Statistics for Psychologists

John Towse, Tom Beesley

2022-12-07

Contents

Chapter 1

Week 1 - Introduction to PSYC121

Written by John Towse & Tom Beesley

1.1 Analysis labs and 'pre-lab' activity

For each lab session that you have been scheduled to attend, please come "prepared'. By this we mean

- You have watched the lecture video, made notes, and developed questions where you have them.
- You have worked through the online "prelab" preparation materials - Here is the one for Week 1
- You read the lab sheet (this one) to get a sense of what we're going to be doing, and you anticipate potential problems so that you can focus on these in the session.

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.

Want additional support? Keep in mind that the Department has endorsed and will use the Statistics textbook by David Howell called "**Fundamental statistics for the behavioral sciences"**

The book covers principles of statistics as well as some tutorials on using R. You can access a library copy from the library pages

1.2 Activities for this week

1.3 Task 1 - check-in with the University attendance register

When you arrive, make sure you have checked-in to your Analysis session in the Levy lab. All students are required by the University to confirm attendance at taught session

Staff will remind you of this in your class.

1.4 Task 2 - Getting dicy

Here's a simple task for you to complete as a group around each of the workstations;

You will be given a pair of dice

1. Working in pairs, one person rolls both dice.
2. Add up the total on each of them and have someone record that total (if you don't have some spare paper or a pen, use your computer)
3. Repeat those steps 20 times.
4. Then swap over your roles (the person rolling the dice, the person recording the outcome)

5. Once everyone at the workstation has had a turn at this, each person should attempt to work out (a) the mean and (b) the median of their dice roll total.
6. Check each others working, and discuss any differences or problems you have.

Are all your answers the same? Why / why not? If not, are they very different or very similar?

1.5 Task 3 - Using RStudio

1.5.1 Introducing R Studio

R and RStudio 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, RStudio is the interface for interacting with it. Put another way, *R is the engine, RStudio 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 make R your ally. Over the next year, in these labs we will increasingly be using RStudio 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.

1.5.2 Getting started with RStudio

For Lancaster University Psychology Students in 2022, 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 RStudio. 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. We're using the web server to make sure everyone has the same, controlled experience.

You will have received an email with your account information to log onto From a computer on the campus wifi, you can access R Studio at:

Lancaster Psychology R Studio Server

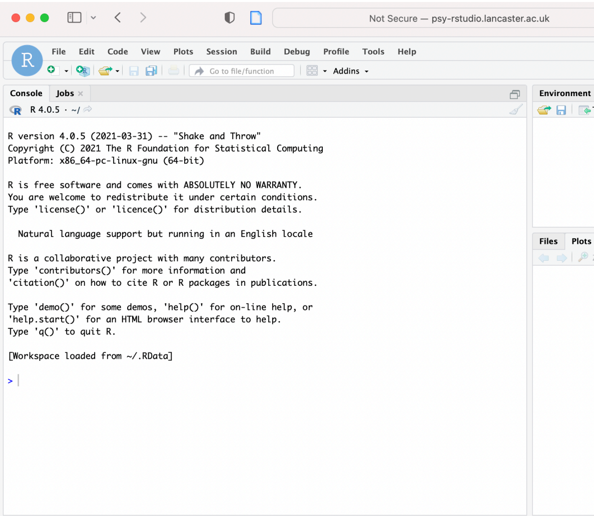
At the login screen, use your university username (e.g., bloggsj)

Your password for R Studio is: [password here]

As it says in the subject line, please keep your account details safe!

1.5.3 What does RStudio look like?

When RStudio starts, it will look something like this:



```
R version 4.0.5 (2021-03-31) -- "Shake and Throw"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> 
```

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).

1.5.4 Let's get started!

The first thing we want to do in RStudio is to create a folder for this week so that we can put the relevant material there and keep it tidy.

From the lower-right panel of RStudio, click the files tab.

Select the "new folder" option and create a new folder (eg "week 1")

Click on that folder to open it

Next, we've prepared some *instructions* for RStudio to use - this is called a "script". So we need to get this script into the server for you to explore and play with

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
4. Click "Upload"
5. Click choose file and find the file on your computer.
6. 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 t.beesley@lancaster.ac.uk for support.

You should now have the script available in RStudio.

Use "Save…As" to create a new version of the script. By doing this, you'll be able to have a "before" and "after" version of the script and can go back over the changes

In the script, select or highlight the first line of text, which is this one:

1.5.4.1 Run your first ever R instruction!

```
5 + 5
```

and "run" this line. That tells RStudio to carry out the instruction.

You should see that in the console tab, RStudio calculates the answer to this incredibly hard maths challenge! (amazing huh? OK, maybe not *that* amazing…).

1.5.4.2 Modify your first ever R instruction!

Use your imagination – add a new line to the script and ask a different simple arithmetic question of your own choosing! What happens?

1.5.4.3 Calculate descriptive stats in R for the first time!

In this week's 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 used in the lecture. We've already created the instruction that will do exacly this and it is in the script, so run this line from the script

```
week_1_lecture_data <- c(7,8,8,7,3,1,6,9,3,8)
```

This creates an "object" called `week_1_lecture_data`. We can then perform calculations on this object. For example, we can find the mean by running the following command (use the script to do this)

```
mean(week_1_lecture_data)
```

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

Next, let's ask for the median by running this line from the script:

```
median(week_1_lecture_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 the block of code in the script for this that starts and ends with the "getmode" text (there are 6 lines of text)

This is just a bit of clever jiggery-pokery that gets the mode. What does R say the mode is?

1.5.4.4 Your challenge

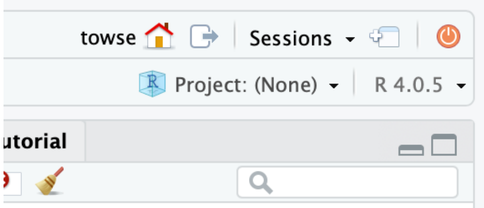
How can you get RStudio to verify / check the dice calculations that you attempted earlier? Think about how you might solve this problem, on the basis of what we have covered so far.

We will discuss this in class and attempt to get RStudio to check your answers. In doing so, annotate the script (add notes for you - not RStudio) using the "#" command

1.6 Before you finish

a) Make sure you save a copy of the script that you have been working on by the end of the session. This provides you with the record - the digital trace - on what you have done. And it means you can come back and repeat any of the work you have performed.

b) End your session on the RStudio server, this logs you out of the server and stops any ongoing activities and tasks you have set up, maybe in the background.



There is a red "power" button near the top right of the R studio window (do ask for help if you can't find it). It's a good habit to get into to turn the session off

1.6.1 Extra content for outside the lab class

a) In the Howell text book on statistics, there's some R code on descriptive statistics. It is included in the script for you to look at and play with.

b) 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 numbers on the right. The nameAnalysis_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)

1.7 Task 4 – Review the learnr sample / practice questions

After every block of teaching in part-1 analysis (specifically, we mean in week 5, week 10, week 15 and week 20) there will be a class test. This will assess your knowledge and your understanding of the material that has been covered.

The class test will comprise a set of Multiple Choice Questions (and the set of questions will be different for each student, as the test will involve random selection from a larger pool) under timed conditions.

In order to help you get (a) a broad or basic feel for the sort of questions you might get in the class test (b) self-review your progress through the term, we will provide MCQs each week for you to attempt.

So these are for your benefit… you can take the questions when you choose to, and the learnr quiz will provide feedback on the answers your provide. Just bear in mind:

a) We place a set of questions at the end of the learnr pages so that you can attempt these at the end of each week, after you've completed lab activities, follow-up work, weekly Q&As etc. But it's up to you when you answer the questions

b) These are meant as *indicative questions*. There's no point in learning/ memorising these questions (they won't be on the quiz!) and our advice is to reflect on how the teaching and content links to the sorts of questions that get posed.

1.8 Task 5 – Data collection exercise

In order to learn about psychology and data analysis techniques, we need data! Rather than rely too much on artificial data (certainly it is sometimes useful to say "Here are a bunch of numbers and this is what we can do with them" – think about the R Studio example for this week's lab) for the most part, we prefer to draw on datasets that are a bit more engaging and meaningful that you have a stake in yourself! By using a common data set, that we can return to over the year, we can also build up familiarity and confidence in the data and remove a potential obstacle to thinking about the more important analysis part.

So a key task will be for everyone to have a go at taking our online survey, and contribute to a dataset that can be used throughout the year.

We would like you to complete the survey via your Department Sona system account. This way, you will receive a research credit for doing so, to "jump start" your credit account.

The sona system is can be accessed by following this link

Chapter 2

Week 2: Descriptive statistics in RStudio

2.1 Pre-lab work

Last week, among other things, we asked you to

- Roll some dice, carry out some (relatively) straightforward 'hand' calculations of central tendency
- Connect to the RStudio server and create a folder
- Within RStudio upload and run a script - a set of instructions, and explore annotations
- Adapt the script commands to perform calculations on the dice rolls within RStudio
- Complete a survey so that we can collect data for analysis teaching

Your progress was great! We start with small steps and build up - but this is a nice start and we're pleased how things went!

Before the lab, make sure you have worked through the material in the week2 learnr tutorial. The link is here

2.1.1 R Studio tasks

For a reminder of how to start RStudio, see week 1 lab sheet

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

> A word of advice (from David Howell's statistics book: "One more word of advice"I can't resist adding what is perhaps the best advice I have. If there is something that you don't understand, just remember that "Google is your friend." She certainly is mine. (Well, maybe Google is getting a bit pushy, but there are many other search sites.) If you don't understand what Fisher's Exact Test is, or you don't like my explanation, go to Google and type in Fisher's Exact Test. I just did that and had 260,000 hits. You can't tell me that there isn't going to be something useful in there.)"

2.1.2 Bringing data, scripts and files into R Studio

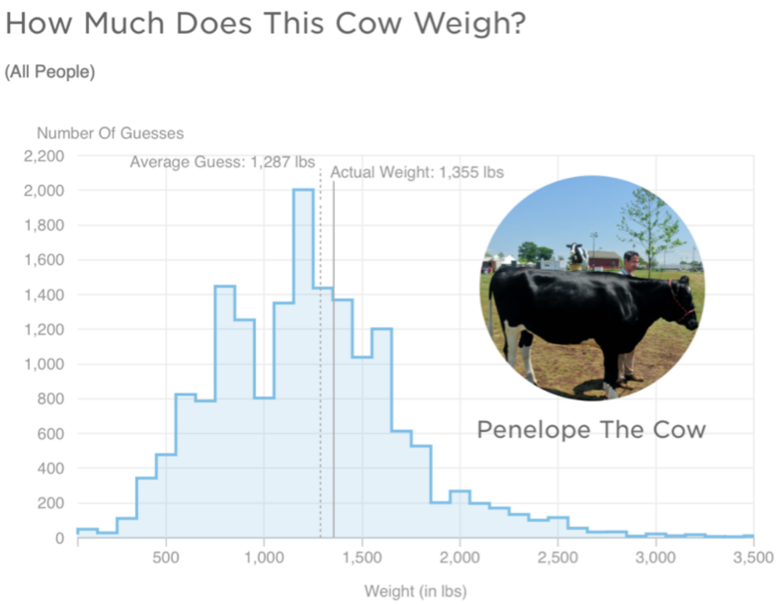
In week 1, we had a tiny dataset (relatively speaking) that we entered into R through a script line. That worked for what is was. But it's going to become painful and tedious when (a) we want to work with larger datasets (b) we have data more complex than a 1-dimensional list of numbers (think about some 2-dimensional data sheets you might have encountered in excel for example)

R can handle data files, and this week we're going to explore them. Within R, we can specify 'data frames' which can have, essentially, multiple columns of data, and we can link data files to data frames for processing

To make things straightforward, each week we'll provide students with a "zip" file that contains the script to start from (which you can expand and annotate etc, and save on your file space). We'll also provide a data file or data files for you to use in the zip file. R can then *import* these files into the RStudio environment. So when you upload the zip file, you can import the data AND you can open up the script

2.2 Lab exercises - descriptive information in R Studio

Some 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 the 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 sample of) the PSYC121 student data collected on guessing the weight of Penelope, and ask whether it resembles the properties of this large dataset.

2.2.1 Loading the data

Using the instructions and advice given on Moodle, get the `week2.zip` file and bring the data and R script into R Studio. The week 2 zip file is here

2.2.2 Using the R script

Step 1. Download the `week_2.zip` file (If you are using a mac, there is a video guide on Moodle to explain how to download the zip file as is)

Step 2. Open the appropriate folder on the RStudio server

Step 3. Upload the zip file

Let's start working with our data, by opening up (clicking on) the script "Week_2.R" file.

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)

The data are on the RStudio server 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

```
cows <- read_csv("~/penelope22.csv")
```

What this command accomplished was to read the spreadsheet called 'penelope22' into an object in R called `cows`. 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(cows)
```

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

2.2.3 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(week_1_lecture_dataa)
```

That gave us the mean of the small dataset `week_1_lecture_data`. This time, we want to explore the penelope dataset. But also, the lecture_data was just a single list of numbers. The penelope22 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**. So run the followinbg line in the script

```
mean(cows$estimate)
```

```
sd(cows$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?

2.2.4 Calculations from a range of columns

We have seen that:

```
mean(cows$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 change the script so it looks like this:

```
mean(cows$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)

2.2.5 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):

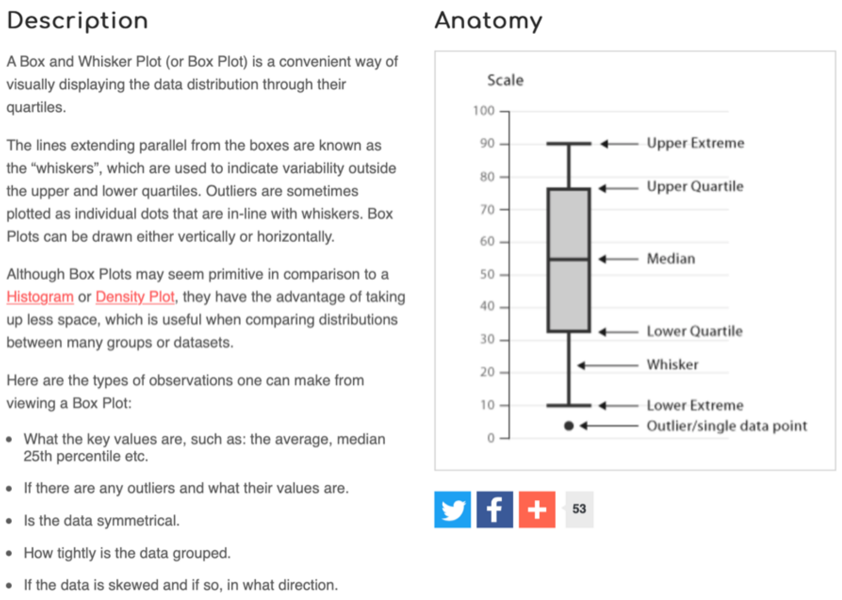
```
hist(cows$estimate)
```

One way to alter or adjust the histogram is to change the width of the bars, the intervals, between each plot section. Try run this line from the script

```
hist(cows$estimate, breaks = ??)
```

Does it work? No? What you need to do is replace the two question marks in the script (or better still, create a new instruction line in which you amend this to have a numerical value representing the number of different plot bars. Try at least 3 different values. Look at and think about how this affects the visual distribution.

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:



Description

A Box and Whisker Plot (or Box Plot) is a convenient way of visually displaying the data distribution through their quartiles.

The lines extending parallel from the boxes are known as the "whiskers", which are used to indicate variability outside the upper and lower quartiles. Outliers are sometimes plotted as individual dots that are in-line with whiskers. Box Plots can be drawn either vertically or horizontally.

Although Box Plots may seem primitive in comparison to a Histogram or Density Plot, they have the advantage of taking up less space, which is useful when comparing distributions between many groups or datasets.

Here are the types of observations one can make from viewing a Box Plot:

- What the key values are, such as: the average, median 25th percentile etc.
- If there are any outliers and what their values are.
- Is the data symmetrical.
- How tightly is the data grouped.
- If the data is skewed and if so, in what direction.

Anatomy

2.2.6 Additional challenge

In the zip file, we also provide data on the estimates of the percentage of immigrants in the UK. This will allow you to explore the data discussed in the analysis lecture, and create visualisations of the data and its spread. Can you apply the analysis of the penelope data to the immigration data?

Chapter 3

Week 3: DVs and IVs in RStudio

Written by John Towse & Tom Beesley

3.1 Pre-lab work

Last week we asked you to

- Use a script to run instructions in RStudio
- Put data into RStudio form a data file and explore how to run descriptive stats

This week - again, there's a learnr tutorial to follow and help prep for this week's activities You can find it here

3.2 R Studio tasks

For the "penelope22" 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...

3.2.1 Task 1 - More with the penelope22 data

Step 1. This week, we again 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)
```

Step 2. Explore help() commands. R can give you more information about how it works.

Step 3. Creating a project and a folder, and setting the working directory

Step 4. Bring the `week3.zip` file into R Studio server. Like last week, upload the zip file, and launch the R script.You can get the file here

Step 5. Read or load the penelope data into R. Have a look at it using the View() command in the script

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

```
aggregate(x = *MISSING*2$estimate, by = list(*MISSING*$identity),
FUN = mean)
```

First, let's try this (you will need to use your object name in place of MISSING). What do you get? Does this match what we did last week when we calculated the mean for the *female* and for the *other* (i.e., non-female) group?

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

`aggregate` This is a command to call for descriptive statistics

`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?)

3.2.1.1 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 called 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)

3.2.1.2 The assignment operator

As well as learning about the pipe operator, we want to remind you /draw attention explicitly to another important element of the R command line syntax: the assignment operator. Uing a command such as

```
cows <- read_csv("penelope22.csv")
```

looks for the csv datafile called 'penelope22', and assign it to an object or variable called 'cows'

We could create any object name we wanted (within limits of names already known to RStudio). It isn't just for reading in data files, we can perform a whole range of functions and assign them to an object.

3.2.1.3 Task 2 - Salary 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, about working with 2-dimensional data, 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 from the file "wages" in the week_3.zip file (you will need to adapt the code in the script for the penelope data so that it will load in the wages data)

Let's take a peek at the dataset with

```
glimpse(wages)
```

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 salary (estimate).

By the way, the govt statistics say the actual median income in 2019 was approx. £30,350 see this link

A) Writing into your script, use the working “aggregate” commands from task 1 with the penelope weight data to find out the salary guesses as a function of where someone lives? That is, can you adapt that code for this problem? First, make sure you read in the data file into an R object.

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 use the `group by()` command to display salary guesses as a function of where someone lives? Check this this gives you the same answer

3.2.2 Task 3 - Phone Use data

E) Datset 3: Use the dataset of phone screen time usage to further explore and consolidate the group_by() command (ie we’ll drop the aggregate command for this task to focus on group_by(). Adapt script lines from the above two tasks to read in and calculate screen time usage as a function of the type of phone.

F) Use RStudio 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?

3.2.3 Task 4 - Final challenge: visualisation

Can you find a way to visualise the screentime usage data that you have been working with above? The script provides two ways to consider doing this - boxplots (which we have looked at in script commands already) and ggplot, which we have spent less time with but is an extremely powerful engine for creating plots. We’ve provided the start of the code in each case, leaving you to work out the specifics. Remember to annotate your creations!

3.2.4 Now you’re finished ...

Remember to complete your notes in the script, save the script file, and end the server session.

Chapter 4

Week 4: Customisation of graphs, and z-scores

Written by John Towse & Tom Beesley

4.1 Pre-lab work

- Complete materials from sessions in previous week (materials from week 1 /2 / 3 remain fully available for anyone who wants them). Consolidate what we have already covered.
- This week - again, there's a learnr tutorial to follow and help prep for what we are covering: You can find it here.
- Make sure you have access to the `week4.zip` file for the RStudio server.You can get the file here.
- If you create a folder and upload the file into RStudio before the lab class you'll be even more ready to follow along!

4.2 R Studio tasks

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

Students were generally very good at utilising each of these;

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

and

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

[tibble = technical name for the data in R]}

This week, we're focusing on how you can edit or customise a graph to be more useful to a viewer.

4.3 Section 1 - Customisation of data plots

Step 1. Set up a folder for this week in the R Project that you created last week.

Step 2. Bring the `week4.zip` file into R Studio server. Like last week, upload the zip file, and load in the data file. Launch the week_4 R script as before.

{If you've done Step 1 &2 already as a pre-lab preparation, super, pat yourself on the back, skip these steps an move on)}

Step 3. 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)
```

Step 4. Read in the datafiles that will be on the server. There's already a script line for this, you just need to change the file name (see the comments for advice on this)

Step 5. We've provided a suggestion of how you can complete the visualisation challenge task from week 3.

Step 6. Customize you graph work. We've provided some suggestions about adding titles and labels for your graph. Edit and play with the script lines to make them useful to you and to understand how they work.

- Try change the text, the colours, and so on of the graphs.
- Add comments for yourself about what the different commands do. The idea is to learn by trying different things out (changing values, taking out elements of the command, putting other is) and record for yourself.
- If you are struggling or not sue, try look at help files.

4.4 Section 2: 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.

4.4.1 z-scores 1

z-score 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?

4.4.2 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?

4.4.3 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)

4.4.4 Extra activity

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

Q7. Tom Bunion has completed a huge research study and 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 5

Week 5: Class test

No materials!

Chapter 6

Week 6: Sampling, probability and binomial tests

Written by Tom Beesley & John Towse

6.1 Pre-lab work

- Ensure you have watched the lecture for Week 6.
- There is a learnr tutorial to complete which will help you to think about **binomial tests**. You : You can find it here.
- Set up a new folder in RStudio and upload the files from the Week 6 zip file
- If you create a folder and upload the file into RStudio before the lab class you'll be even more ready to follow along!

6.2 Card sampling task

In the first task this week we will look at the sampling of events and we will apply the basic statistical test of the binomial test: `binom.test()`

Each table has a set of cards. These will be 13 red cards and 13 black cards - please check your set to ensure you have the right number of each colour (it doesn't matter what suit the cards are).

In this task we will be drawing samples from the deck, and trying to tell whether the deck is biased towards red or black. Think of the cards as the population of scores out there in world, and when you can only see back of the cards, that data is unobserved.

As an experimenter we are trying to estimate what is true about the world, and in order to do this we need to draw a sample. So each time you draw a card, you are observing one data point from the population, and based on a collection of data (the sample) you are going to draw an inference about what is true about the population.

6.2.1 Set up and instructions:

1. One person on each table should act as the "world" (the dealer). Congratulations, you are God. That is, this person will determine what is true about the state of things in the world. In this example, that means they control what is contained in the deck of cards. For each experiment, secretly remove some cards from the deck and place those face down. It's important that no one sees what these cards are. For example, you might take out 5 red cards, so that 13 black and 8 red are left; this deck is now

biased to black. Or you might take out 3 red and 3 black; this deck is not biased.

2. The remaining people (1 or more) will act as the experimenters. Your job is to draw samples and work out whether you think the deck is biased or not towards either red or black.
3. For each experiment, go through the following steps:

- The world removes some cards from the full deck (the number and colour of the cards removed is up to them) and they place these face down on the table. They shuffle the deck ready to start the "experiment".
- The experimenters "pre-register" their sample size. That is, they state how many cards they are going to draw.
- Draw samples one at a time. Importantly - make sure you replace the cards each time and the world/dealer should give the pack a quick shuffle.
- Mark down whether the card was red or black in your logbook
- The world should shuffle the cards after each draw. Repeat 2.3 and 2.4 until you have reached your sample size.
- At the end of each experiment, the experimenters should draw a conclusion based initially on their own "gut feeling" about the data. Do you think the deck was biased towards red, black, or was it unbiased?
- As a group, use RStudio to run a `binom.test()` to provide a statistical result (you can do this in the console, or create a new script to save your tests and results - it's up to you). Was this result unusual? How likely were the data given the null hypothesis? Note down the p value that this test gives you.
- The "world" can then reveal the hidden cards. Was the deck actually biased or not? How does this sit with a) your initial conclusions, and b) the result of the binomial test?

4. Repeat all steps in part 3 again for a new experiment, **making sure that you try different parameters for the experiment**. So vary a) how many cards are removed from the deck, b) the combination of cards removed from the deck, and c) the pre-registered sample size. Feel free to swap the roles around.
5. Once you've conducted a few experiments, discuss on your table the results you found. It might be useful to think about the following things:

- were there times when your intuitions were different to the statistical result? For example, you were sure there was a bias, but in fact the statistics told you this was not that unusual (p was > .05)?

- were there times when the deck was actually biased, but you failed to prove this with your experiment (you failed to see p < .05)? Do you remember what this type of error is called?
- were there times when the deck was *not* biased, but the test result suggested it was (p < .05)? Do you remember what type of error this is called?

6.3 RStudio tasks

For the second exercise today we will take a look at the phone data again. If you haven't already, set up a new folder for Week 6 and upload the data from the Week 6 zip (see pre-lab instructions).

6.3.1 Read in the csv data file

You should know how to do this by now. But if not, try searching "csv" at the top. Remember that what you name your data set is important for the following commands.

6.3.2 Take a look at the data

There are lots of ways to get a quick look at the data. Here are a few useful ones (some you've come across, some that might be new): `glimpse()`, `summary()`, `View()`, `head()`.

6.3.3 Create a boxplot of the phone data

Complete the code to create a boxplot of the estimated phone use. Note that you can put the boxes on either the x or you y axis. Copy and paste the code and edit it so you can also plot the actual phone use.

6.3.4 Create a density plot and/or hisotgram of the phone data

It's very easy to convert the boxplot code into either a density plot (`geom_density()`) or a histogram (`geom_histogram()`). Have a play around with these different types of graphs. Which one communicates the spread of the data most clearly? Is it better to plot these on the x or y axis?

6.3.5 Plot the relationship between estimated and actual phone use

So far we've looked at the phone use estimate and the actual phone use separately. But if people are at all accurate in their estimates, we'd expect these two things to be related (those people who use their phone more probably know they do). Let's use a scatter plot to see if this relationship exists in our data. In `ggplot()` we can do with with `geom_point()`. Each point on the graph needs an x and y value, so with the code you've been given, you just need to add in the two variables we want to plot.

- Is there a relationship between these variables?
- How would you describe this relationship in words?

6.3.6 Binomial test of the accuracy of phone use estimates

So did people overestimate or underestimate their phone use on average? We have given you a column called *accuracy* which simply says whether each participant underestimated or overestimated. Add your dataset name to this code to use the `count()` function to get the total who overestimated and underestimated. With these totals, you have enough information to run a binomial test:

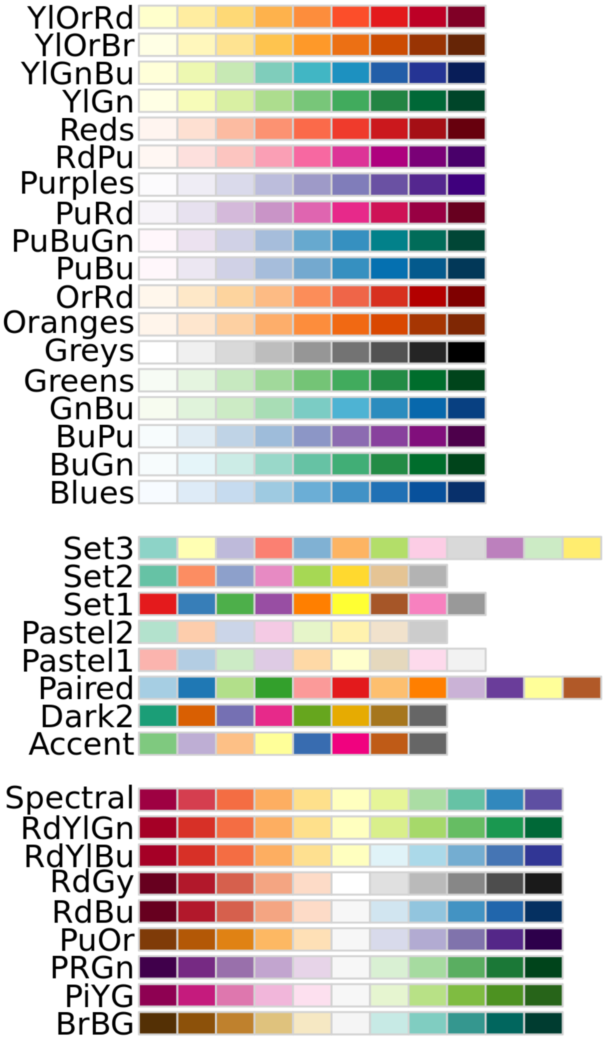
- for this test focus on the totals for "underestimate" and "overestimate" (for simplicity we can ignore people who were "accurate" and the NA values)
- These give you your first two parameters for `binom.test()`
- to get the probability, we consider the null hypothesis
- that is what's the probability of overestimating or underestimating under the null hypothesis?
- what is the result of the binomial test?
- You can also include the *UK_region* within your count, and then run binomial tests on these sub-groups. Do you find any interesting reuslts? What issues might we have with running the `binom.test()` on these sub groups?

6.3.7 More things to try with your scatter plot

We can customise our plot even further:

- Try adding/changing the `size =` within `geom_point()` to make the points bigger or smaller (values from .1 to 30)

- Try adding/changing the `alpha =` within `geom_point()` to make the points transparent (try values between 0.1 and 1)
- try adding `colour =` within `ggplot(aes( ))` to *map* the colour to the 'UK_region' variable.
- you can change the colours used by ggplot by adding `+ scale_colour_brewer()` to your plot code. Within this, try setting the pallete parameter to one of these options (e.g., `scale_colour_brewer(palette = "Set3")`)


YlOrRd
YlOrBr
YlGnBu
YlGn
Reds
RdPu
Purples
PuRd
PuBuGn
PuBu
OrRd
Oranges
Greys
Greens
GnBu
BuPu
BuGn
Blues
Set3
Set2
Set1
Pastel2
Pastel1
Paired
Dark2
Accent
Spectral
RdYlGn
RdYlBu
RdGy
RdBu
PuOr
PRGn
PiYG
BrBG

- remember you can add labels using `+labs()`
- remember you can set a new theme, such as `+ theme_minimal()`

6.4 Week 6 Quiz

You can access a set of quiz questions related to this week here.

Chapter 7

Week 7: Filtering data and testing means (one-sample t-test)

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

- There is an online tutorial. Please make every attempt to complete this before the lab!

- Create a folder for Week 7

- Download the Week_7.zip and upload it into this new folder in RStudio Server.

7.2 RStudio tasks

7.2.1 Plotting and filtering

1. Open the Week_7_script and run the `library`, `options` 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. Add a `read_csv()` command to get the data - you've done this every week, so this should be familiar.

2. Complete the `geom_histogram()` code to plot the distribution of salary data. Try setting `bins=20` within your `geom_histogram()` code. Play around with the number of bins for the histogram.

3. OK - we've got some pretty funky values here! Some people think the average salary is over £400,000!!! 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 (<). Remember that the filter command *keeps* the data that is *TRUE* according to the expression.

5. STOP! OK, **this next bit is very important**. If you completed that last step correctly you'll see an output in the console showing a "tibble" (a data frame) which has 194 rows and 5 columns. However, your object has not actually changed in the environment. This will still be showing as 197 observations (row). So the filter command will take out those large estimations, but we haven't actually changed the data object. To do this we need to assign (`<-`)the result of our filter command to the object. To do this, you need to put in some code at the start of this small chunk of code you've just run, so that the result of your commands will be assigned to the object (you can make this the same object name you've been using -overwrite it-, or call it a new name).

6. When you run this command again you should see the (new) object has changed to 194 rows. We can now plot the data again as a histogram. To do this, copy the earlier code for the histogram and edit it as necessary (if you're using a new object for the filtered data, you'll need to edit the code to reflect that). .

7. And as you know, we can also look at the distribution as a boxplot, a violin, or a density plot. Feel free to add in your own code for other visualisations you might like to try.

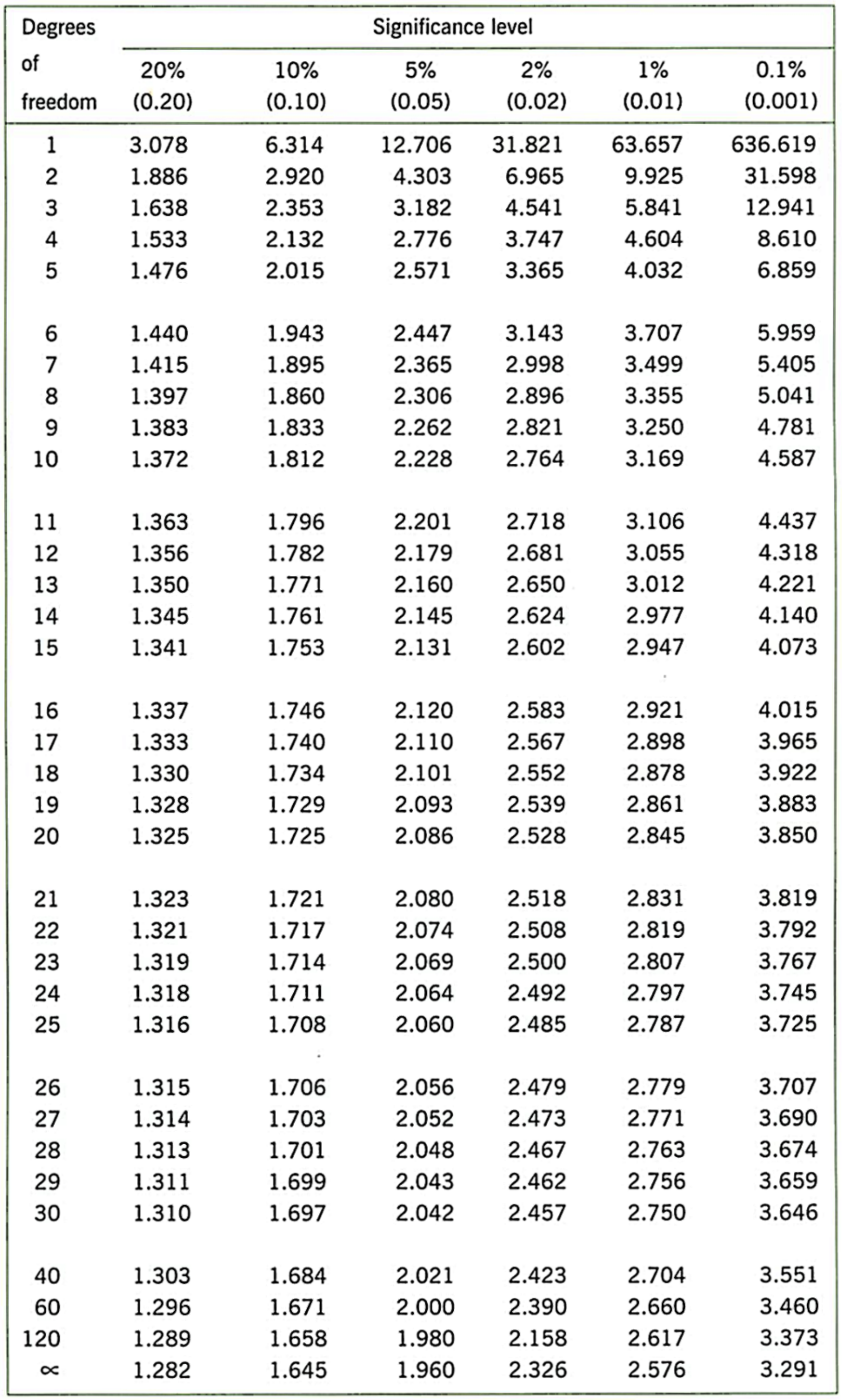
7.2.2 One-sample t-test

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.

8. First, let's calculate the mean and sd of the column.

9. Now we can compute a t-statistic and check whether this mean is significantly different from the expected mean. We do this 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 the 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.*

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



Degrees	Significance level					
of	20%	10%	5%	2%	1%	0.1%
freedom	(0.20)	(0.10)	(0.05)	(0.02)	(0.01)	(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
∝	1.282	1.645	1.960	2.326	2.576	3.291

7.2.3 Practising filtering

11. Filtering can also be useful for selecting certain sub-sets of our data. In the script we have given you an example of how we select a sub-set of data based on two conditions from two different columns:

```
data_set_name %>% filter(home_location_in_UK == "NW" & sibling_order
== "oldest")
```

We have given you a few different columns to look at and to use in practicing your filter commands:

`sibling_order`: what position in age was the respondent within their siblings `home_location`: UK / Asia / Europe, etc `home_location_in_UK`: NW, NE, etc (NA is non-UK residents) `attention_check`: respondents were asked "click strongly agree to show you are paying attention" - some people failed this!!!

Complete the following filters. We've put in `()` the number of rows you should see in the resulting object

12. Just those people who come from the North East (16 rows)
13. Those people who come from Wales and are the middle child within their siblings (2 rows)
14. Those people passed the attention check, are from the UK, and are an only child (21 rows)
15. Those people who are NOT from the North West; you'll need to use `!=` (84 rows)
16. Those people who failed the attention check (that is, did not say strongly agree) (14 rows)
17. Those people who are from the South East or (`|`) the South West (38 rows)

7.3 Sample size, size of effect, and the one sample t-test

In the lecture this week, Tom used an application to show the process of sampling data. You can access this application at the link below. There are three "parameters" you can change in this:

- **The true mean of the effect:** Think of this like the bias that was set up in your deck of cards last week. There is some true state out there in the world, and we are going to draw samples from a distribution of data that has a mean that equals this value. If you make this 100, then the true mean is equal to that under the null hypothesis (there is no effect).

- **The standard deviation of the data:** This sets how variable the data are in this population. If the data are more variable, then our samples will produce estimations that are less accurate of the true mean value.
- **The sample size:** How many observations are drawn in the sample. These are represented by the yellow circles in the plot.

Each time you draw a sample the data points are plotted in yellow and the mean of the sample is marked with the red line. The application also runs a one-sample t-test against the expected mean under the null hypothesis, of 100. The null hypothesis is also represented by the static distribution presented in grey, centred on 100.

Things to try:

1. Start with a sample size of 10, and a mean of the effect of 110 (SD = 15). How often do you get a significant result (p < .05) when you draw a new sample?
2. Now try changing the mean of the effect to 120. Does this increase or decrease the likelihood of getting significant results? What about changing to 130?
3. Now keep the mean effect constant (say 110), but increase the sample size. Try 5, then 10, 15, and so on. Does this increase or decrease the likelihood of getting significant results?
4. Set the mean of the effect to 100 and the sample size to 10. Keep drawing new samples, noting each time the p value. You will evenutally get a p value of < .05. What type of error is this?

Click here for the one-sample t-test application

7.4 Week 6 Quiz

You can access a set of quiz questions related to this week here.

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 geoms. 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

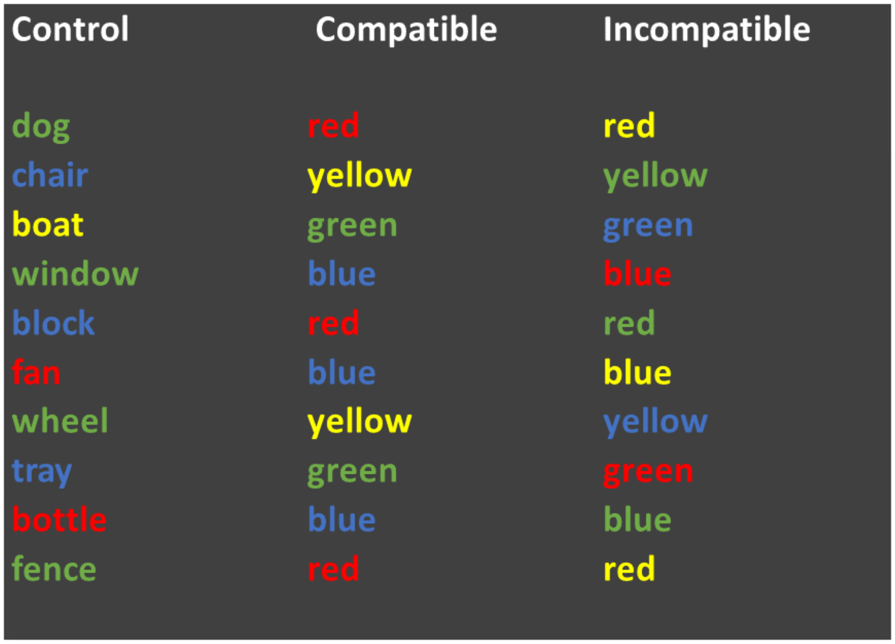
Online tutorial: You must make every attempt to complete this before the lab! To access the **pre-lab tutorial click here** (on campus, or VPN required)

Getting ready for the lab class

1. Create a folder for Week 8 and download the Week_8.zip file and upload it into this new folder in RStudio Server.

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 and then we will then plot them using ggplot(). First though, we’ll need to inspect the data and maybe do a bit of data wrangling by using our `filter()` command.

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

2. Run the `library` and add a `read_csv` line to read in the data set. Call your data something meaningful (perhaps *data_w8* or *data_stroop* but maybe not *bestest_most_fantastic_data_on_the_stroop_test_eva_init*)

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 and 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. You can play around with the *alpha* parameter, setting it to a value 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. Here we will introduce you to a new function called `mutate()`. This function will create a new variable (column) from an old one. Run this next block of code (you don't need to edit this one) to create the new column, *avg_time*. We are using this in combination with `group_by(pID)` because we want to calculate the average time for each participant. Use `view()` to take a look at the data to check the column has been created correctly.

6. Edit the `geom_histogram()` code to plot the distribution of values in the new *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 (for now) 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 you need to think about how you are storing the result of this filter process. Do you want to create a new object, or overwrite the existing object?

8. Check the new distribution of average times after this filter has been applied to the data.

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 **is the same as** (==) 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.

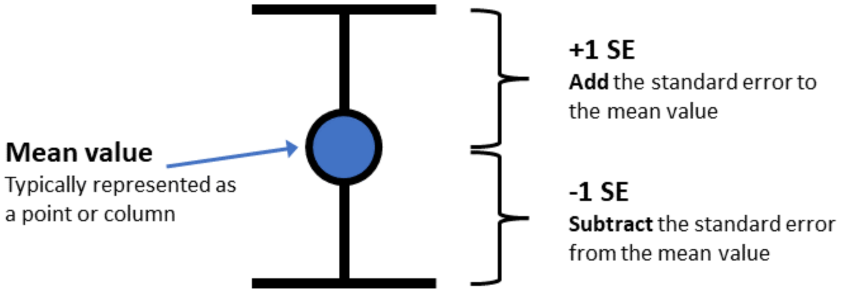
5. Write out a statement to express this result. Here's a template you can use, where you need to edit the bits in the []:

   > "There [was a / was no] significant difference between the [describe the variables that were compared], t([degrees of freedom here]) = [t value here], p < [p value here]."

6. 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. Write out a report statement (Q5) for each of your comparisons.

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. A very typical way to plot a mean value is to plot it with the standard error of the mean (SEM):


+1 SE
Add the standard error to the mean value
Mean value
Typically represented as a point or column
-1 SE
Subtract the standard error from the mean value

2. The code from Task 2-Q1 will give the mean. We will now complete the second line of this code to give the standard error values. TO do this, 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).

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

4. We will now plot these 3 mean values in a figure. Let's use `geom_point()` so that our means and SEs look a bit like the figure above. Complete the ggplot command to plot our summarised value called *stroop_mean* (y), as a function of the IV, *condition* (x).

5. Now we need to add some "error bars" which provide a visual guide as to how much uncertainty we have in our mean value. Edit the code for the `ggplot()` command to plot both `geom_point()` (same as Q4) 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 (hint: add or subtract) 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_point` 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_brewer()`. Take a look at the Week 6 (6.3.7) for instructions on setting colours.

10. Try changing your `geom_point()` to `geom_col`.

8.5 Saving your work

Scripts: By now you are hopefully getting used to editing and working within the script. As you know, 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) only exist within RStudio, and are temporary (with a script and the csv file, you can always redo the analysis). But what if you want to use the data elsewhere? For example you may want to share the data with your project (PEP?) 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. Or maybe you want to make the csv file available to other researchers. To get files out of RStudio, 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)

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

Create a folder for Week 9. and download the week_9.zip file and upload it into this new folder in RStudio Server.

9.2 RStudio tasks

9.2.1 Exploring the data

In this class we will be exploring some data on people's estimations on aspects of the UK population. We asked people 4 different questions:

> Out of every 100 people, about how many do you think are:
>
> 1. Christian?
> 2. Muslim?
> 3. Over the age of 65?

We also asked a related question about immigration:

> What percentage of the UK population do you think are immigrants to this country? (i.e. not born in UK)

1. Read in the data "data_wk9.csv" using `read_csv()`.

2. Take a look at the summary statistics for all of the columns in our data using `summary()`

3. At this stage you could try using `ggplot()` with `geom_histogram()` (hint: not sure? look at last week) or `geom_density()` to explore the different columns.

4. You may be tempted at this stage to apply the `filter()` commands to remove any outliers. If so, yes, that's entirely sensible. But note that we are going to remove outliers a little further into our tasks after we've first done some visualisation.

9.2.2 Visualising relationships in the data

To what extent are people's estimations of these population parameters related? Let's look at this by plotting these data as `geom_point()`:

1. Complete the code to add one numeric column in the data to x and another numeric column to y. You can pick any of the columns you like, but it's important that you understand what research question you are asking with your choice. For example, you might be asking "Do people who estimate there are more Christians in the population also think there are more people over the age of 65?"

2. Note the general pattern in the data. Is there a *postive relationship*: do people who give high estimations for one variable tend to give high estimations for the other variable? Or is there a *negative relationship*: do people who give high estimations for one variable tend to give lower estimations for the other variable? Or is there no discernable relationship at all?

3. Copy the code and edit it to explore relationships between the other numeric variables, each time noting the research question you are asking, and discussing on your table what kind of relationship you can see in the data.

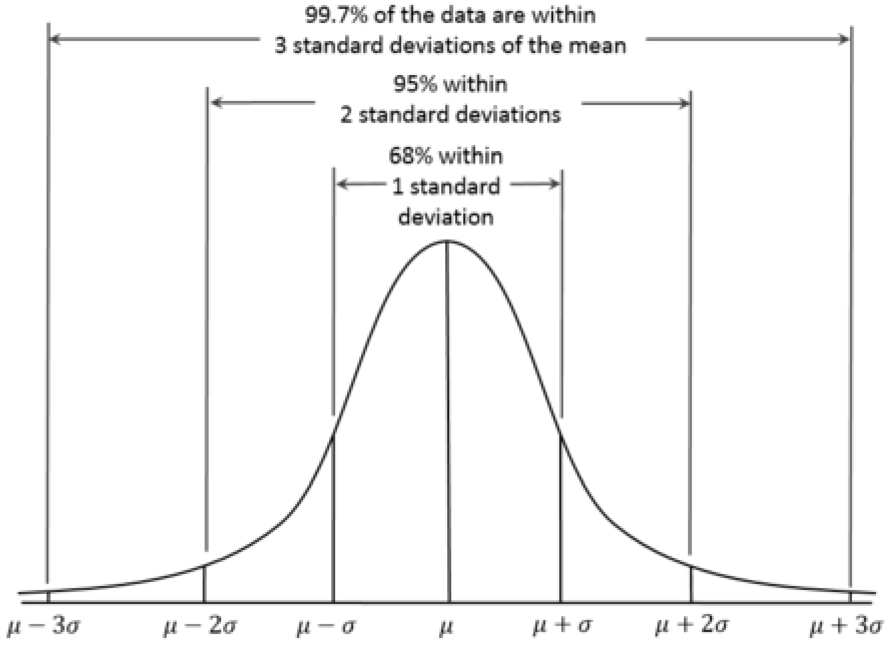
9.2.3 Using z-scores to remove outliers

1. You may have noticed that there are some fairly extreme values in some of these numeric estimations of the population. As we've discussed in previous weeks, 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.

2. Let's create a "z-transform column" called *z_imm* for the estimates of the percentage of immigrants. Complete the code you have been given by adding the relevant variable (column) name to the code. You may want to create a new data object at this point.

3. View the new data object to check this column has 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 it conforms to what we know about z-scores (e.g., `mean()` should be very close to 0, `sd()` = 1). Note, if you want to change the output in r from scientific notation to non-scientific notation, you can run the command `options(scipen=999)`.

4. 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:



5. Let's consider an outlier any value that has a z of 2.5 (a conventional cutoff). Plot a histogram of the *z_imm* column in order to inspect whether there are data that are above 2.5 or below -2.5.

6. Add a filter command to remove the values in the *z_imm* column are greater than 2.5 or less than -2.5.

9.3 Unrelated samples t-test

We have also included a categorical variable in our data this week, which is one you have seen before in our analysis classes: the home location in the UK of the respondent, `home_location_in_UK`. For this data object we have included only those responses from those people from the "North" (NW and NE) and those from the "South" (SW and SE). Other respondents from elsewhere have been removed from the data. We can therefore look at whether people's home location determines their population estimations.

1. First we will look at the mean population estimations, split by home location. To do this, complete the `group_by()` and `summarise()` commands to give the mean estimates of the proportion of muslims in the population by home location. You don't need to edit the `N = n()` line - this provides the number of participants at each level of the *home_location_in_UK* variable.

2. What do the means suggest? Do people in the North and South give different estimations?

3. Let's test if these differences are real. First, it is worth noting that many more respondents originate from the North than from the South (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.

4. Now let's run the t-test. This week we are comparing data from different samples of participants (those who are from the North and South). 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 parameters, 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.

5. In that t-test we looked at the *pop_est_muslim* variable, but we can do this test for all of our population estimates. Copy the code to run another `var.test()` and `t.test()`, for the *pop_est_christian* variable. Note the t and p values; What do these tell us about the relationship between music preference and social media use? You can also run two further tests on the other population estimations.

9.4 Power and effect size (d) calculations

1. 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.

2. Import the stroop data. We have reduced the data down to just the *compatible* and *incompatible* conditions.

3. 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.

4. We already know that this large effect size was significant with our 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 that you calculated (Cohen's d) and set the N to 20. What **power** would we have achieved with this sample size, to detect this large effect? Discuss with your table, or staff, what this power means.

5. 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.

6. 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 (this is a real "stressor task" people use!). We are unsure of what consequence this will have for our effect size, but we want to estimate the effect size that could be detected in our experiment. 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?