# Statistics for Psychologists

John Towse, Tom Beesley, Margriet Groen, Rob Davies

2022-10-07

# Contents

# Chapter 1

# Intro

This is a collection of tuition material written for Psychology undergraduates at Lancaster University. At the moment the content represents the "lab materials" for the PSYC121 and PSYC122 modules in first year. They feature tuition of programming with R, with material designed to be accessible for students without any programming background.

## 1.1 Analysis labs and 'pre-lab work'

Where the 121 Analysis lab includes a "pre-lab" section – such as this one! – please ensure you have (a) watched the relevant pre-recorded lecture material (b) read through *this section* before your assigned lab class session and (c) completed any specified exercises.

The lecture is designed to deliver important ideas and procedures for learning about analysis. Pre-lab material is then designed to help consolidate this learning, or enhance, expand and apply it in ways that set the scene for the lab session activity. We want to prepare you to be ready to go in the session itself and make the most of your time there.

> Top tip for part 1: Going into sessions prepared will help you get more out of the teaching.'
>
> Getting an overview or briefing on what you will be doing by reading relevant material etc, means you know what to expect, and where to focus your energy. Don't be shy about coming with questions!

## 1.2 Computing systems

For all University systems you will need to be a registered user and have received a 'username' and details about how to generate your 'password'. If you are struggling to set this up, then contact ISS ASAP.

All staff and students have e-mail accounts on the University system. Normally an e-mail address takes the form: first initial.surname@lancaster.ac.uk

For example, to contact the Content Director of PSYC121, Tom Beesley, you use t.beesley@lancaster.ac.uk (usually you can also use the shortened version, t.beesley@lancs.ac.uk)

When the department needs to contact you individually we send a message to your **university e-mail account**. In addition, the University e-mail security system, specifically the SPAM prevention procedures, will frequently "junk" e-mail from external accounts from free systems (e.g., hotmail, google, etc.). It's always preferable to use your University email account so staff know it is a genuine student request.

Consequently we stress that you:

- DO NOT use external accounts when contacting the Department or Staff as messages from these are not guaranteed to be delivered (and are routinely not replied to by Staff)
- Check your university account daily, as important University and Departmental messages may be in this mailbox

***Moodle***

Moodle plays a key role in your time at Lancaster. Moodle is the University VLE (Virtual Learning Environment). Moodle is used to:

- Communicate course information and urgent messages to all students on a course. As a result **it is essential for all students to log in at least once a day**
- Make course materials, such as lectures, lecture slides/notes, and lab handouts, accessible to students prior to these sessions
- Provide the web links for each Web-Based Assessment (WBA)
- Provide forums for online discussions regarding course material (students are encouraged to use these forums to raise questions about course material that they seek clarity on)

## 1.3 Other Department research systems

***The Department SONA system***

In your full online Part I handbook you can read about the Research Participation Scheme (run through the SONA system). This scheme offers students the opportunity to take part in research studies. The studies cover a range of research topics and are interesting to take part in. They provide an invaluable learning opportunity for students in terms of how to design experiments, how to treat participants in an ethical manner and how to run an experiment. Many research studies in the first term will be carried out by final-year undergraduates collecting data for their research projects. This may well be you in a couple of years time!

Psychology students registered on PSYC121 are strongly encouraged to take part in these studies and collect a minimum of 20 credits* [* this is an estimate at the time of writing - the target will be communicated when confirmed within the department]. There are multiple incentives.

First, completion of these credits is an essential pre-requisite to use the SONA system to recruit participants to your own study when you enter your third year. (nb you are required to carry out your own research project as part of your Psychology degree). This benefit of this access should NOT be underestimated.

Second, once you have collected 20 credits, you will be able to collect additional credits, and earn further project credits for yourself in the third year. But also, taking part in studies, run by other undergraduates, or postgraduates or Research Assistants or staff, is a fantastic way to learn about research methods and experimental designs and psychology in general. Time and again, many students tell us that they got great ideas for projects and for good experiments from their experiences as participants. Being a research participant provides unique insights into what goes on in experiments and what we can learn from them.

You will receive more information about SONA – such as how to register to gain access to studies – in the next couple of weeks.

# Chapter 2

# Week 1: Introduction to Statistics with R Studio

Written by John Towse & Tom Beesley

The exercises in this section are designed to familiarise you with working in R Studio.

## 2.1 Lab Work

### 2.1.1 Introducing R Studio

R and R Studio is the software that we will be using to explore and learn about analysis in your Psychology degree. It's a computational engine: a very snazzy calculator that you should see as your friend and ally in the journey to understand and appreciate psychology. It sits *alongside* what we teach about the concepts and interpretation of statistical analysis.

R is the core software, R Studio is the interface for interacting with it. Put another way, *R is the engine, R Studio is the cockpit.*

Like even a simplest calculator, it just does what you ask (at least when you ask nicely!) but it requires the user to know what they want from it and to understand what it is telling you. A calculator can't help a kid get the right answer to a multiplication problem if they don't know the difference between multiplication and division and addition etc. And whilst a calculator is brilliant at doing the number crunching (and as a bonus, R Studio can help with turning the numbers into beautiful graphs and images too), even a calculator requires

a thoughtful person to take the answers and make sensible interpretations from them.

Therefore, we need to learn both about the concepts of statistical analysis on the one hand, and the processing of statistical information -through R- on the other. The lectures will provide the starting point and the direction for statistical concepts, whilst these analysis labs provide the more practical experiences in how to use R, and how to R your ally. Over the next year, in these labs we will increasingly be using R Studio to focus on the latter, processing side, which will allow you to focus your energies on the conceptual side and its relevance for appreciating psychology.

### 2.1.2 Getting started with R Studio

For Lancaster University Psychology Students in 2021, we will be learning about R Studio through a simple but powerful web server architecture. That is, through the power of the internet, you can access and use R Studio by logging into a free account that we have provided and we will maintain for your use.

> Here's a little secret: There are several different ways to access R Studio. For example, you can download a copy of the software onto your computer, or use a Virtual Machine set up to run a copy. There's nothing to stop you having your local copy, but please note - we can't support your own version through lab classes

So why are we using an R Studio web server?

- Uniformity of experience. It doesn't matter whether you have a PC, a Mac, a Chromebook or whatever. The server provides a consistent, uniform experience. That also means we can spend more time helping you master the software, rather than translating between versions on different platforms
- Ubiquity of experience. There are **lots** of computers on campus. There are computer rooms, there are pods in the learning centre, there are bookable laptops in the Levy Lab, and so on. Providing a web interface from a browser, means you can practice R from all the different machines. Your not limited to a particular copy on a single machine.
- Consistency of performance. Since everyone logs into the same installation of R Studio, we can ensure that things work the way we expect. This allows staff to focus on teaching you the key issues, without worrying about software conflicts because different machines have different configurations, or different versions with different features etc. We'll have more time to focus on the important stuff!

- Quality of performance. Some installations of R Studio and some commercial versions of the server are set up, well, to be the basic version rather than the premium version. To draw a papelle: do you want to have to put up with the cheap, low-bandwidth broadband package? We've got control of the server set up, so we'll try and make this the most effective way to do this.

We've moved to the R Studio server approach this year. We've tried hard to set it up to work well- Apologies in advance for any teething issues, we will try to resolve them as quickly as possible.

You will have received an email with your account information to log onto the R Studio server. Please keep your account details safe.

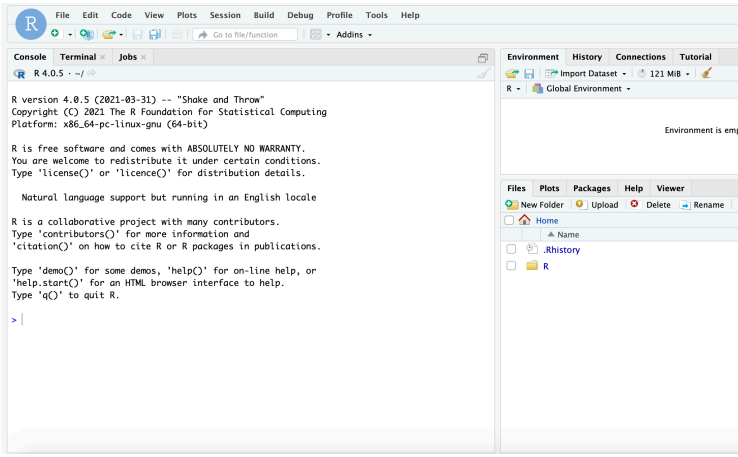
From a computer on the campus wifi, you can access R Studio at:

psy-rstudio.lancaster.ac.uk (off campus, you will need to be on the VPN)

At the login screen, use your university username (e.g., bloggsj)

Your password for R Studio is: [password here]

#### 2.1.2.1 What does it look like?




When RStudio starts, it will look something like this:

RStudio has three panels or windows: there are tabs for Console (taking up the left hand side), Environment (and History top right) , Current file (bottom right). You will also see a 4th window for a script or set of commands you develop, also (on the left hand side).

### 2.1.3 Let's do something!

In the console window on the left hand side, there's a command prompt ">". This is where we ask RStudio to do our bidding!

1. Click in the console window and we will get R to work as a calculator. Type in:

```
5 + 5
```

and press enter. You should get the answer (amazing huh? OK, maybe not *that* amazing...). Use your imagination – ask a simple arithmetic question of your own choosing!

2. In the first analysis lecture, we looked at measures of central tendency and how to calculate them. So let's get R to do these calculations also!

First, we tell R about the data from the lecture. Copy the following line and paste it into the console, then press enter to run it:

```
PSYC121_week_1_data <- c(7,8,8,7,3,1,6,9,3,8)
```

This creates an "object" called `Analysis_week1_data`. We can then perform calculations on this object. For example, we can find the mean by using the following command (again, copy and paste)

```
mean(PSYC121_week_1_data)
```

Check the answer is the same we found in the lecture (it should be 6!).

Next, let's ask for the median:

```
median(PSYC121_week_1_data)
```

This also should be the answer from the lecture (7)

R doesn't have a single corresponding command for the *mode*, but we can use this series of commands:

```
getmode <-
  function(PSYC121_week_1_dataa) {
    uniqv <- unique(PSYC121_week_1_data)
    uniqv[which.max(tabulate(match(PSYC121_week_1_data, uniqv)))]
    }

getmode(PSYC121_week_1_data)
```

This is just a bit of clever jiggery-pokery that gets the mode.

### 2.1.4 Extra content

The commands above are designed to show you that with RStudio active, you can get quick and accurate answers to material covered in PSYC121 lectures.

All we have asked is that you write (or copy in) text to get the information. However, after the lab, you could play around with RStudio in your own time and think about the following:

In R, "<-" is the assignment operator as in the command we used:

```
PSYC121_week_1_data <- c(7,8,8,7,3,1,6,9,3,8)
```

We create the variable label on the left (`Analysis_week1_data`) and we give it those number on the right. The name `Analysis_week1_data` is largely arbitrary: try use a variable of your own naming (your own name?) instead - and then use that alternative name for the other commands.

---

**Throughout this year, we'll use the convention of the "underscore" to separate words in labels (it_makes_them_easier_to_read than ifyoudidn'thaveanyspaces)**

---

What does that tell you about the text used to get the mode? Can you figure out what each line does?

Also, once you have created a variable, you can check what the variable comprises by calling it at the command line. Just write its name, and R will respond with all the data points (all the X values it knows about). Take note that, as you write the variable label, R studio should offer to "auto-complete" the name. It only does that if it you have defined the variable, or are writing a known command.

# Chapter 3

# Week 2: Descriptive statistics in R Studio*

Written by John Towse & Tom Beesley

## 3.1 Pre-lab work

Last week we asked you to

- Practice a WBA task (the math skill task)
- Connect to the R Studio server and enter some simple commands
- Complete a survey so that we can collect data for analysis teaching

This week - we are introducing the `learnr` tutorial system for lab preparation. This will allow you to try out some useful lab skills before your main lab.

**Please make sure you've looked at and followed through the exercises here: https://ma-rconnect.lancs.ac.uk/PSYC121_W2_labprep/**

### 3.1.1 File organisation

We also recommend that you set up your University "H drive" by following the instructions here

If you do this, you can organise your files in one place and access them from any university computer that you log in to, including the lab machines in the Levy Lab and the virtual machine (see above). So this effectively acts as a free, cloud-based storage solution for your work (and even personal stuff).

However you choose to store your files, it's essential that you are careful about file organisation. As you progress through these exercises, you will find yourself downloading many data files, and creating many R scripts.

Consider each week a new set of materials that should have it's own folder. Get used to organising your work and you will be well placed to revise the material at a later date.

### 3.1.2 R Studio tasks

For a reminder of how to start R Studio, see the Week 1 content.

You can access the RStudio Server at: http://psy-rstudio.lancaster.ac.uk/

(remember: off campus, you will need to be on the VPN)

### 3.1.3 Bringing data, scripts and files into R Studio

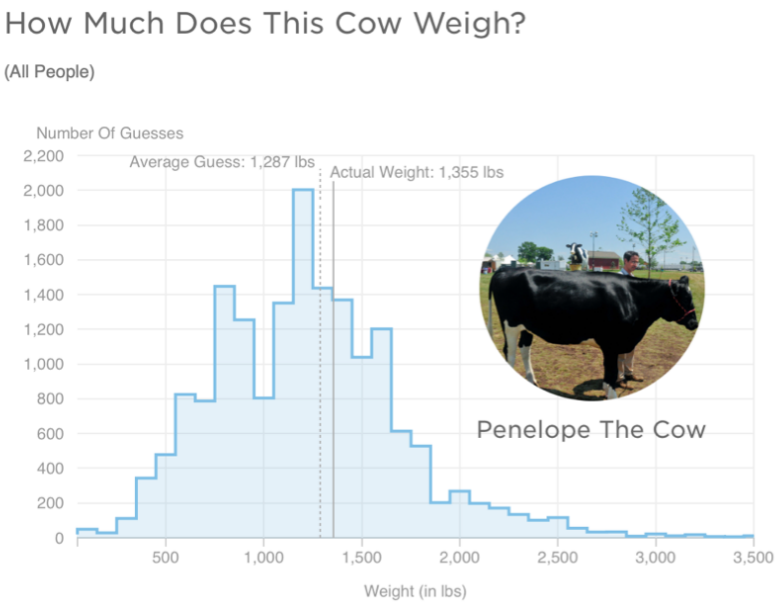
In week 1, we had a tiny dataset that we entered into R through command line. We're going to need a way to tell R about bigger datasets though, so we need to work with datafiles, and bring them into R so that they can form data frames and other forms of data objects in R.

We've provided (on moodle) a walk-though of how to get a data file (in a zip compressed file) into the R environment. Note, once we have imported it into R, it's no longer the original file started with, but within R is has become a named data frame.

## 3.2 Lab exercises

### 3.2.1 Descriptive information in R Studio

Some 5 years ago, a large group of participants gave an estimate of the weight of Penelope the cow. Just over 17,000 guesses. And the distribution of guesses was




something like this:

What we can see from this graph is that:

1. Guesses formed a roughly normal distribution. There is a bit of a skew with a right-hand tail, but this is inevitable as a weight of less than 0 is physically impossible, but there is no limit of the semantics of a large guess.
2. The mean guess weight (1,287 lbs) is very close to the actual (true) weight of the cow (1,355 lbs). So even though lots of people were inaccurate, a central tendency measure has a pretty good alignment with te true weight. This is known as the Wisdom of Crowds phenomenon, first identified by Galton in 1907 (though he suggested using the median weight)

Let's look at (a smaple of) the PSYC121 student data from 2021 collected on guessing the weight of Penelope, and ask whether it resembles the properties of this large dataset.

### 3.2.2 Loading the data

To get the data into R Studio, you need to complete the following steps:

1. Download the "zip" file by clicking this link
2. Find the location of the file on your computer and check it is saved as a ".zip" file
3. Return to RStudio and navigate to the folder you want to upload the data into (you could create a new folder for this week).

4. Click "Upload"
5. Check the target directory is correct
6. Click choose file and find the file on your computer.
7. Select the file and click "Open". Click "OK"

You should now see the files extracted in the directory. If you receive an "unexpected server error" please try this process in a different browser. If you still have trouble, send your username to us t.beesley@lancaster.ac.uk for support.

### 3.2.3 Using the R script

Let's start working with our data, by opening up the "Week_2_script.R" file. It's part of the zip file for week 2. This is an "R script" that contains a series of commands. We could type each of these in the console window (like we did in Week 1), but by having them in a script like this, it saves us typing them or copying and pasting them into the console. It also means we have a record of all the commands we have run.

The first command is to load a library of functions:

```
library(tidyverse)
```

To run this, simply click anywhere on line 1 of the R script to put the cursor there, and press ctrl+enter (cmd+enter on a mac) or click the button called run. You will see a number of messages appear in the console. Don't worry about these, or worry too much about what exactly this command is doing. Essentially this is giving us some useful tools for our analysis. We will introduce the features of the **tidyverse** gradually during this course.

(side note: If you were using a local version of R studio on your computer, it might not have the 'tidyverse' library already installed. You would need to install the package first)

To import the data into RStudio, from the csv file, simply click the "Import Dataset" button in the "Enviornment" pane. John has a video available to follow along with, if you have trouble with this bit.

The data are in R studio if you have followed all the lab sheet to this point. Note that when you imported the data into the R environment, a command line was generated at the console

```
penelope21 <- read.csv("~/penelope21.csv")
```

What this command accomplished was to read the spreadsheet called 'penelope21' into an object in R called penelope21. You could use any object label, but it's important to then keep that name consistent in what you do next.

The command was also generated

```
View(penelope21)
```

which presents the data in a windo of R studio. Note that "NA" means not available or missing data. Does this file structure make some sense to you?

### 3.2.4 Finding the mean and median estimates

Use the data to answer the following questions…

1. What is the mean weight estimates?
2. What is the standard deviation of the estimates?
3. What is the median weight of the estimates?
4. Which of these central tendency measures is the more accurate measure of the true cow weight? (make a judgement)
5. What is the mean weight estimate (and standard deviation) for female respondents and non-female (male / non-binary /prefer not to say) respondents?

You may be thinking, how do I possibly do any of this?! Well this week most of the commands you need are contained in the R script you have downloaded. Also, remember from last week, we explored the R command:

```
mean(lecture1_data)
```

That gave us the mean of the small dataset "lecture1_data". This time, we want to explore the penelope dataset. But also, the lecture_data was just a single list of numbers. The penelope21 object is more like a datasheet. So we need to tell R Studio which **column** we are interested in. RStudio uses the format **data$column**. Hence, we can ask

```
mean(penelope21$estimate)
```

(this command is in the r script so you don't need to write it out) and to get a standard deviation we can use the command:

```
sd(penelope21$estimate)
```

So from this, can you work out what you would do to get the median value (remember from last week how we got the median value?). Part fo the command is given to you, can you change the text so that it works?

### 3.2.5 Calculations from a range of columns

We have seen that:

```
mean(penelope21$estimate)
```

will provide a mean of the column "estimate". In the third column, named "female_estimate", we have the estimates of just the female respondents. In the fourth column, named "other_estimate", we have the estimates of the "other" respondents (males and non-binary and prefer not to say).

So can you now figure out how you might get information about the estimate from the female data (only) or the non-female data? Try it, based on what you have just done. Does it work?

You will find that the result of the this command produces an "NA" result. This means that the answer is "Not Available", or in other words, is a "missing value". This is because some of the values in this column are NA, and the mean of a column with NAs will always lead to the result NA.

Instead, try this command:

```
mean(penelope21$female_estimate, na.rm = TRUE )
```

Any different? The `na.rm = TRUE` instruction tells RStudio that missing data can be ignored in this mean calculation. (in technical language, `na.rm` is a parameter of the function `mean` that removes the NAs if set to TRUE)

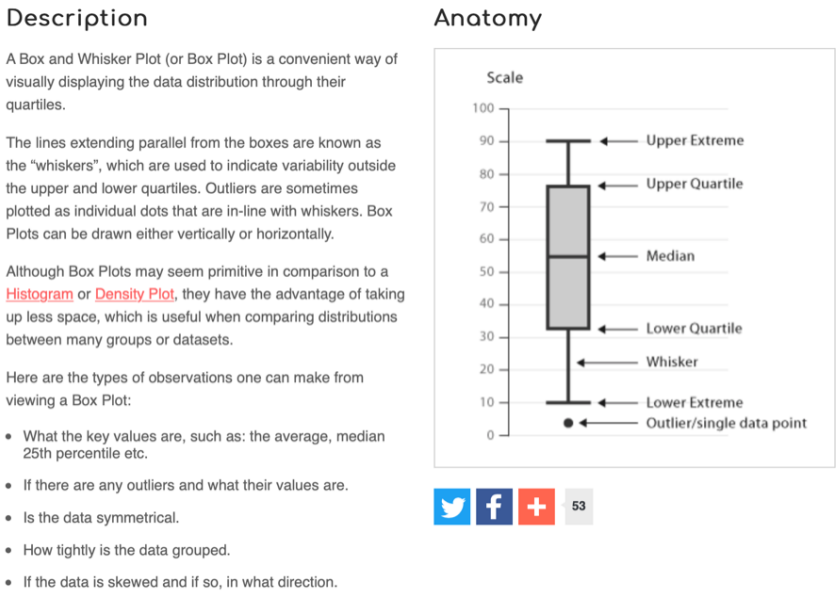
### 3.2.6 Simple graphs

RStudio can be used to create graphical data plots that can help interpret datasets

The first thing we can do is create a histogram distribution of guesses from the sample student data to compare with the previous large sample study (i.e. the 17,000 guesses):

```
hist(penelope21$estimate)
```

We can also create a "box and whisker plot". Here's a general simple description of a box-and-whisker plot as a graphical representation of data:




We can create a box and whisker plot for the estimate column using the following command:

```
boxplot(penelope21$estimate)
```

## 3.3 Extra content!

Want to try explore some more outside of the analysis class before next week and improve your R skillset? Try these exercises:

### 3.3.1 Helping yourself!

If you're not sure how a command works, or some detail of command, then R itself may be able to help you. You can ask for help for example by entering the command

help(mean)

Try get help on other commands we have explored this week. Does the help info converge with your growing knowledge of the commands? Does the help show you some of the additional options with the commands? Can you use the examples which may accompany the help info?

You can also use help() to find out more about packages rather than just commands (see next challenge). And most help info will give some example, but you can ask for examples specifically with example() (e.g. example(mean))

#### 3.3.1.1 Create a box-and-whisker plot of BOTH the female and non-female data on the same figure.

To do so, you can use a really flexible toolkit (in R-speak, called a package) with publisher-quality graphics capabilities called "ggplot2". We turned this on with the command before

```
library(tidyverse)
```

Have a look at the help information about for example, "geom_boxplot", and try out the script information provided for week 2. The challenge is to figure out how the script instructions produce the graph – what do the elements do (maybe fiddle around with it to find out!).

Good luck!

# Chapter 4

# Week 3: DVs and IVs in R Studio

Written by John Towse & Tom Beesley

## 4.1 Pre-lab work

Last week we asked you to

- Complete the teaching dataset survey
- Prepare for the lab class with the learnr tutorial
- Complete some R Studio exercises working with data and a script to get some descriptives and some data

This week - again, there's a learnr tutorial to follow and help prep for this week's activties : https://ma-rconnect.lancs.ac.uk/W3_LabPrep/

About 1/3 of students at the time of writing this had yet to complete the teaching survey. We'd really appreciate more completions so we can work with a good sample of data this term, and as a reminder, the data are here: https://lancasteruni.eu.qualtrics.com/jfe/form/SV_6rLPbl1YI8Y2WXj

Since we've got less data than we would expect, we can't yet use the data in this term's activities.

## 4.2 R Studio tasks

For the "penelope21" data in week 2, we provided you with estimates data, and you were able to generate descriptive statistics for the estimates (if not, please

go back and work through that part of the week 2 lab sheet again). You also found the weight estimates for the female and non-female guesses, right?

However, in order to find the estimates separately for gender identity, we needed to have a column for each gender category. Whilst that worked, it could get cumbersome over time always to work with data created like that.

*There is a better way*

### 4.2.1 Task 1 - penelope21 data

Download the week3.zip file and upload it into R Studio server. If you need them, here are the key instructions from Week 2.

Import the "penelope21" data using the `read.csv()` command (if unsure, see here in Week 2), and launch the R script.

This week, we want to explore commands from the tidyverse library (toolkit) which can help us do more powerful things more elegantly. So let's get R to work with the tidyverse library

```
library(tidyverse)
```

(Nb. This command is the accompanying R script. You can access it from there.)

This time, let's ask for the estimate data arranged by identity:

```
aggregate(x = penelope21$estimate, by = list(penelope21$identity),
FUN = mean)
```

First, let's try this! What do you get? Does this match what we did last week when we calculated the mean for the female and for the other group?

Second, let's look at what is happening here:

`aggregate` This is a command to get descriptive statistics Remember we can use help to find more about this command!

`x=`
This defines what column we are analyzing

`by=list`
Now we tell R how to group the estimate data, and which column does that

`FUN=mean`
Specifies which descriptive function is being asked for Can you explore whether you can call on alternate measures?)

### 4.2.2 group_by()

There's another way that also allows us to group scores by a (nominal) variable. This is explored in the learnr tutorial, which should help you create the command the get weight estimates broken down by gender identity. You need to define the data frame for the estimates data, and the gender IV and the estimates DV

```
*MISSING* %>% group_by(*MISSING*) %>% summarise(mean_estimate =
mean(*MISSING*))
```

First, try this command and see what you get. If you run this command as entered, it won't work. So now use your experience at skills from the above and the learnr tutorial to work out what is required.

Note

`%>%`
This is a "pipe operator", basically take the output from the left and feed it into the requests on the right. `Summarise`
Provide summary statistics information for the specified variable as specified (whether mean, median etc)

### 4.2.3 The assignment operator

As well as learning about the pipe operator, we want to introduce another important element of the R command line syntax: the assignment operator.

Probably without knowing about it, we've been using this already! When a dataset is imported into R Studio from the menu, a command is created such as

penelope21 <- read.csv("~/penelope21.csv")

What this does is look for the csv datafile called 'penelope21', and assign it to an object / variable called 'penelope21'

We could create any object name we wanted (within limits of names already known to R Studio). If we wanted to work with something called "week_3_data" we could use

week_3_data <- read.csv("~/penelope21.csv")

In R Studio, we would call on "week_3_data" rather than "penelope21" in the command lines.

### 4.2.4 Task 2 - Salary20 data

Using aggregate and summarise may not seem like much progress, because they are just replicating what we had already done with mean() is week 2. However

(a) this emphasizes that there are often several ways to get at the same thing in R (b) now we know about grouping, we can start to do more efficient and informative things.

Now, let's turn to the guesses made about median salary in the UK. We can get the data (created in 2020) from the file "salary20" in the week 3.zip file;

Let's take a peek at the dataset with

```
glimpse(salary20)
```

Glimpse pretty much does what you might think from the meaning of the word – it just gives us a data sample (handy because this is a much bigger dataset) and shows that we have 3 columns; UK_region (where someone lives, note 'other' probably equals Northern Ireland, Europe, China, etc, family_position (age relationship with siblings), and median_salary guess.

By the way, the govt statistics say the actual median income in 2019 was approx. £30,350 https://www.statista.com/statistics/1002964/average-full-time-annual-earnings-in-the-uk/

A) Can you use the "aggregate" command from task 1 to find out the salary guesses as a function of where someone lives? That is, can you adapt that code for this problem?
B) Can you use the aggregate command to find out salary guess as a function of family relationships? (if you are the youngest child maybe you have older siblings earning money that changes your evaluation?)
C) Can you get a breakdown of guess as a function of BOTH UK region AND family relationship together?
D) Can you adapt the "summarise" command to display salary guesses as a function of where someone lives?
E) Use R Studio to figure out the overall mean salary estimate and the standard deviation. Calculate by hand what salary estimate would have a z score of z=-1.5?

# Chapter 5

# Week 4: Z-score calculations and DVs/IVs again

Written by John Towse & Tom Beesley

## 5.1 Pre-lab work

This week - again, there's a learnr tutorial to follow and help prep for what we are covering: https://ma-rconnect.lancs.ac.uk/W4_LabPrep/

## 5.2 R Studio tasks

Last week we introduced two different ways to get a variable / column of scores investigated as a function of a separate column of data. In others words, describe the DV a function of an IV

Those two approaches involved

```
aggregate(x = DV, by = list(IV), FUN = mean)
```

and

```
data.frame %>% group_by(IV) %>% summarise(mean_estimate = mean(DV))
```

Use the week 3 learnr tutorial from last week to get more hints and practice with this, if you need to, alongside the week 4 tutorial.

### 5.2.1 Task 1 - Climate change data

1. Download the week4.zip file and upload it into R Studio server. If you need them, here are the key instructions from Week 2. This week the data is called "climate". Launch the week_4 R script as before.

Step 2. Once again, we're gong to be using commands from the tidyverse library (the pipe operator is one example) so we need to ensure that it's active. Run the command:

```
library(tidyverse)
```

(Nb. This command is the accompanying R script. You can access it from there.)

In the climate data frame, there are the following variables:

*human_CC* - A response to the question: How much do you think human activity contributes to climate change, as a percentage of overall climate change? Please enter a value between 0 (humans do not contribute to climate change at all) and 100 (human contribute entirely to climate change).

*UK_region* - where in the UK you used to live

*country_visits* - the number of countries you have visited

*visit_category* - a number between 1-3

- 1 means you have visited 1-4 countries
- 2 means you have visited 5-9 countries
- 3 means you have visited more than 9 countries

Step 3. Maybe belief in human activity as a contributor to climate change varies across the UK? Find out which region of the UK has the highest and the lowest belief in the relative role of human activity in climate change. Change the following command based on what was learnt last week and the current data described above

```
Data_frame_name %>% group_by(IV) %>% summarise(mean_estimate =
mean(DV))
```

Step 4. Maybe views about climate change are affected by how well travelled you are? Because you see the global impacts of clmate change, or because travel (often producing carbon emissions) is a sign of not being worried. So use the climate_change_category to look at whether there are differences in rating of human contribution to climate change.

### 5.2.2 Extra activity.

Come back to this afterwards for some extra practice if you want: Use the learnr activities to think about visualisation of the climate data. What graph(s) would be useful? How can you customise them?

## 5.3 Z-scores

> Hint / Reminder: Sketch a normal (z score) distribution and mark the mean/mode, and mark off the relevant parts of the question so you know what you are trying to achieve and how to interpret any calculations you make.
>
> Hint/ Reminder 2. For questions 6 & 7, remember that from the week 4 lecture material, typically in psychology we use the 5% level as a cutoff to decide, in broadly described terms, whether something is extreme or unlikely vs. at least somewhat plausible or likely.

### 5.3.1 z-scores 1: distributions

Q1. What is the relationship between the sign of a z-score and its position in a distribution?

Q2. If a distribution has a mean of 100 and a standard deviation of 10, what is the raw score equivalent to a z-score of 1.96?

Q3. If a distribution has a mean of 157 and a standard deviation of 19, what is the raw score equivalent to a z-score of 1?

### 5.3.2 z-scores 2: Using z-score tables

Q4. What proportion of scores lie between the mean and a z-score of 0.5?

Q5. What is the combined proportion of scores lying between z=-1.2 and z=.85?

### 5.3.3 z-scores 3: Applying z-scores to inferential problems

Q6. A Neuropsychologist has presented a test of face recognition to 200 neurotypical participants and finds that the scores are normally distributed with a mean of 85 and the standard deviation of 12. Two brain-damaged patients are also given the test. The one with right hemisphere brain damage scored 58 and the one with left hemisphere damage scored 67.

1. What is the z score of the right hemisphere patient when compared to the neurotypical group?
2. What proportion of neurotypical participants score lower that this patient?
3. Is this patient likely to belong to the population of neurotypical participants? (justify your answer)
4. What is the z score of the left hemisphere patient when compared to the neurotypical group?
5. What proportion of neurotypical participants score lower than this patient?
6. Is this patient likely to belong to the population of neurotypical participants? (justify your answer)

### 5.3.4 Extra activity

Come back to this afterwards for some extra practice if you want:

Q7. Tim Bizzley has measured the foot size of men and women and found each to be normally distributed. The men have a mean size of 55 with a standard deviation of 5 and the women a mean of 33 and a standard deviation of 5. Joanna Toes has foolishly measured two individuals but forgotten to note their gender. These have foot sizes of 37 and 47. To which gender is each more likely to belong? What evidence is there for this?

# Chapter 6

# Week 6: Visualising data and binomial tests

Written by Tom Beesley & John Towse

They say a picture paints a thousand words, so in this week's lab we will be learning some fundamental skills in **data visualisation** with the `ggplot()` commands. We will then conduct our first statisitical test to tell whether there is a *real difference* in our data. You must ensure you have watched the lecture and done the pre-lab work before coming to class.

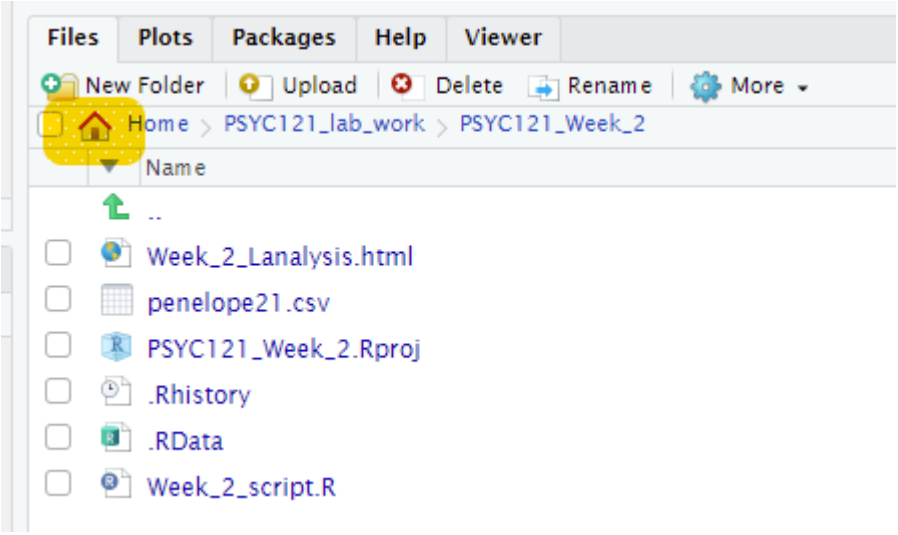
## 6.1 Pre-lab work: online tutorial

To access the **pre-lab tutorial click here** (on campus, or VPN required)

## 6.2 Creating a new folder and project

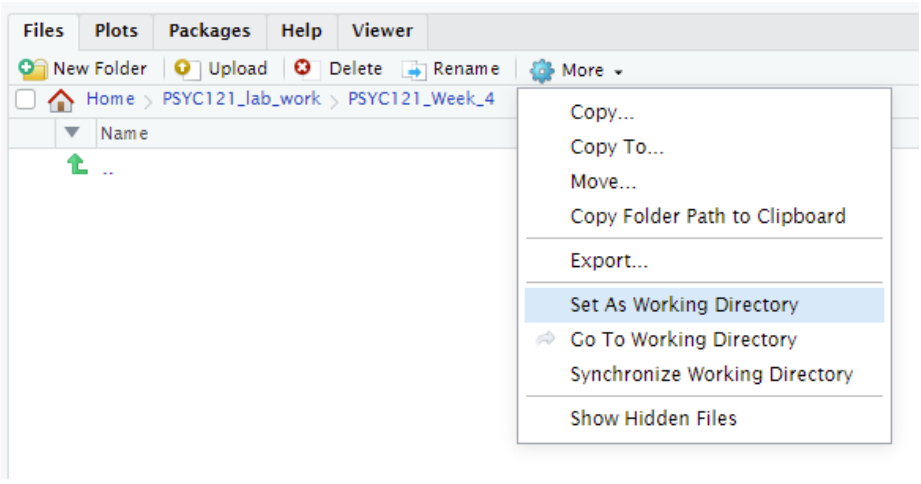
We are going to set up a new folder for this week and an RStudio *Project*. This is a good practice for organising your scripts and data.

1. When you are logged on to the **RStudio Server**, navigate to your home directory by clicking on the small house icon:



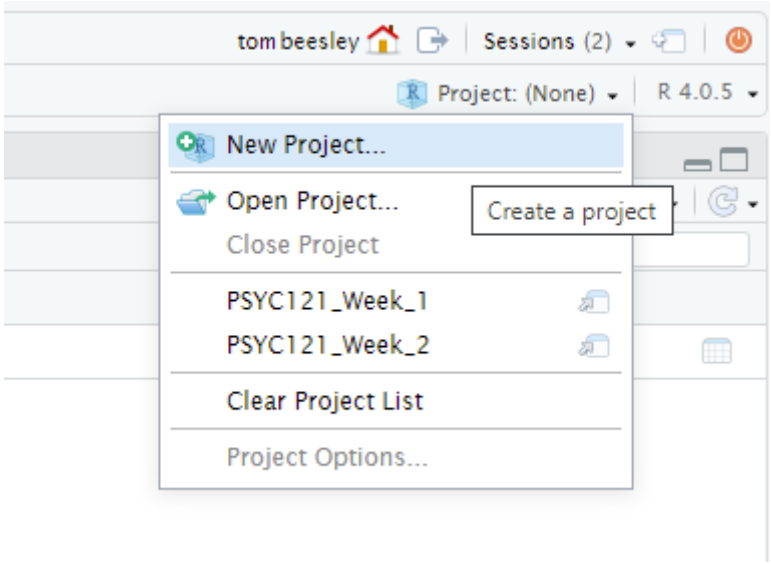

2. Click the "New Folder" button just above the home button. Name the folder something sensible (e.g., Week_6) and click OK.

3. Click on this folder and then click "More" and "Set As Working Directory". This tells RStudio that you are now working in this directory. When you read in data using `read.csv` it will know where to look for the file name that you provide.




4. Let's add the files we need for this week. Download the Week_6.zip and upload it into RStudio Server. If you need them, here are the key instructions from Week 2.

5. Now let's finish this process by making this a *Project*. Click on the small blue icon in the top right of RStudio, and clicking "New Project":




6. You may be asked to save the workspace or data - you should do this. Then select "Existing Directory" and make sure that the new directory you have created is selected as the "Project Working Directory" - it should be, if you set the working directory correctly above. If not, navigate to the correct directory here.

7. And you're done! This should now appear as a project in your front page on RStudio. You can get back to that front page by clicking the red power button in the top right corner. Using projects has many benefits. It will keep all the content for the week in one place, and save the commands you've used in the console. You can also use the *Project Menu* to navigate quickly between different projects.

## 6.3 Visualising phone use

It's time to put the skills you learnt in the online tutorial into practice with a new data set. In the survey, people estimated their daily phone use, and then looked up the actual time their phones were on. This data is provided in the zip file.

1. Open the script "Week_6_script.R". This is the script file you will use for this task and the next one (3).

2. Run the first two lines of code `library(tidyverse)` and `data_w6 <- read_csv("data_phone.csv")`. Note that the `read_csv()` function is another way to import data. A shortcut version of using the "Import Dataset" action.

3. You should now have an object in environment called `data_w6`. You can view this (the "spreadsheet view") by clicking on the object.

4. Try running the ggplot command. You will see that this generates an error in red. Oh no!! Don't panic. Look at what it's telling you is wrong. Let's look at the ggplot command and edit it to make it display something. As a reminder from the online tutorial:

   > The main graphing command is `ggplot()` which tells R we want to draw some kind of plot. If you want to, you can just run `data_w6 %>% ggplot()`, and you'll see that R produces just a blank plot window. The plot is blank because we haven't told it a) what type of plot we want, and b) what data to plot.
   >
   > Note that the next line is **added** to `ggplot()` (i.e., `ggplot() + geom_point()`). Plotting with `ggplot()` involves adding elements (geoms) together. Think of it as adding layers on top of each other: the ggplot() is your pizza base, and the geoms and other features are the toppings!
   >
   > Within the `geom_point()` command we have the very important command of `aes()`. This stands for **aesthetics**. `aes()` tells ggplot how the data should be represented on the features of the plot.

5. Add y and x components inside the `aes()` command. These are used to map variables in your data to the features of the graph. If you are stuck on this, it's time to go back and look at the

6. Inside the `aes()` command, map the variable *screen_time_estimate* to the x axis, and the variable *screen_time_actual* to the y axis. In general, does it look like people's estimates were accurate?

7. Add a setting for *colour* outside of the `aes()`, to make all the points red. Why outside of `aes()`? Well remember from the online tutorial:

   > Careful observation of the code here will show you that these new commands are written **outside** of the `aes()` command. This is really important to note. We write these outside of `aes()` because these features of the plot **are not related to the data**. We are NOT *mapping* any of our data to these features. Instead, we are simply specifying one particular value for each of these "settings" in the plot

8. Now map the *colour* to the variable *phone_type* (within `aes()`). Can you see any differences between people who have different phones?

9. Try changing your `geom_point()` command to `geom_jitter()` (keeping the mappings the same). Suddenly this reveals many more points! Why is this? Check the help to find out more about this geom: `?geom_jitter`.

## 6.4 Conducting a binomial test

Let's now look at whether these differences we see are meaningful. Do people estimate their daily screen time to be more than it actually is? There are two blocks of code to run here. You don't have to edit these, but it's important to look at what they are doing:

1. Run the first block of code. This adds a new column to the data, *est_diff*, which means the "estimate difference": actual screen time minus estimated screen time. You can take a look at this code by clicking on the new object in the environment, *data_phone_diff*. What do the positive and negative values reflect?

2. Run the second block of code. This filters out (!= means "not equal to") the zero values in the est_diff column. Why is this important for the sign test?

3. The result of the final line of code (`count()`) will be output in the console. This tells us how many values were above 0 (TRUE) and how many below zero (FALSE). This is all the information we need to conduct our statistical test:

- How many positive values were there (above 0)?.
- How many observations were there in total?
- Add these two values to the two "MISSING" values in code for the binomial test
- Run the test and read the output - and take note of the "p-value"
- This tells us how likely such a result would be if the null hypothesis was true.
- Does this mean there is a significant over or underestimate in people's estimates of phone time usage?

## 6.5 Plot labels and themes

Visualising data is all about communication: how can I communicate my results so that it makes the data very easy to interpret. Try these additional steps:

1. Add a `labs()` layer to modify text on the graph. If you're not sure how to do this, then feel free to go back to the online tutorial, or take a look at the help wiht `?labs`. Try changing the axis titles and add a title for the graph.
2. Change the theme of the graph. Try `?theme_classic` for help.
3. Graphs can be saved by clicking *Export*, then *Save as image*. Set the dimensions to how you like it to look, give it a suitable filename, and click save.

## 6.6 Answers

When you have completed all of the lab content, you may want to check your answers with our completed version of the script for this week. **Remember**, looking at this script (studying/revising it) does not replace the process of working through the lab activities, trying them out for yourself, getting stuck, asking questions, finding solutions, adding your own comments, etc. **Actively engaging** with the material is the way to learn these analysis skills, not by looking at someone else's completed code…

Download the answers script

# Chapter 7

# Week 7: One-sample t-test on salary estimates

Written by Tom Beesley & John Towse

Today we will look in a bit more detail at people's estimates of the average UK salary. We will first plot this data using `geom_histogram()` and also `geom_boxplot()`. When we do this, we'll see that there are some unusual values, and we'll need to do a bit of **data wrangling** to remove them, using the `filter()` command. We'll then turn to the conceptual ideas of the lecture - how can we tell if the mean of our sample is unusual, or whether we would actually expect this mean value under the null hypothesis? Finally, we'll continue to develop our skills in **data visualisation** by exploring `geom_density()` plots.

## 7.1 Pre-lab work: online tutorial

**Online tutorial**: You must make every attempt to complete this before the lab! To access the **pre-lab tutorial click here** (on campus, or VPN required)

**Getting ready for the lab class**

1. Create a folder and a Project for Week 7. Click here for the instructions from last week if you are unsure.

2. Download the Week_7.zip and upload it into this new folder in RStudio Server. If you need them, here are the instructions from Week 2.

## 7.2 Plotting and filtering

**Important!** - You should be using the Week_7_script as you work through these tasks. Edit the script to complete the tasks. Running code from the script is easy - place your cursor on the line or block of code you want to run and press "run" (of ctrl/cmd+enter). See the video on Moodle if you're unsure. Save your script as you go to keep a record of your work.

1. Open the Week_7_script and run the `library`, `options` and `read_csv` commands. The `options` command is new. It is very cryptic and you don't need to worry too much about this - it is making sure that the values in the graphs are displayed as regular numbers and not as scientific notation.

2. Complete the `geom_histogram()` code to plot the distribution of salary data

3. OK - we've got some pretty funky values here! Some people think the average salary is over £300,000!!! Well, maybe they just added too many zeros (let's give them the benefit of the doubt). Quite often when we get our "raw" data, it contains weird values like this that we need to consider removing. Let's run the `arrange()` code now to see what exactly those high values are.

4. We'll need to remove these high values to get a better sense of the distribution. Let's use a `filter()` command to do this. We need to make a decision about what values to exclude. In later labs we'll look at a more systematic process of removing *outliers*, but for now, let's just remove any that are over £200,000. Edit the `filter()` command to keep only those estimates that are *below* £200,000 (<). Remeber that the filter command *keeps* the data that is *TRUE* according to the expression. Also note that you are making a new object at this step: *data_w7_f* .

5. Now your filter has done its job (check the Environment to make sure *data_w7_f* has fewer rows than *data_w7*), let's plot the data again. Edit the `aes()` command of this next code chunk to draw a new histogram.

6. And as you know, we can also look at the distribution as a boxplot. Edit the `geom_boxplot()` code to do this.

## 7.3 One-sample t-test

We now want to know if the salary estimates are different to the actual average salary in the UK (which is approx. £30,000). Our hypothesis might be that people are inaccurate - they overestimate or underestimate the average UK salary. Let's test that.

7. The first step towards this is to calculate a mean of the column of salary estimates. Edit the line of code with `mean()` to tell R to compute this value. Remember to use the new data object you created after you removed the outliers. If you're struggling to remember how to compute a `mean()`, jump back to the Week 2 content

8. Now we can compute a t-statistic and check its significance with `t.test()`. Edit the code on this line to conduct a one-sample t-test. You need to provide **the sample of data on which you want to conduct the test**, and the **expected mean under the null hypothesis**. Remember our hypothesis is that people are not accurate. Your calculation of the mean should tell you whether they numerically overestimated or underestimated. But would we expect such a result under the null hypothesis? Run the t-test and **note the p value**. How likely is it that we would see this sample of data (this mean value and the distribution of data - the SD) under null hypothesis? The p value ranges from 0 to 1. If it is very low - typically we say $p < .05$ - then we conclude our result is unlikely under the null hypothesis and it is therefore a *significant result*.

9. What is the critical value of t in the t-distribution table, for this sample size? Degrees of freedom is N - 1.

| Degrees of freedom | Significance level | | | | | |
|---|---|---|---|---|---|---|
| | 20% (0.20) | 10% (0.10) | 5% (0.05) | 2% (0.02) | 1% (0.01) | 0.1% (0.001) |
| 1 | 3.078 | 6.314 | 12.706 | 31.821 | 63.657 | 636.619 |
| 2 | 1.886 | 2.920 | 4.303 | 6.965 | 9.925 | 31.598 |
| 3 | 1.638 | 2.353 | 3.182 | 4.541 | 5.841 | 12.941 |
| 4 | 1.533 | 2.132 | 2.776 | 3.747 | 4.604 | 8.610 |
| 5 | 1.476 | 2.015 | 2.571 | 3.365 | 4.032 | 6.859 |
| 6 | 1.440 | 1.943 | 2.447 | 3.143 | 3.707 | 5.959 |
| 7 | 1.415 | 1.895 | 2.365 | 2.998 | 3.499 | 5.405 |
| 8 | 1.397 | 1.860 | 2.306 | 2.896 | 3.355 | 5.041 |
| 9 | 1.383 | 1.833 | 2.262 | 2.821 | 3.250 | 4.781 |
| 10 | 1.372 | 1.812 | 2.228 | 2.764 | 3.169 | 4.587 |
| 11 | 1.363 | 1.796 | 2.201 | 2.718 | 3.106 | 4.437 |
| 12 | 1.356 | 1.782 | 2.179 | 2.681 | 3.055 | 4.318 |
| 13 | 1.350 | 1.771 | 2.160 | 2.650 | 3.012 | 4.221 |
| 14 | 1.345 | 1.761 | 2.145 | 2.624 | 2.977 | 4.140 |
| 15 | 1.341 | 1.753 | 2.131 | 2.602 | 2.947 | 4.073 |
| 16 | 1.337 | 1.746 | 2.120 | 2.583 | 2.921 | 4.015 |
| 17 | 1.333 | 1.740 | 2.110 | 2.567 | 2.898 | 3.965 |
| 18 | 1.330 | 1.734 | 2.101 | 2.552 | 2.878 | 3.922 |
| 19 | 1.328 | 1.729 | 2.093 | 2.539 | 2.861 | 3.883 |
| 20 | 1.325 | 1.725 | 2.086 | 2.528 | 2.845 | 3.850 |
| 21 | 1.323 | 1.721 | 2.080 | 2.518 | 2.831 | 3.819 |
| 22 | 1.321 | 1.717 | 2.074 | 2.508 | 2.819 | 3.792 |
| 23 | 1.319 | 1.714 | 2.069 | 2.500 | 2.807 | 3.767 |
| 24 | 1.318 | 1.711 | 2.064 | 2.492 | 2.797 | 3.745 |
| 25 | 1.316 | 1.708 | 2.060 | 2.485 | 2.787 | 3.725 |
| 26 | 1.315 | 1.706 | 2.056 | 2.479 | 2.779 | 3.707 |
| 27 | 1.314 | 1.703 | 2.052 | 2.473 | 2.771 | 3.690 |
| 28 | 1.313 | 1.701 | 2.048 | 2.467 | 2.763 | 3.674 |
| 29 | 1.311 | 1.699 | 2.043 | 2.462 | 2.756 | 3.659 |
| 30 | 1.310 | 1.697 | 2.042 | 2.457 | 2.750 | 3.646 |
| 40 | 1.303 | 1.684 | 2.021 | 2.423 | 2.704 | 3.551 |
| 60 | 1.296 | 1.671 | 2.000 | 2.390 | 2.660 | 3.460 |
| 120 | 1.289 | 1.658 | 1.980 | 2.158 | 2.617 | 3.373 |
| $\infty$ | 1.282 | 1.645 | 1.960 | 2.326 | 2.576 | 3.291 |

## 7.4 Density plots: salary estimates by home location

11. Draw a `geom_density()` (density plot) by adding an $x$ mapping for the *uk_salary* column inside `aes()`. This plot can be thought of as a smoothed version of the histogram.

12. Now map the *fill* feature to the variable *home_location*. This should create 3 density plots, one for each home location. You might find that it's a bit difficult to see the shape of all three distributions. To make things clearer, add `alpha = .5` **outside** of the `aes()` command in `geom_denisty()`. This should make the plots a bit more transparent. Try playing around with different values of alpha between 0 and 1.

13. Manually change the colours of the columns by adding (+) this code as a layer to your ggplot code: `scale_fill_manual(values = c("darkgreen", "darkblue", "darkred"))`. Take a look at this PDF for more colour options than you'll probably ever want!

14. You can save your graph outside of R by clicking Export -> Save as Image.

## 7.5 Answers

When you have completed all of the lab content, you may want to check your answers with our completed version of the script for this week. **Remember**, looking at this script (studying/revising it) does not replace the process of working through the lab activities, trying them out for yourself, getting stuck, asking questions, finding solutions, adding your own comments, etc. **Actively engaging** with the material is the way to learn these analysis skills, not by looking at someone else's completed code…

Download the answers script

# Chapter 8

# Week 8: Related-samples t-tests, plotting means and SE bars

Written by Tom Beesley & John Towse

Today we will take a look at summarising means and standard errors (SEs) from our data. We will look at how we plot these together on the one graph (using `ggplot()` commands that allow us to share mappings between different geoms0. We will explore our data on the famous "Stroop Task" and we will use a related-samples t-test to examine the differences between the means of our different conditions in this task.

## 8.1 Pre-lab work: online tutorial

**Online tutorial**: You must make every attempt to complete this before the lab! To access the **pre-lab tutorial click here** (on campus, or VPN required)

**Getting ready for the lab class**

1. Create a folder and a Project for Week 8. Click here for the instructions from Week 6 if you are unsure.

2. Download the Week_8.zip file and upload it into this new folder in RStudio Server. If you need them, here are the instructions from Week 2.

## 8.2 Calculating means and SEs

The "Stroop Effect" is a classic demonstration of automaticity of behaviour. Participants have to say the colour a word is printed in, which is an easy task for a "compatible" stimulus like **GREEN**, and a much more difficult task for an "incompatible" stimulus like **BLUE**. We can't help but read the text - it has seemingly become an automatic process.

| Control | Compatible | Incompatible |
|---|---|---|
| dog | red | red |
| chair | yellow | yellow |
| boat | green | green |
| window | blue | blue |
| block | red | red |
| fan | blue | blue |
| wheel | yellow | yellow |
| tray | green | green |
| bottle | blue | blue |
| fence | red | red |

In this task we will calculate the means and standard errors of the means. We will then plot them using ggplot().

1. Open the script "Week_8_script.R" (see prep work)

2. Run the `library` and `read_csv` lines of code. You should see an object in the environment called *data_w8*

3. View the data with view(data). You will see that the data are a little different from the data we have worked with previously. We have an *pID* variable, which gives a unique number for each person. Each person has 3 rows. This is because the different conditions of the Stroop task reflect a **within-subjects variable** (related samples). For data like this it is often useful to have them arranged in what is referred to as "long format", with multiple rows for each response the participant provides. For the current data that means we have a variable called *condition*, which is our IV, and one called *time* which is our DV.

4. Let's look at the distribution of *time* (our DV) as a function of *condition*. Complete the next chunk of code by mapping *x* to *time* and *fill* to *condition* for our `geom_density()` plot. As per last week, you will want to set the *alpha* parameter to something between 0 and 1 - note that this is done OUTSIDE of the `aes()` command.

5. We seem to have some outlier values at both the high and the low ends. It's probably best if we remove data for the whole participant if their *average* time is unusual. To do that, we'll need to create a new column (remember `mutate()`?) to give us these average time values for each participant. Run the next block of code (you don't need to edit this one) to create the new column, then run the next line to `view()` it.

6. Edit the `geom_histogram()` command to plot the values of the *avg_time* column.

7. We now need the filter out the values in our data that we feel are unusual. Like last week, we will do this in a fairly unprincipled manner, by "eyeballing" the data (next week we'll consider something a bit more "scientific"). Complete the filter command so that it removes both the very low values in the *avg_time* column, and also those that are very high. Because you want to filter out low AND high values, you are using an **AND** expression (**&**). You will therefore need to enter in two numerical values, based on your assessment of the histogram produced for Q6. Note that the filtered data is stored as a new object: *data_w8_f* ("f", for "filtered").

## 8.3 Running related samples t-tests

We have seen in our density plots that the reaction times (DV) look different in the three different Stroop conditions (our IV). But now we need to look at whether there are **statistically significant differences** between the means of the three conditions.

To do this, we will first summarise the mean time taken by each condition in the Stroop task. Remember from Week 3 that we can use `group_by()` and `summarise()` to get summary stats (e.g., mean, sd) at each level of the IV. That's what we want to do now:

1. Edit the `group_by()` code to specify the IV and the `summarise()` to calculate the `mean()` of our DV. If you do this correctly, you'll get three values - a mean value for each level (condition) of our IV. Do these means reflect what you would expect in the Stroop task?

2. Next we need to test if these differences between our means are real. To do that, we can run a related samples t-test; remember that the data for each level of the IV in this experiment came from the same person. We

must use a `filter()` to restrict the data to just two levels of the IV: the *condition* column/variable in the data. This is because a related samples t-test looks at the difference between two means (and only two), so the column we use for the t-test needs to have just two levels of the IV (two of the conditions).

3. The filter command is already set up to restrict the data to two of the conditions. Note that the filter uses an "|" symbol, which means "or", because we want the data that **is the same as** (==) one condition OR the other condition.

4. Run the t-test on this selection of data, to compare the means from these two levels of the IV. Is the result significant? Note the t-value and the p-value. How would you express this as a statement in a report?

5. With 3 levels to the IV *condition* there are 3 possible comparisons we can make (1 vs. 2; 1 vs. 3; 2 vs. 3). Complete all three tests, by copying and pasting the commands, editing each to make a different filter selection, and then to run the t-test. Make sure you interpret the results of the t-tests.

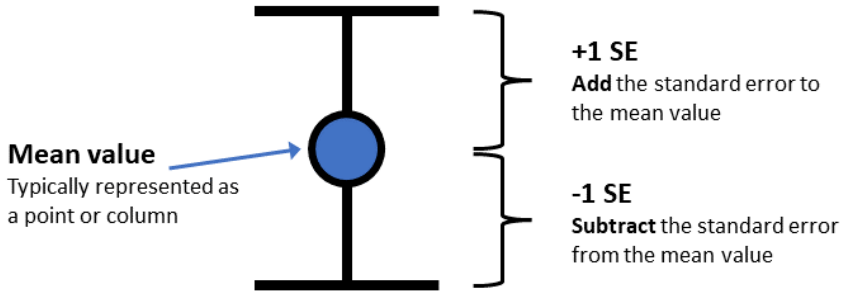
## 8.4 Plotting the means and SEs

1. In Task 2 you calculated the means for each condition in the Stroop task. We've seen in lectures that "standard error" provides an estimate of how variable that mean will be across the samples we collect. In the online tutorial you were shown the code for creating the SE. Complete the code to add in the relevant variables to calculate both means and SEs. The code from Task 2-Q1 will give the mean; you simply need to add the correct variable (DV) to the `sd()` command to calculate the SE values (note that you don't need to put anything in `n()`, as this simply calculates how many rows there are).

2. View the new object *data_w8_summary*. Check that the means and SEs are different for the 3 conditions. If they are the same, you probably summarised the wrong column!

3. Now complete the ggplot command to first plot our DV, the *stroop_mean* (y), as a function of the IV, *condition* (x).

4. Now we need to add some "error bars" which provide a visual guide as to how much variance there is in our data (or how much uncertainty we have in our mean value). This figure illustrates how we would typically present error bars with mean values.




5. Edit the code for the `ggplot()` command that plots both `geom_col()` and `geom_errorbar`. You will need to calculate a `ymin` and a `ymax` value. Use the illustration above to work out how to combine the mean value and the SE value to create the right `ymin` and `ymax`.

EXTRA: These next steps can be completed to practice customising your plot

6. Add a `labs()` layer to the plot to change the axis titles, and the title of the plot.
7. Change the theme of the plot (e.g., `theme_void()`)
8. Map the *fill* aesthetic to the variable *stroop_condition*. You can do this for `geom_col` or `geom_errorbar` or both at once by putting it in the `aes()` within the `ggplot()` command.
9. Manually change the colours of the columns with using `scale_fill_manual()`. Take a look at the Week 7 instructions on how to do this.

## 8.5 Saving your work

**Scripts:** By now you are hopefully getting used to editing and working within the script (if not, see the video on working in the script vs. console on Moodle). To save a script, you simply click the save icon, or press ctrl+S (cmd+s on a mac).

**Plots:** To save a graph you have produced, click the "Export" button in the plot window, then "Save as Image". You can resize the graph and give it an appropriate filename. If you've set the working directory correctly, then the new file should appear in the current folder.

**Data:** The data objects you create (in the Environment) will be saved automatically if you have created a project. However, this data only exists within

RStudio. What if you want to use the data elsewhere? For example you may want to share the data with a project supervisor. To do this, we need to write the data to a csv file (like those we use to import the data). You can do this with the following command: `write_csv(the_data_object, "the_filename.csv")`

**Exporting from RStudio:** The above save operations save files to a folder within RStudio Server. At some stage you will need to get these files out of RStudio Server, for example if you need a graph for your report, or you need to share the data or the scripts. To do this, simply select the files you want in the Files pane, click "More" and then "Export". Selecting multiple files will produce a ".zip" file, which will need to be "unzipped" on your computer to access the individual files (instructions for Windows and instructions for Mac)

## 8.6 Answers

When you have completed all of the lab content, you may want to check your answers with our completed version of the script for this week. **Remember**, looking at this script (studying/revising it) does not replace the process of working through the lab activities, trying them out for yourself, getting stuck, asking questions, finding solutions, adding your own comments, etc. **Actively engaging** with the material is the way to learn these analysis skills, not by looking at someone else's completed code…

Download the answers script

# Chapter 9

# Week 9: Unrelated-samples t-test and Power

Written by Tom Beesley & John Towse

## 9.1 Pre-lab work: online tutorial

**Online tutorial**: You must make every attempt to complete this before the lab! To access the **pre-lab tutorial click here** (on campus, or VPN required)

**Getting ready for the lab class**

1. Create a folder and a Project for Week 9. Click here for the instructions from Week 6 if you are unsure.
2. Download the Week_9.zip file and upload it into this new folder in RStudio Server. If you need them, here are the instructions from Week 2.

## 9.2 Data visualisation and cleaning

In this class we will be exploring some data on music preferences and social media use. In our survey we asked people to rate how much they liked various music artists out of 5. There were some "pop" artists like Beyonce and Taylor Swift, and some "rock" artists like Coldplay and Nirvana. In the data we have grouped the ratings for pop artists together, and similarly for the rock artists: each respondent therefore gets a *pop_score* and a *rock_score*. We've also turned these ratings into a *categorial variable*, which reflects whether someone liked pop

more than rock ("pop-tastic"), or not ("rock-on" - either they preferred rock, or they had no preference).

The other variables relate to social media use. We have the *instagram_followers* and *facebook_friends*, which provide a (fairly crude) measure of how much someone engages with these social media platforms.

1. Let's first take a look at the data on music preferences. We might think that people who like rock music are less likely to be in to pop music (and vice versa). To look at this *relationship* we can plot one score on the x axis and one on the y axis. Read in the data (make sure it's called "data_w9") and complete the `geom_jitter()` code to do this (we use `geom_jitter()` here because there are overlapping points).

2. Consider the pattern of data you see. In general, do people who really like pop (high scores) like or dislike rock? Conversely, do those who like rock dislike pop? What pattern would you expect for these different relationships in preference? Talk about it with your table.

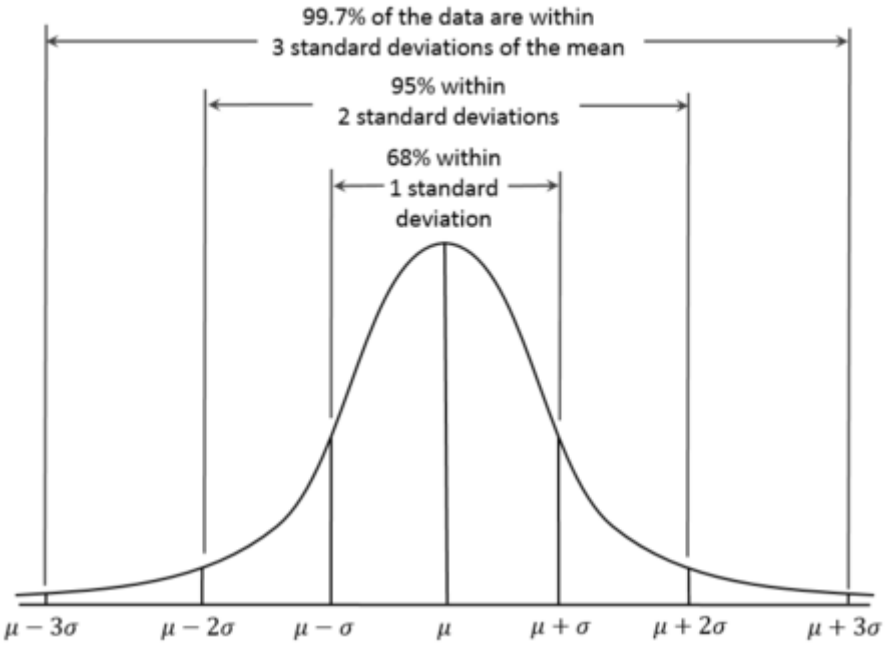
3. Next you can add a mapping between *music_pref* and *colour* to show our boundaries for the two levels of the categorical variable

4. Let's take a look at the two social media variables. Complete the code for the two histogram plots to view the data on facebook friends and instagram followers.

5. You'll see that for both variables, there are some extreme values (some people have >1000 friends and > 2000 followers). These outlier values can be problematic when we run our statistical tests, so (like last week) we probably want to control their influence by removing them. As you saw in your online tutorial, we can convert the data to z scores, and then remove z values above and below certain values.

6. Let's create two "z-transform columns". Complete the code by adding the two variable (column) names to the code. View the *data_w9* object to check these have been created correctly. Like in the online tutorial, it would be a good idea to calculate some descriptive statistics for these new columns to check them (e.g., `mean()`, `sd()`, `min()` and `max()`).

7. We know from our lectures on the z distribution that values of greater than 2 (or less than -2) reflect around 5% of the distribution, and values greater than 3 (or less than -3) represent less than 1% of the distribution:




8. Let's consider an outlier any value that has a z of 2.5 (a conventional cutoff). Complete the filter command to remove the data where the z value is greater than 2.5 for both the *z_FF* column and the *z_IF* column. Note you only need to look at positive values: our histogram has already shown that these data are positively skewed with outliers only at the positive end (you cannot have fewer than 0 friends/followers; these are ratio scales with a true 0 value). Complete the filter command to remove the positive z values above 2.5. When completing this, note the use of the AND (&) symbol: We want to KEEP z values that are below 2.5 for both the *z_FF* and *z_IF* variables.

## 9.3 Unrelated samples t-test

9. Let's have a look at whether people's music preferences relate to their social media use. To do this, let's use our category variable *music_pref*. Complete the `group_by()` and `summarise()` commands to give the mean values of *facebook_friends* (meanFF) and *instagram_followers* (meanIF) for each level of the *music_pref* variable. You don't need to edit the `N = n()` line - this provides the number of participants at each level of the *music_pref* variable

10. What do the means suggest? Are those rock-grandads stuck in the last decade on Facebook? Are the pop-divas insta-addicts?

11. Let's test if these differences are real. First, note that most people liked pop music more than rock music (see the $N$ column in the summary). We have unequal sample sizes, and potentially unequal variances. Run the `var.test()` code to check if the variances of the two samples are similar (homogeneity of variance). If this test produces a p value less that .05, then the variances in the two samples are unequal. That will have consequences for how we run the `t-test()` in the next step.

12. Now let's run the t-test. This week we are comparing data from different samples of participants (those who preferred pop and those who preferred rock). We need to tell the t-test that the data are NOT paired (`paired = FALSE`). The result of the `var.test()` in the last step will tell you whether the `var.equal` value should be `TRUE` or `FALSE`. Set `var.equal = FALSE` or `var.equal = TRUE` depending on whether the variances are equal. When you're happy with the values, run the t-test. What result did you get and what does this mean? Discuss this with your table, or with the staff in the lab.

13. In that t-test we looked at the *facebook_friends* variable, but what about the *instagram_followers* variable? Copy the code to run another `var.test()` and `t.test()`, for the *instagram_followers* variable. Note the t and p values; What do these tell us about the relationship between music preference and social media use?

## 9.4 Power and effect size (d) calculations

14. We saw in last week's lab tasks that there was a significant effect in our Stroop task data: participants were faster to say the colour names of the compatible list compared to the incompatible list (there were significant differences with the control list too). We will now use these data to calculate an **effect size** (*Cohen's d*) for the t-statistic that we observed in that test.

15. Import the stroop data. It's always a good idea to `view()` it (you can do this by clicking on it in the environment) to remind yourself what it looks like.

16. Run the code that will `filter()` the data to the two conditions we are interested in (compatible and incompatible).

17. Run the `cohens_d()` code to calculate the effect size, which is reported as *effsize*. You can ignore any negative sign, taking note of the absolute value.

18. So we know that this large effect size was significant with our fairly large sample of participants. What might we have expected with a much smaller

sample size. Use the `pwr.t.test()` function to add in the effect size (d) and an $N$ of 20. What **power** would we have achieved with this sample size, to detect this large effect?

19. Let's say we wanted our next experiment to have an 80% chance of finding an effect at least as large as the one we found. Complete the `pwr.t.test()` function to work out the minimum sample size we would need to achieve power of .8, with this effect size.

20. Let's say we are looking to run a new experiment in which we give people a stressful task to complete simultaneously. We will ask them to put their hands in a bucket of ice cold water while doing the Stroop task. We are unsure of what consequence this will have for our effect size, but we want to estimate the effect size that could be achieved. We decide to run 40 participants, and want to achieve a power of .90 (90% chance to find an effect at least this large). What is the minimum effect size we could hope to detect under these conditions?

## 9.5 Answers

When you have completed all of the lab content, you may want to check your answers with our completed version of the script for this week. **Remember**, looking at this script (studying/revising it) does not replace the process of working through the lab activities, trying them out for yourself, getting stuck, asking questions, finding solutions, adding your own comments, etc. **Actively engaging** with the material is the way to learn these analysis skills, not by looking at someone else's completed code…

Download the answers script