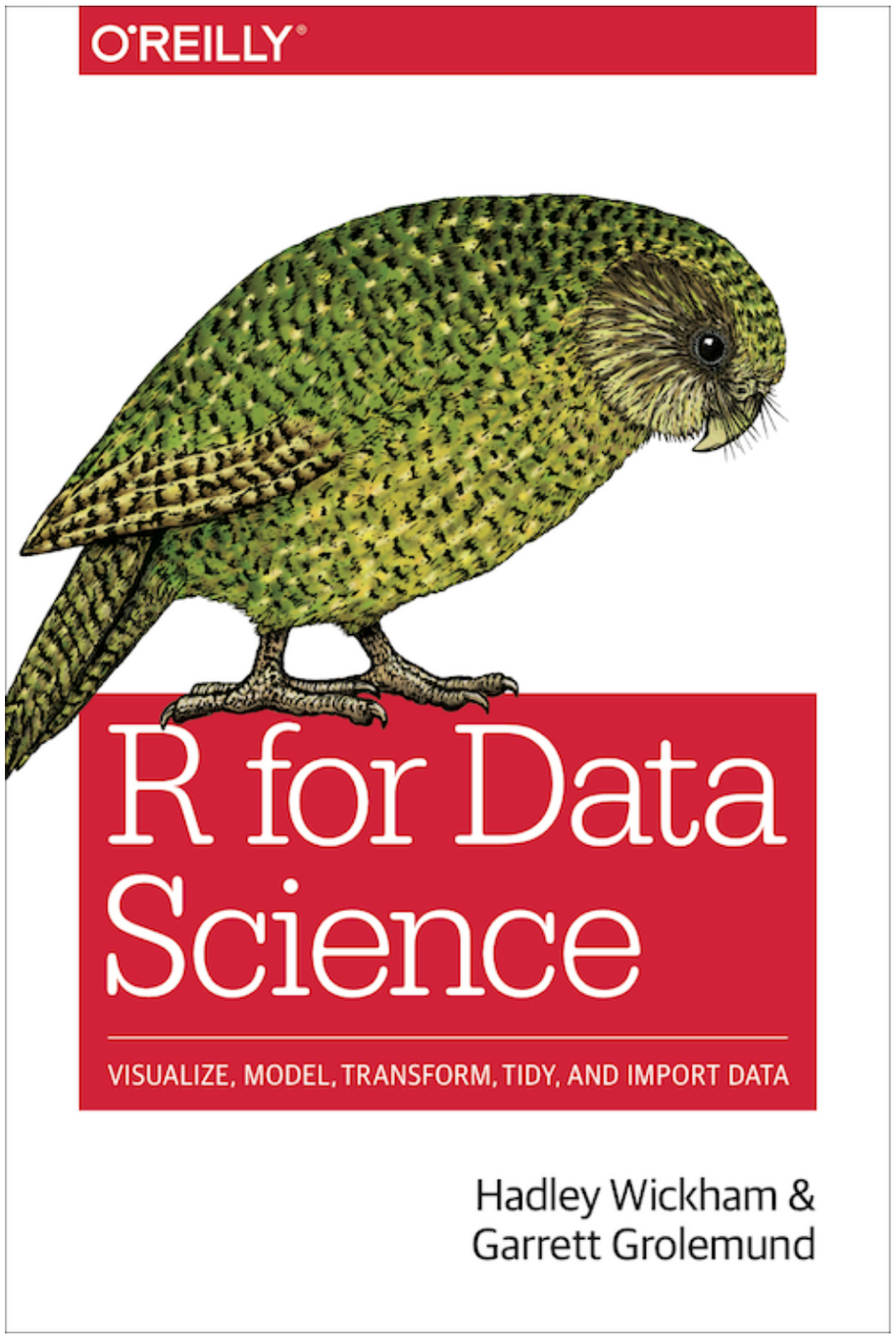
"SPSS is like a bus - easy to use for the standard things, but very frustrating if you want to do something that is not already pre-programmed.

R is a 4-wheel drive off-roader, with a bike on the back, a kayak on top, good walking and running shoes in the passenger seat, and mountain climbing and spelunking gear in the back.

R can take you anywhere you want to go if you take time to learn how to use the equipment, but that is going to take longer than learning where the bus stops are in SPSS." (Greg Snow, 2010, stackoverflow.com).

In meme form…


If statistics programs/languages were cars...
Excel
SPSS
SAS
STATA
R



Available electronically for free at:

http://r4ds.had.co.nz

“Hadley Wickham, the Man Who Revolutionized R”

Chief Scientist at RStudio, author of key R packages incl. `ggplot2`, `plyr`, `dplyr` - all components of the `tidyverse`.

https://priceonomics.com/hadley-wickham-the-man-who-revolutionized-r/

R User Groups…

R user groups exist all over the world with many in the UK (incl. Manchester)…

UK

England

- Birmingham: BRUM
- Canterbury: CanterburyR
- Cambridge: CambR
- Coventry: Warwick R User Group; @WarwickRUG
- Exeter: Exeter R Users Group
- London: LondonR
- Manchester: ManchesterR
- Newcastle: R North East; @RstatsNE
- Nottingham: Nottingham R User group
- Oxford: R user group Oxford; @rusersoxford
- Sheffield: SheffieldR;@Sheffield_R_

You can even buy the t-shirt…

RStudio

$17

magrittr

$17

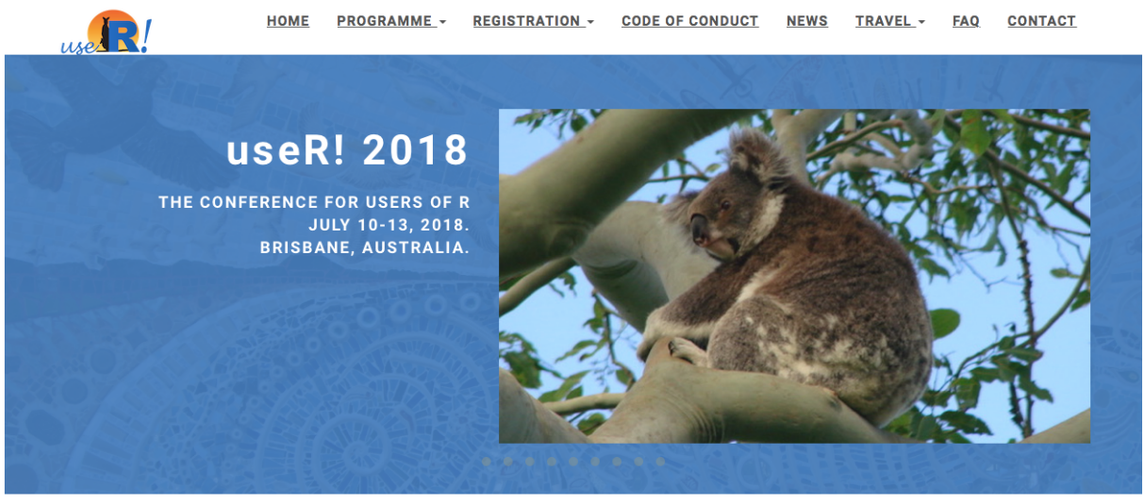
Shiny

$17

I like the way you R

$17

…and conferences



R skills are in high demand…

February 11, 2014

R skills attract the highest salaries

Two recent salary surveys have shown that R language skills attract median salaries in excess of $110,000 in the United States.

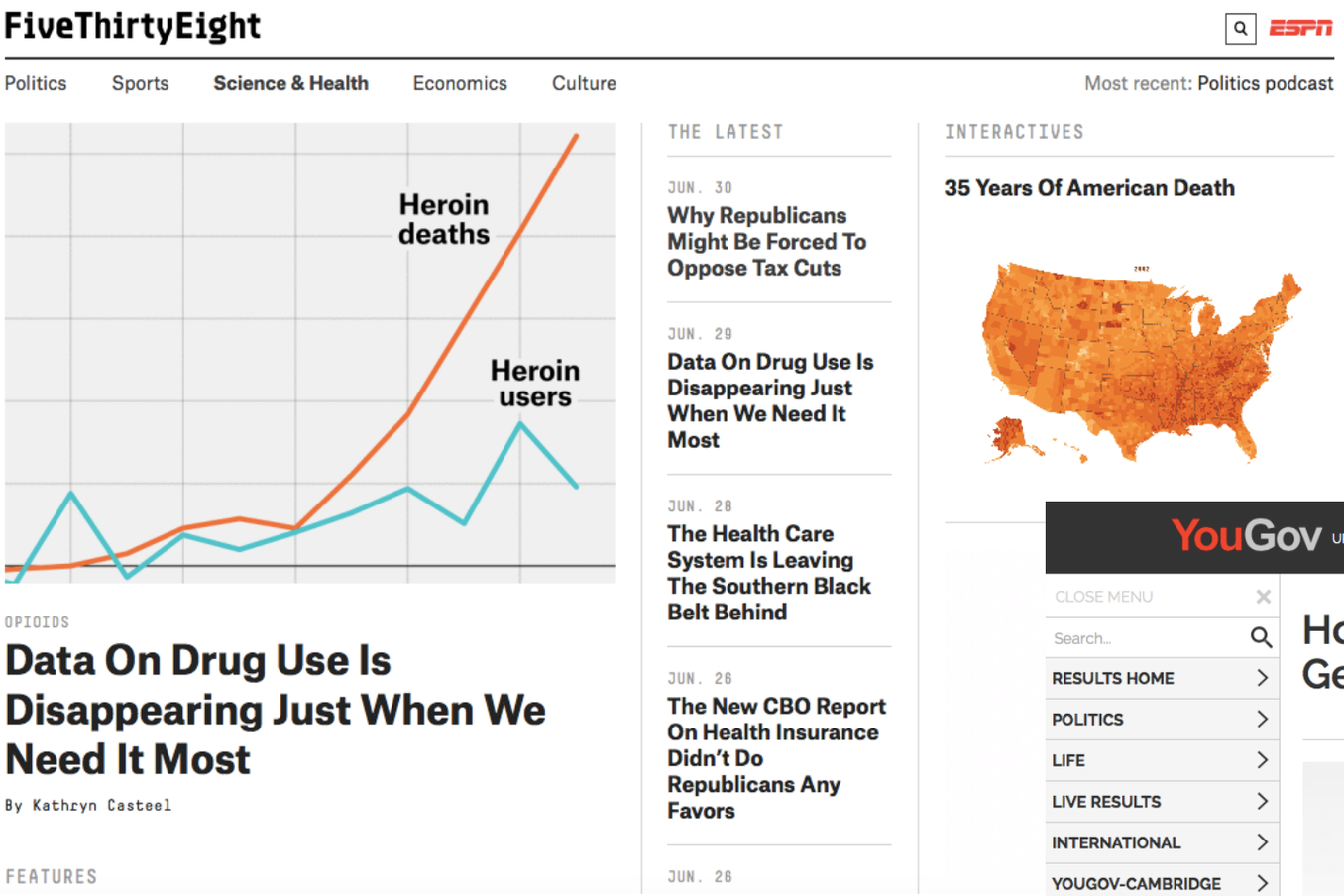
In the 2014 Dice Tech Salary Survey of over 17,000 technology professionals, the highest-paid IT skill was R programming. While big-data skills in general featured strongly in the top tier, having R at the top of the list reflects the strong demand for skills to make sense of, and extract value from big data.

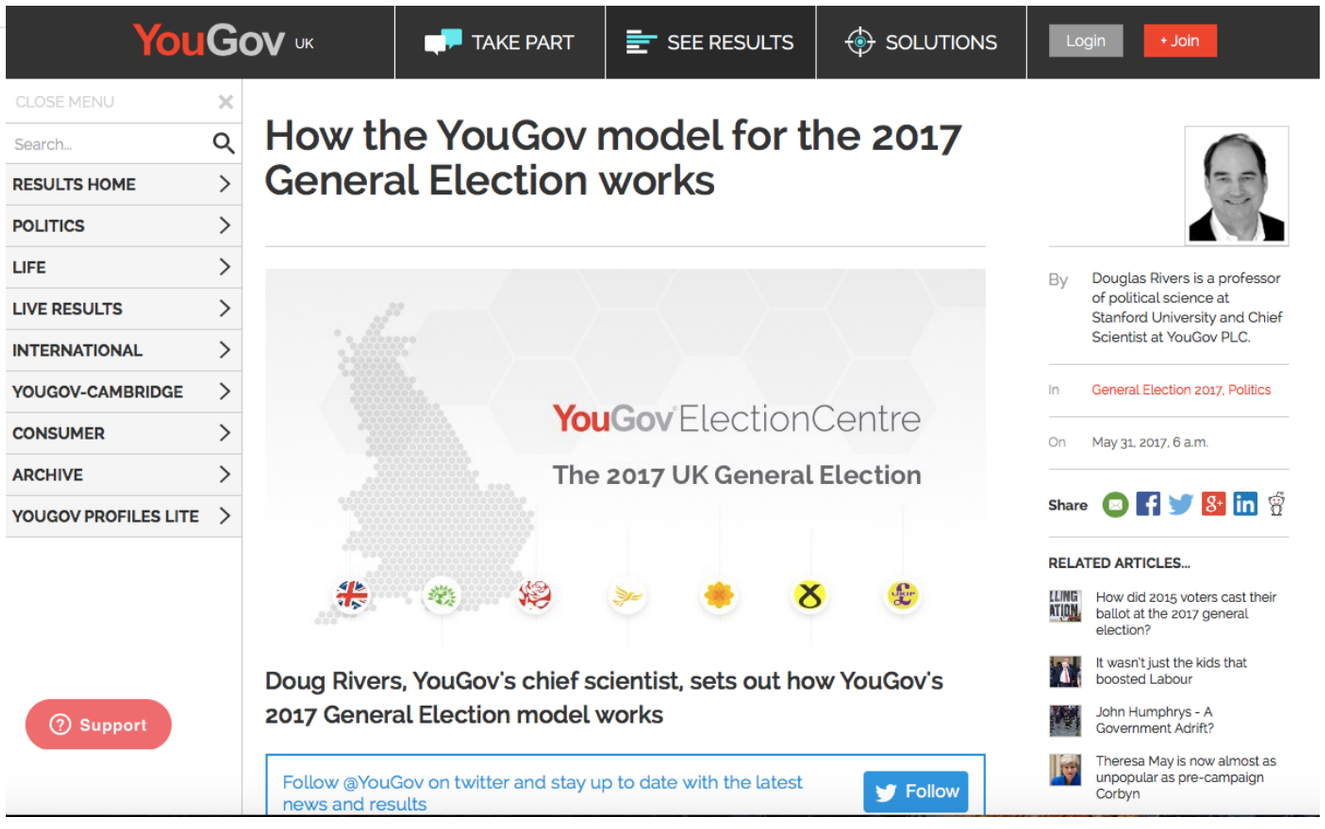
AVERAGE SALARY FOR High Paying Skills and Experience

SKILL	2013	YR/YR CHANGE
R	$ 115,531	n/a
NoSQL	$ 114,796	1.6%
MapReduce	$ 114,396	n/a
PMBok	$ 112,382	1.3%
Cassandra	$ 112,382	n/a
Omnigraffle	$ 111,039	0.3%
Pig	$ 109,561	n/a
SOA (Service Oriented Architecture)	$ 108,997	-0.5%
Hadoop	$ 108,669	-5.6%
Mongo DB	$ 107,825	-0.4%

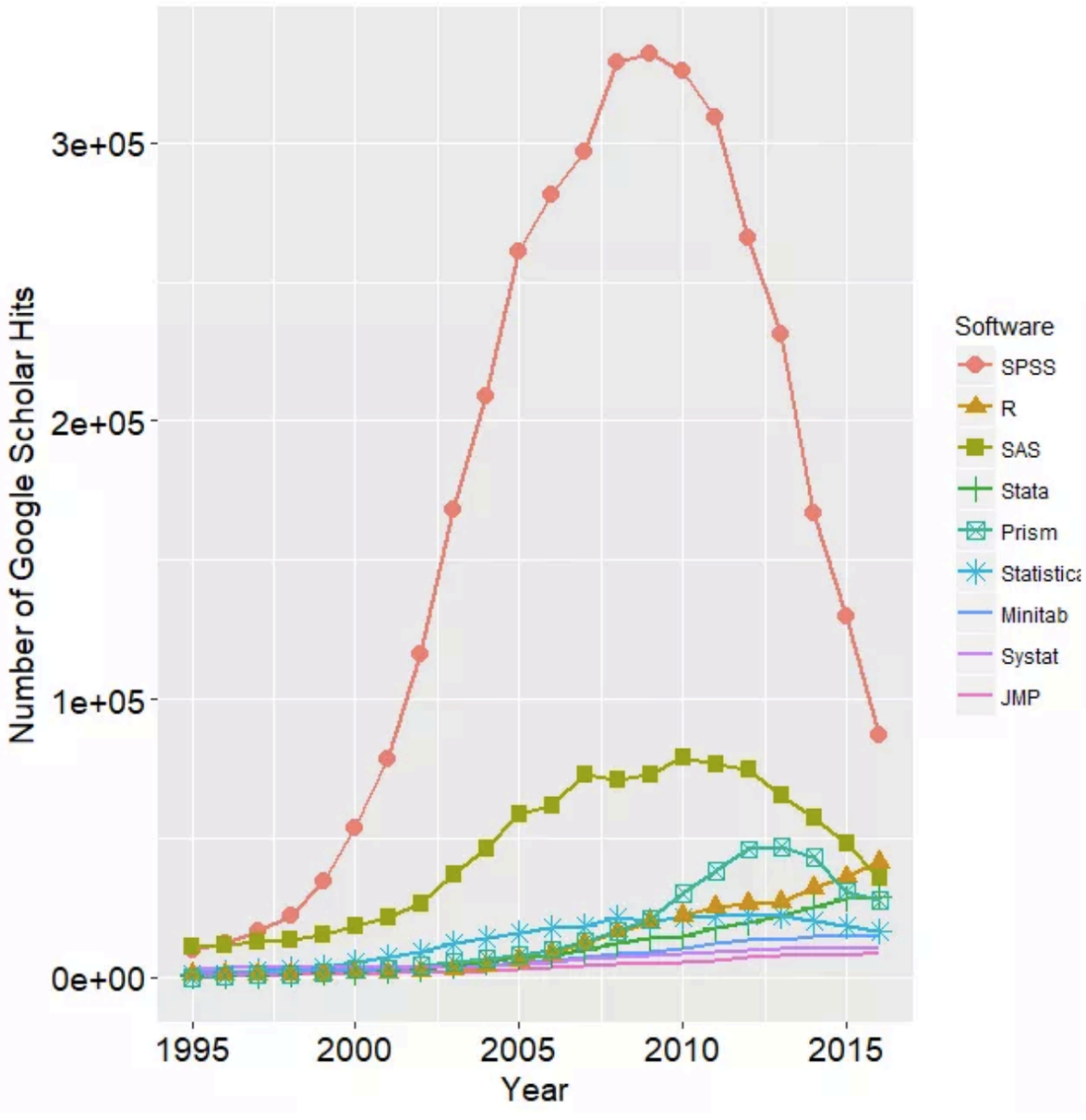
Similarly, the recent O'Reilly Data Scientist Survey also found R skills amongst those that pay in the $110,000-$125,000 range (albeit amongst a much smaller and specialized sample of respondents).

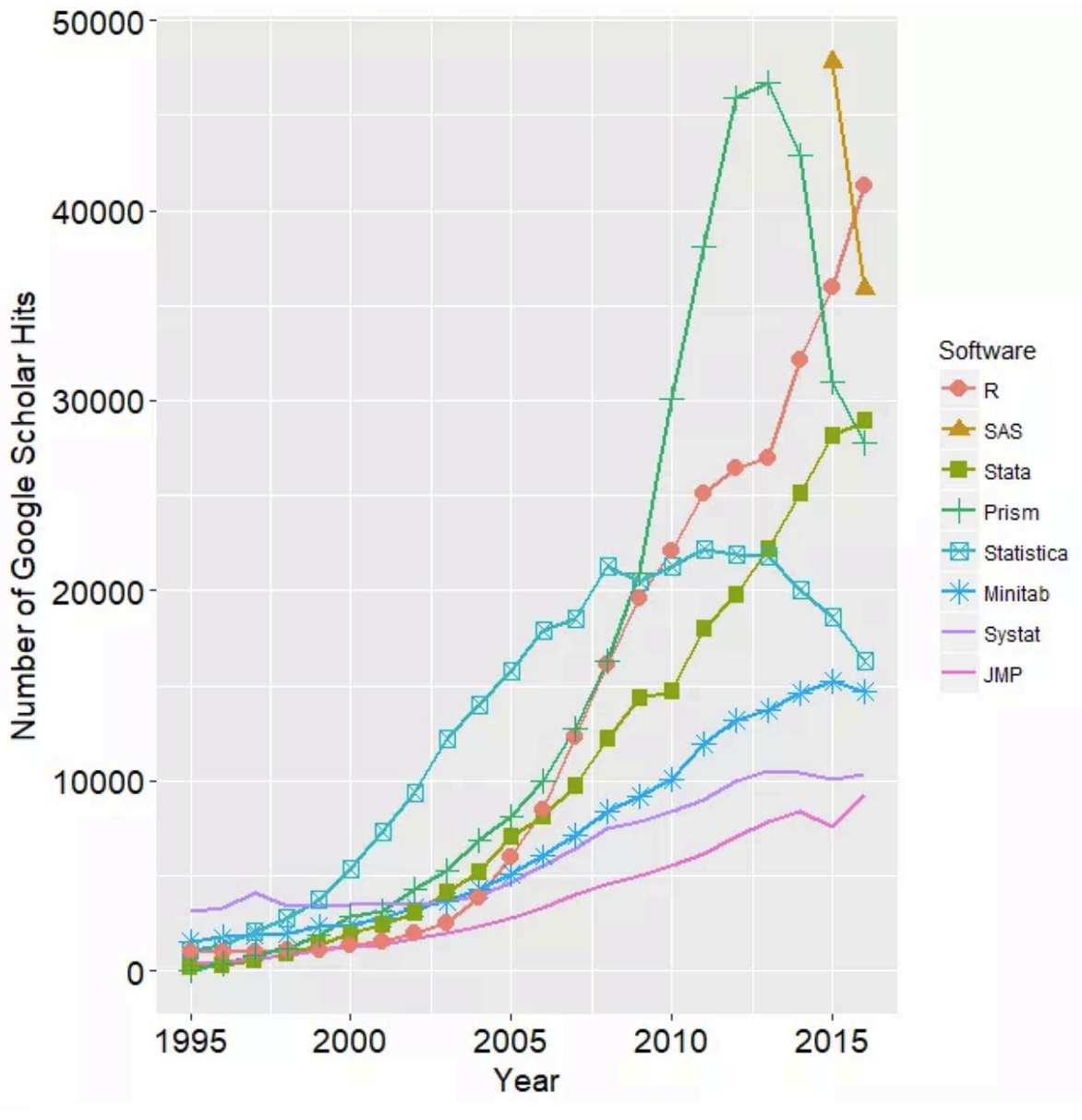
and R is widely used by a number of organisations…





and across academia…





with Data Science a growing employment destination for Psychologists.

CAREER CENTER

Hot jobs: Big-data psychologists

Wanted: Scientists who can help companies make sense of consumer and employee data.

By Rebecca Voelker

Print version: page 18

Every time you use an Internet search engine, sign up for a company "rewards" program or swipe your credit card, that information is saved and stored. The industries that amass these billions of bytes of data are increasingly hiring psychologists to help make sense of it, says Douglas Reynolds, PhD, president of APA's Div. 14 (Society for Industrial and Organizational Psychology).

"Big data, and what it means for business, is a hot topic right now," says Reynolds, who also serves as vice president of assessment technology at Development Dimensions International Inc., a human resources consulting firm. "We're in high demand."

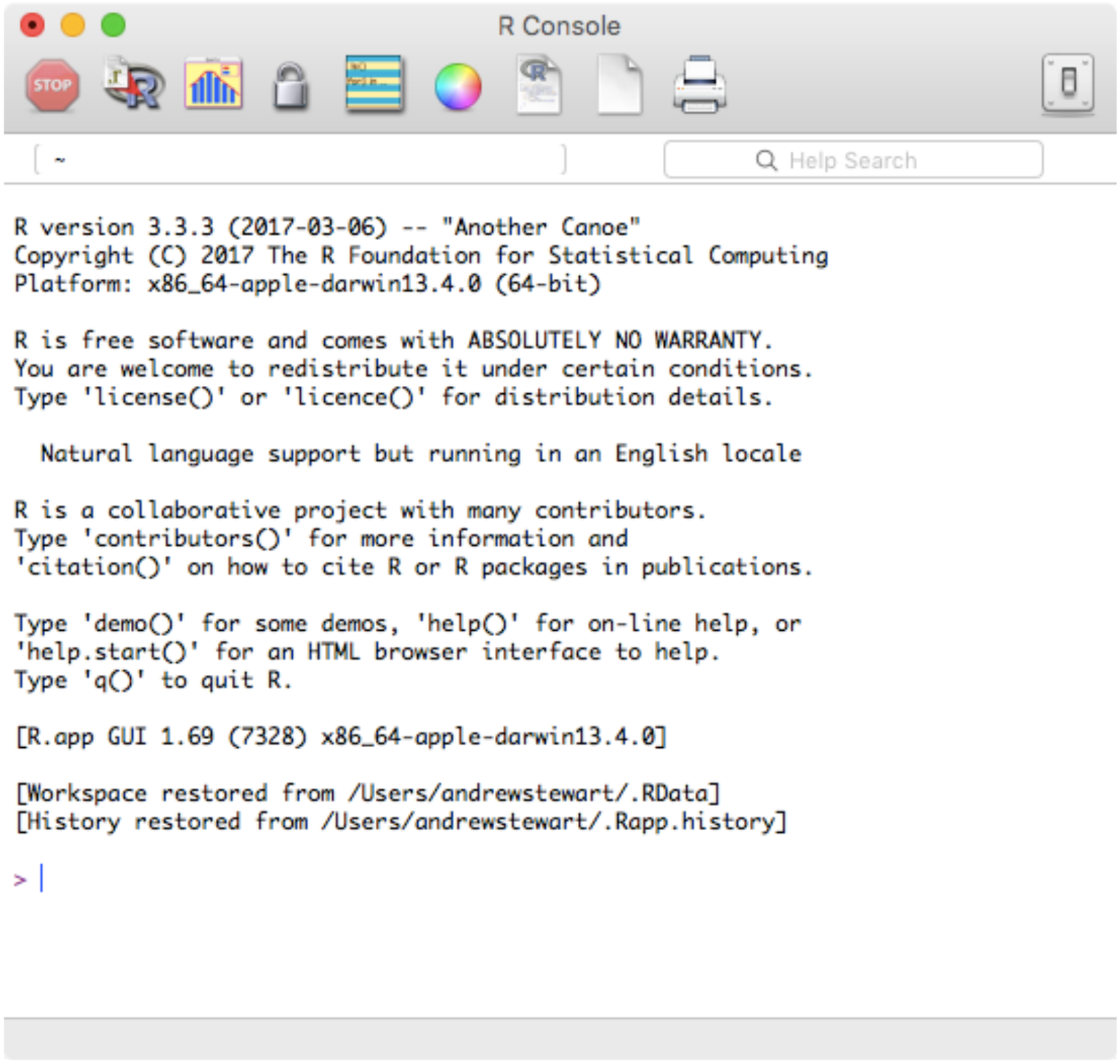
Psychologists' ability to interpret numbers and human behavior makes them key members of many industry analytics teams, adds Suzie Weaver, PhD, a psychologist and senior analytic consultant with Epsilon, a global marketing and analytics company. "Analytics is at the core of everything we do, whether it's in research, the academic sphere or the business world."

Starting R

- R is open source (i.e., free to download and add to). Main R site is:
- *www.r-project.org*
- From here you can download R (from one of the CRAN[1] mirrors for Windows, Mac and UNIX).
- R updates regularly (you need to update manually).

[1] *CRAN = The Comprehensive R Archive Network*

- When you first load R it looks like this:



- Rather than using the R interface, you should use the RStudio graphical interface.

- Download it from www.rstudio.com

- When you run RStudio, it looks like this:

RStudio File Edit Code View Plots Session Build Debug Tools Window Help — Thu 15:21

~/Desktop/Air Work/MRes 2016:17/R workshop MRes/R workshop directory/Workshop - RStudio

Go to file/function | Addins | Workshop

Workshop script2.R | DV

Source on Save | Run | Source

```
library (lmerTest)
library (lsmeans)
library (pbkrtest)

#Read in First Pass Data
FPs <- read.csv("~/Desktop/Air Work/R analyses/Indirect Request Expt/Experiment 1 - probability of success - Libby's data/FP

#this sets up the contrasts so that the intercept in the mixed LMM is the grand mean (i.e., the mean of all conditions)
my.coding <- matrix (c(.5, -.5))

contrasts (DV$Context)<-matrix (c(.5, -.5))
contrasts (DV$Sentence)<-matrix (c(.5, -.5))

#construct the modelswith crossed random effects for subjects and items for the pre-critical, critical and post-crtical regi

model.full <- lmer (RT ~ Context*Sentence + (1+Context*Sentence |Subject) + (1+Context*Sentence |Item), data=DV, REML=TRUE)
model.null <- lmer (RT ~ (1+Context*Sentence |Subject) + (1+Context*Sentence |Item), data=DV, REML=TRUE)

summary (model.full)
lsmeans (model.full, pairwise~Context*Sentence, adjust="none")

fpmodels <- lmer (statement ~ Meaning*Probability + (1+Meaning*Probability |R_s) + (1+Meaning*Probability |Item), data=FPs
```

23:1 (Top Level) — R Script

Console ~/Desktop/Air Work/MRes 2016:17/R workshop MRes/R workshop directory/Workshop/

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

    step

> library("lsmeans", lib.loc="/Library/Frameworks/R.framework/Versions/3.3/Resources/library")
Loading required package: estimability

Attaching package: 'lsmeans'

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

    lsmeans

> model.full <- lmer (RT ~ Context*Sentence + (1+Context*Sentence |Subject) + (1+Context*Sentence |Item), data=DV, REML=TRUE)
```

Error in strsplit(keys, " *, *") : non-character argument

Environment | History

Import Dataset | List

Global Environment

Data

DV	1680 obs. of 5 variables
my.coding	num [1:2, 1] 0.5 -0.5

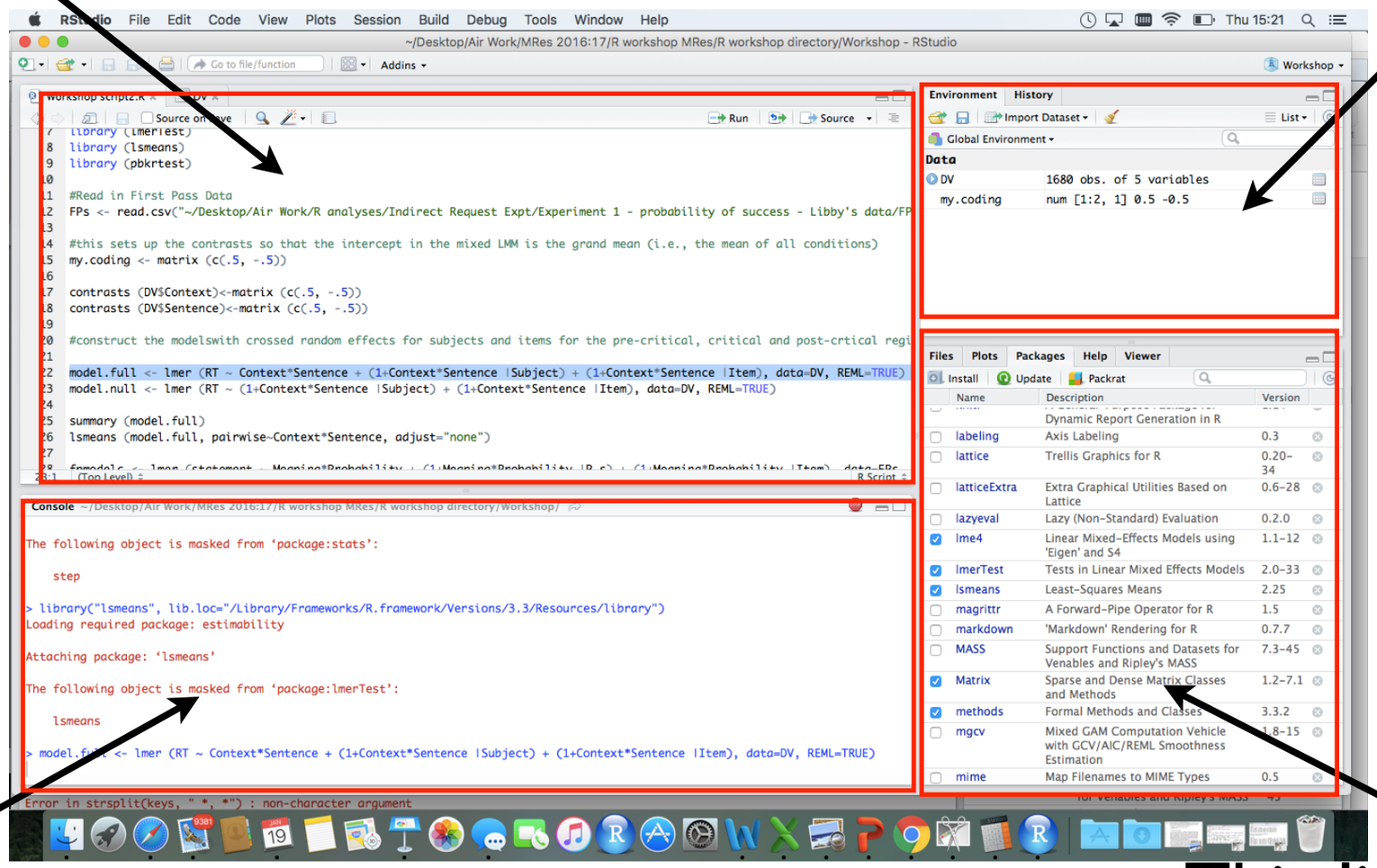
Files | Plots | Packages | Help | Viewer

Install | Update | Packrat

Name	Description	Version
	Dynamic Report Generation in R	
labeling	Axis Labeling	0.3
lattice	Trellis Graphics for R	0.20-34
latticeExtra	Extra Graphical Utilities Based on Lattice	0.6-28
lazyeval	Lazy (Non-Standard) Evaluation	0.2.0
lme4	Linear Mixed-Effects Models using 'Eigen' and S4	1.1-12
lmerTest	Tests in Linear Mixed Effects Models	2.0-33
lsmeans	Least-Squares Means	2.25
magrittr	A Forward-Pipe Operator for R	1.5
markdown	'Markdown' Rendering for R	0.7.7
MASS	Support Functions and Datasets for Venables and Ripley's MASS	7.3-45
Matrix	Sparse and Dense Matrix Classes and Methods	1.2-7.1
methods	Formal Methods and Classes	3.3.2
mgcv	Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation	1.8-15
mime	Map Filenames to MIME Types	0.5

This is where you build your script and where data can be seen.

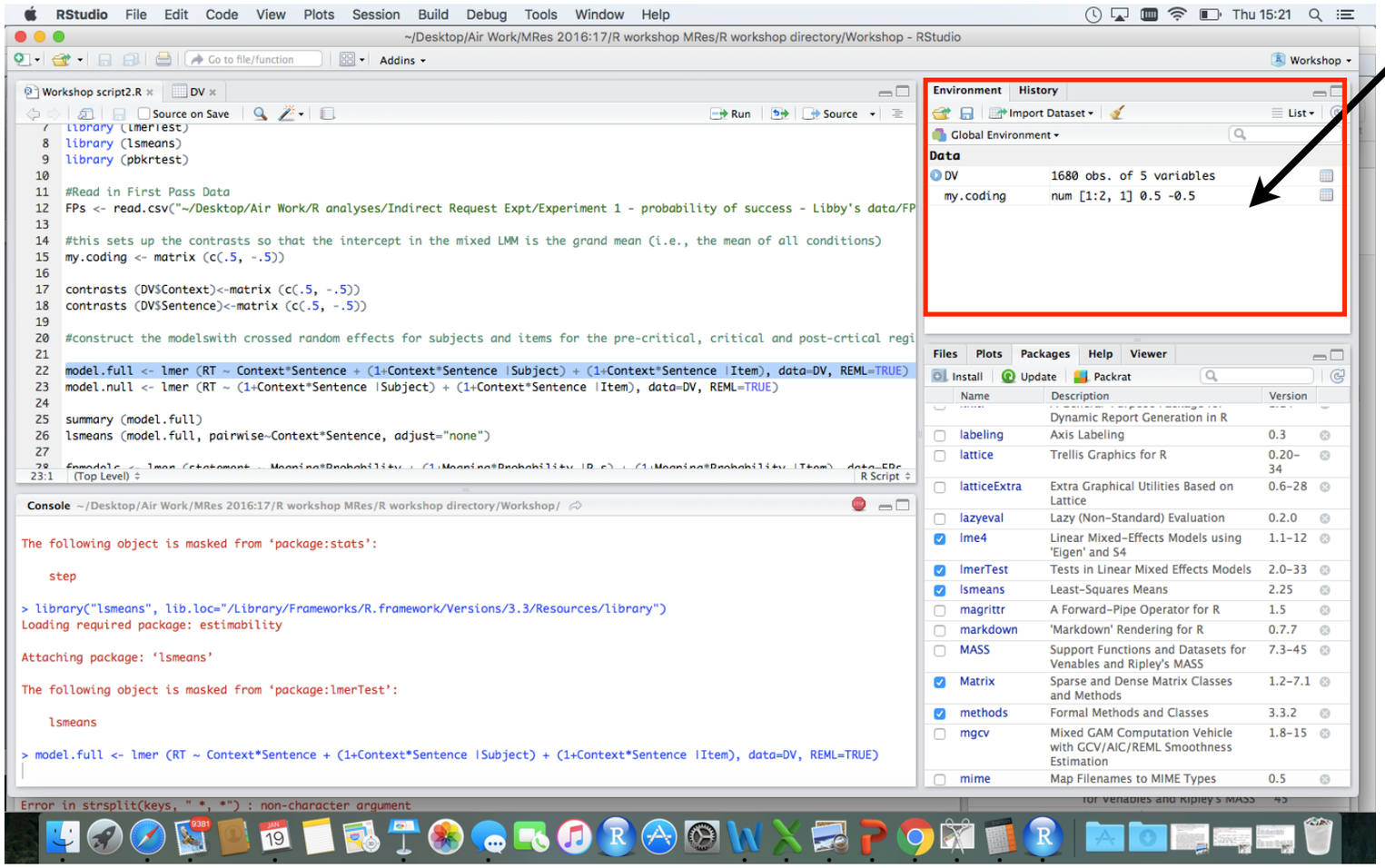
This contains information about your variables and open data sets.



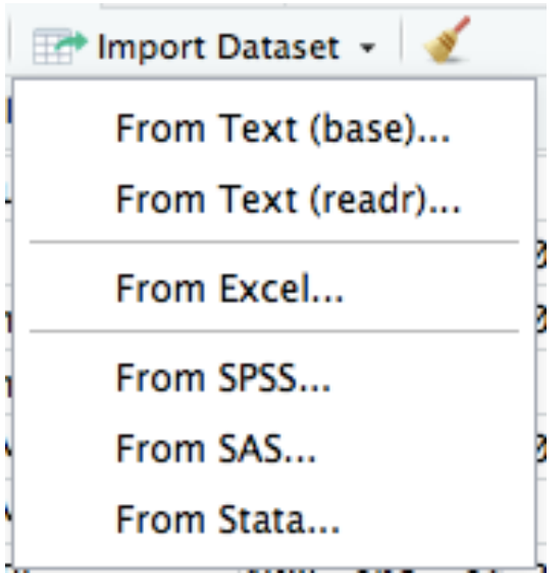
This is where you type commands.

This lists the packages you have loaded, and has tabs for help, graphs etc.

Here is where you import your data.



- Importing data (CSV (Text), Excel, SPSS, SAS and Stata format). CSV can be delimited by commas, tabs etc. Most of the time you'll use the `readr` import function…



Import Text Data

File/Url:

~/Desktop/Air Work/R analyses/R courses/R course/R Work Folder/priming data.txt

Browse...

Data Preview:

Subject *(integer)*	Priming Condition *(character)*	RT *(integer)*
1	LONG	500
2	LONG	551
3	LONG	479
4	LONG	583
5	LONG	522
6	SHORT	399
7	SHORT	423
8	SHORT	444
9	SHORT	410
10	SHORT	398

Previewing first 50 entries.

You can change the type of each variable by clicking on the down arrow.

You should change this to type Factor (LONG vs. SHORT).

Import Options:

Name: priming_data

Skip: 0

First Row as Names

Trim Spaces

Open Data Viewer

Delimiter: Tab

Quotes: Default

Locale: Configure...

Escape: None

Comment: Default

NA: Default

Code Preview:

```
library(readr)
priming_data <- read_delim("~/Desktop/Air Work/R analyses/R courses
/R course/R Work Folder/priming data.txt",
    "\t", escape_double = FALSE, trim_ws = TRUE)
View(priming_data)
```

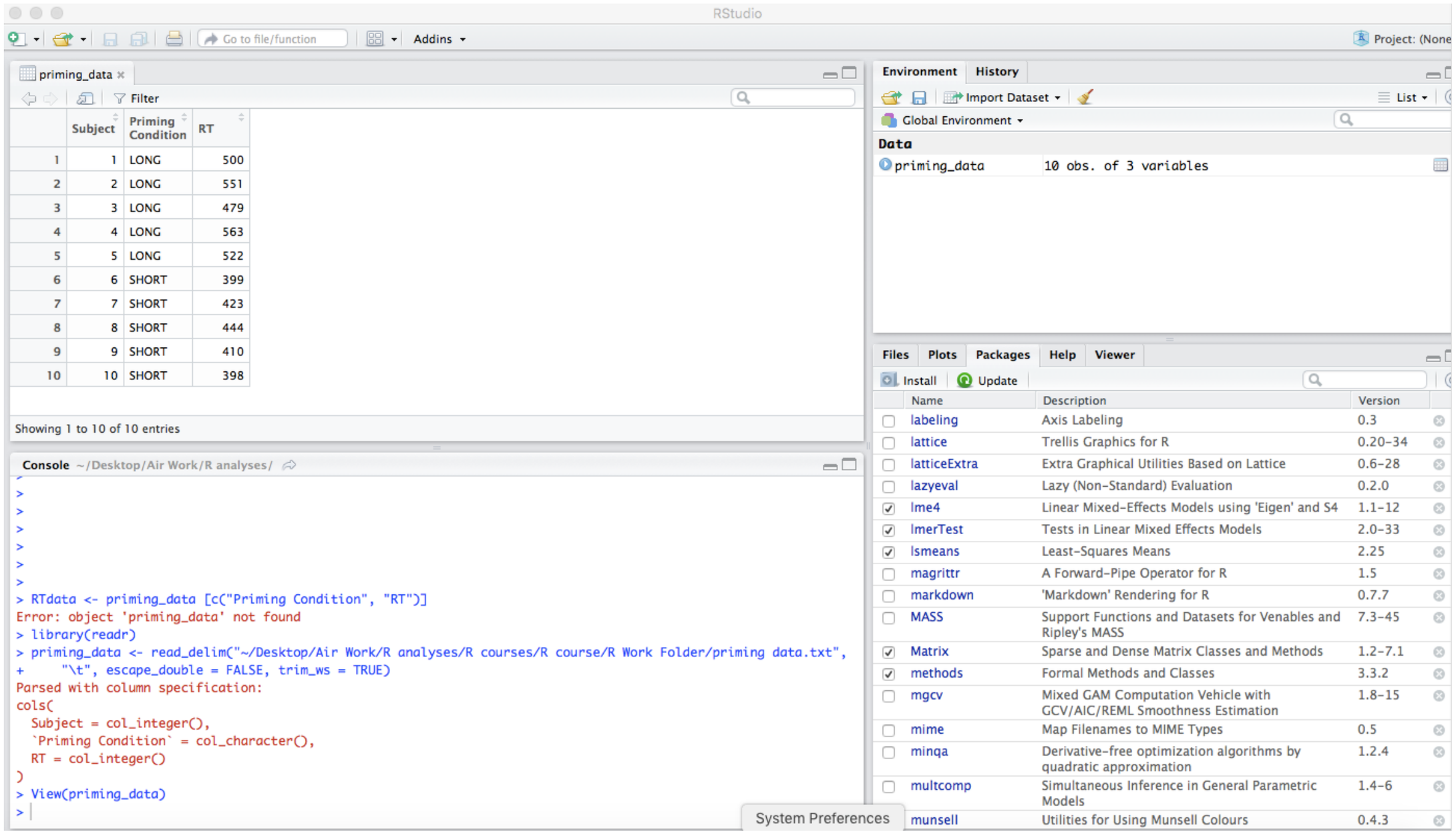
Import

Cancel

Can change here how the columns are delimited.

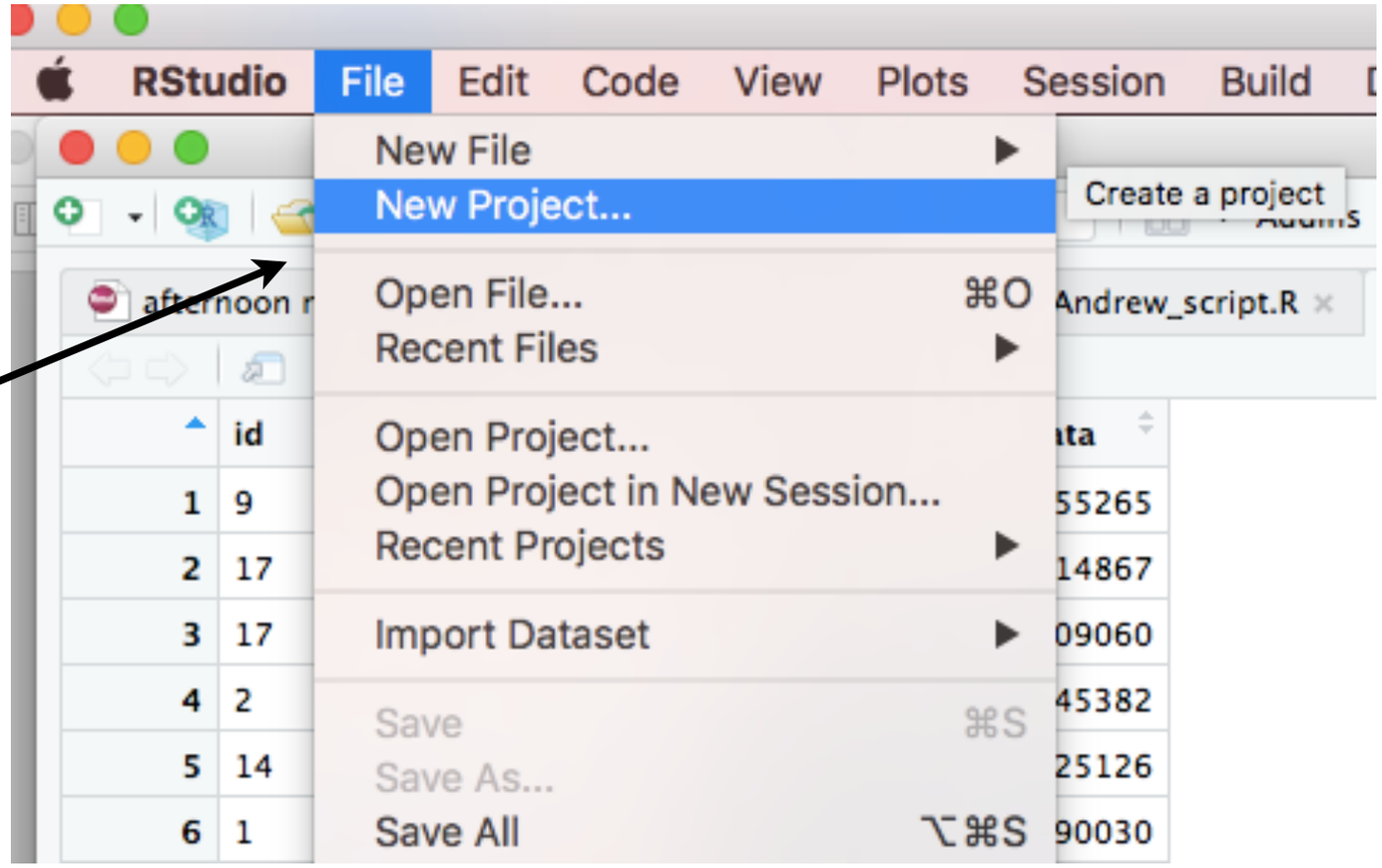
This is the code that corresponds to what you're getting R Studio to do.

- R Studio now looks like this:



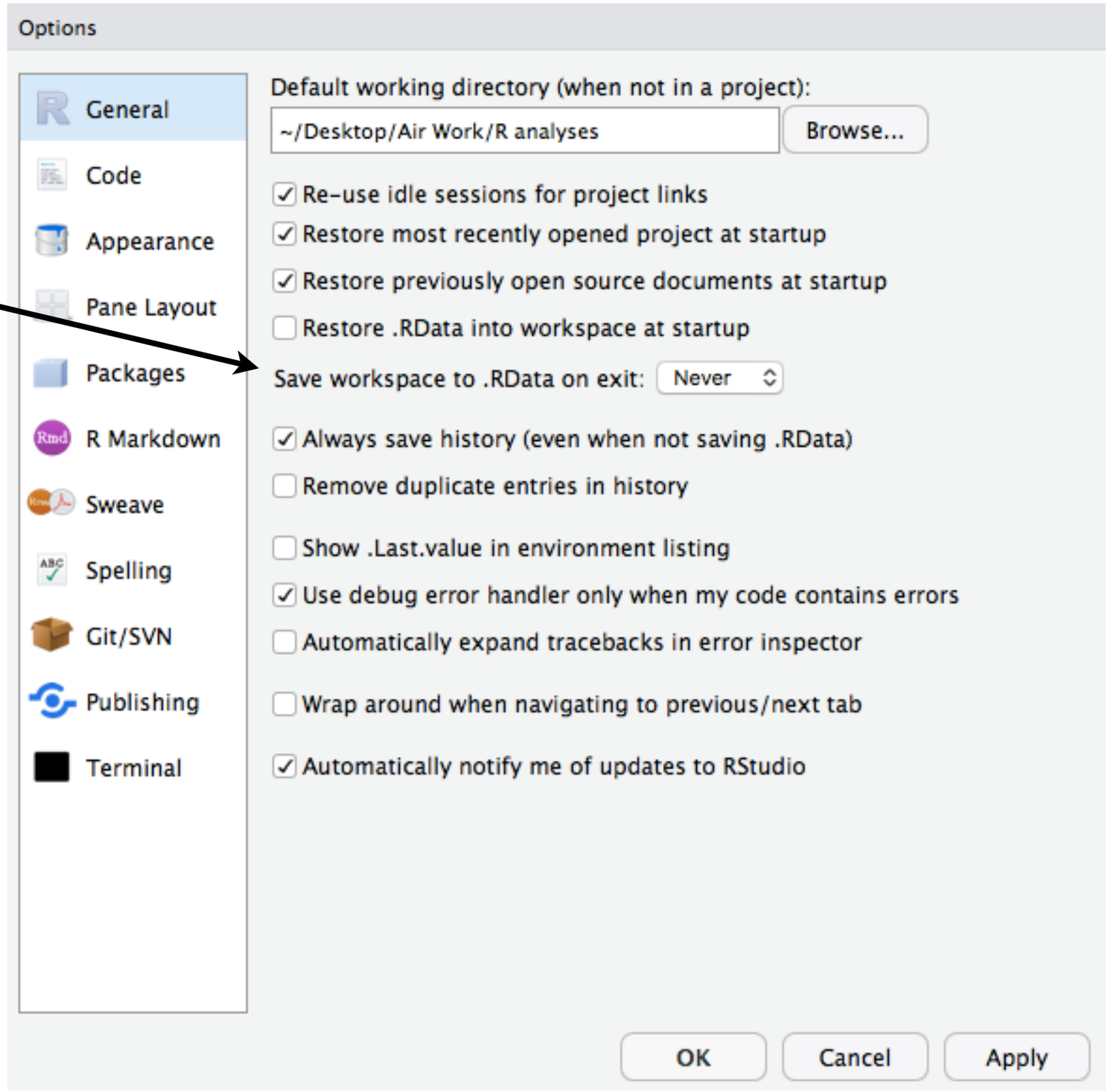
When Starting R

Always create a new project - this will keep all your files nice and organised.

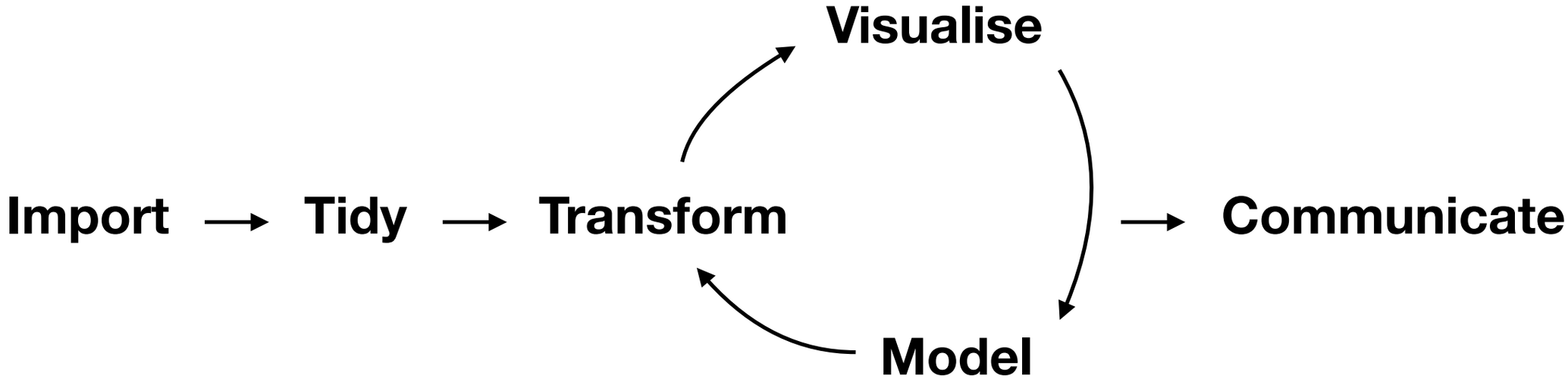


When Starting R

Under preferences, make sure you <u>uncheck</u> the saving workspace option…



Analysis Workflow in the Tidyverse (Garrett Grolemund and Hadley Wickham) - from Data to Write-up



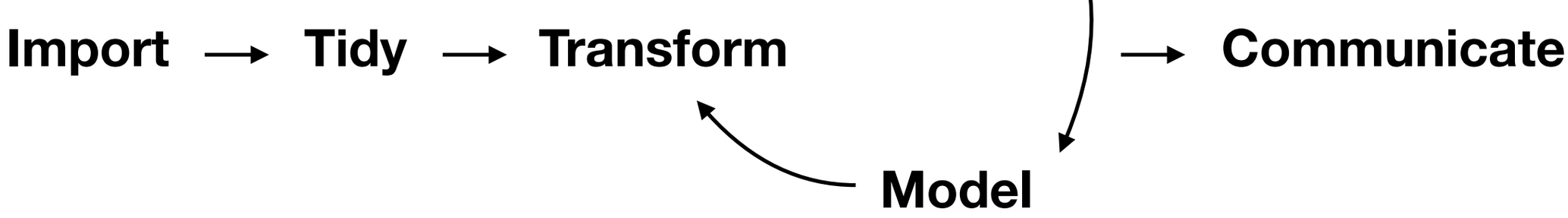
Workflow

Data plots using `ggplot()` **and** `pirateplot()`

Tidying and transforming using `dplyr` **and** `tidyr`

Visualise

html, pdf and doc reports using `RMarkdown`



ANO(C)VA using `afex()` **and** `aov()`
Linear regression using `lm()` **and** `step()`
(Generalised) linear mixed models using `lmer()` **and** `glmer()`

Packages in R

R has a number of functions already built in - but for much of what we do we need to load particular packages to let us use additional functions. We do this by:

```
> install.packages (“packagename”)
```

```
> library (packagename)
```

Every time you start R afresh, you need to load the packages you will use by using the `library` function or the `require` function.

Writing Scripts in R

Source on Save | Run | Source

```
1  library (lme4)
2  library (lmerTest)
3  library (emmeans)
4  library (pbkrtest)
5  library (readr)
6  library (ggplot2)
7  library (ordinal)
8  library (plyr)
9  library (psych)
10 library (yarrr)
11 library (car)
12 library (RVAideMemoire)
13
14 Main <- read_csv("Main.csv")
15
16 Main$Subject <- as.factor (Main$Subject)
17 Main$Image <- as.factor (Main$Image)
18 Main$SportType <- as.factor (Main$SportType)
19 Main$VideoCondition <- as.factor (Main$VideoCondition)
20
21 #VideoCondition 2=congruent, 3=Incongruent, 4=neutral
22 Main$VideoCondition <- revalue (Main$VideoCondition, c("2"="Congruent", "3"="Incongruent", "4"="Neutral"))
23
```

23:1 (Top Level) R Script

Ultimately you will be writing scripts that will allow yourself and others to recreate your analysis just by running the script - the code at the start of the script will load the packages your analysis needs, then load your data, tidy, transform and visualise your data, and then code for your model…

Writing Scripts in R

Scripts should have lots of comments, indicated by a # symbol - this helps others read your code, and also yourself when you look back at it later.

Scripts can be saved and uploaded to (e.g.) OSF, GitHub, or submitted as an electronic supplement alongside your journal submission. Even when journals don't require you to adhere to Open Science practices, it's a good idea to make your code and data available during the reviewing process - and publicly available once your paper is published.

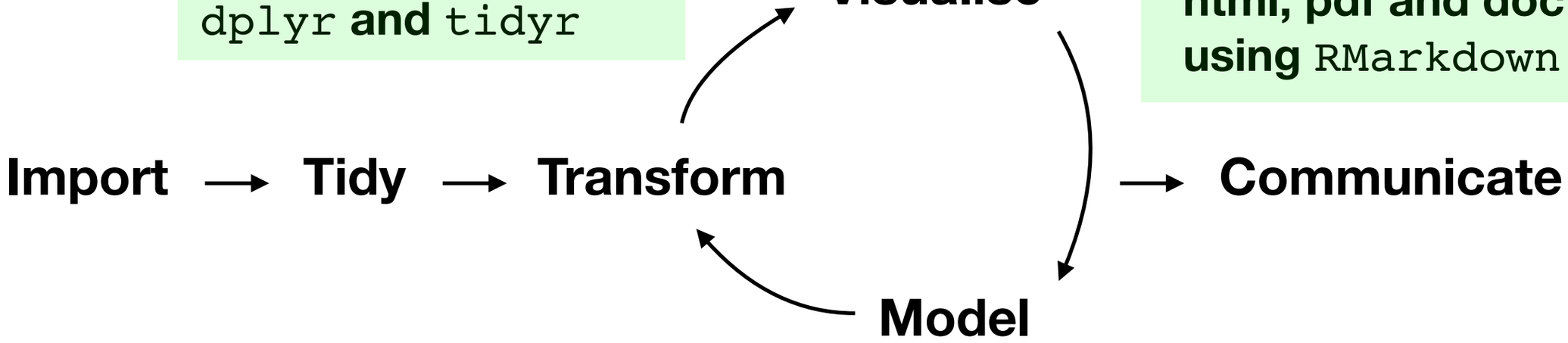
If you aren't sharing your data and code, you aren't engaged in generating reproducible or open research...

Workflow

Tidying and transforming using `dplyr` **and** `tidyr`

Data plots using `ggplot()` **and** `pirateplot()`

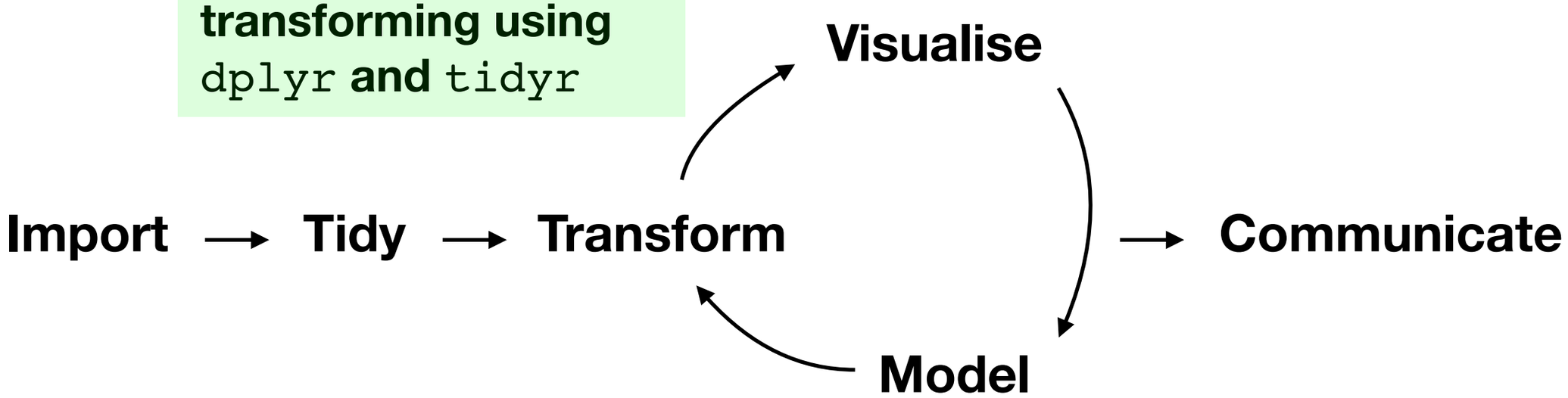
html, pdf and doc reports using `RMarkdown`



ANO(C)VA using `afex()` **and** `aov()`
Linear regression using `lm()` **and** `step()`
(Generalised) linear mixed models using `lmer()` **and** `glmer()`

Tidying and Transforming Data

Tidying and transforming using `dplyr` and `tidyr`



Tidying and Transforming Data

Imagine we have two datasets - one (called `data`) contains a large number of records of individual participants with measures of Working Memory, IQ, and reading comprehension.

If we type into the Console:

```
> View (data)
```

the data frame is displayed like this…

Filter

	ID	WM	IQ	Comp
1	1	43	72	16
2	2	51	109	18
3	3	55	107	18
4	4	38	102	20
5	5	52	121	17
6	6	52	92	16
7	7	47	68	21
8	8	47	97	23
9	9	47	93	22
10	10	45	101	17
11	11	47	86	17
12	12	45	113	21
13	13	46	114	19
14	14	50	99	20

Showing 1 to 14 of 10,000 entries

To get more information about the structure of our data frame we can type:

```
> str (data)
'data.frame':   10000 obs. of  4 variables:
 $ ID  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ WM  : int  43 51 55 38 52 52 47 47 47 45 ...
 $ IQ  : int  72 109 107 102 121 92 68 97 93 101 ...
 $ Comp: int  16 18 18 20 17 16 21 23 22 17 ...
```

So we have 10,000 observations with 4 variables associated with each observation - all of them of type integer.

Imagine that 48 of these 10,000 people also took part in a reading time experiment and we have their reading data (called `dataRT`) for Simple Sentence and Complex Sentence reading conditions:

```
> str (dataRT)
'data.frame':    48 obs. of  3 variables:
 $ ID              : int  9937 1506 5212 374 6757 1778 9421 5576 7326 4166 ...
 $ Simple Sentence : int  1996 2235 2177 1824 2113 2056 2037 2073 1830 1824 ...
 $ Complex Sentence: int  2551 2310 2244 2483 2567 2791 2226 2270 2640 2386 ...
```

We are interested in analysing the data of these 48 people in the data frame called `dataRT` but covarying out the effect of IQ captured in our data frame called `data`.

Problem - how can we combine these two data frames so that we end up with one data frame of 48 people, their reading times plus their individual difference measures?

Manually, in Excel we could open the two data frames as spreadsheets and cut and paste cases where the id number matches…

Probably ok for 48 participants, but what if you had 200 or 2,000?

In R, we can use the `inner_join` function from the `dplyr` package where we join the two data frames matched by ID.

```
> inner_join (data, dataRT, by=(c("ID")))
      ID WM  IQ Comp Simple Sentence Complex Sentence
1    374 49  80    19            1824              2483
2    516 45 108    23            1800              2348
3    570 45 114    17            1942              2409
4    648 50 102    22            2167              2308
5    691 49 112    20            1816              2313
6    882 45 113    24            1848              2237
7   1033 44 100    19            1882              2219
8   1191 50  89    17            1731              2417
9   1338 50  82    21            1868              2462
10 1506 48  96    20            2235              2310
```

We can use the assignment symbol <- to assign the output of this `inner_join` function to a new variable I'm calling `dataRT_all`. We can ask for the structure of this new data frame using the `str()` function:

```
> dataRT_all <- inner_join (data, dataRT, by=(c("ID")))
> str (dataRT_all)
'data.frame':    48 obs. of  6 variables:
 $ ID              : int  374 516 570 648 691 882 1033 1191 1338 1506 ...
 $ WM              : int  49 45 45 50 49 45 44 50 50 48 ...
 $ IQ              : int  80 108 114 102 112 113 100 89 82 96 ...
 $ Comp            : int  19 23 17 22 20 24 19 17 21 20 ...
 $ Simple Sentence : int  1824 1800 1942 2167 1816 1848 1882 1731 1868 2235 ...
 $ Complex Sentence: int  2483 2348 2409 2308 2313 2237 2219 2417 2462 2310 ...
```

So we have created a new data frame of 48 participants comprised of their reading times and their individual difference measures from two separate (and different sized) data frames…with one line of code…

	ID	WM	IQ	Comp	Simple Sentence	Complex Sentence
1	374	49	80	19	1824	2483
2	516	45	108	23	1800	2348
3	570	45	114	17	1942	2409
4	648	50	102	22	2167	2308

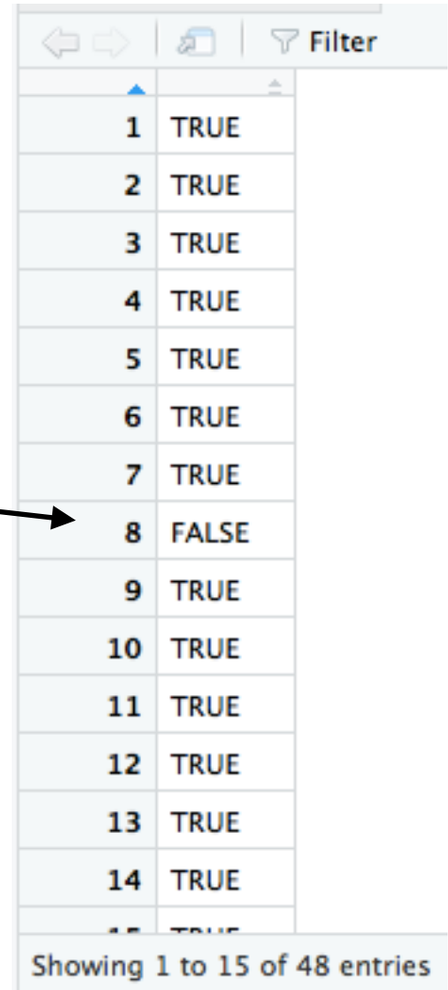
Now imagine we find the distributions of reading times for our two conditions are positively skewed (and we discover the residuals are non-normal). We could log transform these two columns and have two new columns in our data frame - let's call them `log_Simple` and `log_Complex`. We can use the `mutate` function in the `dplyr` package to create two new columns.

```
> data_transformed <- mutate (dataRT_all, log_Simple=log (dataRT_all$`Simple
Sentence`), log_Complex=log (dataRT$`Complex Sentence`))
> data_transformed
     ID WM  IQ Comp Simple Sentence Complex Sentence log_Simple log_Complex
1   374 49  80   19            1824             2483   7.508787    7.844241
2   516 45 108   23            1800             2348   7.495542    7.745003
3   570 45 114   17            1942             2409   7.571474    7.716015
4   648 50 102   22            2167             2308   7.681099    7.817223
5   691 49 112   20            1816             2313   7.504392    7.850493
6   882 45 113   24            1848             2237   7.521859    7.934155
7  1033 44 100   19            1882             2219   7.540090    7.707962
8  1191 50  89   17            1731             2417   7.456455    7.727535
9  1338 50  82   21            1868             2462   7.532624    7.878534
10 1506 48  96   20            2235             2310   7.711997    7.777374
```

Perhaps we have a reason to exclude a particular participant - number 1191 for example. We can set up a logical vector which gives us a value of TRUE for all participants other than 1191:

```
> index <- data_transformed$ID!=1191
> View (index)
```

Values are all set to TRUE expect the participant who has the ID value of 1191



!= stands for “not equal to”- here is the full list of logical operators in R:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

We can now apply our logical vector to our `dataRT_all` data frame and create a new filtered data frame (which I am calling `filtered_data`):

```
> index <- data_transformed$ID!=1191
> filtered_data <- data_transformed[index,]
> filtered_data
      ID WM  IQ Comp Simple Sentence Complex Sentence log_Simple log_Complex
1    374 49  80   19             1824              2483   7.508787    7.844241
2    516 45 108   23             1800              2348   7.495542    7.745003
3    570 45 114   17             1942              2409   7.571474    7.716015
4    648 50 102   22             2167              2308   7.681099    7.817223
5    691 49 112   20             1816              2313   7.504392    7.850493
6    882 45 113   24             1848              2237   7.521859    7.934155
7   1033 44 100   19             1882              2219   7.540090    7.707962
9   1338 50  82   21             1868              2462   7.532624    7.878534
10 1506 48  96   20             2235              2310   7.711997    7.777374
```

We could then run an ANCOVA over the log transformed RTs while covarying out the individual participant effects…

Problem - imagine our data are in the wrong 'shape' - they are in Wide format (each row is one *participant*) but we need them in Long format (each row is one *observation*).

In SPSS, most data will be in Wide format with each experimental condition its own column:

```
> dataRT
   ID Simple Sentence Complex Sentence
1  9937          1996             2551
2  1506          2235             2310
3  5212          2177             2244
4   374          1824             2483
5  6757          2113             2567
6  1778          2056             2791
7  9421          2037             2226
8  5576          2073             2270
9  7326          1830             2640
10 4166          1824             2386
```

For many analyses in R, data need to be in Long format with each row being one observation. So, we want to transform our `dataRT` data frame so it looks like this:

ID	Condition	RT
…	…	…

To do this we can use the `gather()` function in the `tidyr` package.

```
> data_long <- gather (dataRT, "Condition", "RT", c("Simple Sentence", "Complex
Sentence"))
```

The first parameter is the name of the data frame we want to reshape, the second is the name of the new 'Key' column, the third is the name of the new value column and the fourth the names of the columns we want to collapse.

We can use this to create a new data frame called `data_long` which looks like this:

	ID	Condition	RT
1	9937	Simple Sentence	1996
2	1506	Simple Sentence	2235
3	5212	Simple Sentence	2177
4	374	Simple Sentence	1824
5	6757	Simple Sentence	2113
6	1778	Simple Sentence	2056
7	9421	Simple Sentence	2037
8	5576	Simple Sentence	2073
9	7326	Simple Sentence	1830
10	4166	Simple Sentence	1824
11	8817	Simple Sentence	1917
12	7645	Simple Sentence	2008
13	3893	Simple Sentence	2078
14	5199	Simple Sentence	2027

Showing 1 to 14 of 96 entries

And in reverse we can use the `spread()` function to go from Long to Wide data format:

```
> data_wide <- spread (data_long, "Condition", "RT", c("Simple
Sentence", "Complex Sentence”))
> View (data_wide)
```

We’re now back to where we started with data in Wide format:

	ID	Complex Sentence	Simple Sentence
1	374	2483	1824
2	516	2348	1800
3	570	2409	1942
4	648	2308	2167
5	691	2313	1816
6	882	2237	1848
7	1033	2219	1882
8	1191	2417	1731
9	1338	2462	1868
10	1506	2310	2235
11	1521	2452	1858
12	1776	2081	1925
13	1778	2791	2056

Showing 1 to 14 of 48 entries

This is just a small example of functions in the `dplyr` and `tidyr` packages that allow you to tidy, transform, and reshape your data. All of your code for doing this should appear at the start of your analysis script so that others (and you in 5 years or 5 days time) can see exactly what you did.

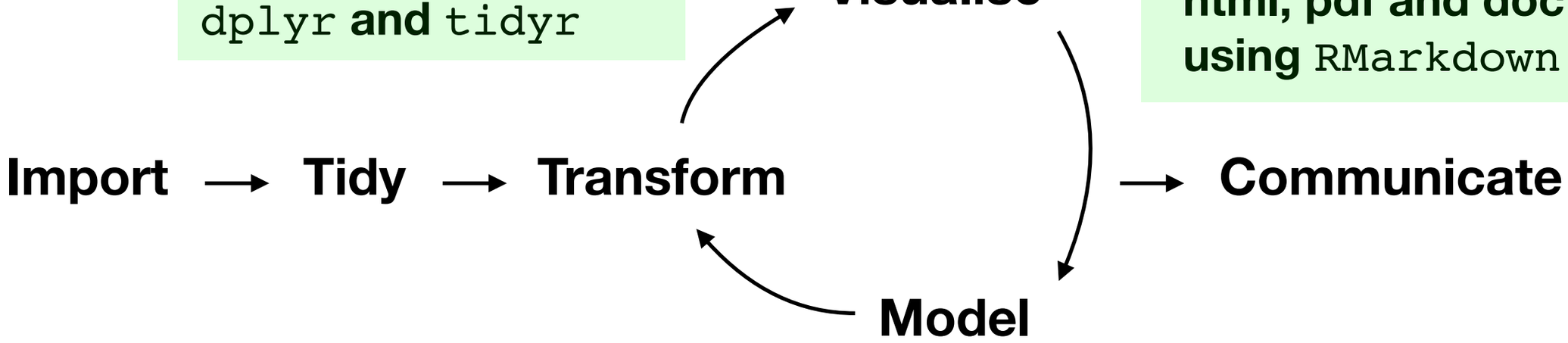
This allows for fully reproducible data preparation in the first part of your analysis workflow (important for Open Science and transparency).

Workflow

Tidying and transforming using `dplyr` **and** `tidyr`

Data plots using `ggplot()` **and** `pirateplot()`

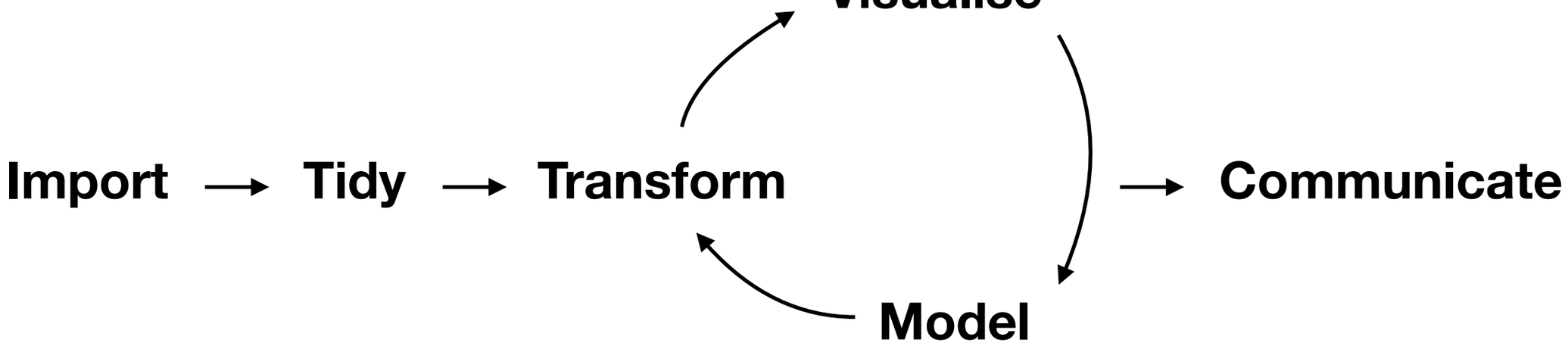
html, pdf and doc reports using `RMarkdown`



ANO(C)VA using `afex()` **and** `aov()`
Linear regression using `lm()` **and** `step()`
(Generalised) linear mixed models using `lmer()` **and** `glmer()`

Visualise

Data plots using `ggplot()` **and** `pirateplot()`



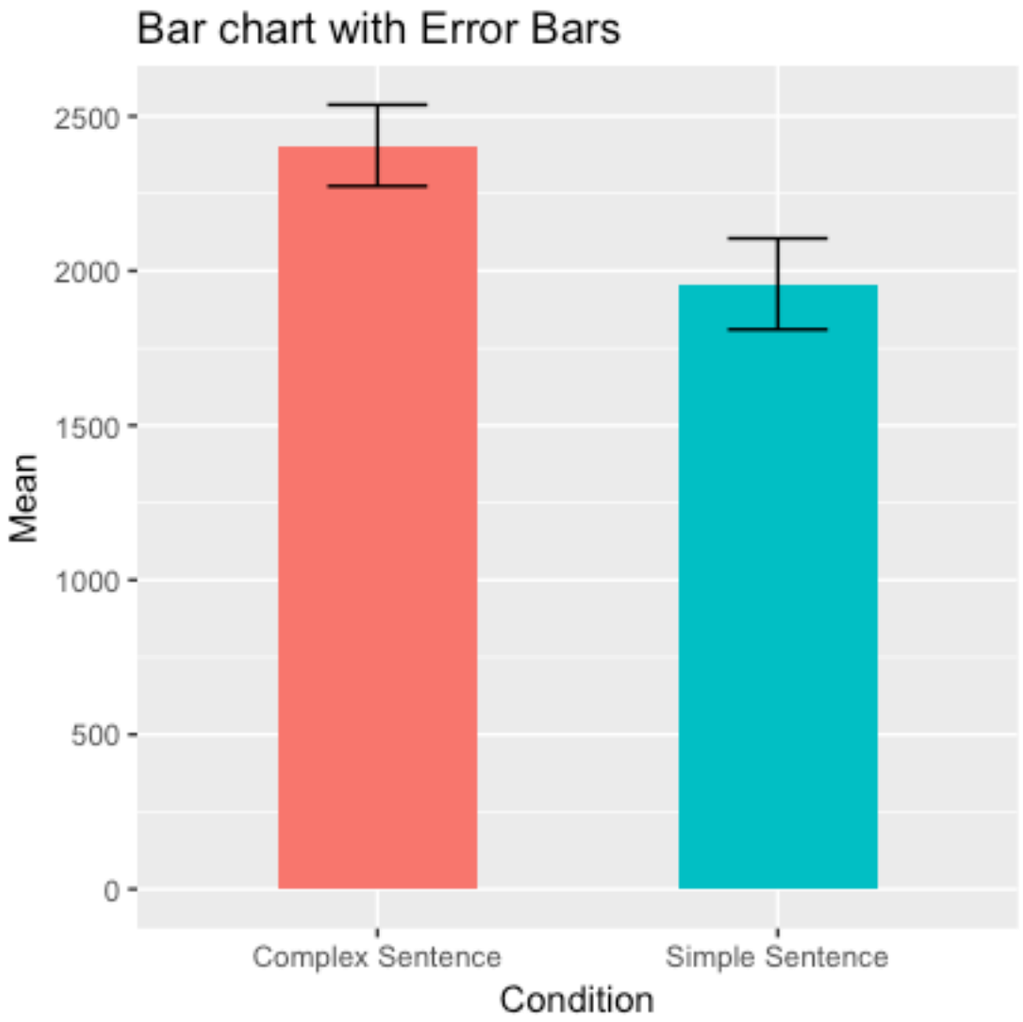
Visualising Your Data

- R has a number of in built graphics functions, but you're more likely to use functions from within the `ggplot2` and `yarrr` packages.

```
> library (ggplot2)

> library (yarrr)
```

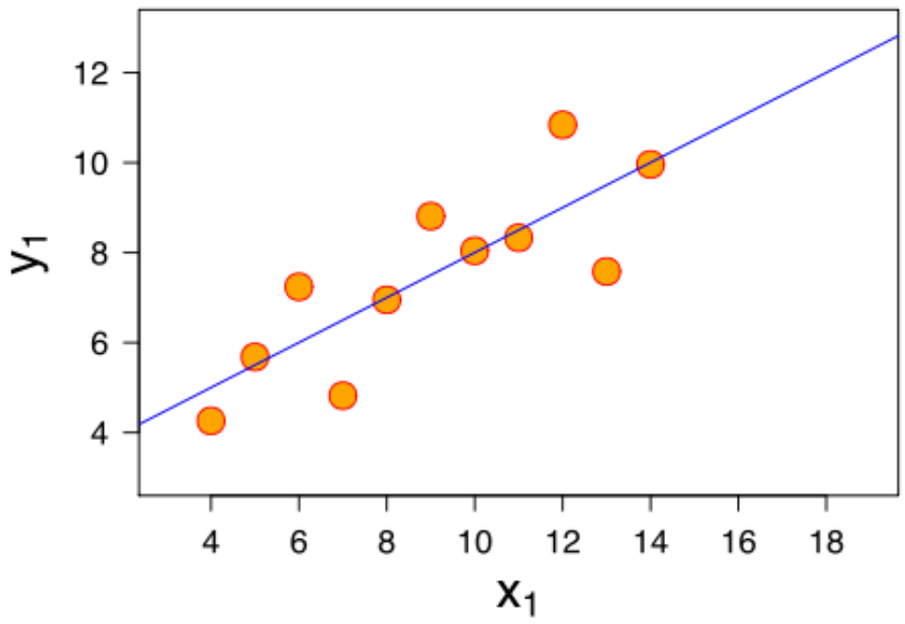
Bar Graphs

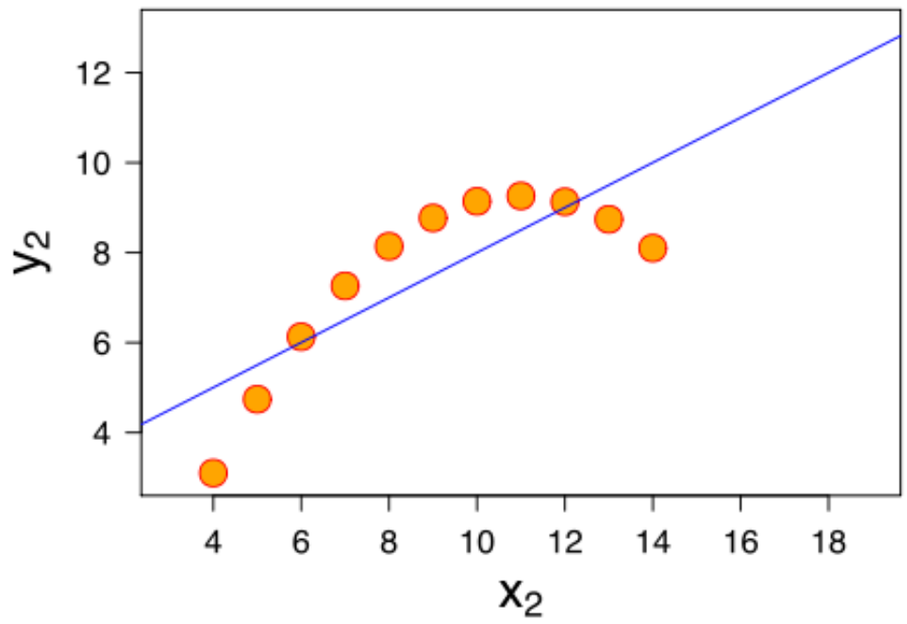


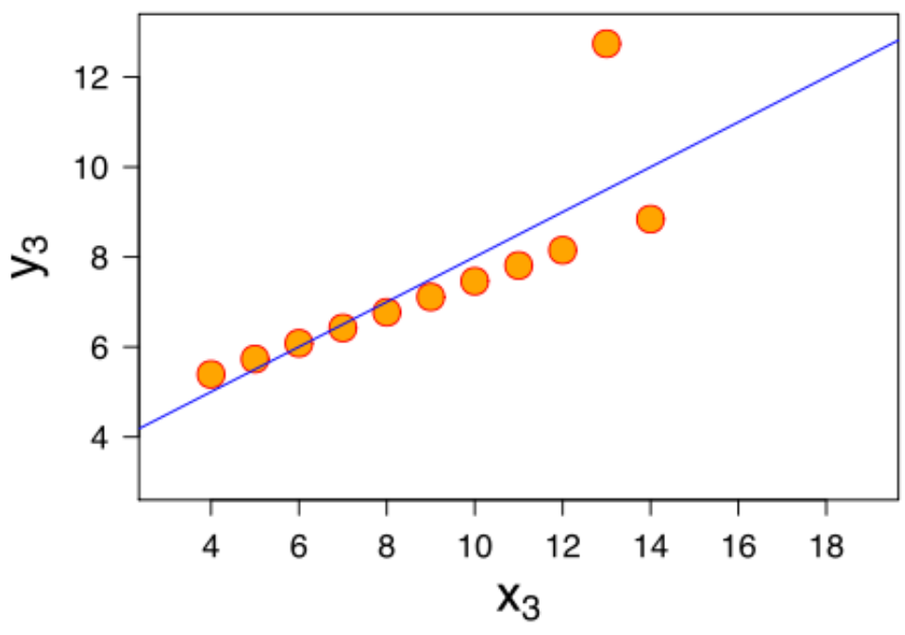
Bar graphs tend to be quite limited in terms of what they communicate. Here they communicate the means for levels of a factor and information about variance. But they don't tell us anything about the *distribution* of the data.

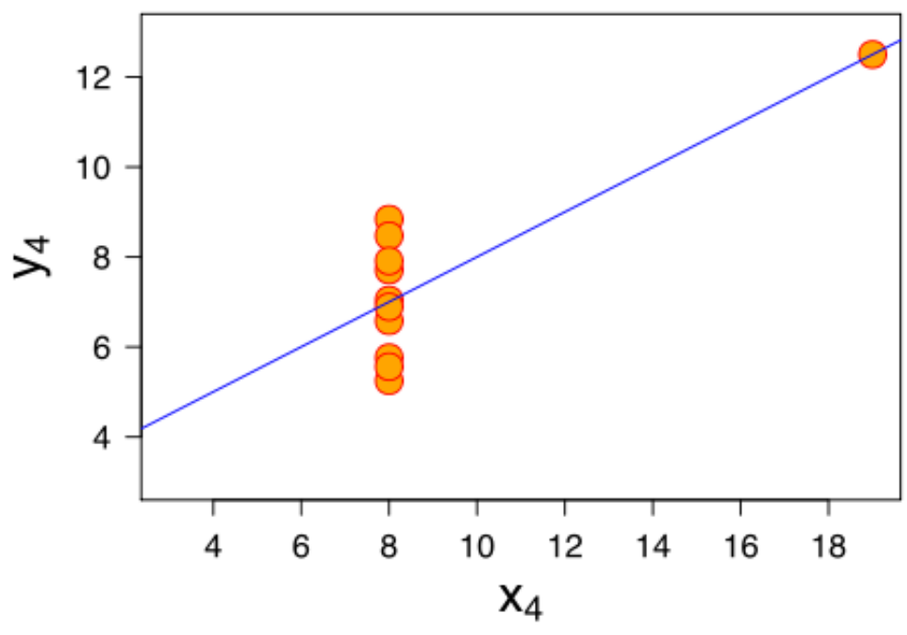
```
> data_summ <- data_long %>% group_by(Condition) %>% summarise(Mean = mean(RT), sd = sd(RT))
> ggplot (data_summ, aes (x=Condition, y=Mean, group=Condition, fill=Condition, ymin=Mean-sd,
ymax=Mean+sd)) + geom_bar(stat = "identity", width=.5) + geom_errorbar(width=.25) +
ggtitle("Bar chart with Error Bars") + guides(fill=FALSE)
```

Anscombe's Quartet







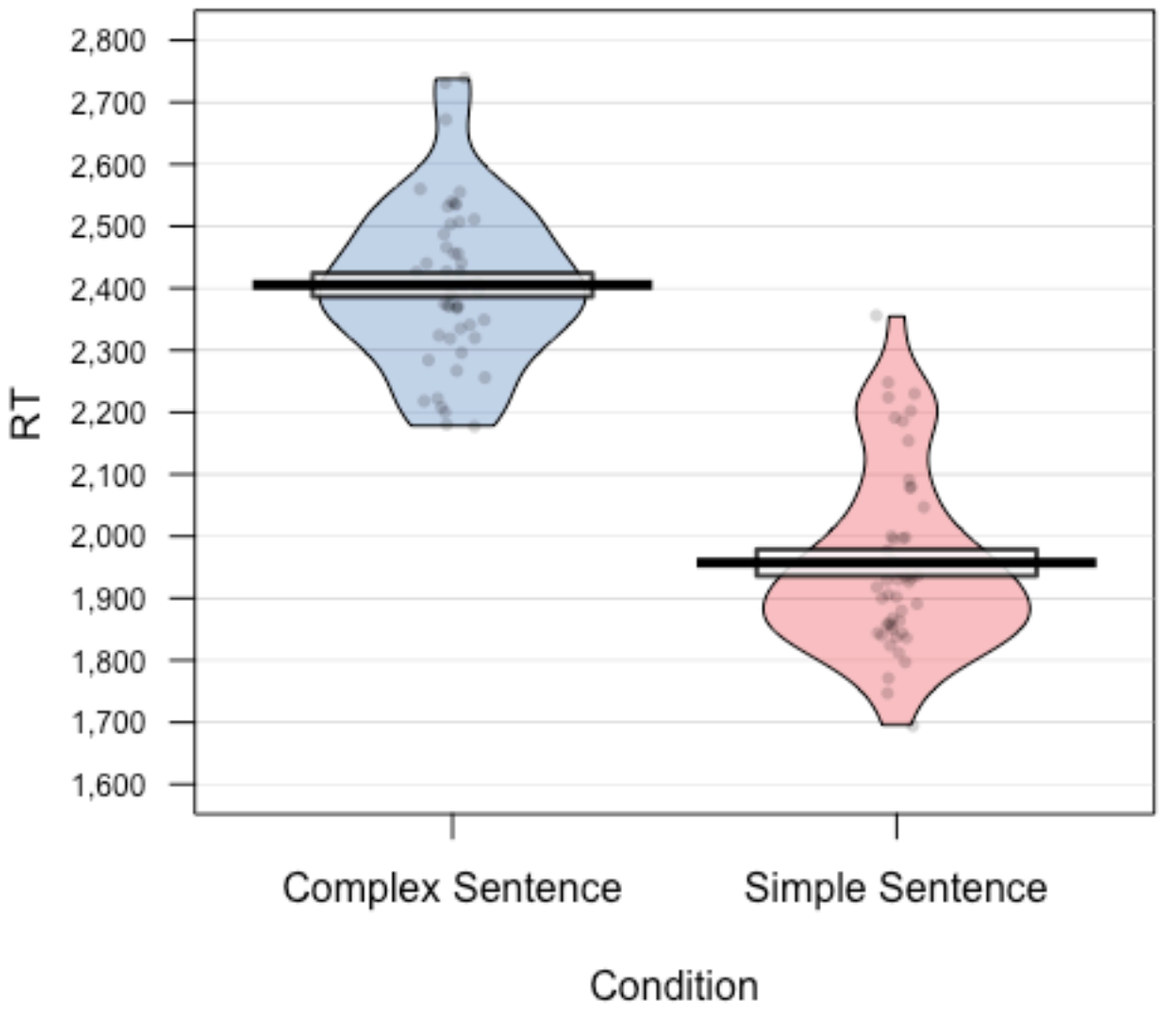


Violin Plots

Violin plots tell us about the distribution of the data. The width at any point corresponds to the *density* of the data at that value.

```
> ggplot (data_long, aes (x=Condition, y=RT, group=Condition, colour=Condition)) +
geom_violin() + geom_jitter(alpha=.5, position=position_jitter(0.05)) +
guides(colour=FALSE)
```

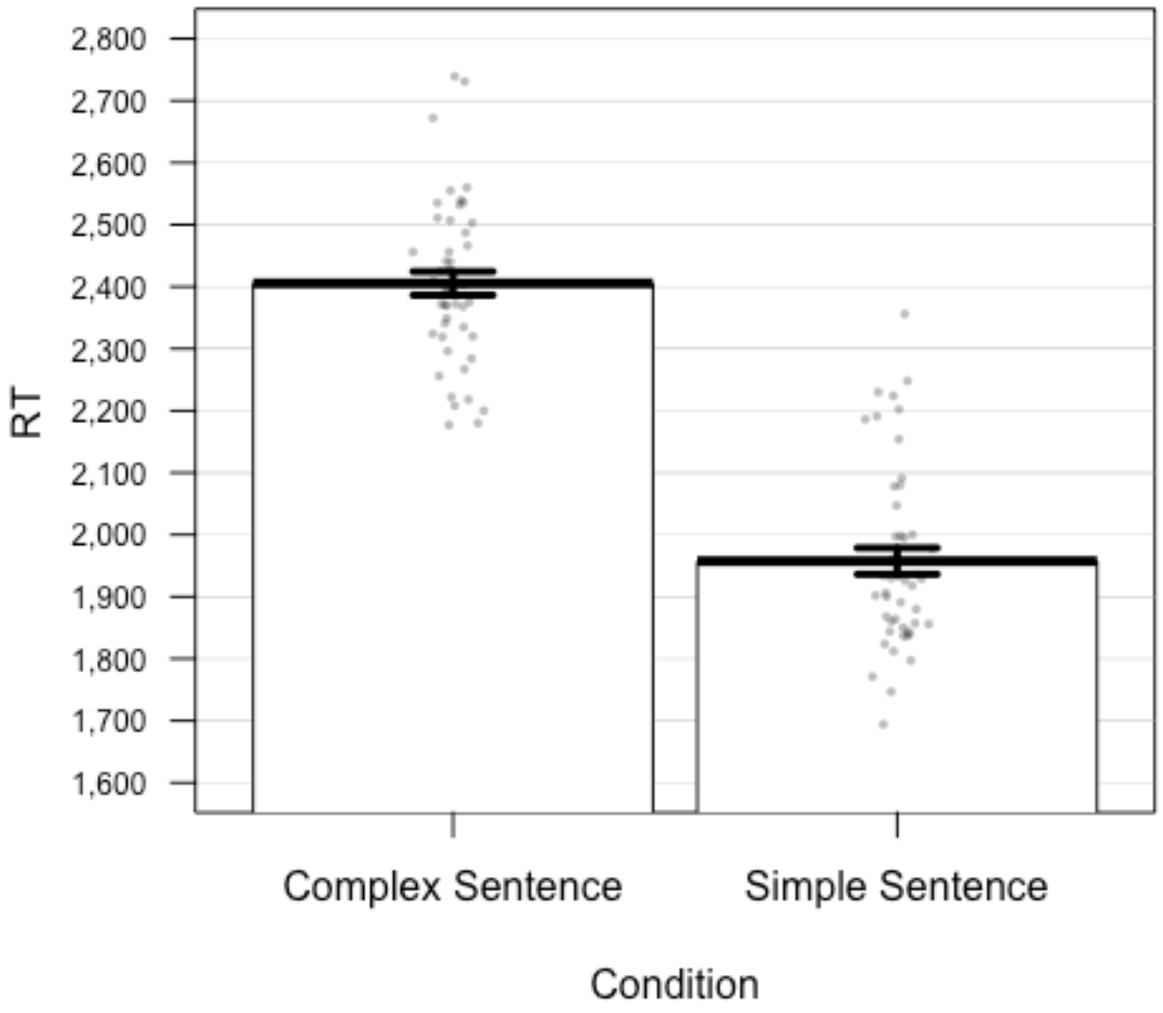
Pirate Plots



Pirate Plots are an example of RDI (Raw data, Descriptive and Inferential statistic) plots. Available in the package `yarrr`. Plots include shape of distribution, mean, and SE (all changeable as parameters).

```
> pirateplot (formula=RT~Condition, data=data_long,
inf.method="se", cex.axis=.75, theme=1)
```

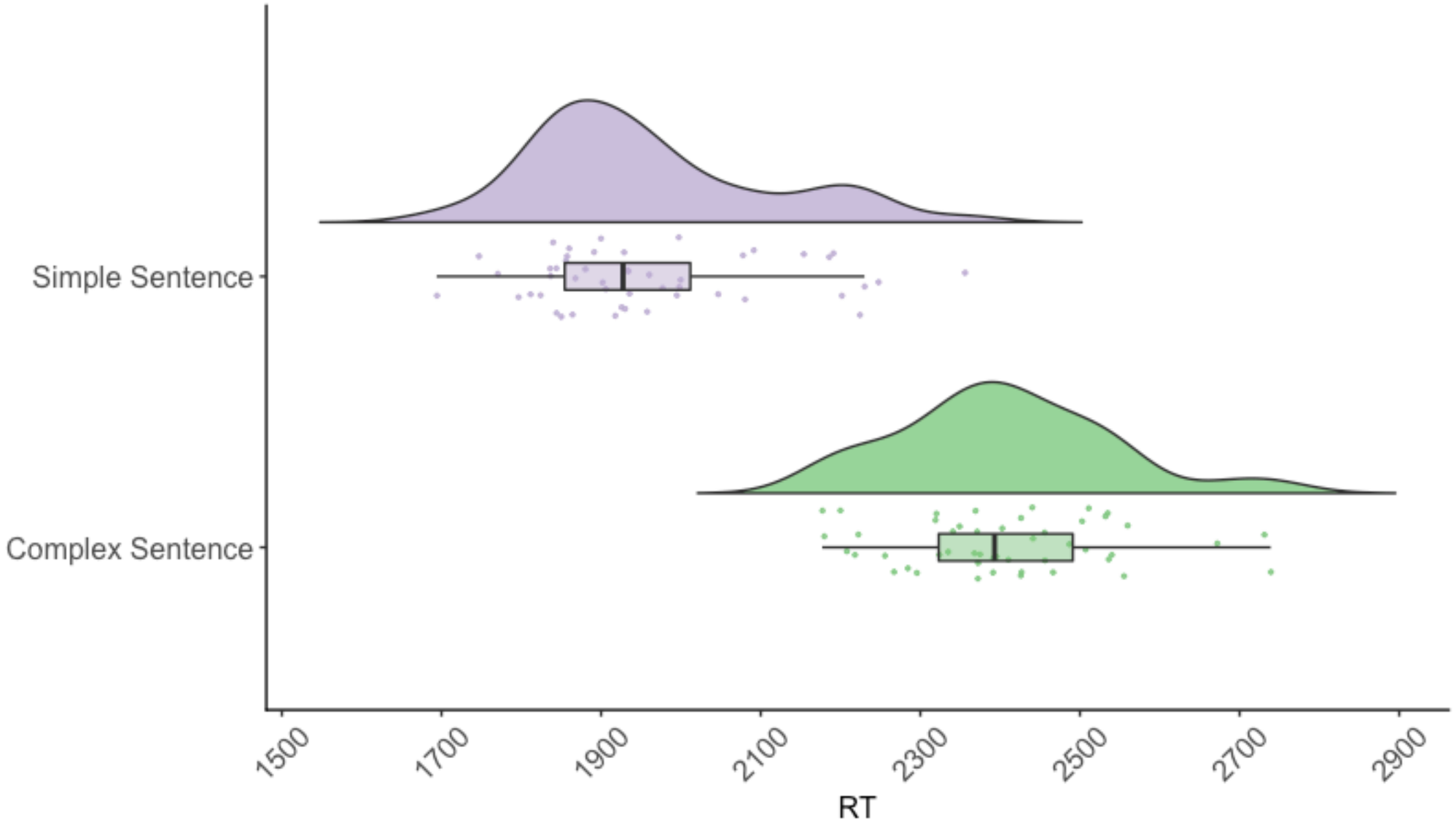
Combining Bar Graphs with Raw Data Plots



Bar graph plus the raw data.

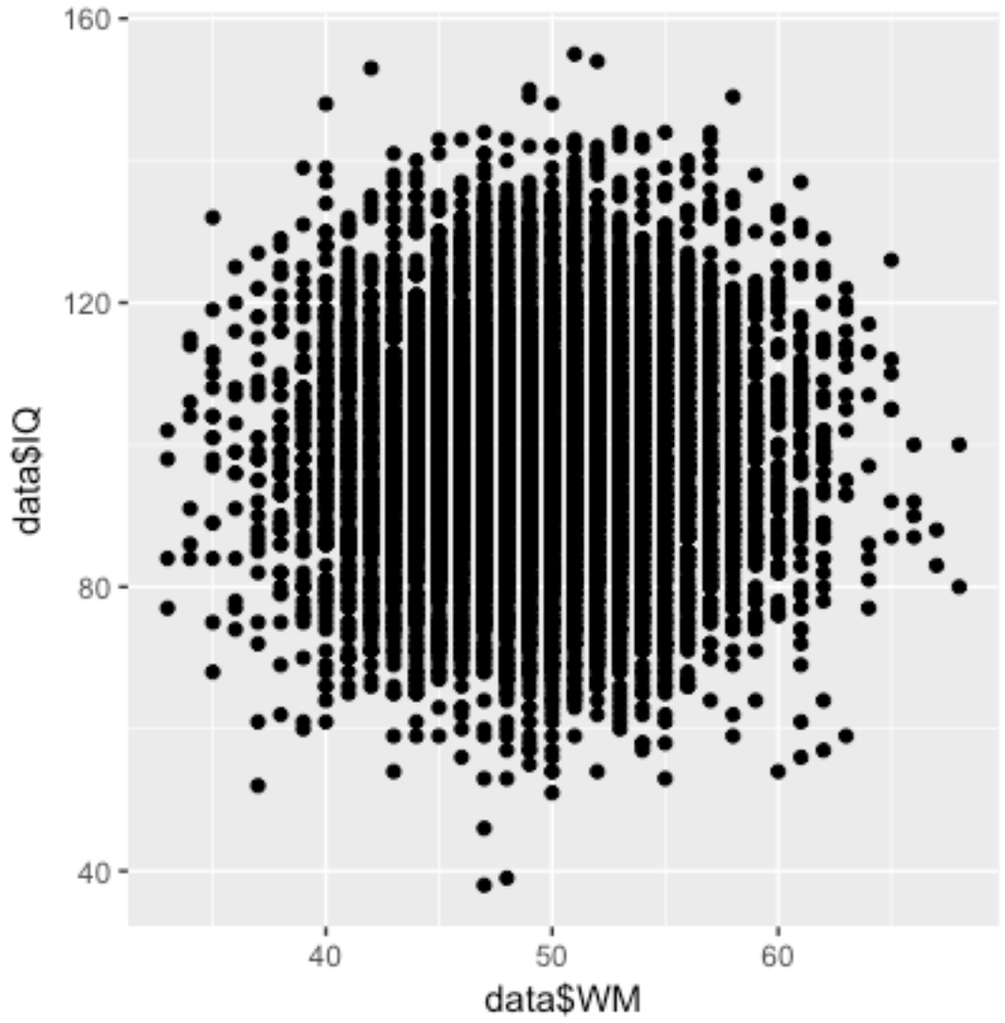
```
> pirateplot(formula=RT~Condition, data=data_long, inf.method="se",
cex.axis=.75, theme=4)
```

Raincloud Plots



Developed by Micah Allen (UCL), raincloud plots allow you to see the raw data, and the shape of the distribution alongside a box plot (capturing the median, 25th and 75th percentiles as hinges, and 1.5 * IQR from the hinges as the whisker length.)

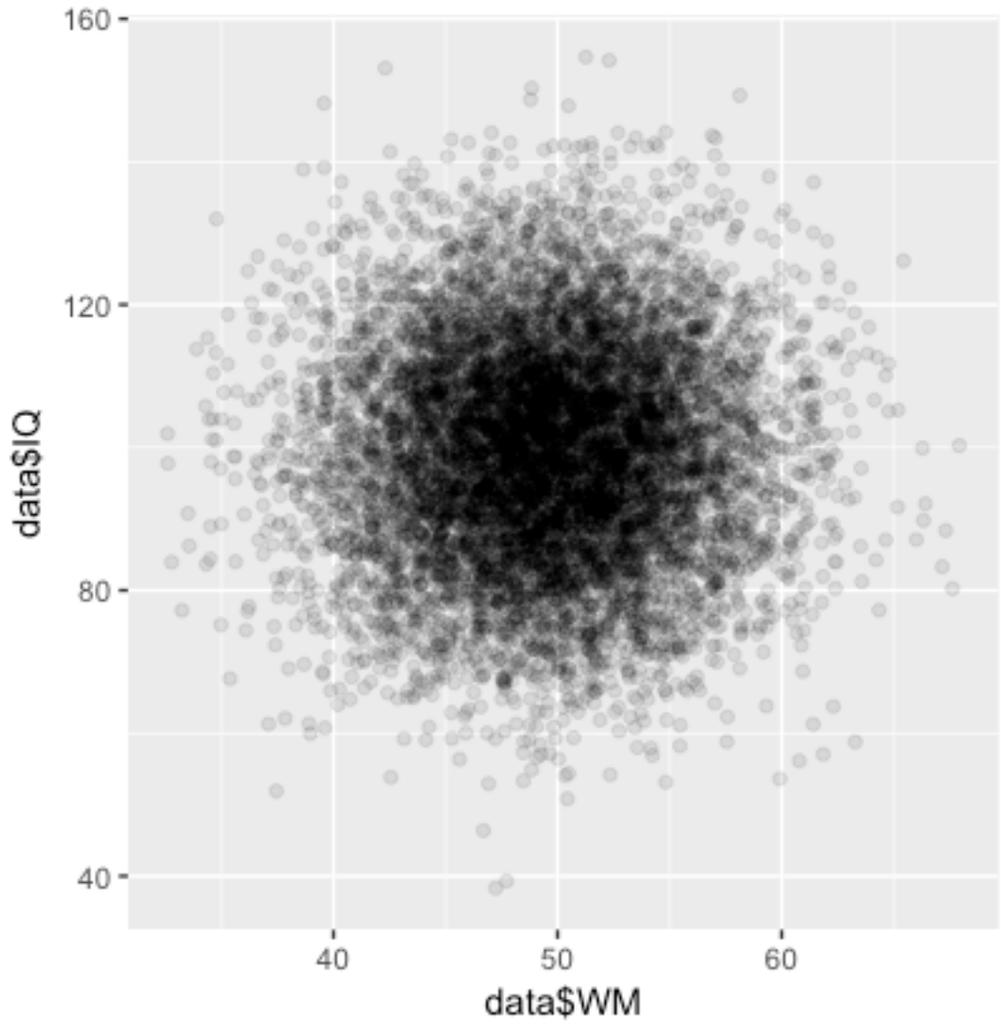
Plotting IQ against WM for our 10,000 participants



The problem of over-plotting - as we have many data points, a number are plotted on top of each other so it is tricky to get a feel for the data.

```
ggplot (data, aes (x=data$WM, y=data$IQ)) + geom_point()
```

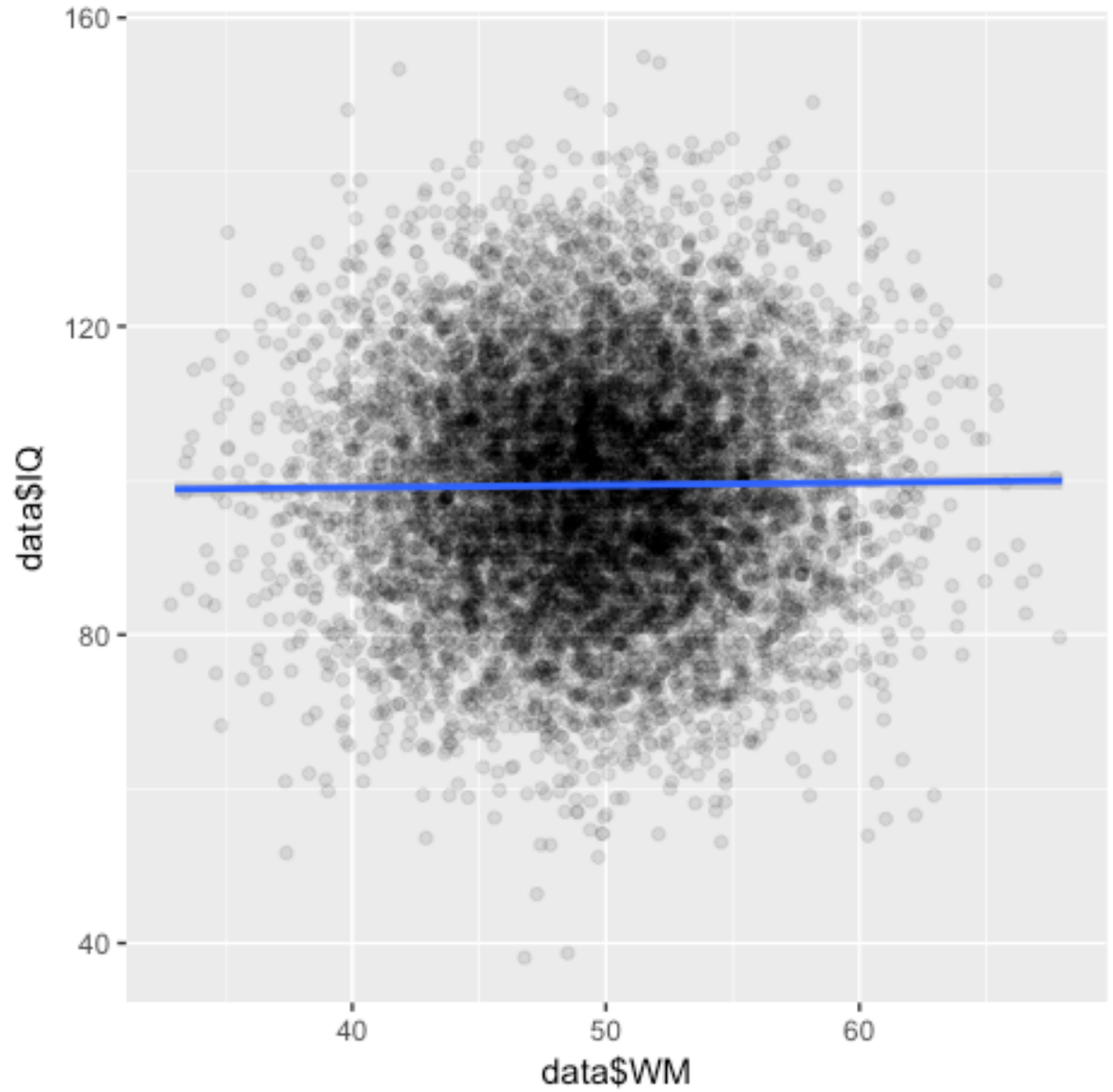
Plotting IQ against WM for our 10,000 participants



To avoid over-plotting, you can jitter the points and set them to be translucent via the alpha parameter.

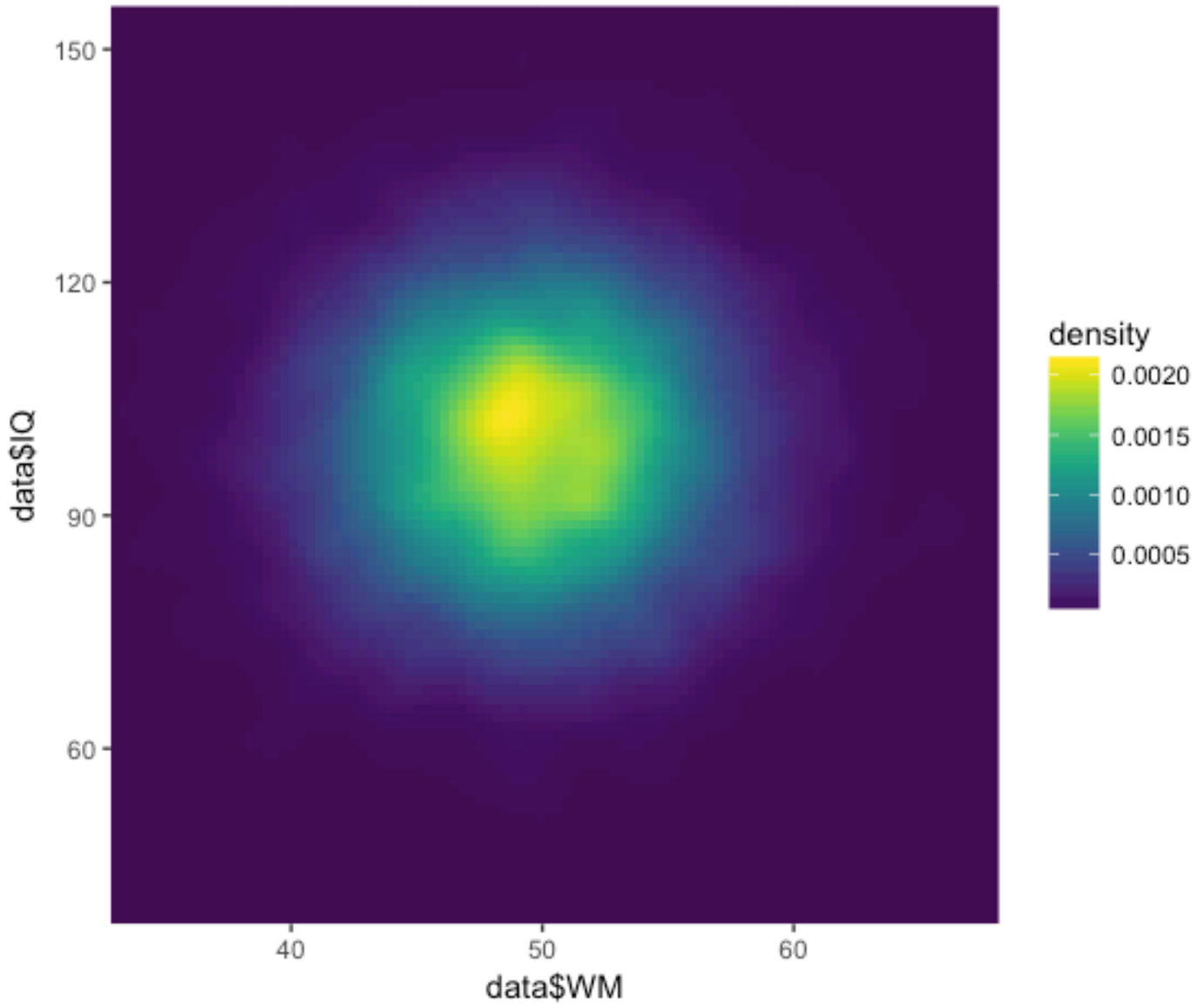
```
> ggplot (data, aes (x=data$WM, y=data$IQ)) + geom_jitter(alpha=.1,
position=position_jitter(0.5))
```

With a regression line




```
> ggplot (data, aes (x=data$WM, y=data$IQ)) + geom_jitter(alpha=.1,
position=position_jitter(0.5)) + geom_smooth(method="lm")
```

And as a Density Heat Map

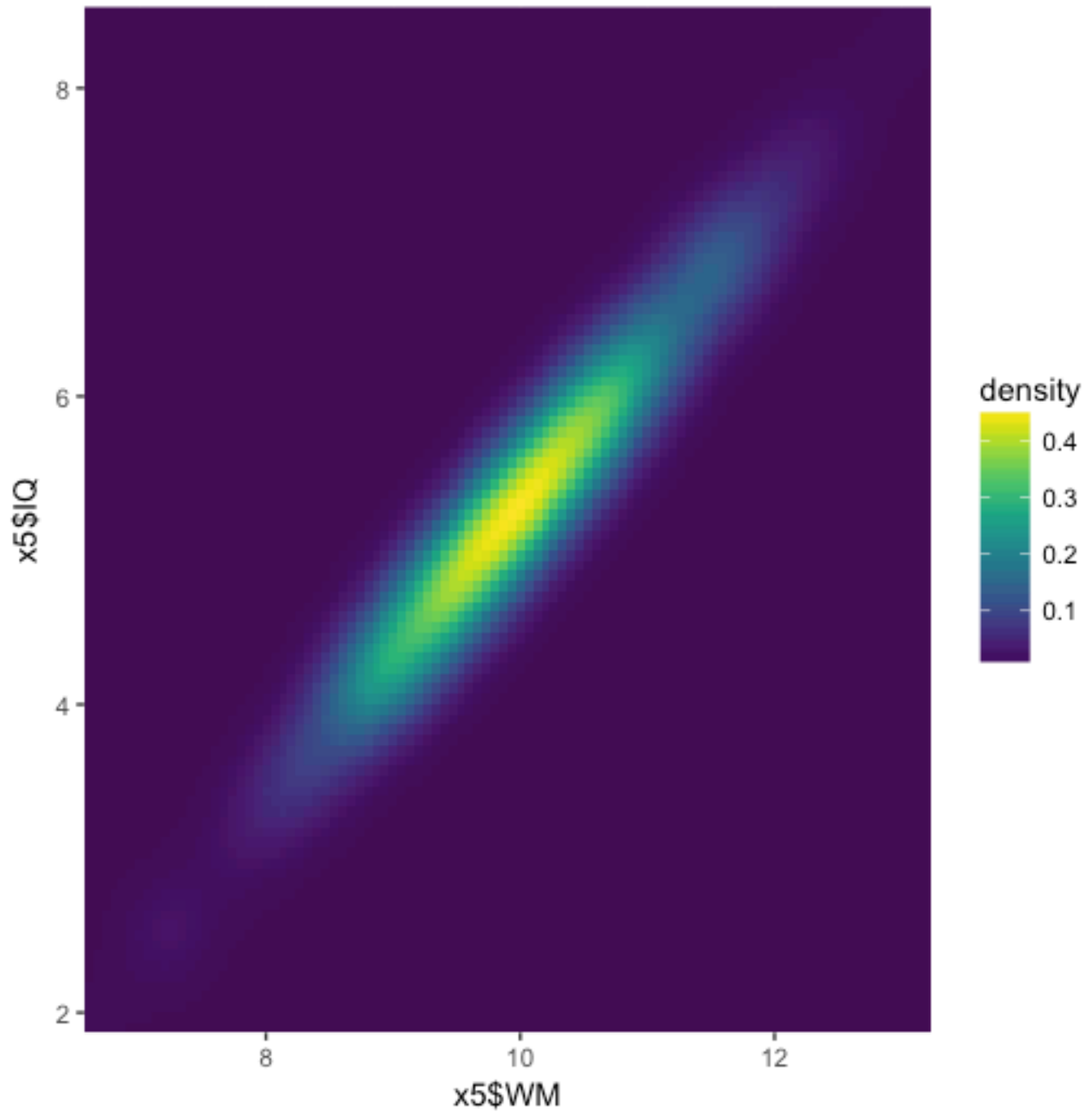



```
> ggplot (data, aes (x=data$WM, y=data$IQ)) + stat_density_2d(aes(fill
= ..density..), geom = 'raster', contour = FALSE) + scale_fill_viridis() +
coord_cartesian(expand = FALSE)
```

If IQ and WM were perfectly (positively) correlated, we'd have something like this…

```
> #creating two perfectly
correlated variables
> set.seed(1234)
> mysigma <- matrix (c(1,1,1,1),
2,2)
> x1 <- mvrnorm(n=1000,
c(5.3,10), mysigma)
> x5 <- as.data.frame (x1)
> colnames(x5) <- c("IQ", "WM")

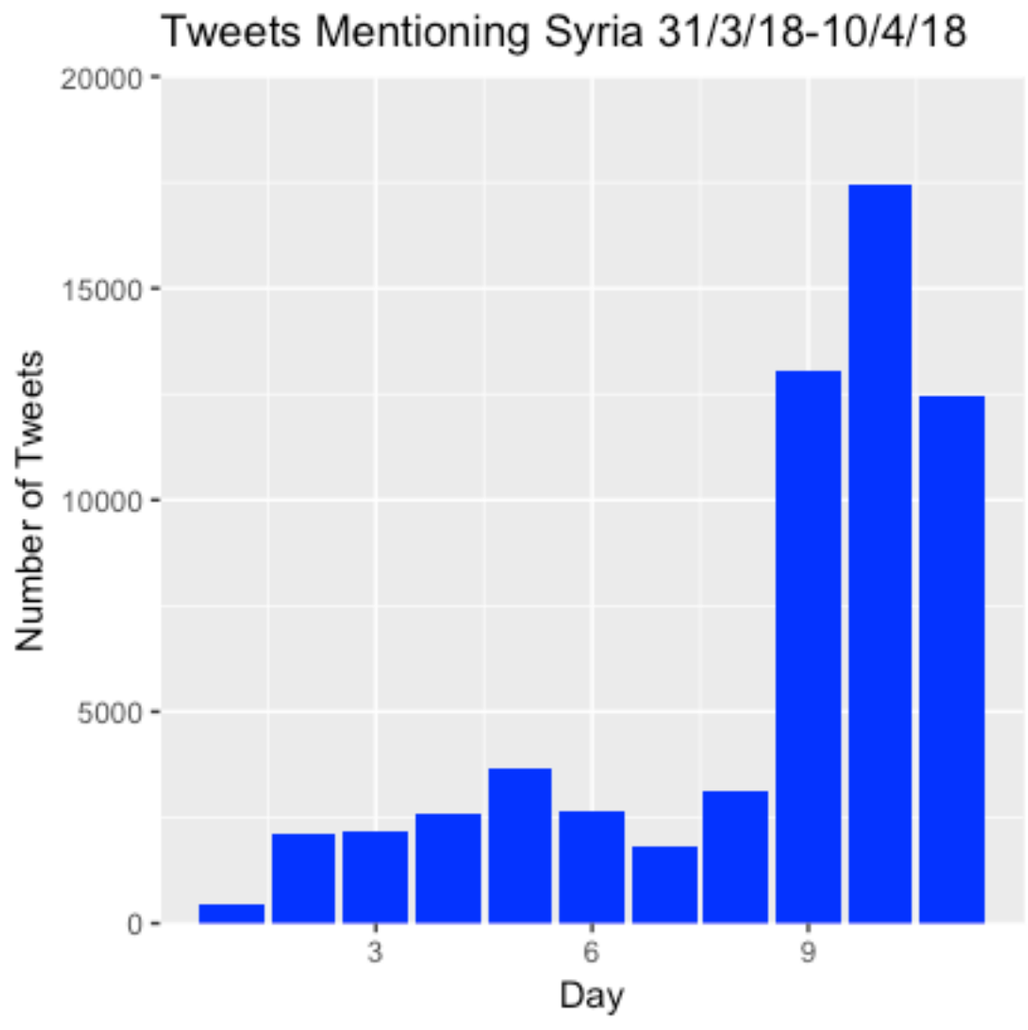
> ggplot (x5, aes (x=x5$WM,
y=x5$IQ)) +
stat_density_2d(aes(fill
= ..density..), geom = 'raster',
contour = FALSE) +
scale_fill_viridis()
+coord_cartesian(expand = FALSE)
```



Visualising ‘Big Data’

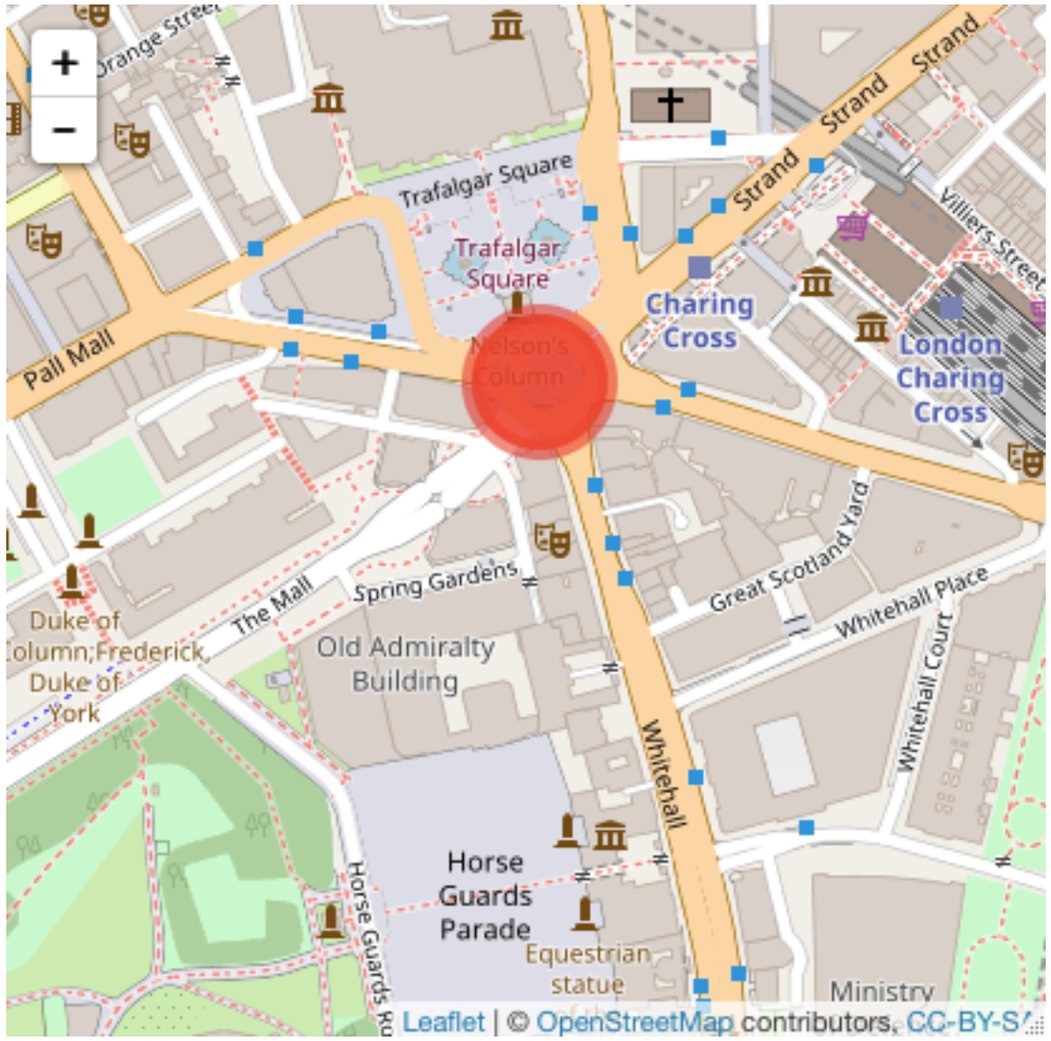
The R package, *rtweet*, can be used to scrape Twitter for mentions of particular topics. Connects via the Twitter Developer API.

In this case I scraped for up to 1,000,000 Tweets mentioning “Syria” around the time of the April chemical attack.

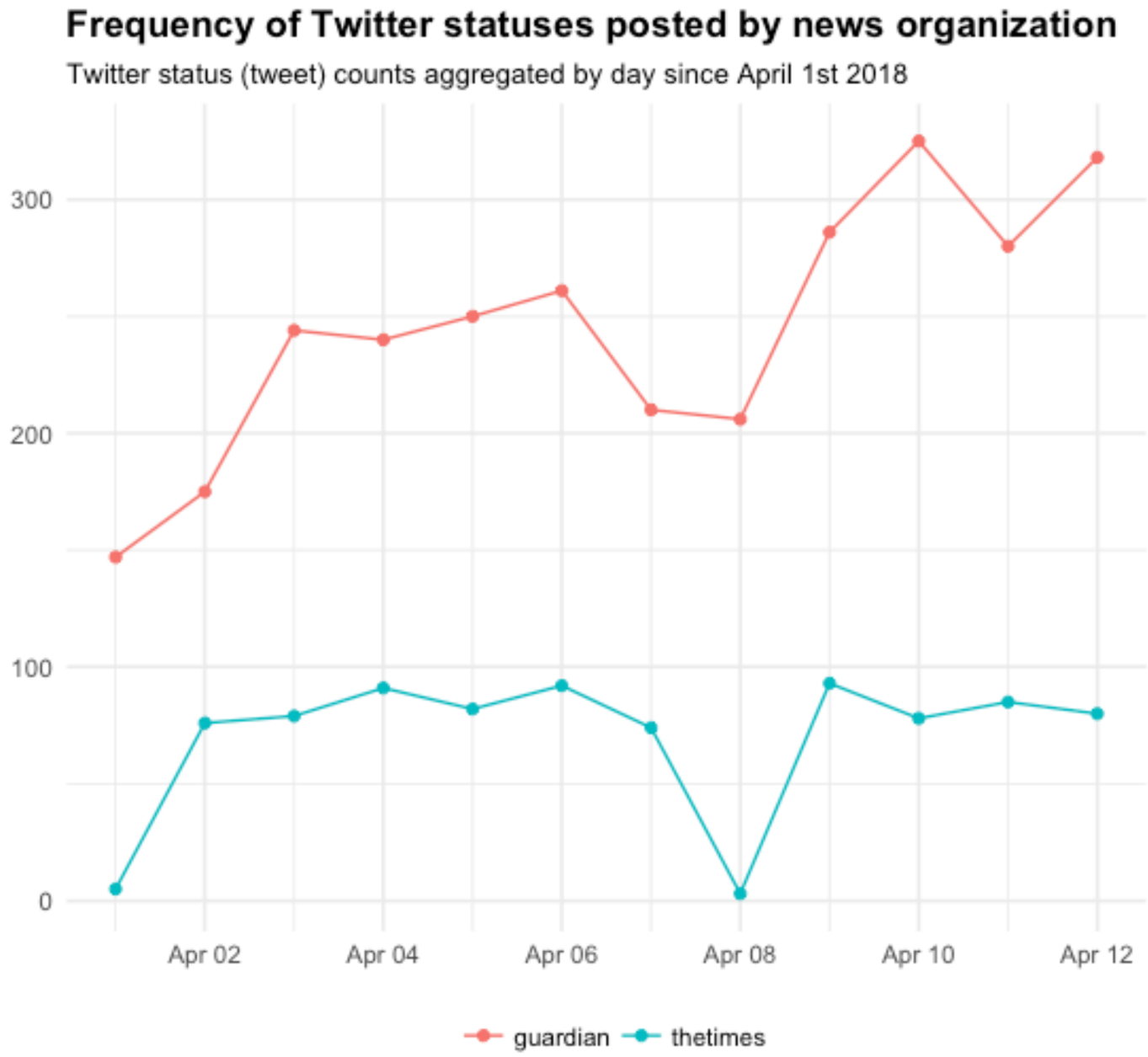


Geospatial mapping

If geospatial tagging has been activated by the user, Tweets will have geospatial co-ordinates attached to them. We can then use the *maps* and *leaflet* packages to plot the location of these Tweets.

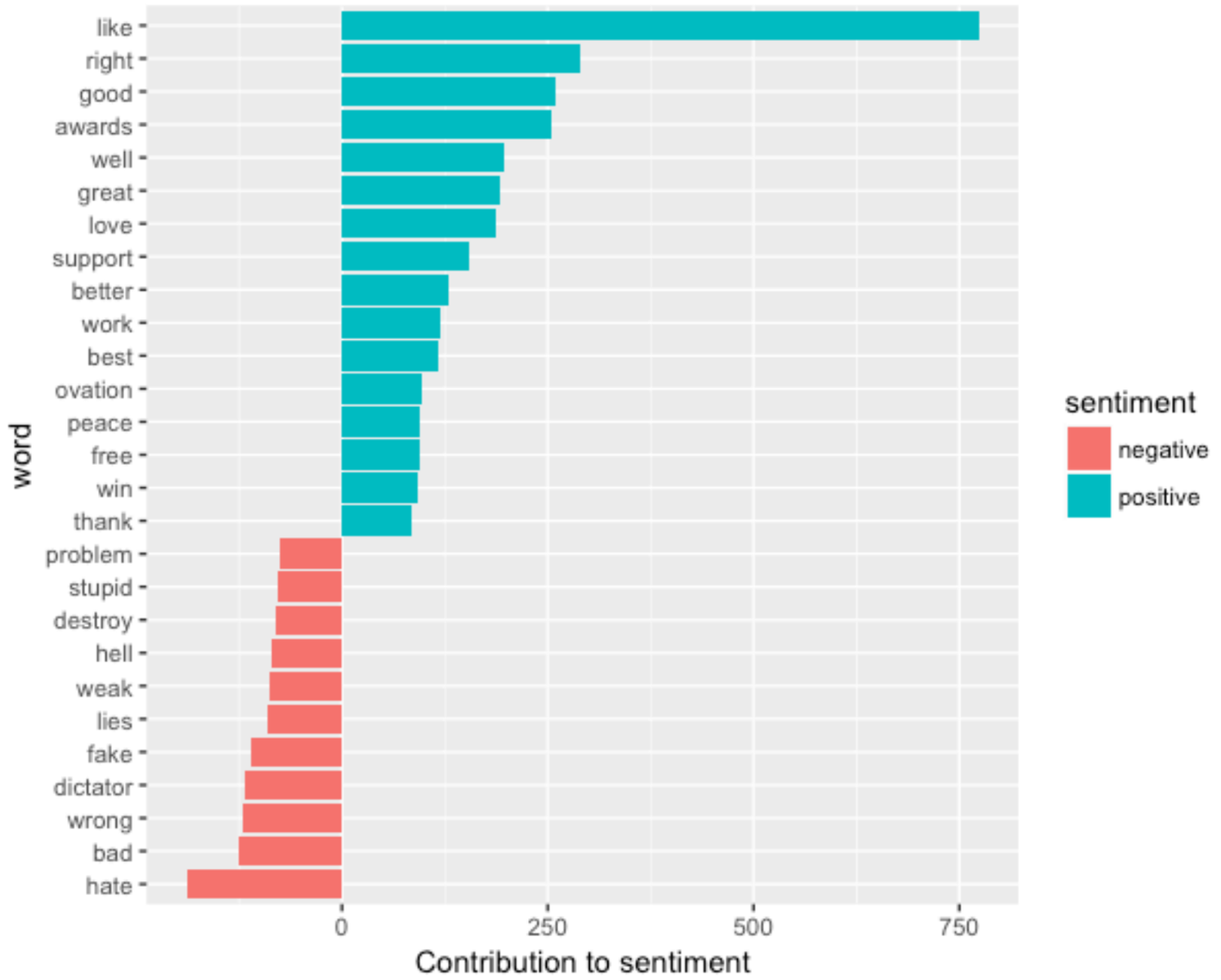


Can also focus on specific Twitter accounts and calculate (e.g.) Tweet frequency.



Tidytext for Sentiment Analysis

We can use the `tidytext` package to visualise the sentiment in Tweets - this visualisation is based on ~10,000 Tweets mentioning “Trump”…

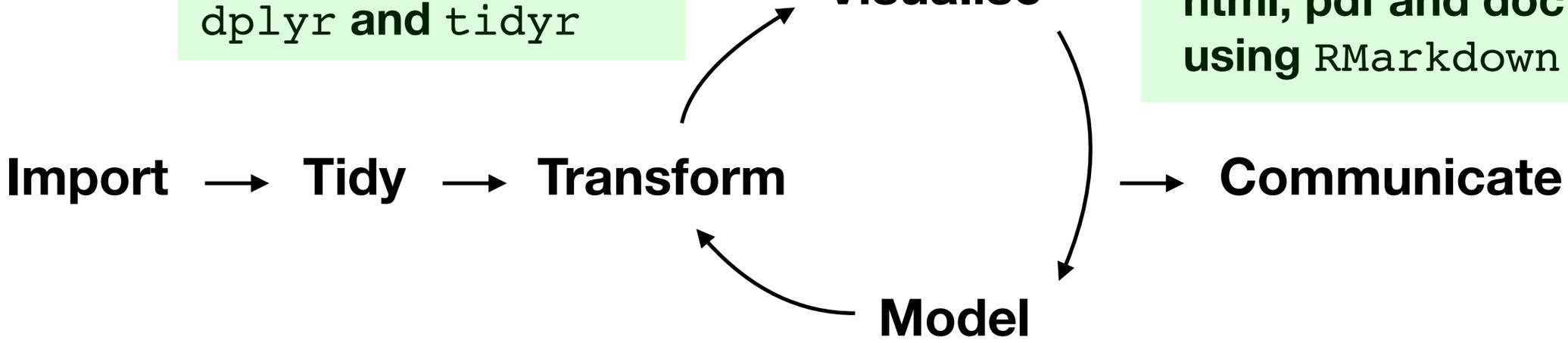


Workflow

Tidying and transforming using `dplyr` **and** `tidyr`

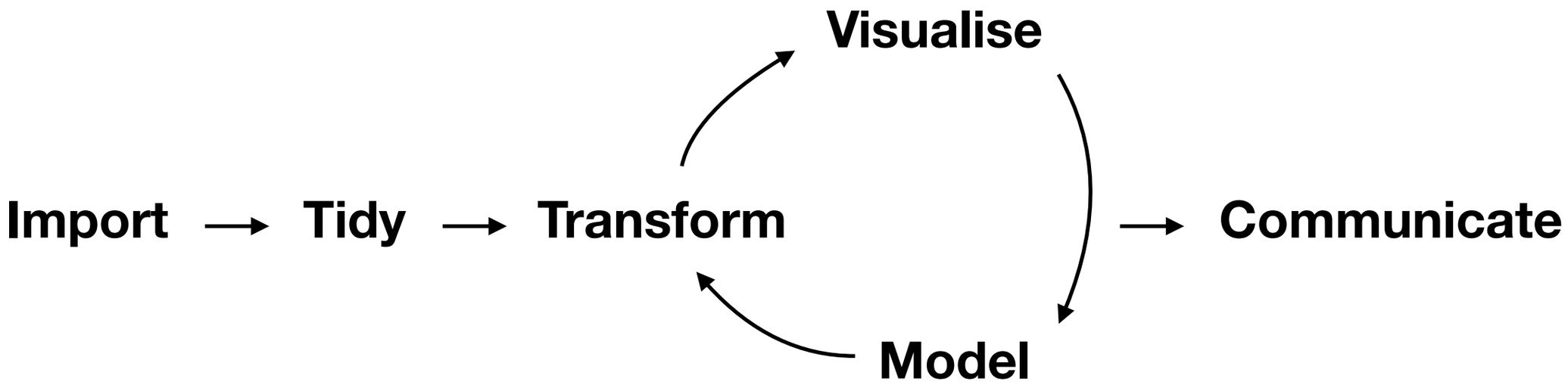
Data plots using `ggplot()` **and** `pirateplot()`

html, pdf and doc reports using `RMarkdown`



ANO(C)VA using `afex()` **and** `aov()`
Linear regression using `lm()` **and** `step()`
(Generalised) linear mixed models using `lmer()` **and** `glmer()`

Workflow



ANO(C)VA using `afex()` and `aov()`
Linear regression using `lm()` and `step()`
(Generalised) linear mixed models using `lmer()` and `glmer()`

Model

- Today we will focus on generating descriptive stats using the `psych` package, AN(C)OVA for different designs, and on simple linear and multiple regression (incl. model selection using AIC).
- Tomorrow we will focus on (Generalised) Linear Mixed Models. Over the last ~ 8 years they have transformed how we analyse experimental data.

You don't need to re-invent the wheel…

- For any problem you want to solve, chances are others have had the same problem, solved it and have created an R package to do exactly what you want…
- One of the many great things about R is that you can add freely available packages to your library.
- ~11,000 R packages currently available
- The sites `r-bloggers.com` and `rweekly.org` are good places to find out about new packages.

The “psych” Package

```
> install.packages (“psych”)

> library (psych)
```

You can read documentation about any package by typing :

```
> help (package name)
```

“psych” contains many helpful functions including *describeBy*.

To get help on any function, just type:

```
> help (function name)
```

This parameter specifies what data frame and variable we want descriptives for.

This parameter specifies how we want our descriptives grouped.

```
> describeBy (data_long$RT, group=data_long$Condition)

 Descriptive statistics by group
group: Complex Sentence
   vars  n    mean     sd median trimmed    mad  min  max range skew kurtosis    se
X1    1 48 2393.04 180.89 2368.5 2390.93 163.09 1926 2791   865 0.11     -0.3 26.11
------------------------------------------------------------------------------------
group: Simple Sentence
   vars  n   mean     sd median trimmed    mad  min  max range skew kurtosis    se
X1    1 48 1986.6 143.07   1992 1981.72 163.83 1731 2326   595 0.27    -0.64 20.65
```

If we had a 2 x 2 design with `Factor_1`, `Factor _2` and one DV in a data frame called `data`, to calculate the descriptives for each of our 4 conditions we would group like this:

```
> describeBy (data$DV, group=list(data$Factor_1, data$Factor_2))
```

You can type the name of the function itself (with no parameters), to see the code underlying it - great for transparency…

```
> describeBy
function (x, group = NULL, mat = FALSE, type = 3, digits = 15,
    ...)
{
    cl <- match.call()
    if (is.null(group)) {
        answer <- describe(x, type = type)
        warning("no grouping variable requested")
    }
    else {
        if (!is.data.frame(group) && !is.list(group) && (length(group) <
            NROW(x)))
            group <- x[, group]
        answer <- by(x, group, describe, type = type, ...)
        class(answer) <- c("psych", "describeBy")
    }
    if (mat) {
        ncol <- length(answer[[1]])
        n.var <- nrow(answer[[1]])
        n.col <- ncol(answer[[1]])
        n.grouping <- length(dim(answer))
        n.groups <- prod(dim(answer))
        names <- names(answer[[1]])
        row.names <- attr(answer[[1]], "row.names")
        dim.names <- attr(answer, "dimnames")
        mat.ans <- matrix(NaN, ncol = ncol, nrow = n.var * n.groups)
        labels.ans <- matrix(NaN, ncol = n.grouping + 1, nrow = n.var *
            n.groups)
        colnames(labels.ans) <- c("item", paste("group", 1:n.grouping,
            sep = ""))
        colnames(mat.ans) <- colnames(answer[[1]])
        rn <- 1:(n.var * n.groups)
        k <- 1
        labels.ans[, 1] <- seq(1, (n.var * n.groups))
        group.scale <- cumprod(c(1, dim(answer)))
        for (var in 1:(n.var * n.groups)) {
            for (group in 1:n.grouping) {
                groupi <- ((trunc((var - 1)/group.scale[group]))%%dim(answer)[group]) +
                  1
                labels.ans[var, group + 1] <- dim.names[[group]][[groupi]]
```

ANOVA

- Imagine we're interested in the impact of caffeine consumption on an individual's motor performance.
- It's a between-subjects design with 3 conditions:
  - low amount of caffeine (single espresso)
  - large amount of caffeine (double espresso)
  - placebo group (water)

Our data look like this:

We have 45 participants, a between participants condition with 3 levels (Water *vs.* Single Espresso *vs.* Double Espresso), and Ability as our DV measured on a scale of 1-10.

Filter

	Participant	Condition	Ability
1	1	Water	4.817174
2	2	Water	5.410972
3	3	Water	5.733776
4	4	Water	4.361721
5	5	Water	5.471650
6	6	Water	5.502422
7	7	Water	5.070104
8	8	Water	5.081347
9	9	Water	5.074219
10	10	Water	4.943985
11	11	Water	5.109123
12	12	Water	4.900645
13	13	Water	4.989498
14	14	Water	5.325784
15	15	Water	5.683798
16	16	Single Espresso	7.050372
17	17	Single Espresso	6.870046
18	18	Single Espresso	6.689962
19	19	Single Espresso	6.723273

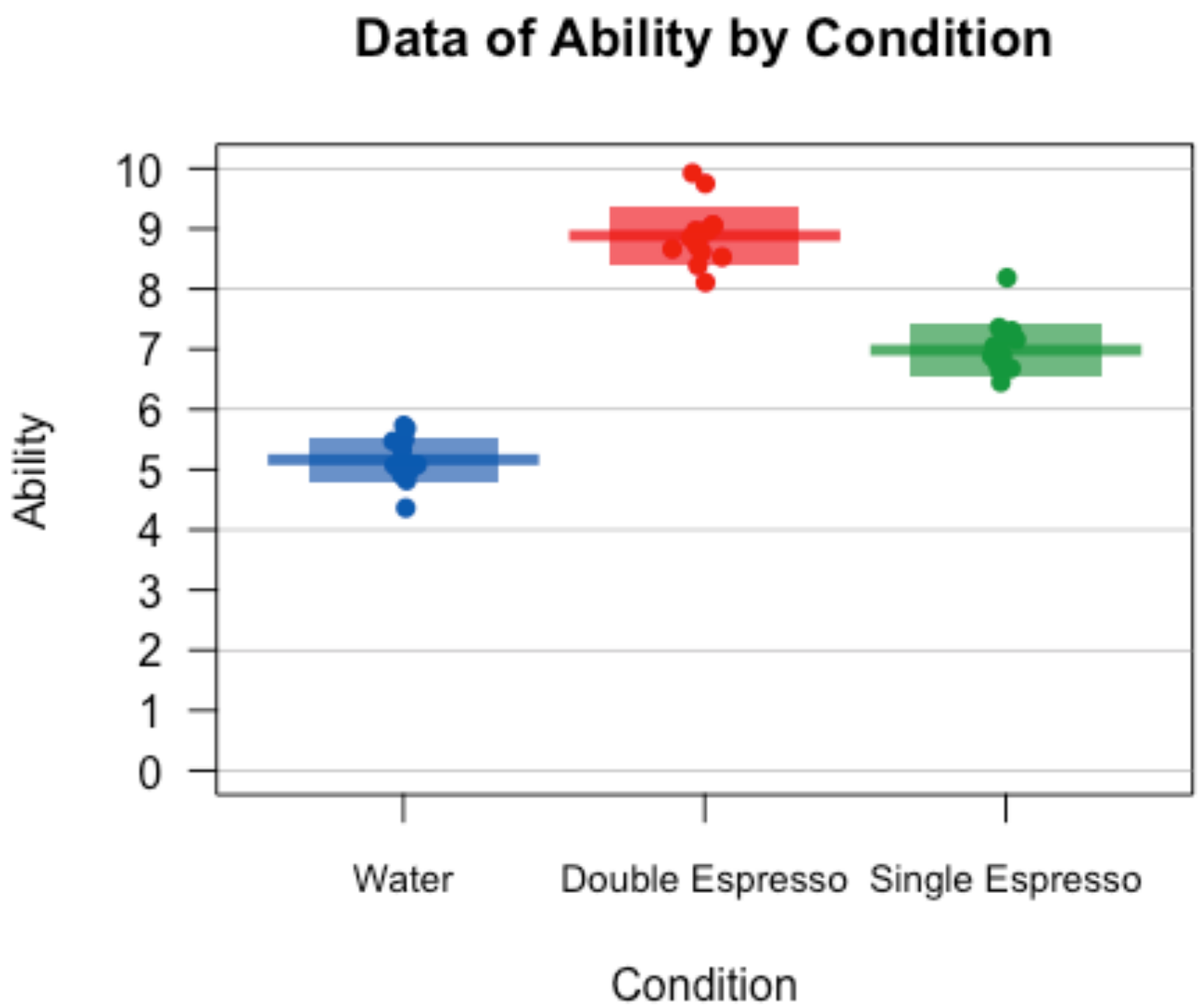
Showing 1 to 20 of 45 entries

First we need to load the packages we're going to use:

```
require (psych) # Gives us descriptive statistics
require (yarrr) # For building pirate plots
require (ggplot2) # For building ggplots
require (afex) # For Factorial ANOVA
require (DescTools) # For generating effect sizes
require (emmeans) # Allows us to run pairwise comparisons
```

If you haven't installed them previously, remember to type
`> install.packages ("packagename")`
first. Note, `require` basically does the same job as `library`.

Let's visualise the data first



Now some descriptives…

We're going to do this by using the *describeBy* function in the *Psych* package.

```
> describeBy (cond$Ability, group=cond$Condition)
```

```
> describeBy (cond$Ability, group=cond$Condition)

 Descriptive statistics by group
group: Water
   vars  n mean   sd median trimmed  mad  min  max range  skew kurtosis   se
X1    1 15 5.17 0.36   5.08    5.18 0.36 4.36 5.73  1.37 -0.27    -0.49 0.09
------------------------------------------------------------------------------
group: Double Espresso
   vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 15 8.89 0.47   8.85    8.87 0.31 8.11 9.92  1.81 0.72     0.05 0.12
------------------------------------------------------------------------------
group: Single Espresso
   vars  n mean   sd median trimmed mad  min  max range skew kurtosis   se
X1    1 15 6.99 0.42   6.88    6.93 0.3 6.45 8.19  1.74  1.4     1.83 0.11
```

Now let's run the 1-way ANOVA using the *aov* function (part of base R). We are going to assign it to a variable we are calling *model*.

```
> model <- aov(Ability ~ Condition, data=cond)
> anova(model)
Analysis of Variance Table

Response: Ability
           Df  Sum Sq Mean Sq F value    Pr(>F)
Condition   2 103.872  51.936  297.05 < 2.2e-16 ***
Residuals  42   7.343   0.175
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Here's the output we get – the F value is the ratio of systematic variance to unsystematic variation. It is the Mean SS of Condition divided by Mean Residual SS.

To get the Mean Square values we divide the Sum of Squares by the associated degrees of freedom (e.g., 7.343 / 42 = 0.175).

The ANOVA tells us we have an effect somewhere of Condition, but we don't yet know which level of this factor differs from which other level(s).

We need to conduct post hoc tests to figure this out. We can conduct both Bonferroni and Tukey pairwise comparisons using the *emmeans* function - Bonferroni is slightly more conservative than Tukey.

```
> emmeans (model, pairwise~Condition, adjust="Bonferroni")
$emmeans
 Condition         emmean        SE df lower.CL upper.CL
 Water           5.165081 0.1079627 42 4.947204 5.382959
 Double Espresso 8.886287 0.1079627 42 8.668409 9.104164
 Single Espresso 6.985001 0.1079627 42 6.767124 7.202879

Confidence level used: 0.95

$contrasts
 contrast                          estimate        SE df t.ratio p.value
 Water - Double Espresso          -3.721205 0.1526824 42 -24.372  <.0001
 Water - Single Espresso          -1.819920 0.1526824 42 -11.920  <.0001
 Double Espresso - Single Espresso  1.901285 0.1526824 42  12.453  <.0001

P value adjustment: bonferroni method for 3 tests
```

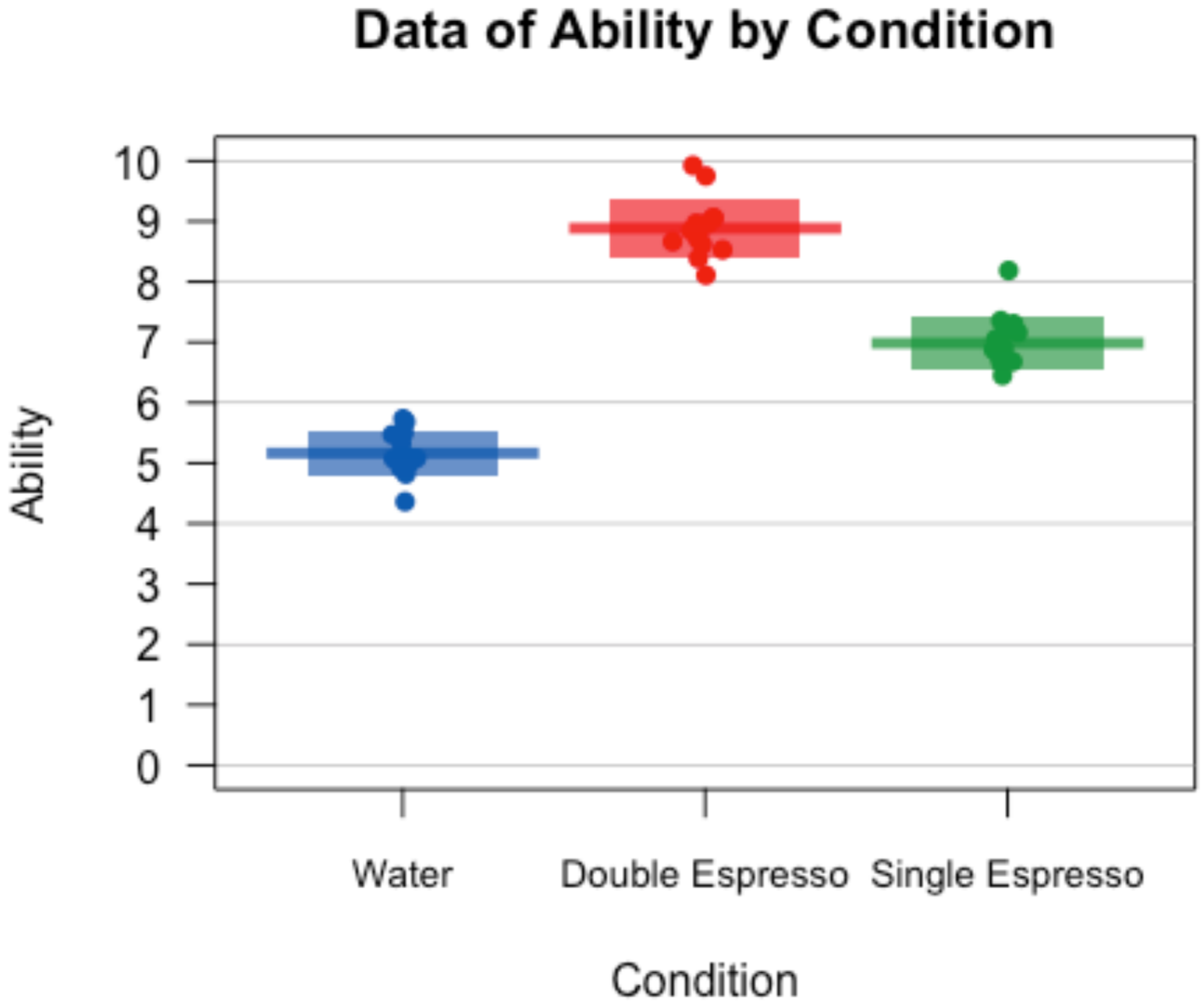
```
> emmeans (model, pairwise~Condition, adjust="Tukey")
$emmeans
 Condition         emmean        SE df lower.CL upper.CL
 Water           5.165081 0.1079627 42 4.947204 5.382959
 Double Espresso 8.886287 0.1079627 42 8.668409 9.104164
 Single Espresso 6.985001 0.1079627 42 6.767124 7.202879

Confidence level used: 0.95

$contrasts
 contrast                          estimate        SE df t.ratio p.value
 Water - Double Espresso           -3.721205 0.1526824 42 -24.372  <.0001
 Water - Single Espresso           -1.819920 0.1526824 42 -11.920  <.0001
 Double Espresso - Single Espresso  1.901285 0.1526824 42  12.453  <.0001

P value adjustment: tukey method for comparing a family of 3 estimates
```

We could set `adjust="none"` if we wanted uncorrected *p*-values. But in this case, both Bonferroni and Tukey comparisons tell us the same thing - each condition differs from each other condition (which fits with what we saw in the graph)…


Data of Ability by Condition
Ability
10
9
8
7
6
5
4
3
2
1
0
Water
Double Espresso
Single Espresso
Condition

Measure of Effect Size

- Effect size measures tell us how much variance can be explained by our experimental factors.
- partial η2 is a correlation between the dependent variable and different levels of a factor.
- For designs with more than one factor it can be a useful indicator of how much variance in the dependent variable can be explained by each factor (plus any interactions between factors).

```
> EtaSq(model, type=3, anova=TRUE)
              eta.sq eta.sq.part          SS df          MS         F  p
Condition 0.93397251   0.9339725 103.871817  2 51.9359084 297.0494  0
Residuals 0.06602749          NA   7.343252 42  0.1748393       NA NA
```

ANOVA for factorial designs

- A particularly good package for factorial ANOVA is by Henrik Singmann and called `afex`.
- Built to work like ANOVA in SPSS - uses Type III Sums of Squares with e*ffect* coding of contrasts. This overrides the default contrast coding in *R* which is for *dummy* coding.

Repeated measures example - 1 Factor, 4 levels

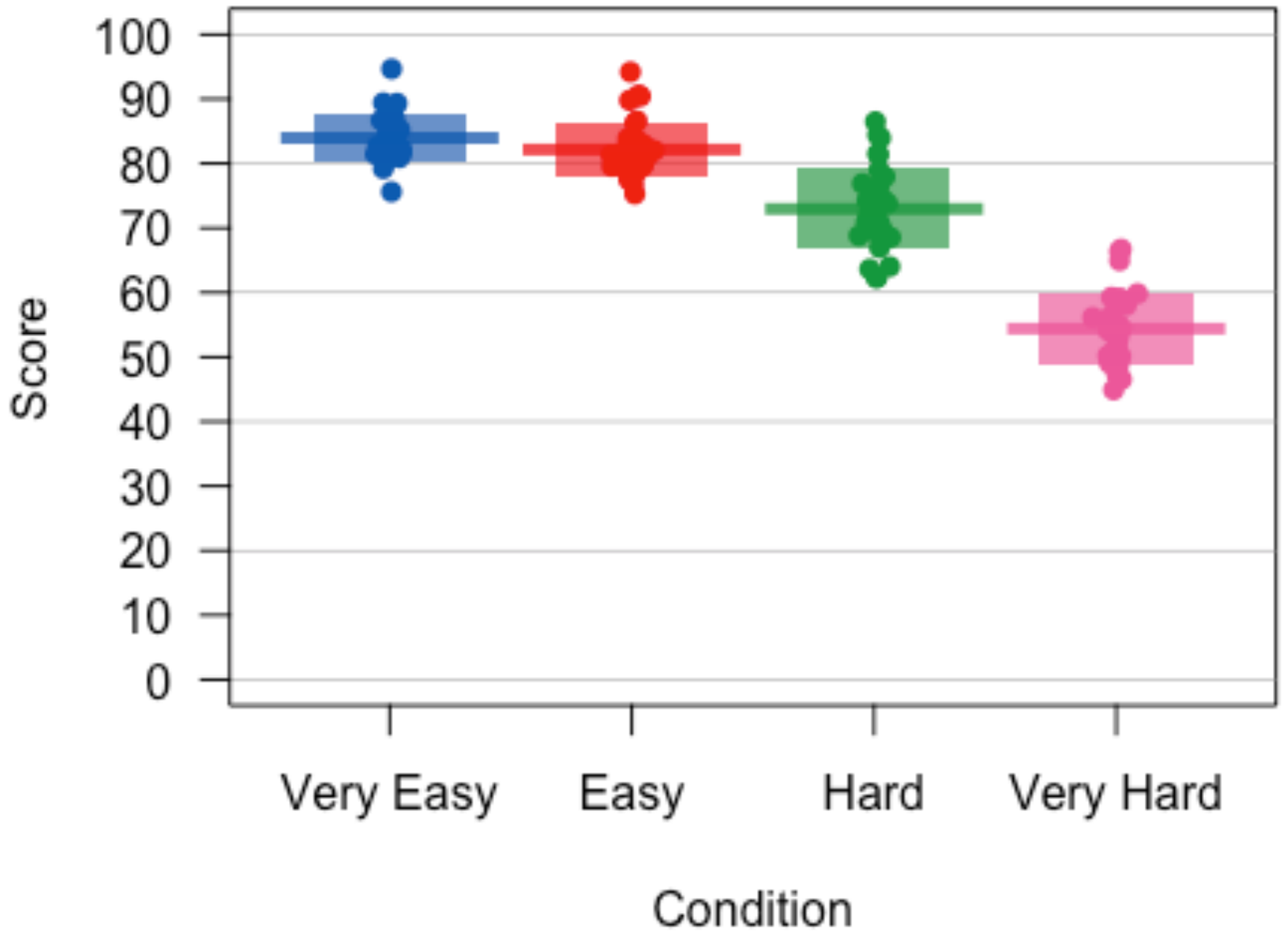
- Let's imagine we have an experiment where we asked 32 participants to memorise words of differing levels of spelling complexity - Very Easy, Easy, Hard, and Very Hard.
- They were presented with these words in an initial exposure phrase. After a 30 minute break we tested them by asking them to write down all the words. We scored them as number correct for each condition.
- We want to know whether there is a difference in the number of words they remembered for each level of spelling complexity.

	Participant	Condition	Score
1	1	Very Easy	80
2	2	Very Easy	86
3	3	Very Easy	89
4	4	Very Easy	75
5	5	Very Easy	86
6	6	Very Easy	87
7	7	Very Easy	82
8	8	Very Easy	82
9	9	Very Easy	82
10	10	Very Easy	81
11	11	Very Easy	83
12	12	Very Easy	81

Showing 1 to 13 of 128 entries

Our data are in long format with three columns - Participant, Condition, and Score.

Let's visualise the data first



We can use the *facet_wrap* function with *ggplot* to plot separate graphs for each participant on the same page:

```
> ggplot (data, aes (Condition, Score,
colour=Condition)) + ylim(0,100) + geom_point() +
facet_wrap(~data$Participant)
```

Now some descriptives…

We're going to do this by using the *describeBy* function in the *Psych* package.

```
> describeBy (data$Score, group=data$Condition)

 Descriptive statistics by group
group: Very Easy
   vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
X1    1 32 83.5 3.62     83   83.31 2.97  75  94    19 0.54     0.83 0.64
------------------------------------------------------------------------
group: Easy
   vars  n  mean   sd median trimmed  mad min max range skew kurtosis   se
X1    1 32 81.62 4.28     81   81.15 2.97  75  94    19 1.14     0.83 0.76
------------------------------------------------------------------------
group: Hard
   vars  n  mean   sd median trimmed  mad min max range skew kurtosis  se
X1    1 32 72.38 6.24     71   72.15 4.45  62  86    24 0.37    -0.56 1.1
------------------------------------------------------------------------
group: Very Hard
   vars  n  mean  sd median trimmed  mad min max range skew kurtosis   se
X1    1 32 53.97 5.5     54   53.62 5.93  44  66    22 0.42    -0.37 0.97
```

This is the our ANOVA model - we have a significant effect of Condition.

```
> model <- aov_4 (Score~Condition + (1+Condition|Participant), data=data)
> summary (model)

Univariate Type III Repeated-Measures ANOVA Assuming Sphericity

                SS num Df Error SS den Df        F    Pr(>F)
(Intercept) 679632      1   936.49     31 22497.36 < 2.2e-16 ***
Condition    17509      3  2179.48     93   249.04 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Mauchly Tests for Sphericity

          Test statistic p-value
Condition        0.90603 0.71042


Greenhouse-Geisser and Huynh-Feldt Corrections
 for Departure from Sphericity

          GG eps Pr(>F[GG])
Condition 0.9401  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

            HF eps   Pr(>F[HF])
Condition 1.043895 2.615157e-44
```

```
> anova (model)
Anova Table (Type 3 tests)

Response: Score
          num Df den Df    MSE      F     ges    Pr(>F)
Condition 2.8203  87.43 24.928 249.04 0.84892 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The effect size is measured by *ges* which standards for generalised effect size (η_G^2) - this is the recommended effect size measure for repeated measures designs (Bakeman, 2005). We get this by using the *anova* function on our model. Note the dfs in this output are always corrected as if there is a violation of sphericity - to be conservative (and to avoid Type I errors) we might be better off to always choose these corrected dfs.

```
> anova (model)
Anova Table (Type 3 tests)

Response: Score
          num Df den Df    MSE      F     ges    Pr(>F)
Condition 2.8203  87.43 24.928 249.04 0.84892 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

So we know we have an effect of Condition, but we don’t know where the difference lies…

Let’s do some post hoc tests with Bonferroni corrected *p*-values…

```
> emmeans (model, pairwise~Condition, adjust="Bonferroni")
$emmeans
 Condition   emmean        SE     df lower.CL upper.CL
 Very.Easy 83.50000 0.8861571 122.33 81.74581 85.25419
 Easy      81.62500 0.8861571 122.33 79.87081 83.37919
 Hard      72.37500 0.8861571 122.33 70.62081 74.12919
 Very.Hard 53.96875 0.8861571 122.33 52.21456 55.72294

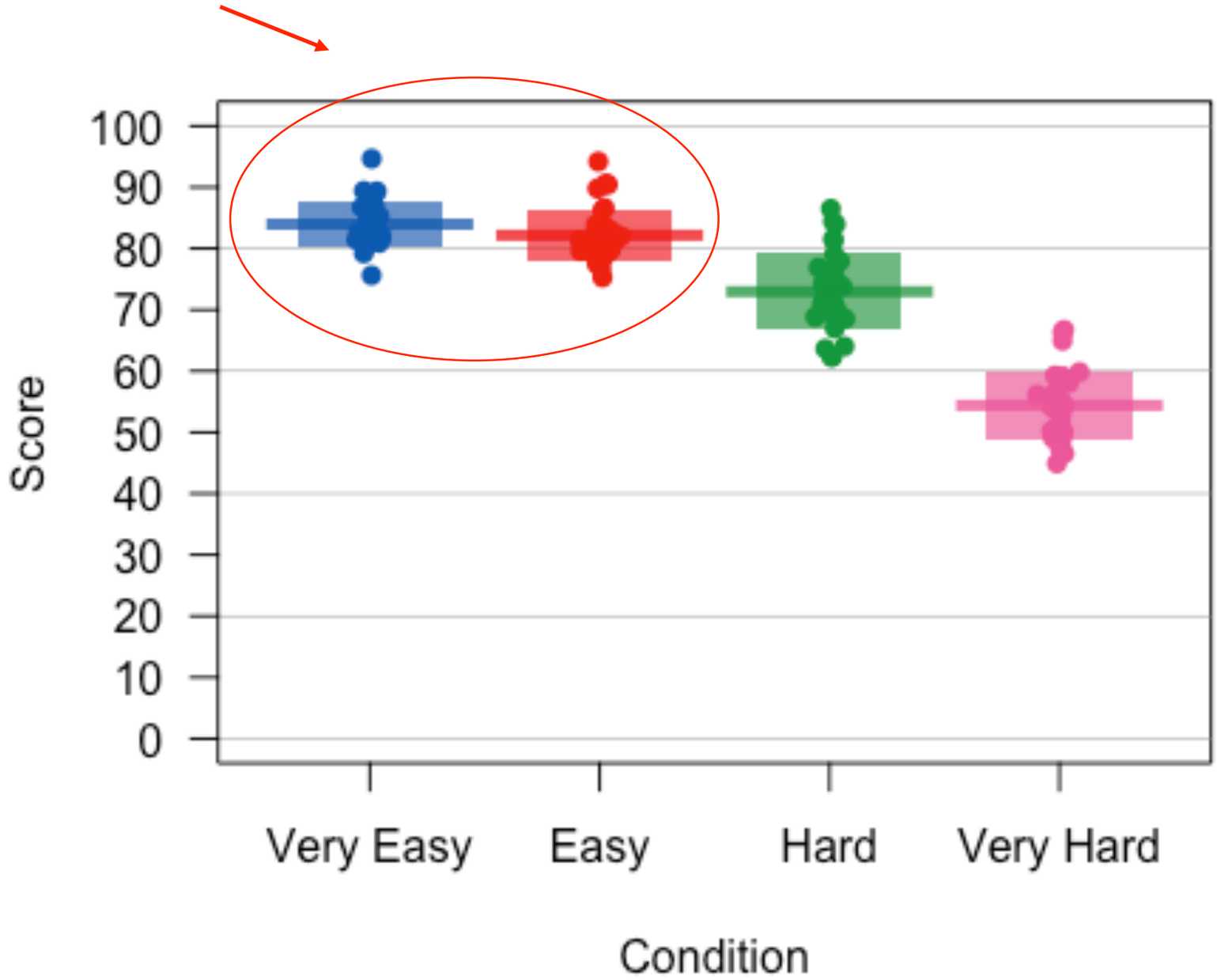
Confidence level used: 0.95

$contrasts
 contrast              estimate       SE df t.ratio p.value
 Very.Easy - Easy       1.87500 1.210249 93   1.549  0.7483
 Very.Easy - Hard      11.12500 1.210249 93   9.192  <.0001
 Very.Easy - Very.Hard 29.53125 1.210249 93  24.401  <.0001
 Easy - Hard            9.25000 1.210249 93   7.643  <.0001
 Easy - Very.Hard      27.65625 1.210249 93  22.852  <.0001
 Hard - Very.Hard      18.40625 1.210249 93  15.209  <.0001

P value adjustment: bonferroni method for 6 tests
```

- We see each level differs from each other, apart from Very Easy *vs*. Easy (where $p = .75$).

These two are equivalent, while other pairwise differences are significant.



2 x 2 Example

- A 2 x 2 repeated measures design with the factors Sentence Type (Positive vs. Negative) and Context (Positive vs. Negative). DV is reaction time (RT).

- When the data file is in *long* format (i.e., each row is one observation):

	Subject	Item	RT	Sentence	Context
1	1	3	1270	Positive	Negative
2	1	7	739	Positive	Negative
3	1	11	982	Positive	Negative
4	1	15	1291	Positive	Negative
5	1	19	1734	Positive	Negative
6	1	23	1757	Positive	Negative
7	1	27	1052	Positive	Negative
8	2	4	1706	Positive	Negative
9	2	8	533	Positive	Negative
10	2	12	1009	Positive	Negative
11	2	16	939	Positive	Negative
12	2	20	1848	Positive	Negative
13	2	24	1435	Positive	Negative

Showing 1 to 14 of 1,680 entries

```
1  install.packages("afex")
2  library (afex)
3
4  aov_4(RT ~ Sentence*Context + (1+Sentence*Context|Subject), data=DV, na.rm=TRUE)
```

- Syntax corresponds to RT being predicted by the two factors (Sentence*Context corresponds to two main effects plus the interaction) plus the random effect by Subjects using the datafile called DV. By setting na.rm to be TRUE, we are telling the analysis to ignore individual trials where there might be missing data - effectively this calculates the condition means over the data that is present (and just ignores trial where it is missing).

- aov_4 aggregates over the grouping term in the random effect. Simply change to (1+Sentence*Context| Item) for by-item (i.e., F2) analysis. This requires the data to contain the individual observations (not aggregated as means).

By Subjects

```
> model <- aov_4 (RT ~ Sentence*Context + (1+Sentence*Context |Subject),
  data=DV, na.rm=TRUE)

> anova (model)
Anova Table (Type 3 tests)

Response: RT
                num Df den Df    MSE      F        ges  Pr(>F)
Sentence             1     59 124547 0.6283 0.0016524 0.43114
Context              1     59  90195 3.1767 0.0060231 0.07984 .
Sentence:Context     1     59  93889 4.5967 0.0090449 0.03616 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- The output contains the main effect of Sentence, the main effect of Context, and the interaction between the two. Associated with each are the dfs, the Mean Squared Error, the F ratio, the generalized eta-squared, and p-value. Note, you can ask for partial eta-squared as effect size measure too.

By Items

```
> model1 <- aov_4 (RT ~ Sentence*Context + (1+Sentence*Context |Item),
data=DV, na.rm=TRUE)

> anova (model1)
Anova Table (Type 3 tests)

Response: RT
                  num Df den Df    MSE      F        ges  Pr(>F)
Sentence              1     27 203164 0.1221 0.0012553 0.72951
Context               1     27  39844 4.0013 0.0080150 0.05561 .
Sentence:Context      1     27  40168 5.7687 0.0116070 0.02346 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- With the same datafile and just by changing *one* word in the analysis code.

Interpreting Interactions

We can build the model as before and pass the model to the function *emmeans* (remember to load the *emmeans* package) and ask for pairwise comparisons with no correction - we need to work out the Bonferroni corrected value ourselves…

```
> emmeans (model, pairwise ~ Sentence*Context, adjust="none")
$emmeans
 Sentence Context    emmean       SE     df lower.CL upper.CL
 Positive Positive 1579.181 57.78624 137.64 1464.917 1693.445
 Negative Positive 1627.877 57.78624 137.64 1513.614 1742.141
 Positive Negative 1594.889 57.78624 137.64 1480.625 1709.152
 Negative Negative 1473.962 57.78624 137.64 1359.698 1588.225

Confidence level used: 0.95

$contrasts
 contrast                                    estimate       SE     df t.ratio p.value
 Positive,Positive - Negative,Positive -48.69643 60.33730 115.72  -0.807  0.4213
 Positive,Positive - Positive,Negative -15.70794 55.39009 117.95  -0.284  0.7772
 Positive,Positive - Negative,Negative 105.21905 59.82499 115.06   1.759  0.0813
 Negative,Positive - Positive,Negative  32.98849 59.82499 115.06   0.551  0.5824
 Negative,Positive - Negative,Negative 153.91548 55.39009 117.95   2.779  0.0064
 Positive,Negative - Negative,Negative 120.92698 60.33730 115.72   2.004  0.0474
```

ANCOVA

- Earlier we looked at how double espresso vs. single espresso vs. water drinking (our IV) might influence people's computer game ability (our DV).

- Imagine we sampled from a new group of participants - and we think other factors that we are not manipulating might also influence the DV – e.g., practice with computer games.

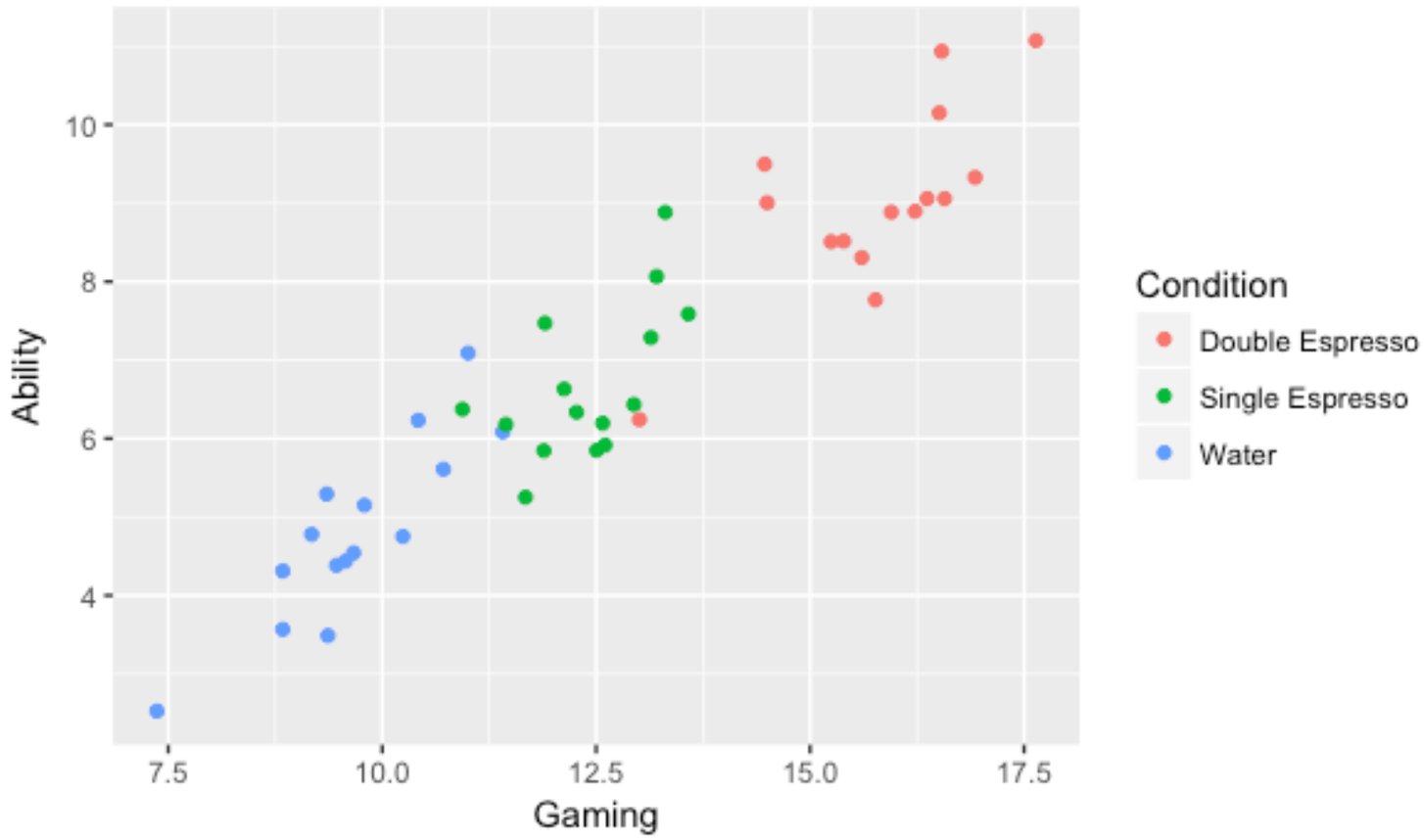
- What we want is to be able to see the effect on our DV of our IV after we have removed the effects of other things (computer gaming frequency in this case).

- Now, imagine we have a measure of computer games frequency - perhaps hours per week people play computer games…

- So, in addition to manipulating the type of beverage we're giving people (i.e., double espresso vs. single espresso vs. water) we also measure how often they play computer games…

- Let's do a plot first with our DV (Ability) on the y-axis, and our covariate (Gaming Frequency) on the x-axis…

```
> ggplot (cond, aes (x=Gaming, y=Ability,  colour=Condition)) + geom_point ()
```

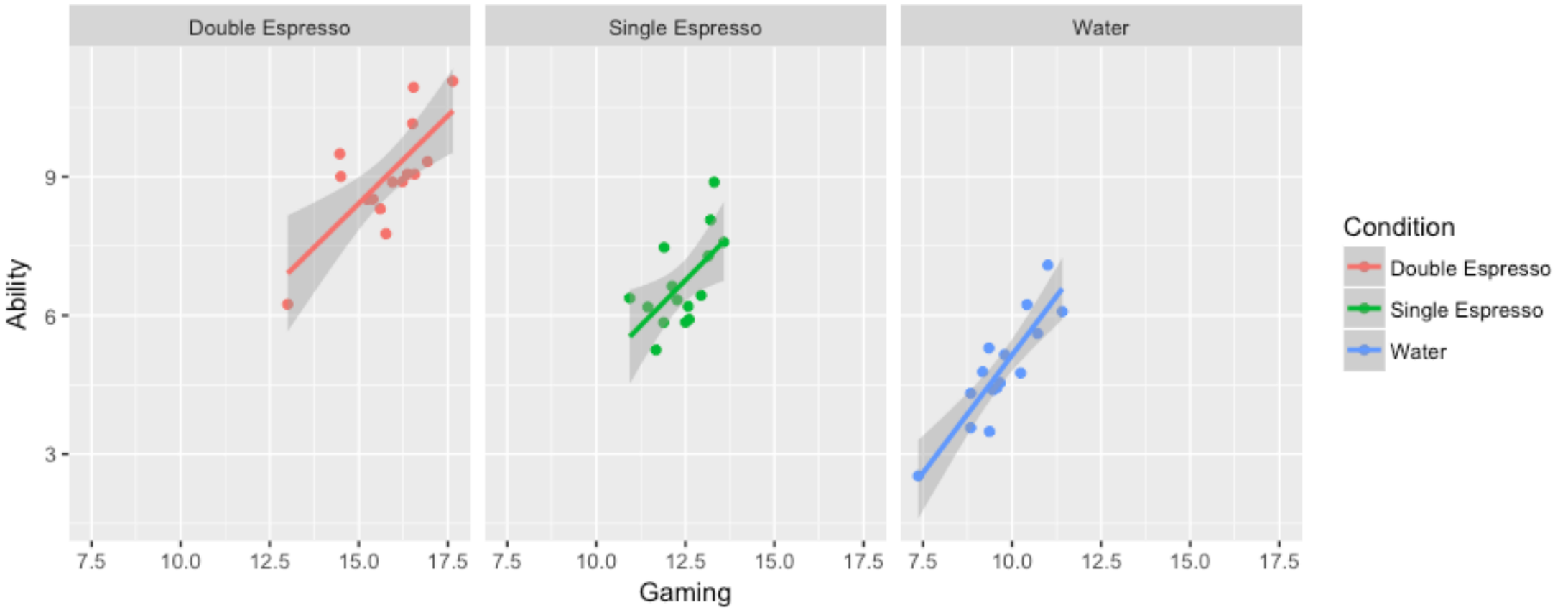


- So we can see there's a relationship between our DV (Ability) and our covariate (Gaming Frequency)…

- We can also see our Gaming Ability groups appear to be clustering in our data by Condition…

We can look at the data separately by condition using the *facet_wrap()* function:

```
> ggplot (cond, aes (x=Gaming, y=Ability,  colour=Condition)) +
geom_point () + facet_wrap('Condition') +
geom_smooth(method='lm')
```



Running a 1-way between participants ANOVA (and ignoring the covariate)…

```
> model <- aov(Ability ~ Condition, data=cond)
> anova(model)
Analysis of Variance Table

Response: Ability
           Df  Sum Sq Mean Sq F value    Pr(>F)
Condition   2 132.746  66.373  53.432 2.882e-12 ***
Residuals 42  52.172   1.242
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor Condition is significant with an F = 53.432. We would erroneously conclude that our manipulation has had an effect…

But now let's control for the effect of our co-variate by adding it before our experimental manipulation…

```
> model_ancova <- aov(Ability ~ Gaming + Condition, data=cond)
> anova(model_ancova)
Analysis of Variance Table

Response: Ability
           Df  Sum Sq Mean Sq  F value Pr(>F)
Gaming      1 161.329 161.329 292.4162 <2e-16 ***
Condition   2   0.968   0.484   0.8771 0.4236
Residuals 41  22.620   0.552
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor Condition is now not significant with an F < 1. However, our covariate *Gaming Frequency* is significant. Adding it means a lot of the variance we previously attributed to our experimental factor is actually explained by our covariate (and our residual error goes down too)…

Rather than calculating over the raw means which are:

Water Group = 4.82

Double Espresso Group = 9.02

Single Espresso Group = 6.69

```
> describeBy (cond$Ability, group=cond$Condition)
 Descriptive statistics by group
group: Double Espresso
   vars  n mean   sd median trimmed  mad  min   max range  skew kurtosis   se
X1    1 15 9.02 1.19   9.01    9.07 0.73 6.24 11.07  4.83 -0.26     0.16 0.31
------------------------------------------------------------------------------
group: Single Espresso
   vars  n mean   sd median trimmed  mad  min  max range skew kurtosis   se
X1    1 15 6.69 0.98   6.37    6.63 0.78 5.25 8.88  3.63 0.69    -0.53 0.25
------------------------------------------------------------------------------
group: Water
   vars  n mean   sd median trimmed mad  min  max range skew kurtosis  se
X1    1 15 4.82 1.16   4.75    4.82 0.8 2.53 7.09  4.56 0.03    -0.57 0.3
```

The calculation is performed over the *adjusted* means (which take into consideration the influence of the covariate):

Water Group = 7.33

Double Espresso Group = 6.32

Single Espresso Group = 6.87

```
> emmeans (model_ancova, pairwise~Condition, adjust="none")
$emmeans
 Condition          emmean        SE df lower.CL upper.CL
 Double Espresso 6.319464 0.4152816 41 5.480786 7.158142
 Single Espresso 6.871614 0.1934303 41 6.480974 7.262255
 Water           7.327960 0.3931110 41 6.534056 8.121864

Confidence level used: 0.95
```

If our experimental factor in the ANCOVA *had* been significant, we could have looked at the pairwise comparisons reported by *emmeans* to determine what condition was different from what other condition…

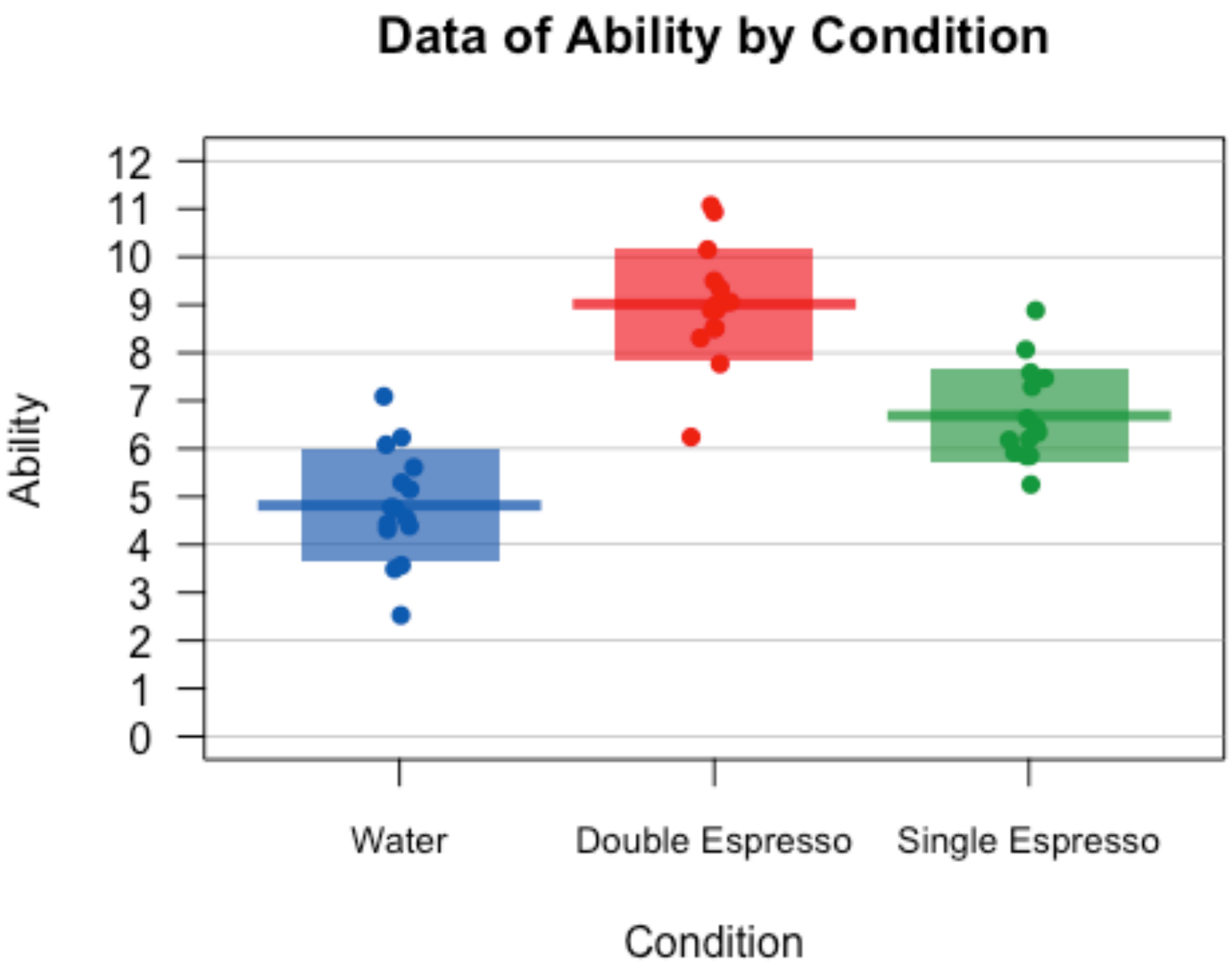
```
$contrasts
 contrast                                estimate        SE df t.ratio p.value
 Double Espresso - Single Espresso -0.5521505 0.4779448 41  -1.155  0.2547
 Double Espresso - Water           -1.0084959 0.7614421 41  -1.324  0.1927
 Single Espresso - Water           -0.4563454 0.4179276 41  -1.092  0.2812
```

But once we take account of the influence of our covariate we found no effect of Condition…

AN(C)OVA as a special case of regression…

- Let’s return to the example we looked at for ANCOVA - and let’s forget the co-variate for a moment…

- We looked at how double espresso vs. single espresso vs. water drinking (our IV) might influence people’s motor task performance (our DV).



Water mean = 4.82
Double Espresso mean = 9.02
Single Espresso mean = 6.69

- First we need to use dummy coding of the levels of our experimental factor - which is the default coding in *R* for factors…

```
> contrasts (cond$Condition)
                 Double Espresso Single Espresso
Water                          0                0
Double Espresso                1                0
Single Espresso                0                1
```

Ability = Intercept + $\beta 1$(Double Espresso) + $\beta 2$(Single Espresso) + ε

The Intercept is our reference category (Water) with coding (0, 0), while the dummy coding for Double Espresso is (1, 0) and for Single Espresso (0, 1)

Ability = Intercept + *β1*(Double Espresso) + *β2*(Single Espresso) + *ε*

We want to calculate *β1* and *β2*

```
> lm1 <- lm(Ability ~ Condition, data=cond)
> lm1

Call:
lm(formula = Ability ~ Condition, data = cond)

Coefficients:
             (Intercept)  ConditionDouble Espresso  ConditionSingle Espresso
                   4.817                     4.199                     1.871
```

The intercept is 4.817 (which is the mean of our Water group), *β1 i*s 4.2, and *β2* is 1.87

To work out the mean Ability of our Double Espresso Group:

Ability = Intercept + $\beta 1$(Double Espresso) + $\beta 2$(Single Espresso) + ε

Ability = 4.82 + 4.2(1) + 1.87(0) + ε

Ability = 4.82 + 4.2 + ε

Ability = 9.02 + ε

To work out the mean Ability of our Single Espresso Group:

Ability = Intercept + $\beta 1$(Double Espresso) + $\beta 2$(Single Espresso) + ε

Ability = 4.82 + 4.2(0) + 1.87(1) + ε

Ability = 4.82 + 1.87 + ε

Ability = 6.69 + ε

Which are the exact same means generated by the ANOVA…

Water mean = 4.82
Double Espresso mean = 9.02
Single Espresso mean = 6.69

We can do ANCOVA like this too - let's consider our co-variate of Gaming frequency…

The *adjusted* means from the ANCOVA (which take into consideration the influence of the covariate) were:

Water Group = 7.33

Double Espresso Group = 6.32

Single Espresso Group = 6.87

Ability = Intercept + *β1(*Gaming) + *β2*(Double Espresso) + *β3*(Single Espresso) + *ε*

Add the covariate to our model *before* the experimental factor:

```
> lm2 <- lm(Ability ~ Gaming + Condition, data=cond)
> lm2

Call:
lm(formula = Ability ~ Gaming + Condition, data = cond)

Coefficients:
             (Intercept)                    Gaming  ConditionDouble Espresso  ConditionSingle Espresso
                 -3.4498                    0.8538                   -1.0085                   -0.4563
```

Ability = Intercept + *β1(*Gaming) + *β2*(Double Espresso) + *β3*(Single Espresso) + *ε*

The *β2* and *β3* coefficients tell us the difference between each group mean (i.e., the adjusted mean) compared to the reference Group (Water) when taking into account the covariate of Gaming frequency:

β2 is the difference between the Double Espresso and Water group adjusted means (= -1.01) while *β3* is the difference between the Double Espresso and Water group adjusted means (= -0.46)…

Let’s check - the following are the adjusted means output by the ANCOVA model:

Water Group = 7.33

Double Espresso Group = 6.32

Single Espresso Group = 6.87

Difference between the Water and Double Espresso Group is 1.01 and the difference between the Water and Single Espress Group is 0.46…

We can work out the mean of our reference group (Water) by plugging in the values to our equation - note that Gaming is not a factor and we need to enter the mean of this variable (which is 12.62296). So,…

Ability = Intercept + *β1(*Gaming) + *β2*(Double Espresso) + *β3*(Single Espresso) + *ε*

Ability = -3.4498 + 0.8538(12.62296) + (- 1.0085)(0) + (-0.4563)(0) + ε

Ability = -3.4498 + 10.777 + ε

Ability = 7.33 + ε

7.33 is the adjusted mean for the Water group…which is what we had from calling the *emmeans* function following the ANCOVA…

Multiple Regression in R

In Formula One, we suspect that the amount of money a team invests in their cars positively influences how many points the team scores in the championship. We also suspect that other factors influence how many points are scored - we have measures of car performance, driver age (in years) and media coverage. We want to determine the regression equation to account for the influence of these 4 predictors on the points scored by a team…

First we will build a linear model with all 4 predictors, then a second model with just the intercept (i.e., the mean) - we then compare them - is the model with the 4 predictors a better fit to our data than the model with just the mean?

```
> model.full <- lm (Points ~ Investment + Car_performance + Age + Media_coverage,
data=regression)
> model.null <- lm (Points ~ 1, data=regression)
> anova (model.full, model.null)
Analysis of Variance Table

Model 1: Points ~ Investment + Car_performance + Age + Media_coverage
Model 2: Points ~ 1
  Res.Df    RSS Df Sum of Sq      F   Pr(>F)
1     95  884.4
2     99 1043.1 -4   -158.73 4.2625 0.003236 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The F-ratio comparing our two models is 4.2625 indicating our model with all the predictors is a better fit than our model with just the intercept (the mean). We then need to get our parameter estimates using the function `summary ()`

```
> summary (model.full)

Call:
lm(formula = Points ~ Investment + Car_performance + Age +
Media_coverage,
    data = regression)

Residuals:
     Min       1Q   Median       3Q      Max
-11.8120  -1.7164  -0.1836   1.9258   7.8733

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -268.72776  106.03555  -2.534  0.01290 *
Investment         0.38025    0.10566   3.599  0.00051 ***
Car_performance   -0.86504    0.56151  -1.541  0.12675
Age               -0.27148    0.26726  -1.016  0.31231
Media_coverage     0.02055    0.04777   0.430  0.66809
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

Residual standard error: 3.051 on 95 degrees of freedom
Multiple R-squared:  0.1522,  Adjusted R-squared:  0.1165
F-statistic: 4.263 on 4 and 95 DF,  p-value: 0.003236
```

Here we have our parameter estimates and the t-tests associated with our predictors.

Here are the R-squared and Adjusted R-squared values (which reflects number of predictors in our model).

```
> #as Investment is the only significant predictor, let's drop the others
> model2 <- lm (Points ~ Investment, data=regression)
> anova (model.full, model2)
```

```
> anova (model.full, model2)
Analysis of Variance Table

Model 1: Points ~ Investment + Car_performance + Age + Media_coverage
Model 2: Points ~ Investment
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     95 884.40
2     98 921.52 -3   -37.119 1.3291 0.2695
```

OK, so the models do not differ significantly by this test - we can also use another measure of goodness-of-fit - AIC (Aikaike Information Criterion). AIC tells us how much information in our data is not captured by each model - lower values are better - can only be interpreted in a relative sense (i.e., comparing one model to another)…

```
> #choose model on basis of lower AIC value
> AIC (model.full)
[1] 513.7622
> AIC (model2)
[1] 511.8735
```

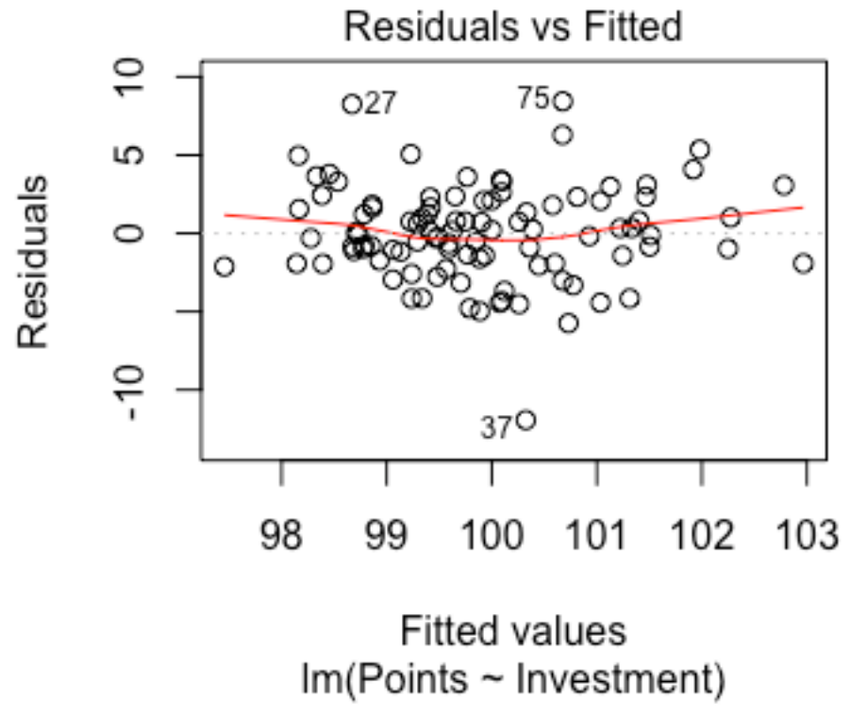
As `model2` has the lower AIC value (so more information in our data is explained by `model2` than by `model.full`), we would be justified in selecting that as our ‘best’ model. AIC penalises models with increasing number of parameters (but not as much as BIC) so gives us a good trade-off of fitting our data and model complexity.

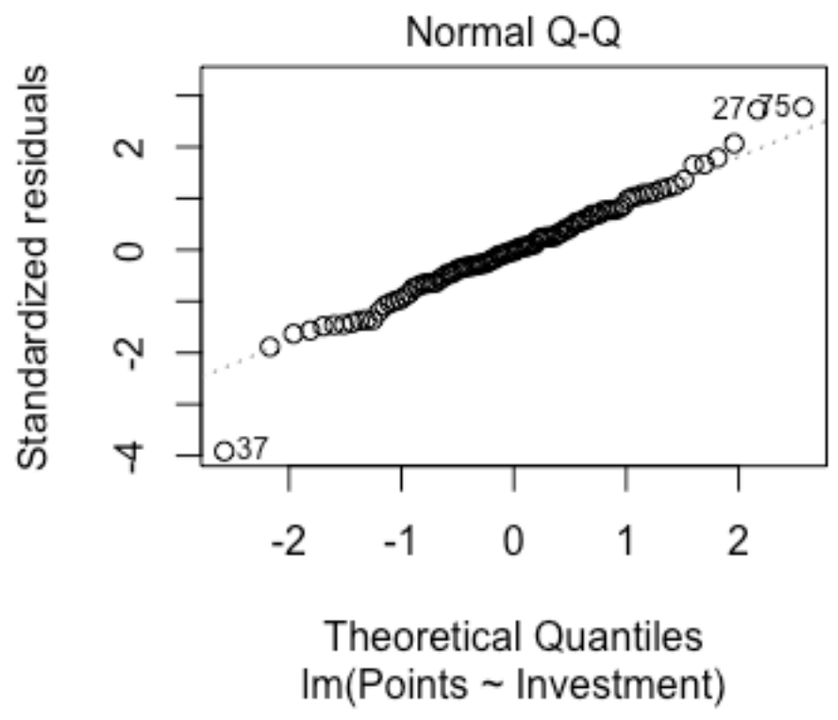
Now let’s look at some diagnostic plots…

We can use the following command to get some visual representations of model fit:

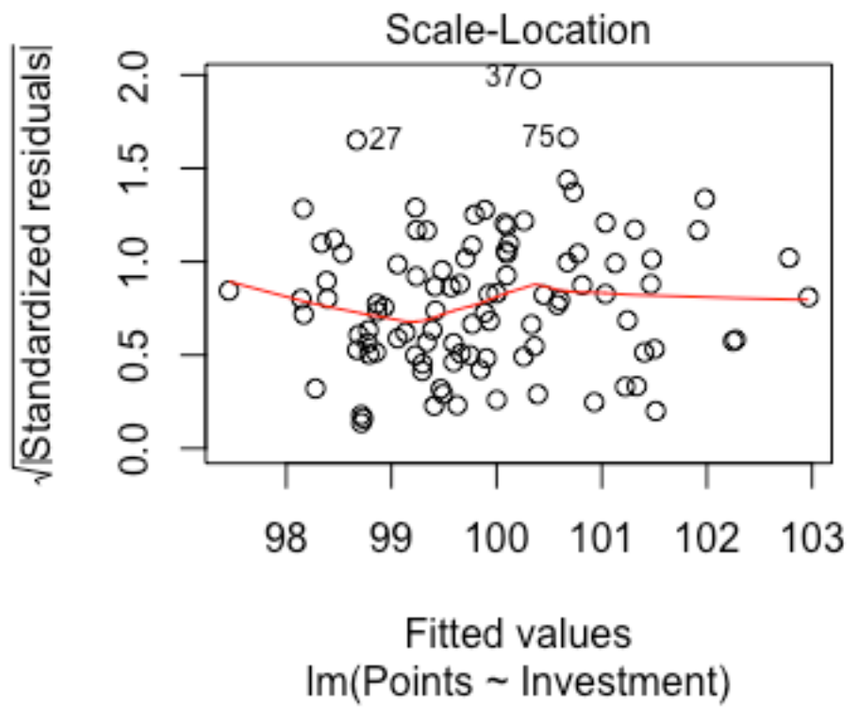
```
> plot (model2)
```

We should have a similar distribution of points (via LOESS Curve Fitting) either side of zero - if we don't it would suggest non-random errors (see Durbin Watson test later). Point 37 looks a bit off both in the Residuals vs. Fitted plot, and also in the Q-Q plot.

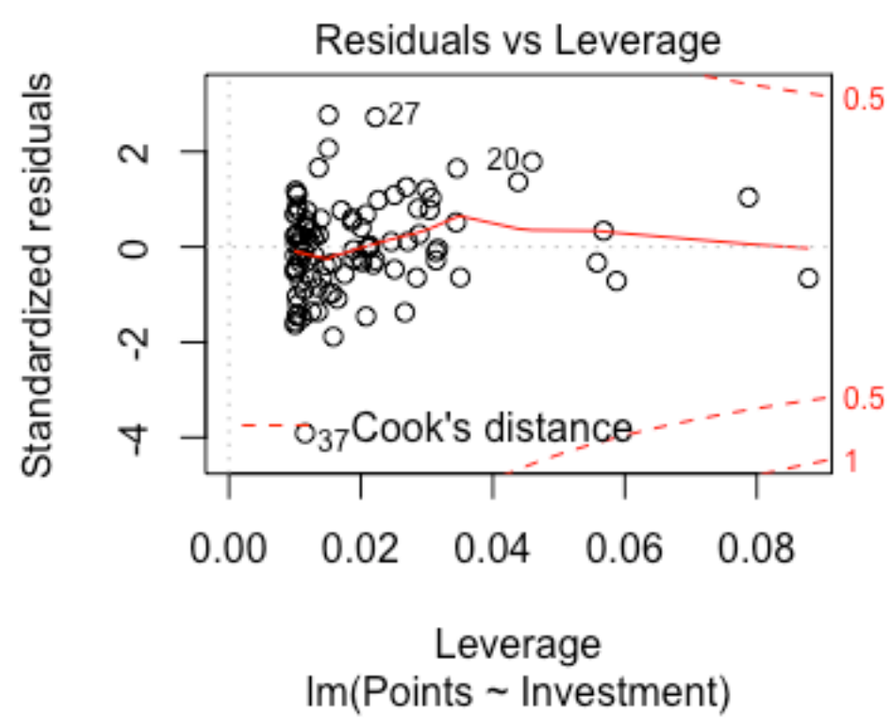




The Scale-Location plot shows if residuals are spread equally along the ranges of predictors. We used this to check the assumption of equal variance (homoscedasticity). We really want to see a horizontal line with equally, randomly spread points.



The Residuals vs. Leverage plot tells us about influential outliers (i.e., outliers that are affecting our model) - when cases are outside of Cook's distance (beyond the dashed line) it means they are having an influential affect on the regression model - we'd might want to exclude these points and rebuild our model.



Durbin Watson Test

- This tests for the non-independence of errors - our errors need to be independent (one of the assumptions of regression). This test needs the `car` package to be loaded.

```
> durbinWatsonTest(model2)
 lag Autocorrelation D-W Statistic p-value
   1       0.07517027      1.830761   0.354
 Alternative hypothesis: rho != 0
```

A D-W value of 2 means that there is no autocorrelation in the sample - our calculated value is pretty close to that - p = .354 so we conclude our errors are independent of each other.

Stepwise Regression Based on AIC Improvement

Rather than building our regression model step by step manually, we can use the `step` function in R - it takes a starting model, and then uses forwards or backwards procedures (or a combination of both) to produce the best model.

We need to install the `MASS` library.

Let's apply the procedure to `model.null` and `model.full` as our limits - we can specify the stepwise procedure with the parameter "direction":

```
> library (MASS)

> steplimitsboth <- step(model.null, scope=list (upper=model.full),
direction = "both")
```

- Let’s focus on the combined method that adds predictors which improve model fit, and removes ones that don’t - based on minimising AIC:

```
> summary (steplimitsboth)

Call:
lm(formula = Points ~ Investment + Car_performance, data = regression)

Residuals:
     Min       1Q   Median       3Q      Max
-11.9048  -1.6602  -0.1308   1.9264   7.7338

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     -251.3167   102.1767  -2.460 0.015678 *
Investment         0.3562     0.1020   3.493 0.000722 ***
Car_performance   -0.8959     0.5489  -1.632 0.105868
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.041 on 97 degrees of freedom
Multiple R-squared:  0.1402,   Adjusted R-squared:  0.1225
F-statistic: 7.908 on 2 and 97 DF,  p-value: 0.0006582
```

We can see the procedure has settled on the model with Investment and Car_performance. AIC value of this model is 511.1639

Stepwise Regression Based on Adjusted R Squared Improvement

- Use the `ols_step_forward` function to work out the model with predictors entered on the basis of improvement via *p*-value and adjusted R^2. For this we need the package `olsrr`.

```
#using ols_step_forward

> install.packages (“olsrr”)
> library (olsrr)
> pmodel <- ols_step_forward(model.full)
> pmodel
```

```
> pmodel
Forward Selection Method

Candidate Terms:

1 . Investment
2 . Car_performance
3 . Age
4 . Media_coverage

-------------------------------------------------------------------------------
                              Selection Summary
-------------------------------------------------------------------------------
        Variable                          Adj.
Step        Entered        R-Square    R-Square     C(p)       AIC         RMSE
-------------------------------------------------------------------------------
   1    Investment           0.1166      0.1076    2.9872    511.8735    3.0665
   2    Car_performance      0.1402      0.1225    2.3410    511.1639    3.0408
   3    Age                  0.1505      0.1240    3.1850    511.9567    3.0382
-------------------------------------------------------------------------------
```

Mallow's Cp is another measure of model fit (low is good).

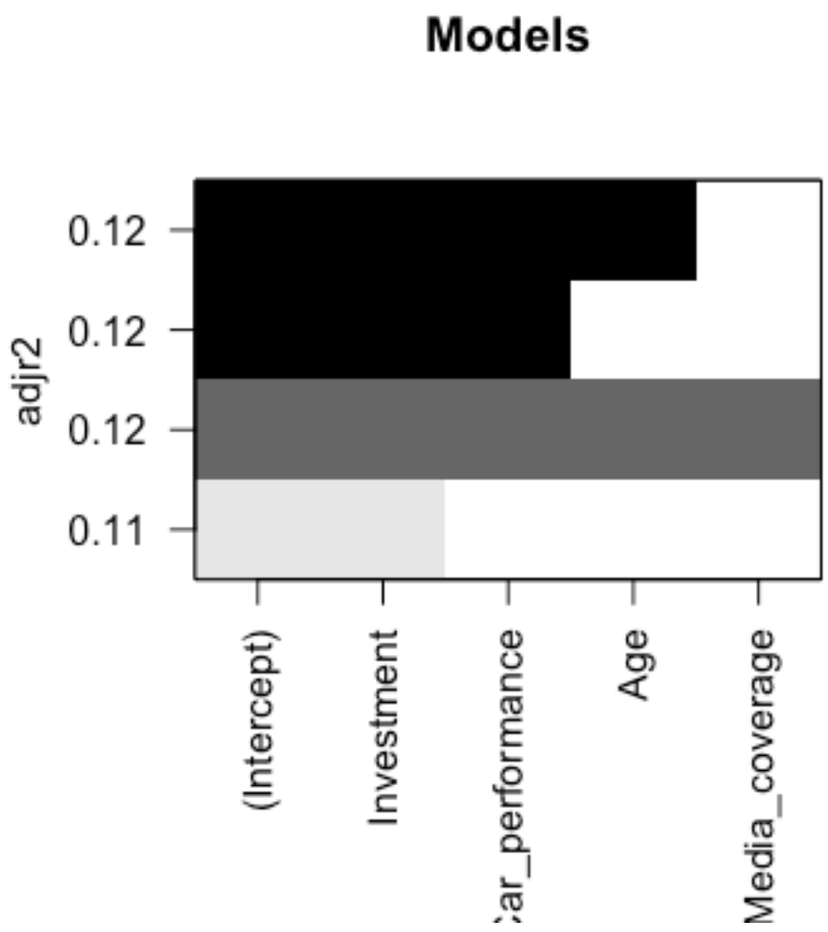
In Step 1 'Investment' is added first, in Step 2 'Car_performance' is then added etc. We end up with the same final model as we had based on AIC improvement.

Adjusted R-squared (which penalises as more predictors are added) is highest for model with the 3 predictors - but AIC tells a slightly different story…

Residual Mean Square Error.

- Visualise the possible models (incl. the one with the largest adjusted R^2 value) using the *leaps* package.

```
> library (leaps)
> leapsmodels <- regsubsets (Points ~
Investment + Car_performance + Age +
Media_coverage, data=regression)
> plot (leapsmodels, scale="adjr2",
main="Models")
```



Collinearity?

- We can apply the `vif()` function to our model - it will work out the VIF values for each of our variables - `vif()` is in the `car` package so don't forget to load that…

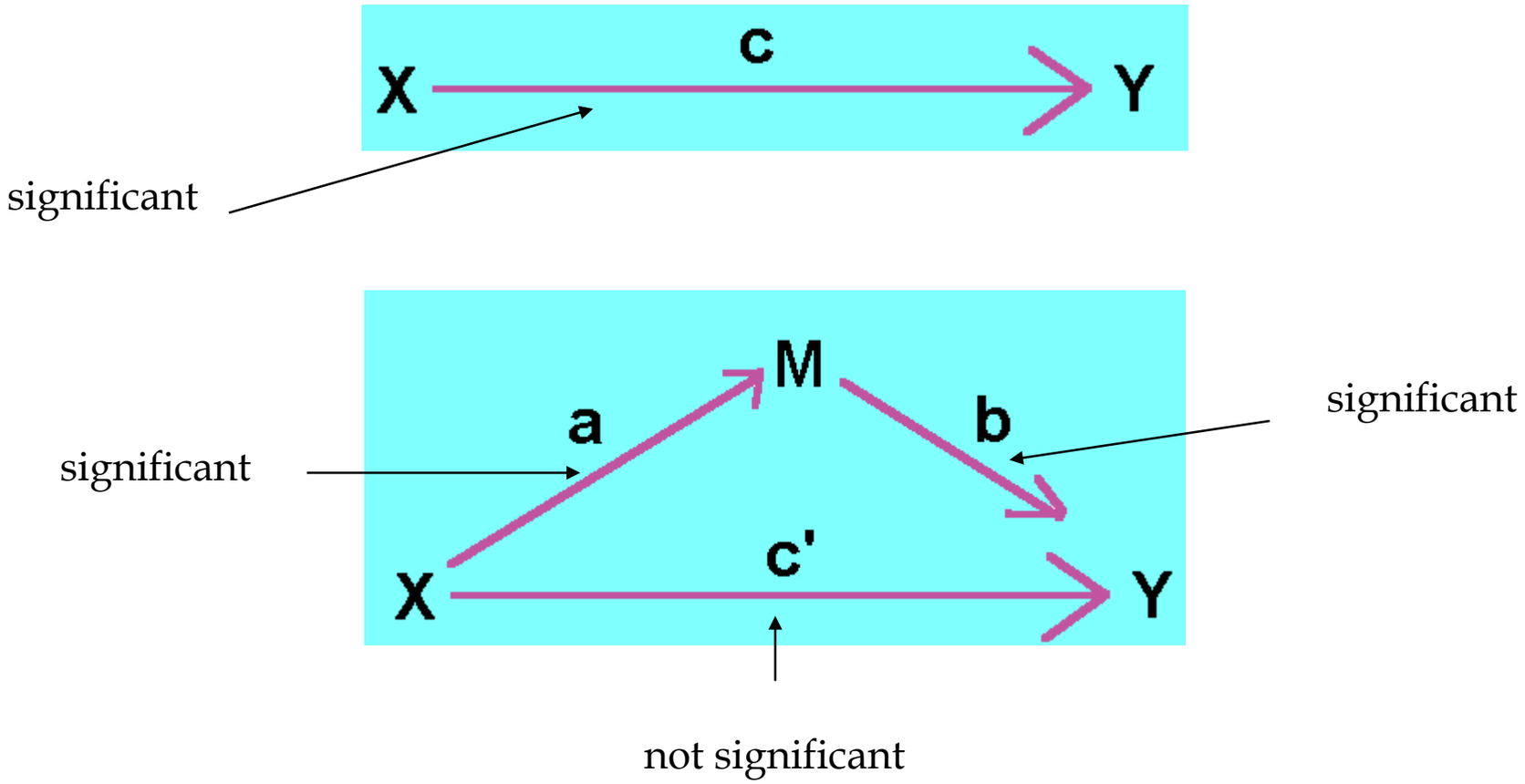
```
> vif (steplimitsboth)
Investment Car_performance
  1.005772        1.005772
```

- As a rule of thumb VIF greater than 10 suggests a multicollinearity issue (although greater than 5 has been suggested too - more conservative).
- For our case, we don't have a collinearity problem as the VIF values are relatively low.

Mediation Effects

- At times, a variable's predicting power of an outcome is lost (complete mediation) or significantly reduced (partial mediation) when another predictor is introduced.

- This added variable mediates the initial effect by its absence and cancels the effect by its presence.



Step 1: The initial variable is correlated with the outcome.

- X is the predictor and Y the outcome variable (path c).

Step 2: Show that the initial variable is correlated with the mediator.

- X is the predictor and M is the outcome variable in the regression equation (path a).

Step 3: Show that the mediator affects the outcome variable.

- X and M are both predictors and Y is the outcome variable (path b).

- X (and not just M) is included so that the initial variable could be controlled in establishing the effect of the mediator on the outcome.

Step 4: After introducing M the effect of X on Y controlling for M (path c') should be zero.

Mediation in R

- Imagine we are interested in whether the number of hours since dawn (X) affects the subjective ratings of wakefulness (Y) of 100 graduate students through the consumption of coffee (M).

- Our datafile is called `Meddata` - we are also using the stargazer package to produce our table - so need to:

```
> install.packages("stargazer")
> library (stargazer)
```

The 4 Steps (Baron and Kenny Method)

1. Estimate the relationship between X on Y (hours since dawn on degree of wakefulness) - Path “c” must be significantly different from 0; must have a total effect between the IV & DV

2. Estimate the relationship between X on M (hours since dawn on coffee consumption) - Path “a” must be significantly different from 0; IV and mediator must be related.

3. Estimate the relationship between M on Y controlling for X (coffee consumption on wakefulness, controlling for hours since dawn) - Path “b” must be significantly different from 0; mediator and DV must be related. - The effect of X on Y decreases with the inclusion of M in the model

4. Estimate the relationship between Y on X controlling for M (wakefulness on hours since dawn, controlling for coffee consumption) - Should be non-significant and nearly 0.

Step 1

```
> #1. Total effect
> fit <- lm(Y ~ X, data=Meddata)
> summary(fit)

Call:
lm(formula = Y ~ X, data = Meddata)

Residuals:
    Min      1Q  Median      3Q     Max
-10.917  -3.738  -0.259   2.910  12.540

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 19.88368   14.26371   1.394   0.1665
X            0.16899    0.08116   2.082   0.0399 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.16 on 98 degrees of freedom
Multiple R-squared:  0.04237, Adjusted R-squared:  0.0326
F-statistic: 4.336 on 1 and 98 DF,  p-value: 0.03993
```

Step 2

```
> #2. Path A (X on M)
> fita <- lm(M ~ X, data=Meddata)
> summary(fita)

Call:
lm(formula = M ~ X, data = Meddata)

Residuals:
    Min      1Q  Median      3Q     Max
-9.5367 -3.4175 -0.4375  2.9032 16.4520

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.04494   13.41692   0.451    0.653
X            0.66252    0.07634   8.678 8.87e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.854 on 98 degrees of freedom
Multiple R-squared:  0.4346,  Adjusted R-squared:  0.4288
F-statistic: 75.31 on 1 and 98 DF,  p-value: 8.872e-14
```

Step 3

```
> #3. Path B (M on Y, controlling for X)
> fitb <- lm(Y ~ M + X, data=Meddata)
> summary(fitb)

Call:
lm(formula = Y ~ M + X, data = Meddata)

Residuals:
    Min      1Q  Median      3Q     Max
-9.3651 -3.3037 -0.6222  3.1068 10.3991

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 17.32177   13.16216   1.316    0.191
M            0.42381    0.09899   4.281 4.37e-05 ***
X           -0.11179    0.09949  -1.124    0.264
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.756 on 97 degrees of freedom
Multiple R-squared:  0.1946,  Adjusted R-squared:  0.1779
F-statistic: 11.72 on 2 and 97 DF,  p-value: 2.771e-05
```

Step 4

```
> #4. Reversed Path C (Y on X, controlling for M)
> fitc <- lm(X ~ Y + M, data=Meddata)
> summary(fitc)

Call:
lm(formula = X ~ Y + M, data = Meddata)

Residuals:
    Min      1Q  Median      3Q     Max 
-14.438  -2.573  -0.030   3.010  11.779 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 96.11234    9.27663  10.361  < 2e-16 ***
Y           -0.11493    0.10229  -1.124    0.264    
M            0.69619    0.08356   8.332 5.27e-13 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.823 on 97 degrees of freedom
Multiple R-squared:  0.4418,  Adjusted R-squared:  0.4303 
F-statistic: 38.39 on 2 and 97 DF,  p-value: 5.233e-13
```

```
> stargazer(fit, fita, fitb, fitc, type = "text", title = "Baron and Kenny Method")

Baron and Kenny Method
================================================================================================
                                              Dependent variable:
                        ------------------------------------------------------------------------
                                Y                    M                    Y                   X
                               (1)                  (2)                  (3)                 (4)
------------------------------------------------------------------------------------------------
Y                                                                                          -0.115
                                                                                           (0.102)

M                                                                        0.424***          0.696***
                                                                         (0.099)           (0.084)

X                           0.169**              0.663***             -0.112
                            (0.081)              (0.076)              (0.099)

Constant                    19.884               6.045                17.322            96.112***
                            (14.264)             (13.417)             (13.162)          (9.277)

------------------------------------------------------------------------------------------------
Observations                  100                  100                  100                 100
R2                           0.042                0.435                0.195               0.442
Adjusted R2                  0.033                0.429                0.178               0.430
Residual Std. Error    5.160 (df = 98)      4.854 (df = 98)      4.756 (df = 97)     4.823 (df = 97)
F Statistic         4.336** (df = 1; 98) 75.313*** (df = 1; 98) 11.715*** (df = 2; 97) 38.389*** (df = 2; 97)
================================================================================================
Note:                                                                  *p<0.1; **p<0.05; ***p<0.01
```

Step 1 shows a significant positive relationship between hours since dawn (X) and wakefulness (Y).

Our Path A model (Step 2) shows that hours since down (X) is also positively related to coffee consumption (M).

Our Path B model (Step 3) shows that coffee consumption (M) positively predicts wakefulness (Y) when controlling for hours since dawn (X).

Finally, (Step 4) wakefulness (Y) does not predict hours since dawn (X) when controlling for coffee consumption (M).

Since the relationship between hours since dawn and wakefulness is no longer significant when controlling for coffee consumption, this suggests that coffee consumption does in fact mediate this relationship.

Determining the significance of our mediator

- With the Baron and Kenny approach, we can use the Sobel test to determine the p-value associated with the influence of our mediator.

- We need to first install the package called `multilevel` so remember then to load it: `library (multilevel)`

- This is what we are interested in - it's the z-value of the effect of our mediator - you can calculate the 2-tailed p-value from the normal distribution like this:

```
> 2*(1-pnorm(3.83939))

[1] 0.0001233404
```

```
> sobel(Meddata$X, Meddata$M, Meddata$Y)
$`Mod1: Y~X`
             Estimate Std. Error  t value   Pr(>|t|)
(Intercept) 19.8836805 14.2637142 1.394004 0.16646905
pred         0.1689931  0.0811601 2.082220 0.03992761

$`Mod2: Y~X+M`
             Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 17.3217682 13.16215851  1.316028 1.912663e-01
pred        -0.1117904  0.09949262 -1.123605 2.639537e-01
med          0.4238113  0.09899469  4.281152 4.371472e-05

$`Mod3: M~X`
             Estimate  Std. Error   t value     Pr(>|t|)
(Intercept) 6.0449365 13.41692114 0.4505457 6.533122e-01
pred        0.6625203  0.07634187 8.6783345 8.871741e-14

$Indirect.Effect
[1] 0.2807836

$SE
[1] 0.07313234

$z.value
[1] 3.83939

$N
[1] 100
```

Mediation via Bootstrapping

- Sobel test can be quite conservative - *bootstrapping* and the *mediation* package a more powerful method to investigate mediation effect. Bootstrapping better for small samples.

- Imagine you have a sample of *n* measurements, you can sample this in many ways as long as you allow some values to appear more than once and others to be left out (sampling with replacement). Calculate the mean lots of times (one for each sampling - often as many as 10,000) and work out the CIs and other population parameter estimates. This is *bootstrapping*.

The Mediation Package

- Based on Preacher & Hayes (2004) - more power than Sobel test - well suited for small sample sizes.

- Need to build two models - one for direct effect of our predictors and one for the effect of our predictor and mediator.

```
> require (mediation)
> fitM <- lm(M ~ X, data=Meddata) #IV on M; Hours since dawn predicting coffee
consumption
> fitY <- lm(Y ~ X + M, data=Meddata) #IV and M on DV; Hours since dawn and
coffee predicting wakefulness
```

- Then use the `mediate` function in the `mediation` package to compare the two models - allows us to estimate effect of the mediator…

```
> fitMed <- mediate(fitM, fitY, treat="X", mediator="M")
> summary(fitMed)
```

```
> summary(fitMed)

Causal Mediation Analysis

Quasi-Bayesian Confidence Intervals

               Estimate 95% CI Lower 95% CI Upper p-value
ACME            0.276803     0.144987         0.43  <2e-16 ***
ADE            -0.115043    -0.316462         0.07   0.268
Total Effect    0.161760     0.000729         0.31   0.048 *
Prop. Mediated  1.653327     0.507091         9.66   0.048 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Sample Size Used: 100


Simulations: 1000
```

- By running this we get the Average Causal Mediation Effects (ACME), our Average Direct Effects (ADE), combined indirect and direct effects (Total Effect), and the ratio of these estimates (Prop. Mediated).
- The ACME is the indirect effect of M (Total Effects - ADE) and the associated *p*-value value tells us if our mediation effect is significant.
- We can bootstrap our data and fit a model based on our estimated population parameters (which is recommend over the default CI estimation method above)…

```
> fitMedBoot <- mediate(fitM, fitY, boot=TRUE, sims=10000,
treat="X", mediator="M")
> summary(fitMedBoot)
```

```
> summary(fitMedBoot)

Causal Mediation Analysis

Nonparametric Bootstrap Confidence Intervals with the Percentile Method

               Estimate 95% CI Lower 95% CI Upper p-value
ACME            0.28078      0.14112         0.43  <2e-16 ***
ADE            -0.11179     -0.29548         0.09   0.273
Total Effect    0.16899     -0.00862         0.35   0.066 .
Prop. Mediated  1.66151     -3.92801        10.91   0.066 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Sample Size Used: 100


Simulations: 10000
```

- We now see that the ACME is the only one that is significant - this tells us we have a significant moderator - with no direct effect of our predictor or combined effect of predictor and moderator when the moderator is taken into consideration. In this case, it’s the same story as we found with the Baron and Kenny method.

Moderation Effects

- Moderation tests whether a variable (Z) affects the direction and/or strength of the relation between an IV (X) and a DV (Y).

- In other words, moderation tests for interactions that affect <u>when</u> relationships between variables occur.

- Imagine we are interested in whether the relationship between the number of hours of sleep (X) a student receives and the attention that they pay to this lecture (Y) is influenced by their consumption of coffee (Z).

Our predictor variables are X and Z - good idea to centre them (i.e., mean of zero) using the `scale` function. It is important to centre both the moderator and predictor to reduce multicolinearity and make interpretation easier.

```
#Centering Data
Xc    <- c(scale(X, center=TRUE, scale=FALSE)) #Centering IV;
hours of sleep
Zc    <- c(scale(Z,  center=TRUE, scale=FALSE)) #Centering
moderator; coffee consumption
```

```
> fitMod <- lm(Y ~ Xc + Zc + Xc:Zc) #Model interacts IV & moderator
> summary(fitMod)

Call:
lm(formula = Y ~ Xc + Zc + Xc : Zc)

Residuals:
    Min      1Q  Median      3Q     Max
-21.466  -8.972  -0.233   6.180  38.051

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 48.54443    1.17286  41.390  < 2e-16 ***
Xc           5.20812    0.34870  14.936  < 2e-16 ***
Zc           1.10443    0.15537   7.108 2.08e-10 ***
Xc:Zc        0.23384    0.04134   5.656 1.59e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

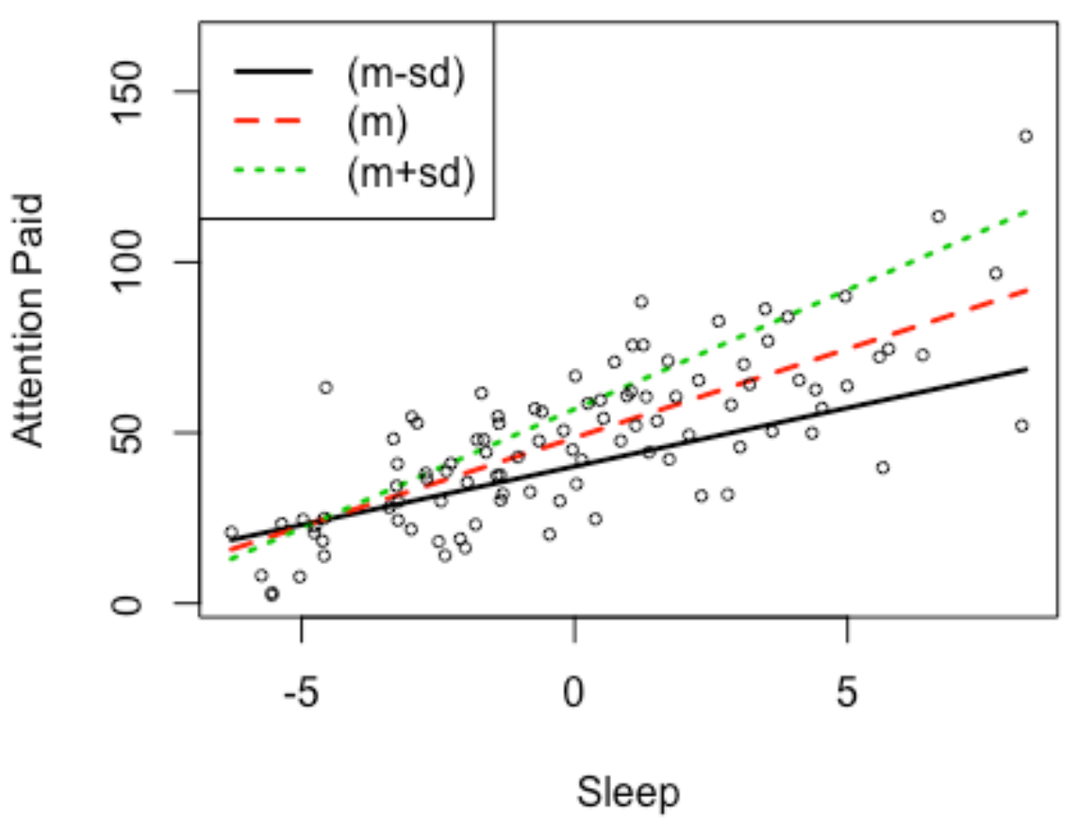
Residual standard error: 11.65 on 96 degrees of freedom
Multiple R-squared:  0.7661,  Adjusted R-squared:  0.7587
F-statistic: 104.8 on 3 and 96 DF,  p-value: < 2.2e-16
```

We can use the `plotSlopes` function in the `rockchalk` package to plot the slopes (1 SD above and 1 SD below the mean) of the moderating effect.

The plot below shows that those who drank less coffee (the black line) paid more attention with the more sleep they got last night, but paid less attention overall than average (the red line).

Those who drank more coffee (the green line) paid more attention when they slept more as well and paid more attention than average.

The difference in the slopes for those who drank more or less coffee shows that coffee consumption moderates the relationship between hours of sleep and attention paid.



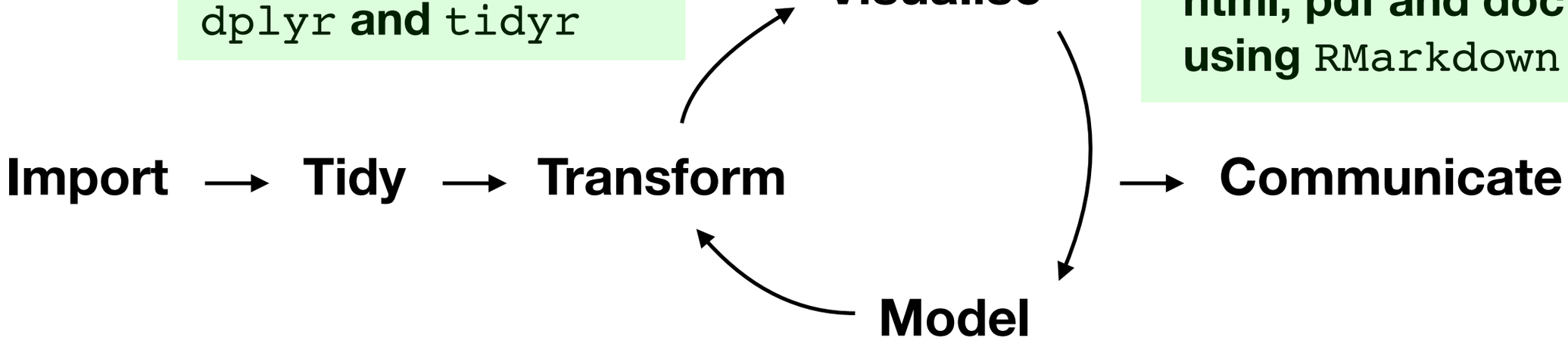

```
ps  <- plotSlopes(fitMod,
plotx="Xc", modx="Zc", xlab =
"Sleep", ylab = "Attention
Paid", modxVals = "std.dev")
```

Workflow

Tidying and transforming using `dplyr` **and** `tidyr`

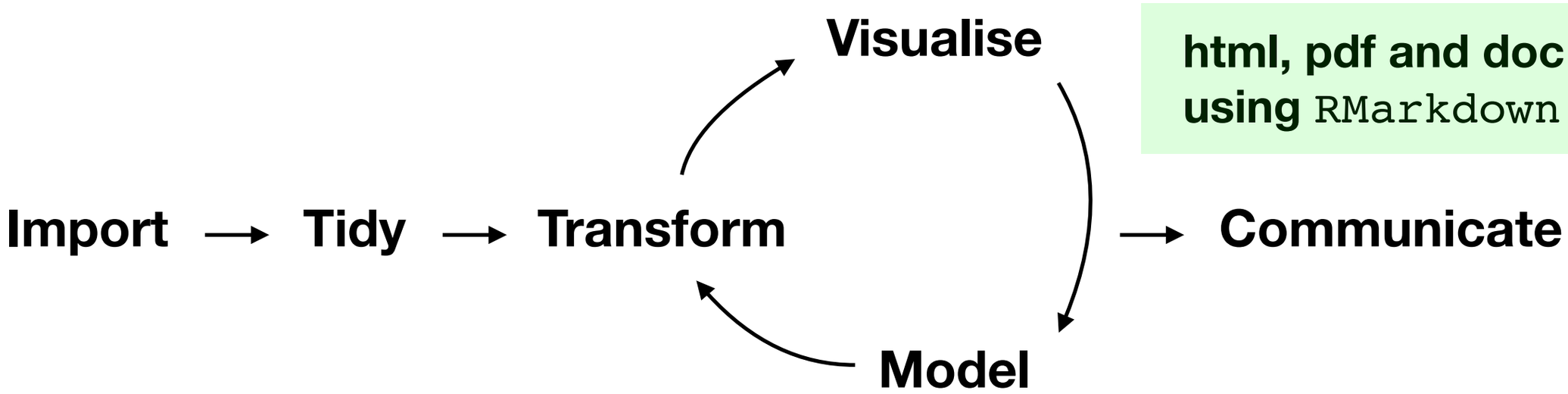
Data plots using `ggplot()` **and** `pirateplot()`

html, pdf and doc reports using `RMarkdown`



ANO(C)VA using `afex()` **and** `aov()`
Linear regression using `lm()` **and** `step()`
(Generalised) linear mixed models using `lmer()` **and** `glmer()`

Communicate

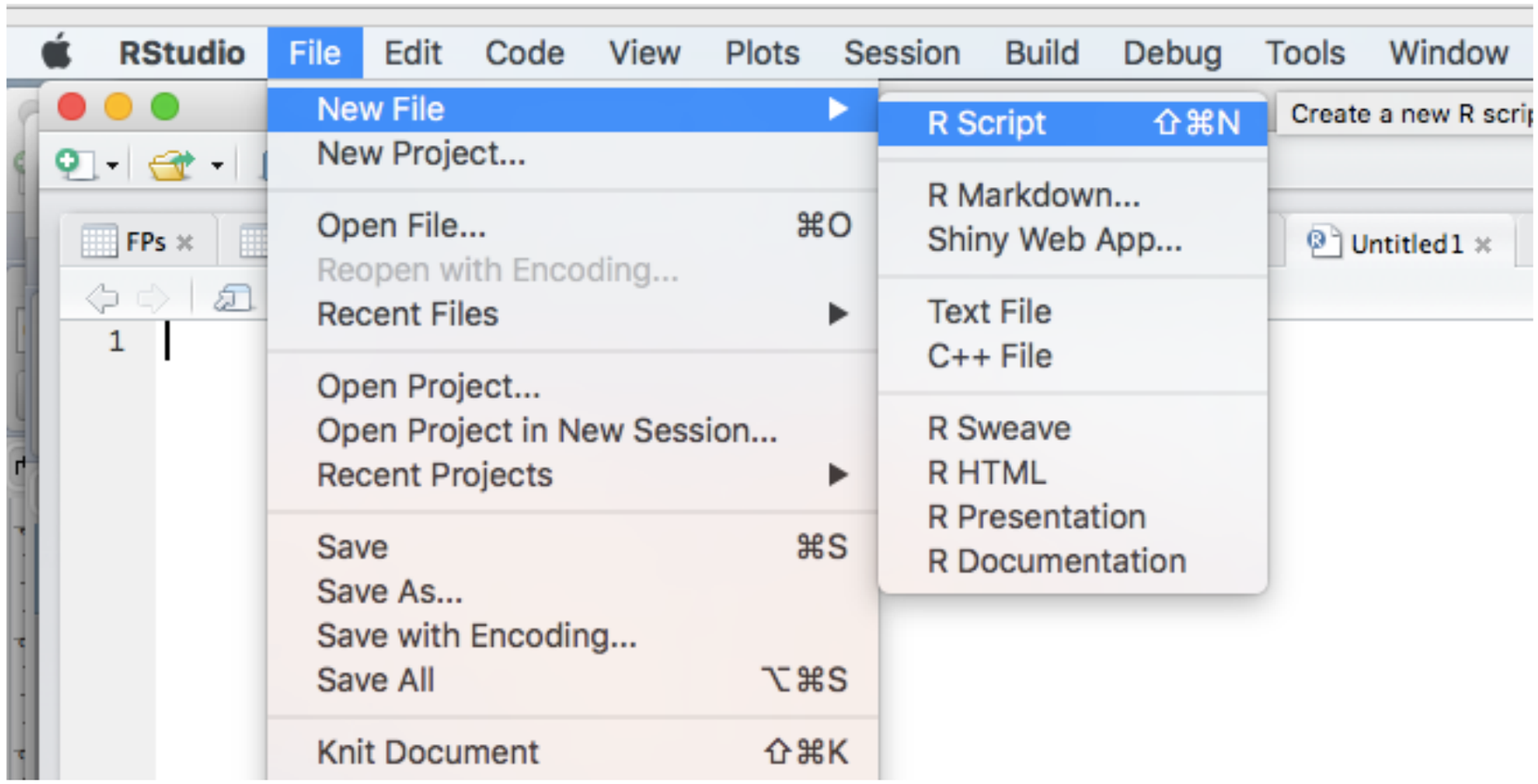


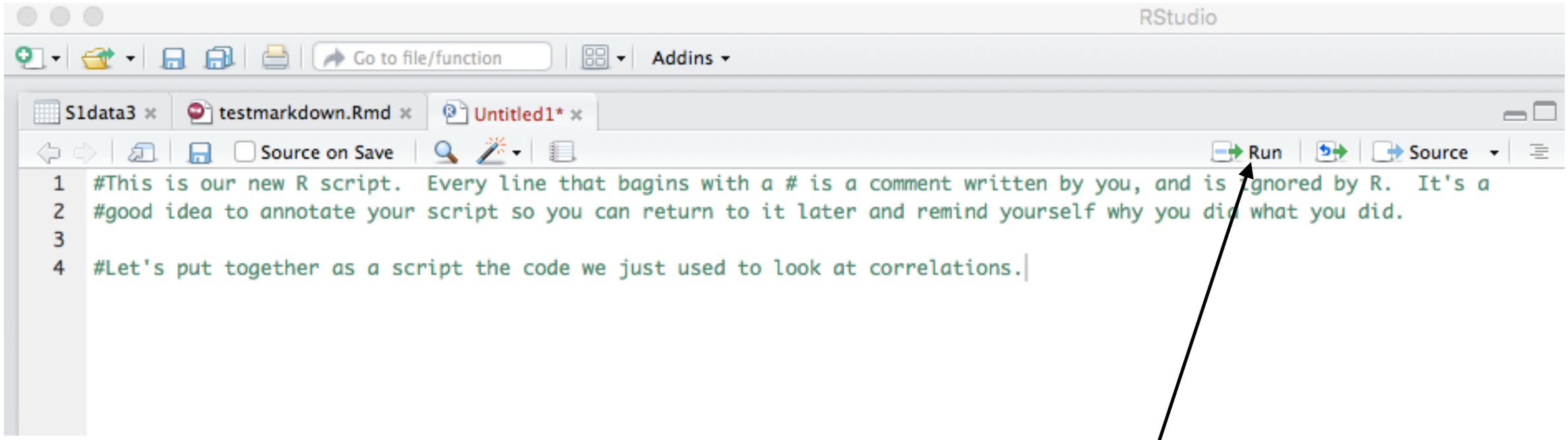
html, pdf and doc reports using `RMarkdown`

Writing Scripts in R

- Rather than simply writing commands line by line in the console, we can create a *script* containing multiple lines of code.
- This script can then be re-run whenever you want, and also could be submitted alongside your paper to a journal for review (along with your data).
- Providing the raw data from your work, plus your analysis script allows other researchers to see *exactly* how you analysed your data. Transparency in the age of the open science is key.

Create a new R script like this:





Once it's written, you can either run your entire script by highlighting it all (CMD-A in OSX) and clicking on 'Run', or you can highlight sections of your script and run only those (by clicking 'Run' once you have highlighted the code you are interested in). CMD-Return will also Run the highlighted code.

S1data3 | testmarkdown.Rmd | Untitled1* | testdata

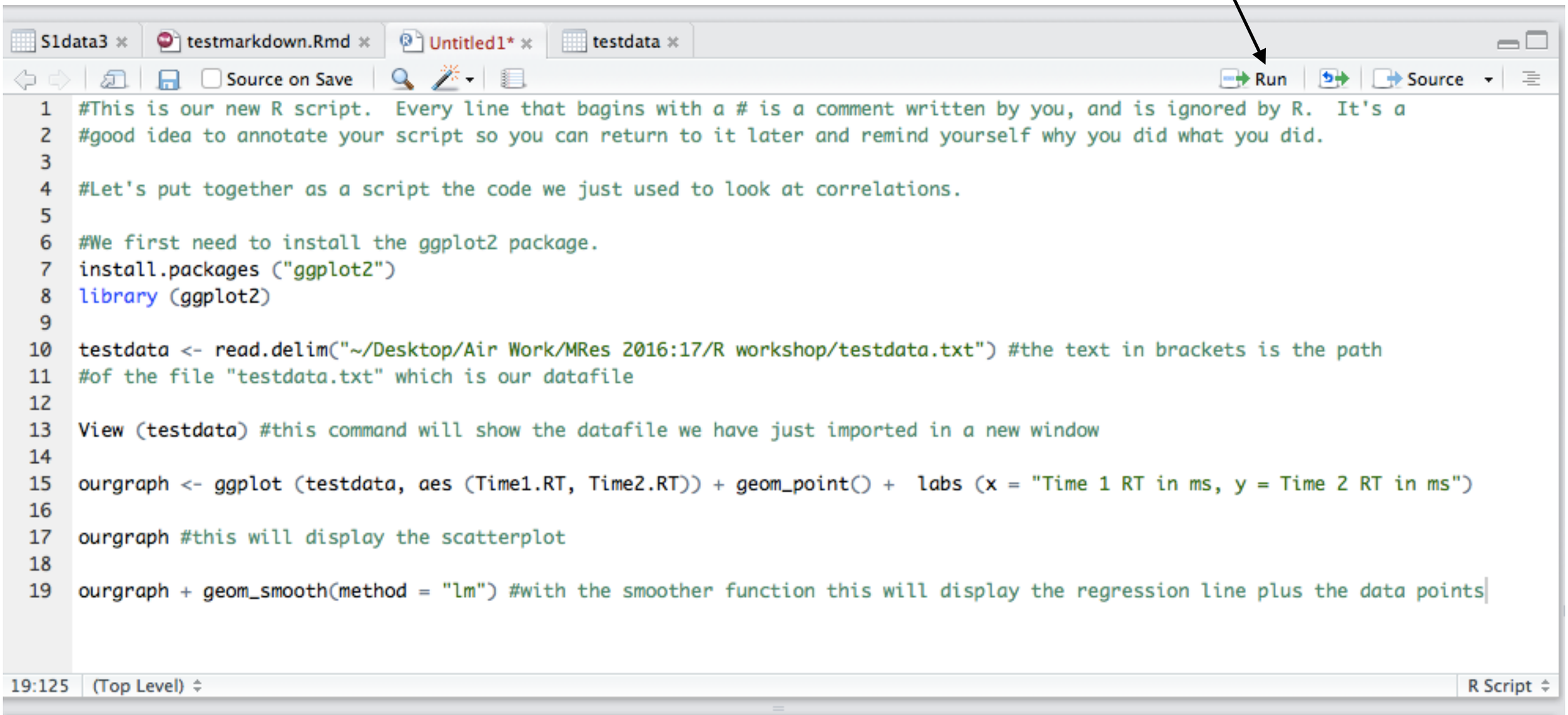
Source on Save | Run | Source

```
1  #This is our new R script.  Every line that bagins with a # is a comment written by you, and is ignored by R.  It's a
2  #good idea to annotate your script so you can return to it later and remind yourself why you did what you did.
3
4  #Let's put together as a script the code we just used to look at correlations.
5
6  #We first need to install the ggplot2 package.
7  install.packages ("ggplot2")
8  library (ggplot2)
9
10 testdata <- read.delim("~/Desktop/Air Work/MRes 2016:17/R workshop/testdata.txt") #the text in brackets is the path
11 #of the file "testdata.txt" which is our datafile
12
13 View (testdata) #this command will show the datafile we have just imported in a new window
14
15 ourgraph <- ggplot (testdata, aes (Time1.RT, Time2.RT)) + geom_point() +  labs (x = "Time 1 RT in ms, y = Time 2 RT in ms")
16
17 ourgraph #this will display the scatterplot
18
19 ourgraph + geom_smooth(method = "lm") #with the smoother function this will display the regression line plus the data points
```

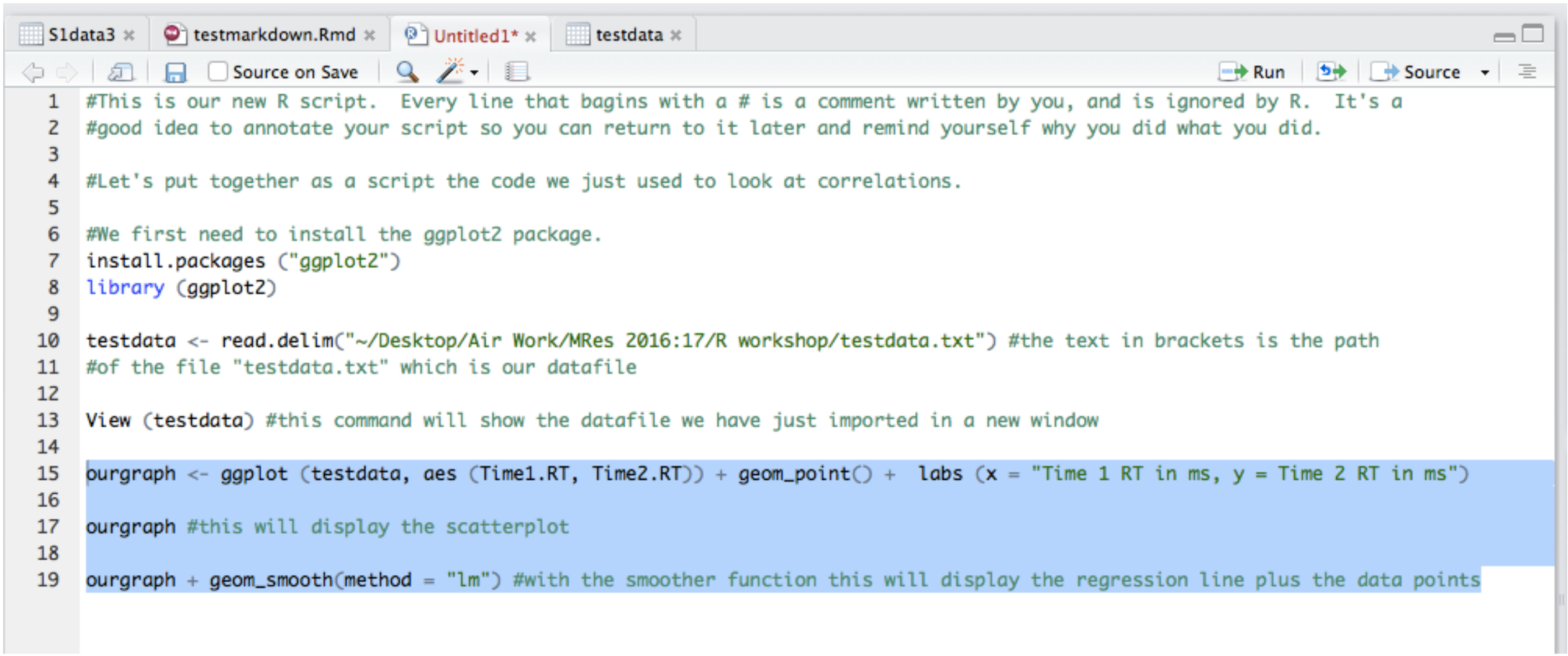
19:125 (Top Level) R Script

We've started writing our script - it includes comments (prefixed by #), the code to load the *ggplot2* package, the code to load our data, and the code to plot our data, first as a scatterplot, then with the regression line.

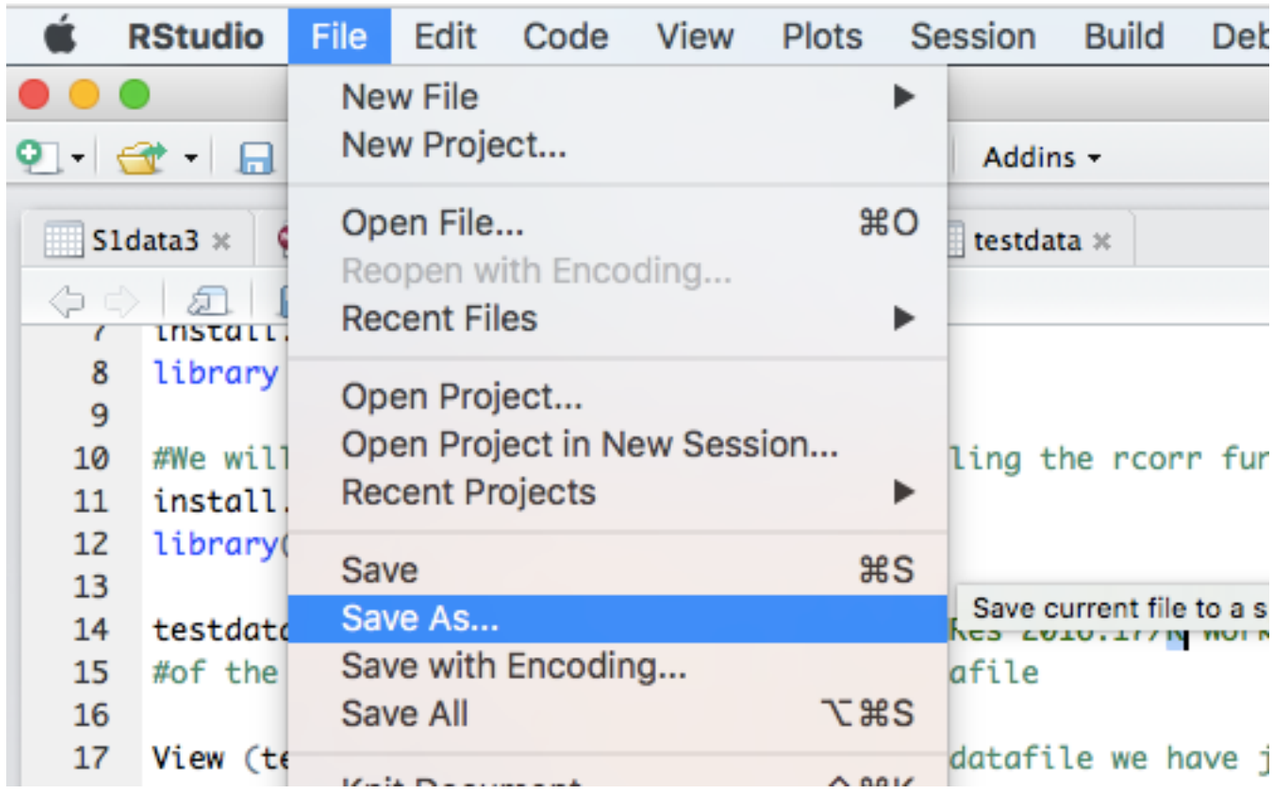
We first need to highlight, and then click on 'Run' to run the whole script.



- But maybe we already have the *ggplot2* package and our data loaded. What if we just want to run the portion of the script associated with generating the graph? We simply highlight it, then click on 'Run'.



Once your script is finished, don't forget to save it…



R Markdown

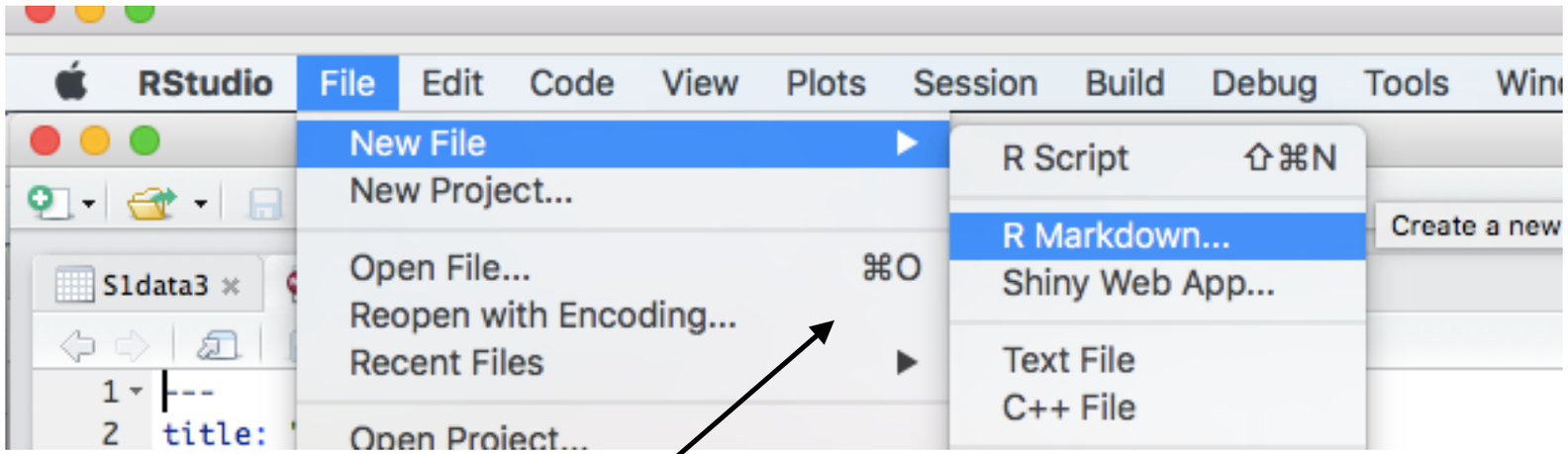
- You can create publishable reports using R Markdown. Such reports contain your code (which you can copy from your R script), the output associated with executing that code, plus comments that you write explaining what you're doing.

- To install it, simply install the R Markdown package:

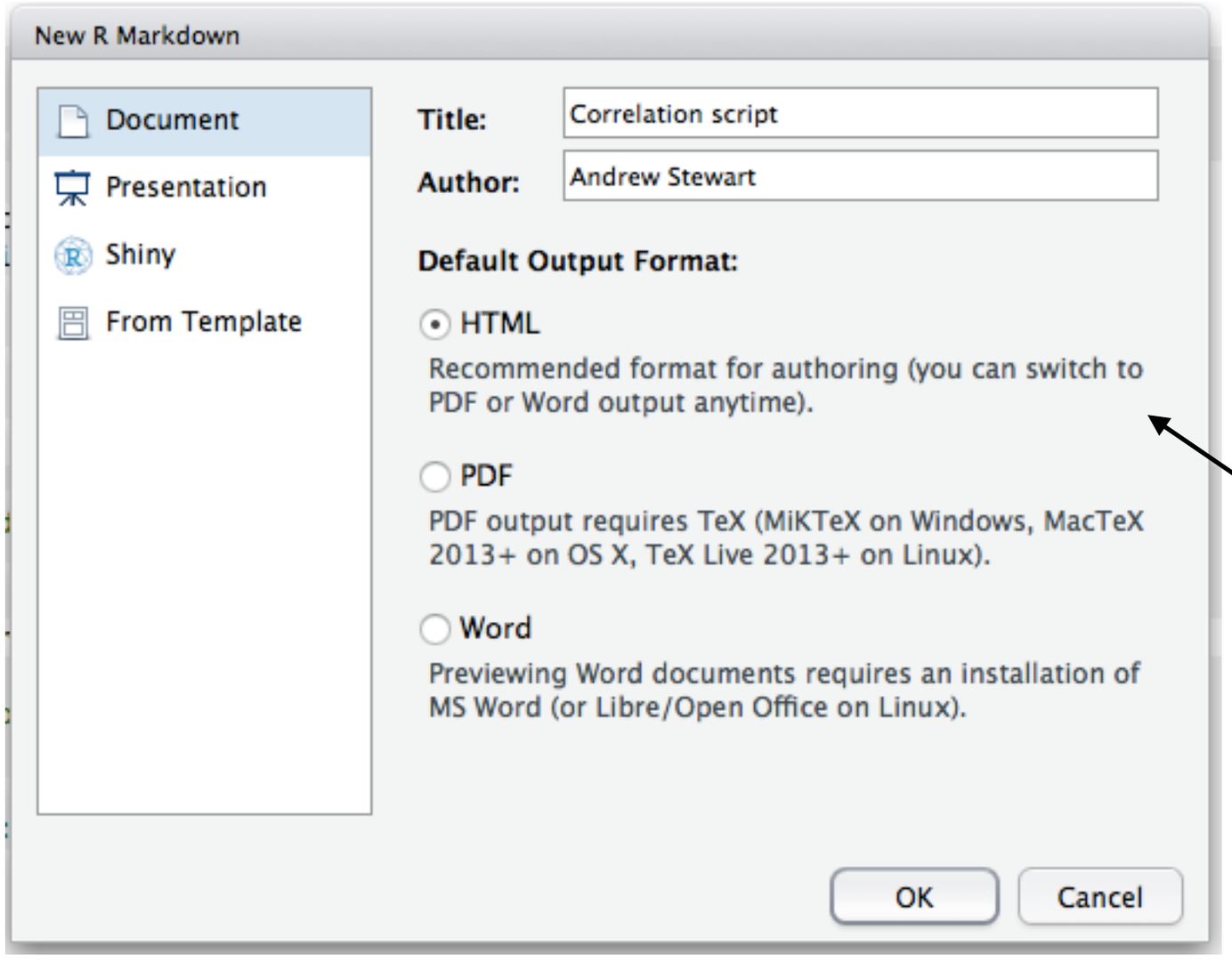
```
> install.packages ("rmarkdown")

> library (rmarkdown)
```

Whenever you want to produce an R Markdown document, make sure you have the *rmarkdown* package loaded.



Start a new R Markdown script. We are going to write a Markdown script incorporating the correlation script we just saw.



Call your document something sensible and specify what type of file it should be. I'm choosing a HTML for this one. You *might* need install a latex file to build a PDF (if so, R will tell you), otherwise you can build a Word file.

Our Markdown script is comprised of R code (grey background) and comments that will appear in the Markdown document (white background).

```
---
title: "Correlation script"
author: "Andrew Stewart"
date: "1/12/2017"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r cars}
summary(cars)
```

Including Plots

You can also embed plots, for example:

```{r pressure, echo=FALSE}
plot(pressure)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

````
---
title: "Correlation script"
author: "Andrew Stewart"
date: "1/12/2017"
output: html_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

This is our new R script that is in an R Markdown document.  All of this text will appear in the Notebook.  Every line that bagins with a # is a comment written by you, and is ignored by R. Howeber, it will appear in the Notebook.

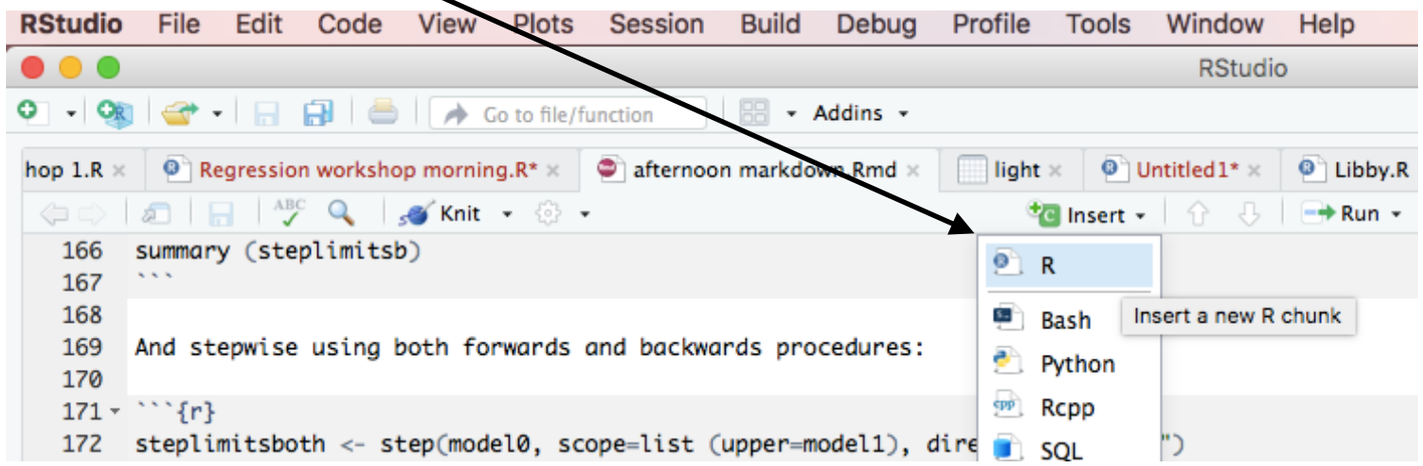
Let's put together as a script the code we just used to look at correlations.

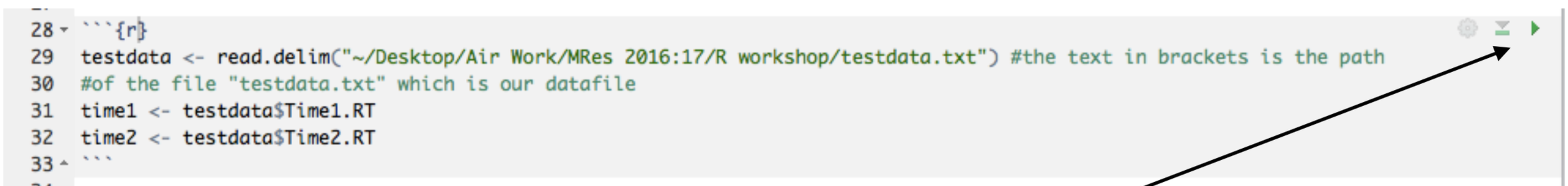
We first need to install the ggplot2 package.

```{r, warning=FALSE, message=FALSE}
install.packages ("ggplot2", repos="http://cran.rstudio.com/")
library (ggplot2)
```
We will also need the Hmisc package for calling the rcorr function
```{r, warning=FALSE, message=FALSE}
install.packages ("Hmisc", repos="http://cran.rstudio.com/")
library(Hmisc)
````

Also note, I am using *warning=FALSE* and *message=FALSE* parameters to stop such text from appearing in the document we're creating. Obviously, if any **have** appeared, you need to be sure they're not serious/relevant. In these case, the warnings relate to a different version number.
````

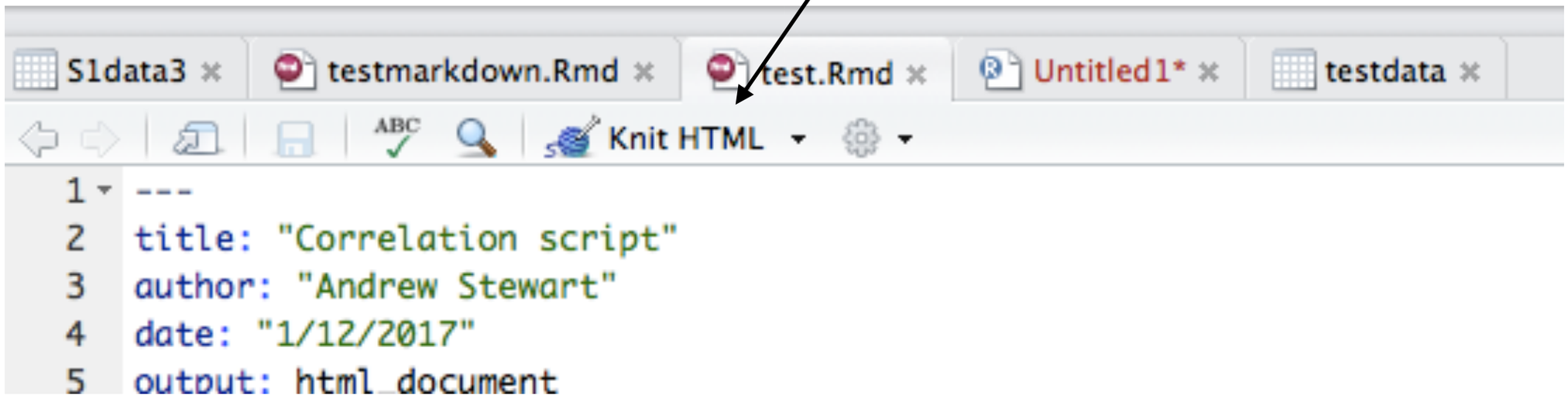
- To insert R code in your Markdown document, click on the Insert button, and select 'R'.





You can click on this arrow , just to run the chunk of R code delimited by ```

- Once you have your R Markdown script written (with appropriate in-text annotations to describe what you're doing, and why) you then need to 'Knit' the code together to produce your report in the format you have chosen:



Correlation script

Andrew Stewart

1/12/2017

This is our new R script that is in an R Markdown document. All of this text will appear in the Notebook. Every line that bagins with a # is a comment written by you, and is ignored by R. Howeber, it will appear in the Notebook.

Let's put together as a script the code we just used to look at correlations.

We first need to install the ggplot2 package.

```
install.packages ("ggplot2", repos="http://cran.rstudio.com/")
```

```
##
## The downloaded binary packages are in
##  /var/folders/95/f9g_lcxj1777n_w285nmptmm0000gn/T//RtmpjNs3Ku/downloaded_packages
```

```
library (ggplot2)
```

We will also need the Hmisc package for calling the rcorr function

```
install.packages ("Hmisc", repos="http://cran.rstudio.com/")
```

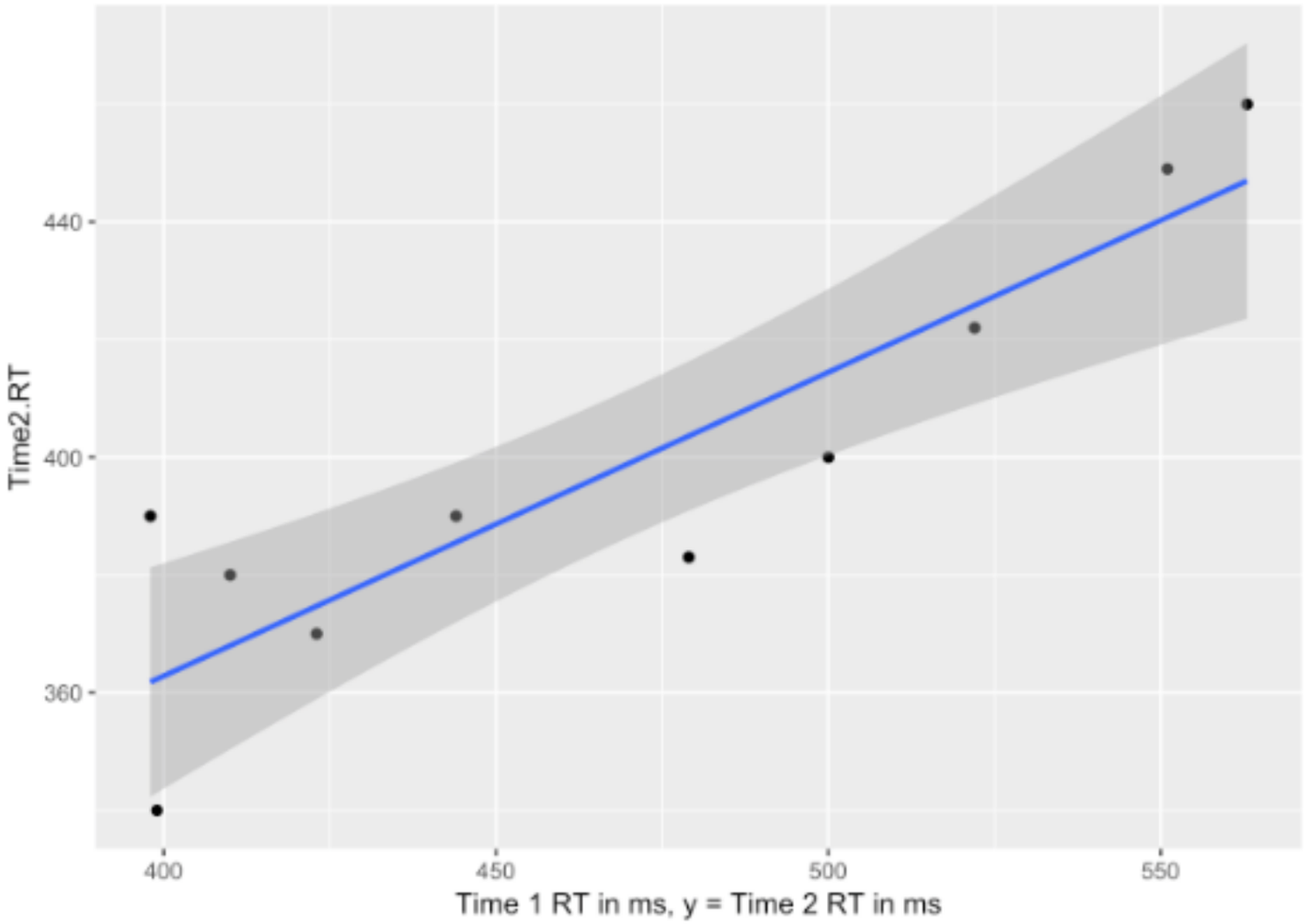
```
##
## The downloaded binary packages are in
##  /var/folders/95/f9g_lcxj1777n_w285nmptmm0000gn/T//RtmpjNs3Ku/downloaded_packages
```

```
library(Hmisc)
```

```
testdata <- read.delim("~/Desktop/Air Work/MRes 2016:17/R workshop/testdata.txt") #the text in brackets is the pat
h
#of the file "testdata.txt" which is our datafile
time1 <- testdata$Time1.RT
time2 <- testdata$Time2.RT
```

You can use double hashes to make a title bigger. Below, you will see our plot plus the regression line.

```
ourgraph <- ggplot (testdata, aes (Time1.RT, Time2.RT)) + geom_point() +  labs (x = "Time 1 RT in ms, y = Time 2 RT in ms")
ourgraph + geom_smooth(method = "lm") #with the smoother function this will display the regression line plus the data points
```



```
time1 <- testdata$Time1.RT
time2 <- testdata$Time2.RT

rcorr (time1, time2)
```

```
##      x    y
## x 1.00 0.89
## y 0.89 1.00
##
## n= 10
##
##
## P
##   x     y
## x       5e-04
## y 5e-04
```

We should probably include some text here which interprets the correlation we have found between our two variables.

Your R Markdown report should contain descriptions of what you're doing which each chunk of code, and any decisions you make as it which analysis to focus on (e.g., interpreting interaction effects). Good commenting helps those read the code, including yourself at some future point.

Sharing your analysis

Journals are increasingly requesting analysis code and data to be published alongside the published paper (and sometimes at the review stage).

You can share your analysis and data even at the submission stage using something like GitHub or OSF.

Git is a powerful tool that allows for the version control in collaborative projects, and the sharing of code and projects.

You can set up a GitHub account, and install GitHub Desktop on your own computer.

GitHub, Inc. github.com/ajstewartlang

Live Trains | Google Scholar | Scopus | jobs.ac.uk | Apple | BBC News | Chester Weather | The Telegraph | The Grauniad | The Independent | Google Maps | Chester Weather Station

Search GitHub | Pull requests | Issues | Gist

Overview | Repositories 4 | Stars 0 | Followers 0 | Following 0

Andrew Stewart

ajstewartlang

Senior Lecturer in the Division of Neuroscience and Experimental Psychology, University of Manchester.

University of Manchester

http://personalpages.manche...

Joined 21 days ago

Pinned repositories

Customize your pinned repositories

List-of-which-repository-goes-with-which-paper

Affective-Theory-of-Mind-Inferences

R code and data for the two experiments

R

Comprehension-of-indirect-requests-is-influenced-by-their-degree-of-imposition

R code and data

R

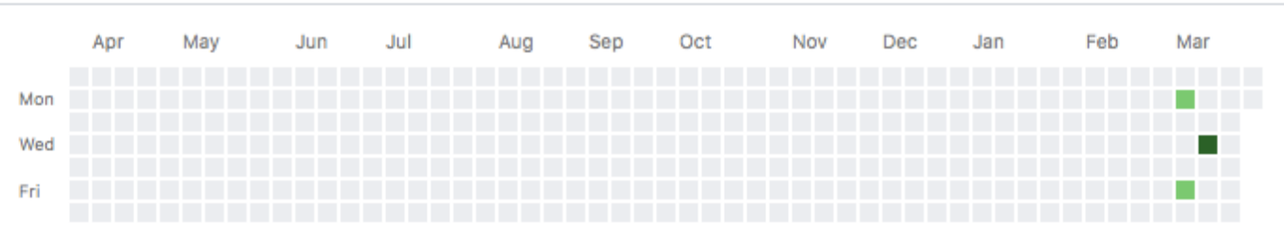
It-s-hard-to-write-a-good-article

R code and data

R

5 contributions in the last year

Contribution settings



Learn how we count contributions.

Less More

This repository Search Pull requests Issues Gist

ajstewartlang / **Affective-Theory-of-Mind-Inferences**

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

R code and data for the two experiments

Edit

Add topics

5 commits 1 branch 0 releases 0 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

ajstewartlang Capitalisation corrected Latest commit 7e14ea2 3 hours ago

Both Expts comparison	Tidied up code with consistent labelling	3 hours ago
Expt 1 Script and data	Tidied up code with consistent labelling	3 hours ago
Expt 2 Script and data	Tidied up code with consistent labelling	3 hours ago
README.txt	Capitalisation corrected	3 hours ago

README.txt

The repository "Both Expts comparison" contains the R analysis script and RT data for both experiments in 1 file. The conditions are relabeled to congruent vs. incongruent to allow a comparison of the magnitude of the congruency effect across the two experiments.

The repository "Expt 1 Script and Data" contains the R analysis script for the Anger/Fear experiment, the RT data, the accuracy data and the two data files (containing the word "graph") used by ggplot2 to graph the means and SEs.

The repository "Expt 2 Script and Data" contains the R analysis script for the Happy/Sad experiment, the RT data, the accuracy data and the two data files (containing the word "graph") used by ggplot2 to graph the means and SEs.

This repository Search Pull requests Issues Gist

ajstewartlang / Affective-Theory-of-Mind-Inferences

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Branch: master

Create new file Upload files Find file History

Affective-Theory-of-Mind-Inferences / Expt 1 Script and data /

ajstewartlang Tidied up code with consistent labelling — Latest commit de7b584 3 hours ago

..

AngerFearAcc.csv	Tidied up code with consistent labelling	3 hours ago
AngerFearRT.csv	Tidied up code with consistent labelling	3 hours ago
AngerFeargraphacc.csv	R code and data	17 days ago
AngerFeargraphdata.csv	R code and data	17 days ago
Expt1_AngerFear_script.R	Tidied up code with consistent labelling	3 hours ago

ajstewartlang / Affective-Theory-of-Mind-Inferences

Branch: master

Find file Copy path

Affective-Theory-of-Mind-Inferences / Expt 1 Script and data / **Expt1_AngerFear_script.R**

ajstewartlang Tidied up code with consistent labelling de7b584 3 hours ago

0 contributors

59 lines (46 sloc) 3.17 KB

Raw Blame History

```
library (lme4)
library (lmerTest)
library (lsmeans)
library (pbkrtest)
library (readr)
library (ggplot2)

#script for AngerFear RT and accuracy data analysis with arousal

#this is the analysis of the RT data
AngerFearRT <- read_csv("~/AngerFearRT.csv")

AngerFearRT$StoryEmotion <- as.factor (AngerFearRT$StoryEmotion)
AngerFearRT$FaceExpression <- as.factor (AngerFearRT$FaceExpression)

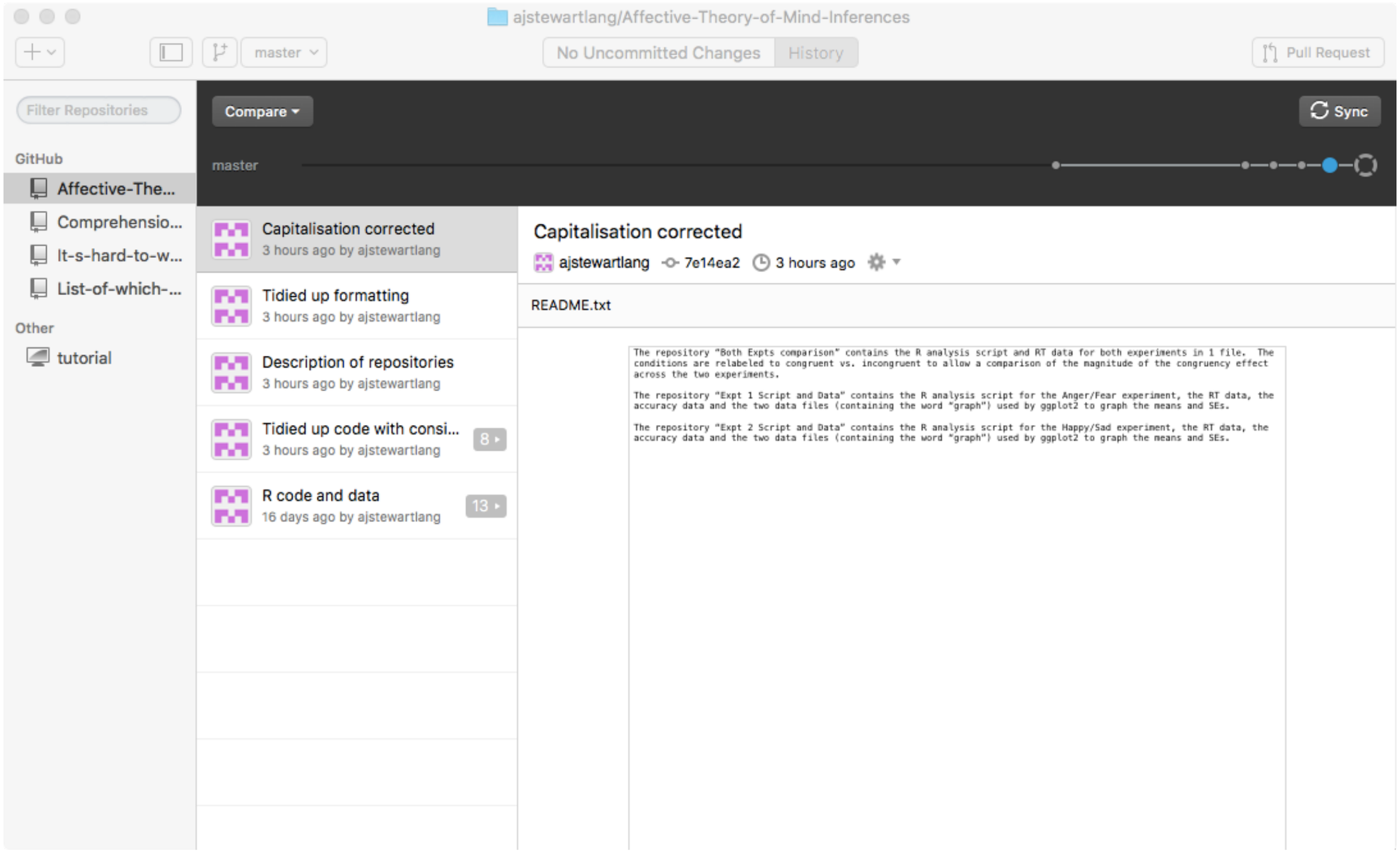
contrasts (AngerFearRT$StoryEmotion) <- matrix (c(.5, -.5))
contrasts (AngerFearRT$FaceExpression) <- matrix (c(.5, -.5))

#with Subject, Vignette, and Face as crossed random effects with arousal
#full model does not converge so need to drop interaction term from the random effects - in addition, Face random effect has only random in
modelRTAr1 <- lmer (RT ~ StoryEmotion*FaceExpression*Arousal + (1+StoryEmotion+FaceExpression|Subject) + (1+StoryEmotion+FaceExpression|Vig
summary (modelRTAr1)
modelRT <- lmer (RT ~ StoryEmotion*FaceExpression + (1+StoryEmotion+FaceExpression|Subject) + (1+StoryEmotion+FaceExpression|Vignette) + (1
anova (modelRTAr1, modelRT)

#difference between models not signif - arousal does not interact with effect so drop arousal from subsequent analysis

modelRTnull <- lmer (RT ~  (1+StoryEmotion+FaceExpression|Subject) + + (1+StoryEmotion+FaceExpression|Vignette) + (1+FaceExpression|Face),
anova (modelRT, modelRTnull)
summary (modelRT)
```

This is GitHub Desktop



This Afternoon

- In the PC cluster, you'll get to work on data visualisation and modelling via ANOVA and Regression using R.
- Remember to write your code in a script - rather than writing it one command at a time.

Tomorrow

- We will focus entirely on (generalised) linear mixed models (GLMMs).

- GLMMs can be used pretty much any time you could use AN(C)OVA or regression. Except they are much more powerful and flexible (allowing for unbalanced designs, missing data, ordinal DVs, and non-normal data).

Recommended Online Resource

The following website is a fantastic interactive book that covers many of analyses (and is what the mediation content is based on) - if you want to expand your R skills, it is well worth a visit:

http://ademos.people.uic.edu/index.html