# Data Fluency

Ben Whalley and Chris Berry

## Contents

# Overview

From the module aims:

> The module aims to foster fluency and confidence in the handling, visualisation and communication of quantitative data, alongside skills and techniques for working with larger corpuses of textual and other data. Data visualisation is taught as a foundational technique for exploring and communicating insights from quantitative data. Existing knowledge of linear models is extended with the introduction of the generalised linear model, and a contemporary approach, emphasising prediction and model evaluation is introduced.

In a nutshell: we want to give you the skills to analyse your data as independent researchers, and to give you confidence in working with data which will stand you in good stead in your future careers.

## Approach of this course

Sometimes, psychology students learn statistics through a "bag of tricks" approach: Workshops might teach how to "do an Anova", or "how run a multiple regression". Or you might be given a checklist of things to do when analysing data of a particular type, but all without any bigger picture of what we are trying to achieve when we collect and analyse data.

To provide a common thread to our teaching, research methods modules at Plymouth adopt the model for the work of data scientists proposed by Wickham, 2017 (see figure):

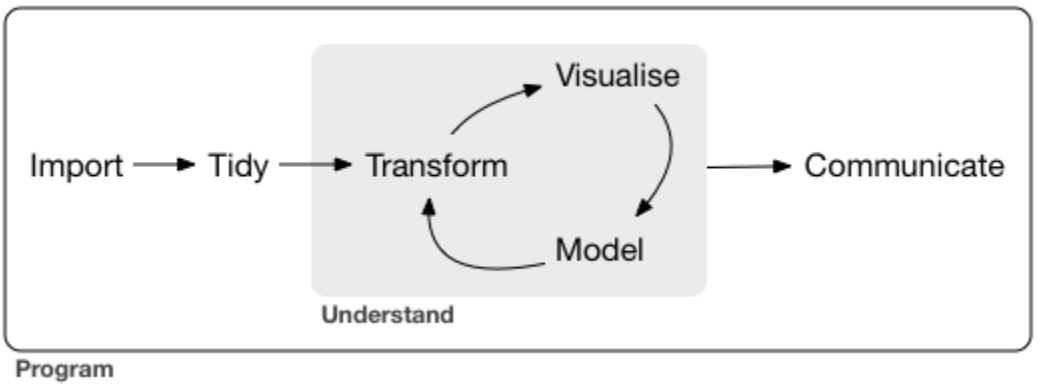



Figure 1: Wickham's model of a data science workflow

In this module we do cover specific skills, but we hope you will also learn about this general approach to working with data, and integrate it into your own research.

## Format of the sessions

We have 16 sessions, which work as follows:

- We avoid extended lectures. This doesn't work well with this subject matter.
- The focus is on learning by doing (this is more like cooking than chemisty).
- In the first hour of each session we will (often) work together.
- In the second hour your work will be self-paced, or in pairs or small groups.
- Activities in the workshops are variable in length, sometimes you will finish early, other times you may be expected to complete the activities outside of class.

## The most important thing of all

The most important thing of all is to **practice**.

These materials provide lots of practice tasks. You **NEED** to work through them all to be able to pass the course effectively.

### Lab diary/R project archive

It's recommended that you keep a running note of all the work you do in class. This can take for form of a notebook in Word, a blog, or an R script (see first session).

Without a running record of what you have/haven't done it's much harder for teaching staff to help you. The record also allows us to review your progress and make suggestions/improvements.

## List of sessions

Don't worry if all the terms in this list don't make sense yet … they will soon!

- BeginneRs 1: Getting started with RStudio
- BeginneRs 2: Basic skills in R
- Visualisation and plotting: Relationships, group differences, color.
- Real world plotting
- Data handling: Working with multiple files; Rmarkdown.
- Regression 1: Core concepts.

- Regression 2: Causes/Multiple regression.
- Regression 3: Tests and variable/effect coding.
- Uncertainty and intervals: Confidence and prediction intervals.

(CHRISTMAS BREAK)

- Fitting curves: Using polynomials.
- Building models 1: Choosing variables in multiple regression
- Building models 2: Anova and Bayes Factors
- Contrasts and tests: Using `emmeans`
- Communicating regression and Anova
- Reproducing an analysis 1: groups work on example, start to finish.
- Reproducing an analysis 2: finding a suitable paper, data handling.
- Reproducing an analysis 3: reproducing the analysis.

A catchup/revision session then follows, but no new material is introduced.

## Access to R

Throughout the module we use R for data processing and analysis.

If you are taking this course at Plymouth University, the easiest way to run the code examples here is to the school's RStudio Server.

- Login to your account on the server here
- To get an account on the server, or reset a password, contact the Psychology technical office

**Installing at home** If you want to install R on your own machine, instructions are available here:

- https://github.com/plymouthPsychology/installR/

Be sure to install the recommended packages or the examples given here won't work.

## Why do we use R?

This material copied from Andy Wills' RminR.

**This document covers some of the reasons we use R in this course. It's not "required reading", but take a look if you're interested.**

### Introduction

R is a piece of software for handling data. It's the one used on this course, but it's not the only option available, others include: Excel, Jamovi, JASP, MATLAB,Stata and, perhaps the most talked-about alternative, SPSS.

### Student experience

Students prefer R. In a recent study, undergraduate psychology students at Glasgow University were given a choice between R and SPSS, having experienced both. Two-thirds of the students chose R. Those who chose R did better in the final assessments and showed lower stats anxiety. R is being used to teach Plymouth University undergraduates (and visiting Year 10 students) across a range of different courses. Read more.

**Employability**

Data science is a graduate skill in high demand, and using R is a key skill in that market. In contrast, demand for SPSS skills has been declining dramatically for a decade. At SPSS's current rate of decline, it'll be gone by the time you graduate. Read more.

**Free**

R is free. You don't need to pay anything to download or use it, and never will. In contrast, once you leave university, SPSS would cost you or your employer around £6000 per person per year.

**Never out of date**

Every analysis you can think of is already available in R, thanks to over 12,000 free packages. As new analyses are developed, they become available in R first. In 2013, SPSS realised it couldn't keep up with R, and admitted defeat.

**Real**

Real data analysis is mainly preprocessing – scientists spend around 80% of their analysis time getting the data into a format where they can apply statistical tests. R is fantastically good at preprocessing. Our course focusses on realistic data analysis, making R the perfect tool for the job.

**Accurate**

The alternatives to R for real data analysis are either kludgy, error prone and have poor reproducibility (e.g. preprocessing in Excel, followed by statistics in SPSS), or are more niche in the graduate jobs market (e.g. MATLAB). In particular, Excel is famously error prone with, for example, 1 in 5 experiments in genetics having been screwed up by Excel and the case for the UK government's policy of financial austerity being based on an Excel screwup.

**Reproducible**

R's use of scripts means that, if you have done the analysis completely in R, you already have a full, reproducible record of your analysis path. Anyone with an internet connection can download R, and reproduce your analysis using your script. Making your analyses reproducible is an essential skill in many areas of research.

**Free as in freedom**

R is "free as in freedom" because all the source code is available to everyone (it's "open source"). Some reasons this is important:

1. All software has bugs; making the source code available means it's more likely that these bugs are found and fixed. In contrast, no one outside of IBM can look at the source code for SPSS, and it's entirely up to IBM whether they fix, or tell you about, the bugs it has.
2. All software is eventually abandoned by the people who wrote it (if for no other reason than their death). Open source software only dies if no one in the world cares enough about it to maintain it. In contrast, closed-source software (e.g. SPSS) dies as soon as the current owners decide to kill it.

**Supported by Plymouth University**

R is already installed on many public machines at the University of Plymouth.

**Runs inside a browser**

You can use R without having to install it, e.g. RStudio Plymouth.

## Teachers' notes

### Course features

**Organization and staffing** The module comprises 17 workshop sessions. Students work individually or in pairs within workshop classes of up to 80. For each session there will be a specified teaching assistant (Ph.D student or TARA) supporting students.

Each session is broken in half, with more intensive exercises in the first session, followed by supported-self study or followup activities (e.g. finding literature, working on problems).

**Resources** All sessions in 2019/20 will be run in the a computer lab. The course requires one computer (PC, Mac, or Linux) per student, in all sessions. Any student with a laptop should bring it.

For activities based around R, students will use the school RStudio server. Accounts will be released in the first session.

There is a need for an online **lab book** system. Students can use the Blog system within the DLE to keep a record of their activities and progress each week.

**Employability skills** Data handling and visualisation. Communication skills. Working to deadline. Critical thinking / analysis.

## Week-by-week resources

### Session 6 (Regression as a fitted line)

Need:

- OHPs
- printable rulers
- Sharpies
- Printed out example-plots.pdf (give several to each group to avoid swapping)

Timings:

- First hour is plotting task
- Second hour is the regression worksheet + prediction using augment

### Session 7 (Causes and effects)

Relates to: second regression session including causal models

### Resources

Bring: - Markers - A3 paper - Cancer plot - Printouts of studies

### Timings

- Introduction… thinking about cause and effect before we dive deeper into techniques
- Explain DAG relationships (but not moderation)
- Students brainstorm one of the topics, then draw a diagram
- Followup questions about how strong relations are, evidence, is there mediation?

30 mins

- Explain effect modification
- Draw in gender and possible moderators
- Draw in other moderators if possible

45 mins

- Distribute cancer plot
- Students draw causal models from cancer plot
- Students add more confounding to their model

60 mins

- Point explaining why correlation would, ideally, be redrawn as directed lines
- Point relating causal to statistical models (t test, regression, multiple regression, mediation etc)
- Stats are just an implementation detail though (focus on your predictions)
- Make the point that inferences require the causal model to be correctly specified
- Representing non-linear effects using $\cup$ and $\cap$ as labels
- Introduce real examples exercise with study1/2.

75 mins

Students work independently.

#### Session 8

- Begin by getting groups to share their models of the papers
- Confounding? Is it addressed in the discussion of the papers? Are the inferences safe?

20 mins

# Assessments

The assessment for PSYC753 includes 3 components:

- A pass fail assessment which will ensure you have acquired the core skills in handling and visualising data.
- A structured data analysis and visualisation task, requiring short answers (50%)
- An authentic analysis assignment, where you will replicate the analysis of a published study (50%).

***For submission dates please check the DLE.*** Don't rely on this website.

## Core skills test

Details of the core skills test will be released in week 13 via the DLE, and will be due around week 20 (early December).

## Data analysis and visualisation task

This assignment is due on 27th February and counts 50% towards your module grade.

Your task is to answer the questions below.

#### Submitting your answers

You should submit exactly 3 files:

1. An rmd file
2. A PDF document, produced by knitting your rmd file.
3. As a separate document, upload a copy of the standard CW coversheet (and complete the feedback section).

Within the Rmd file you should:

- Label each question clearly
- Where necessary, include comments explaining what specific lines of code do.
- Intersperse explanatory text with code chunks (don't put everything in the same code chunk).

Your rmd file should "Just Work" when the marker opens and runs it, and produce the same output as the knitted PDF file you submitted (i.e. there won't be any errors or missing datafiles).

If you work on your own computer at home, you should check your rmd file 'knits' correctly on the online Rstudio server.

**Responses to frequently asked questions will be posted at this FAQ link**

**Questions**

The number of marks each question is worth [out of 50] is given below.

**1. Data handling [10 marks]** Use `dplyr` functions like `group_by` and `summarise` to recreate the two tables below, from the `gapminder::gapminder` dataset.

Your tables should look like the ones below once you've knitted your document to a PDF:

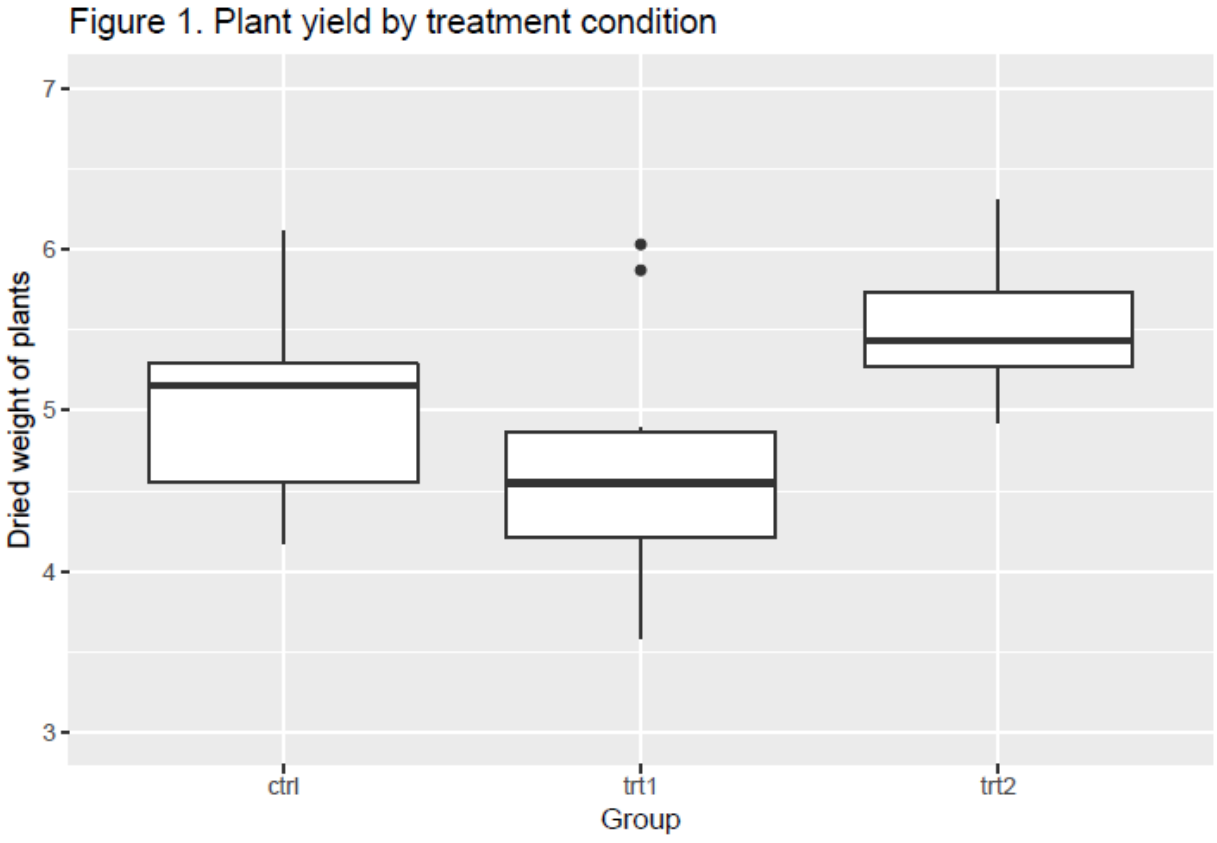
Table 1: Mean Life Expectancy in Europe by Year

| year | mean(lifeExp) |
|---|---|
| 1952 | 64.4 |
| 1957 | 66.7 |
| 1962 | 68.5 |
| 1967 | 69.7 |
| 1972 | 70.8 |
| 1977 | 71.9 |
| 1982 | 72.8 |
| 1987 | 73.6 |
| 1992 | 74.4 |
| 1997 | 75.5 |
| 2002 | 76.7 |
| 2007 | 77.6 |

Table 2: Country With the Highest GDP in 2007, by Continent

| continent | country | gdpPercap |
|---|---|---|
| Africa | Gabon | 13206 |
| Oceania | Australia | 34435 |
| Americas | United States | 42952 |
| Asia | Kuwait | 47307 |
| Europe | Norway | 49357 |

**2. Visualisation and plotting [10 marks]** Recreate the plot below, using the `datasets::PlantGrowth` data:




**3. Regression [20 marks]** Use `read_csv` to load the `climate` dataset held at https://bit.ly/3a1eF7w. The data are responses to a survey concerning people's attitudes to climate change.

These are the variables:

`sex` 0 = male, 1 = female

`age` age in years

`change` To what extent do you agree or disagree with the statement: I can personally help to reduce climate change by changing my behaviour. 1 = strongly disagree…5 = strongly agree

`concern` How concerned are you, if at all, about climate change, sometimes referred to as global warming? 1 = not concerned…4 = very concerned

`nuclear` On a purely emotional level, how do you feel about nuclear power? 1 = very negative…5 = very positive

`exagerate` To what extent do you agree or disagree with the statement: The seriousness of climate change is exaggerated. 1 = strongly disagree…5 = strongly agree

`hedonism` How important to you is the gratification of desires, enjoyment in life, and self-indulgence? 1 = not important…5 = very important

a. Fit a linear model to determine the extent to which `change` is predicted by `concern`. Report the results in APA format, include an appropriate plot of the data, and explain the results.

b. Add `nuclear`, `exagerate`, and `hedonism` to the model in part a. Report the results from this multiple regression in APA format and explain the findings.

c. Does the addition of these variables to the model *improve* the prediction of `change`?

d. Using the model with `concern`, `nuclear`, `exagerate`, and `hedonism` as predictors, derive a prediction for a person who has scores of concern = 2, nuclear = 2.5, exagerate = 3, and hedonism = 3.5.

**4. Fitting curves [10 marks]** Again, using the `climate` dataset at https://bit.ly/3a1eF7w, is there sufficient evidence for a linear, quadratic or cubic component of `age` in the prediction of `change`? Explain your answer.

## Authentic analysis assessment

This task will be due around week 40.

---

Find an empirical paper for which the authors have freely-shared the dataset but not their analysis scripts. Ideally this will link to your research interests, or be similar to your project analysis, but this is not required.

- A good way of finding a suitable paper would be to look through a journal known for publishing papers with open-source datasets, for example PlosOne. Another would be to search Zenodo or a similar repository for recently-published datasets, and look for where the articles reporting thse data were published.
- To keep the assignment simple, choose a paper which uses statistical methods taught as part of the module, or which can be approximated by them. For example: a multiple regression, between-subjects Anova, or repeated measures Anova or mixed model.
- Some papers report many studies, or many different analyses, and may have technical details with which you are not familiar. If this is the case then you do NOT need to deal with the whole paper: Simply **focus on one of the primary analyses**. For example, if one of the main study hypotheses can be answered by a 2x2 Anova or mixed model, but there are lots of other analyses included in the paper, then focus on just this one analysis.

**If you are unsure about the scope of this assignment (e.g. don't know how much of a paper to try and replicate) then please check with the module leader early in the process.**

To check if a paper will be suitable, you can answer these questions:

1. Are the data available?
2. Is the design between-groups? (answer must be yes)
3. Are measurements at more than 2 timepoints per-participant? (answer should be no; if the paper reports "mixed effects" models or repeat-measures Anova it will not be suitable)
4. Is the outcome a continuous variable? (i.e. must not be a binary outcome)

Your task is to:

1. Imagine you were the author of the paper, just prior to collecting data for this study. Complete the form on 'aspredicted.org'. You don't actually need to use the aspredicted website for this — simply copy the heading structure from the form, and include your answers in your submitted manuscript. Comment on whether it is possible and/or straightforward to reconstruct what would have been in the pre-registration from what is written in the journal article. Comments can be either interspersed with the answers or in a separate block at the end.
2. Using `ggplot`, make at least one plot of the data which illustrate the main findings. Explain your design decisions for these plots: what features of the data were you trying to highlight, and how? This plot might replicate a plot from the paper closely, but could also use techniques learnt in the course to increase the information density or provide some other enhancement. One approach to answering this question would be to show several plots representing different tradeoffs/options when plotting the data: Your answer would then justify the selection of the final plot over the others.
3. Replicate the statistical analysis using methods which are appropriate to the data. Report your findings in prose using APA format, but allow yourself additional space for explanation of the models/tests run. Compare/contrast your results with the published manuscript as appropriate. Document your responses and version of the analysis in an Rmd file with accompanying dataset, suitable for upload to Zenodo or OSF.io.
4. Discuss the extent to which the concept of 'researcher degrees of freedom' (Simmons et al 2011; Gelman and Loken 2013) should influence our understanding of your chosen study.

**Important notes**

- Don't assume that a published dataset and manuscript can be easily replicated simply because they have been shared: ***It may well be the case that it is hard to replicate the findings and this is OK (and interesting!) You will not lose marks simply because you cannot replicate the published findings.***
- You do not need to use the *exact* method used in the original paper in all cases; the point of the exercise is to apply methods we have learned in an appropriate way. For example, you may wish to apply Bayesian techniques to a dataset previously analysed with frequentist methods; this would provide additional opportunity to demonstrate understanding of the material and appropriate application of techniques.
- Your mark will be based on how appropriate the analysis you propose is, and the range of skill and understanding you demonstrate in answering the questions.

**Submitting your answers**

You should submit 5 files:

1. An Rmd file and
2. A datafile which when 'knitted' together produce
3. An HTML or PDF document.

4. As a separate document, upload a copy of the standard CW coversheet (and complete the feedback section).
5. Also include a pdf copy of the study you chose to replicate.

If you need to include more than one datafile that is also OK.

Within the Rmd file you should:

- Label each question clearly
- Where necessary, include comments explaining what specific lines of code do.
- Intersperse explanatory text with code chunks (don't put everything in the same code chunk).

Your Rmd file should "Just Work" when the marker opens and runs it, and produce the same output as the knitted HTML or PDF file you submitted (i.e. there won't be any errors or missing datafiles).

If you work on your own computer at home, you should check your Rmd file 'knits' correctly on the online Rstudio server.

See common problems when trying to knit Rmd files

#### Marking scheme

All questions will be marked using the standard school grading scale. Questions 1, 2 and 4 will contribute 60% of the overall mark; question 3 contributes 40%.

#### References

Gelman, A., & Loken, E. (2013). The garden of forking paths: Why multiple comparisons can be a problem, even when there is no "fishing expedition" or "p-hacking" and the research hypothesis was posited ahead of time. Department of Statistics, Columbia University. Chicago

Simmons, J. P., Nelson, L. D., & Simonsohn, U. (2011). False-positive psychology: Undisclosed flexibility in data collection and analysis allows presenting anything as significant. Psychological science, 22(11), 1359-1366. Chicago

## BeginneRs

#### In brief

> A big part of psychology is collecting data about people, visualizing it (graphs etc.), and drawing conclusions. Working with data is a core skill for researchers, and increasingly important in many professions.
>
> RStudio, like Excel, is computer software that helps us to do that. RStudio is rapidly becoming the standard tool for serious data analysis in psychology and other sciences, because it's powerful, relatively easy to use, and free.
>
> In this course we'll learn R as we go, building a little at a time. This and the next session covers the basics (things you would learn as part of an undergraduate statistics course at Plymouth). If some of this is familiar, there will be extension exercises to enhance your knowledge.

---

This material was adapted from Andy Wills' RminR.

Figure 2: Photo: 1ksmiles.com

## Getting started with RStudio

Open a web browser (e.g. Firefox, Safari, Chrome, *not* Edge) and go to an RStudio server, like the one at: https://rstudio.plymouth.ac.uk. Log on, using the username and password you have been given.

If that works, you should see something like this:

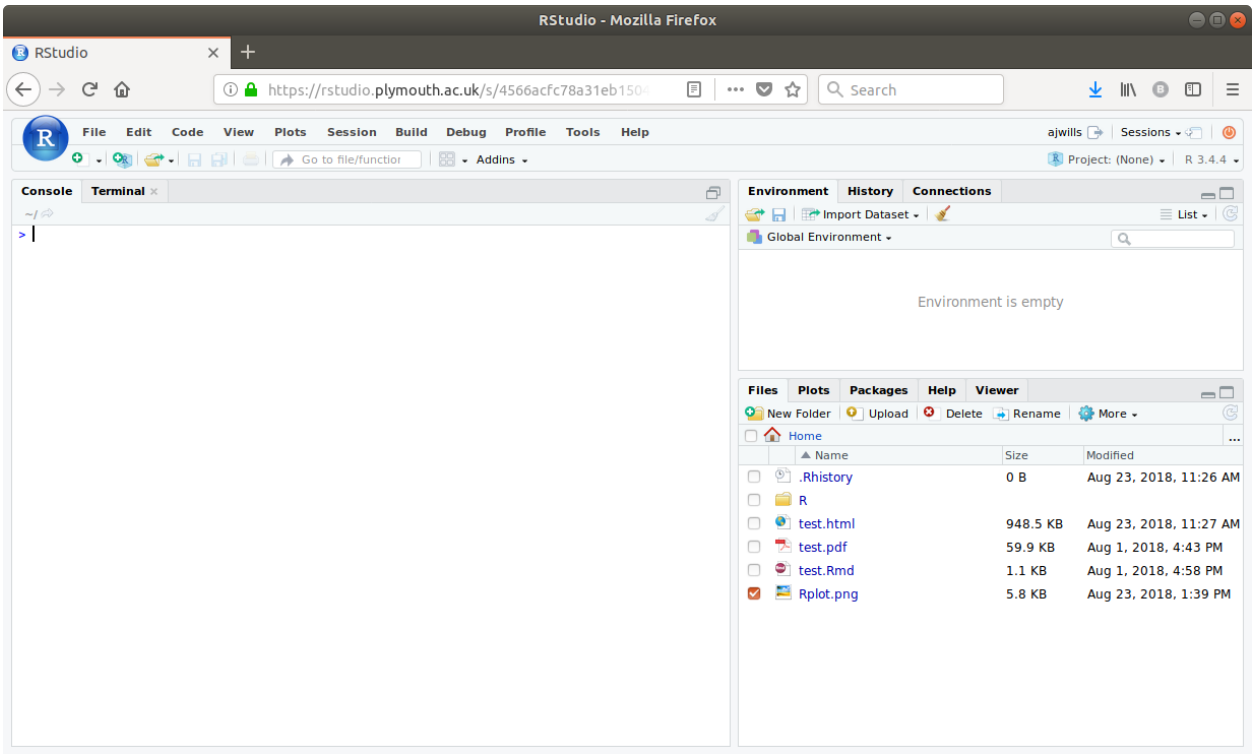



Figure 3: RStudio on first opening

We'll go through what it all means in a bit. But, first, we're going to...

**Create a new project** RStudio uses ***projects*** to help you keep your work organized, and to make sure you have a record of your analyses. You should start a new project each time you start a new module in your degree (possibly more frequently, but we'll come back to that later). Here's how to create a new project:

1. At the top right of RStudio, you will see a little blue cube, with the text "Project: (none)". Click on this, and select "New project".
2. Now click "New Directory"
3. Now click "New Project"
4. Next, type in a name for the project that makes sense to you in the "Directory name" box. I've typed *psyc411*, but you should pick something more meaningful to you (e.g. *beginning-r*). Then click "Create project".
5. Now, create a *R script*. An R script is a record of the analyses you have done. You create an R Script by clicking on the white plus sign on a green background (see below), and then clicking on "R Script".

If everything worked well, your screen should now look like this:

You should be able to see four parts:

1. The **Script** window - This is the rectangle on the top left. This is where you will tell R what to do. It only does what you tell it.

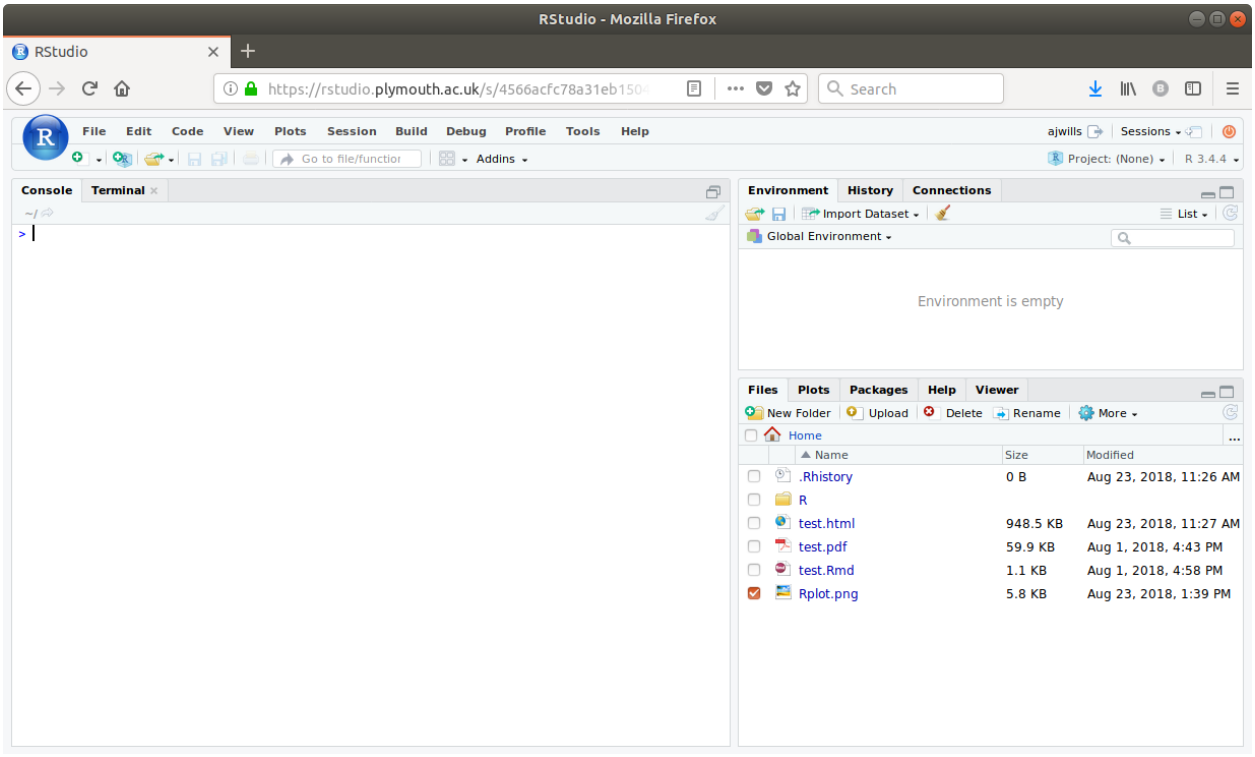



Figure 4: RStudio without a project open

2. The **Console** window - This is the rectangle on the bottom left. This is where R prints the answers to your questions.
3. The **Environment** window - This is the rectangle on the top right. It's where R keeps a list of the data it knows about. It's empty at the moment, because we haven't given R any data yet.
4. The **Files** - This is the rectangle on the bottom right. This is a bit like the *File Explorer* in Windows, or the *Finder* on a Mac. It shows you what files are in your R project.

That's it! You're all set to start learning how to analyse data in R.

## Exploring data (in brief)

Before starting this exercise, you should have had a brief introduction to using RStudio. If not, take a look at the Using RStudio worksheet.

### Contents

- How to use these worksheets
- Loading a package
- Loading data
- Inspecting data
- Calculating a mean
- Dealing with missing data

### How to use these worksheets.

Throughout this worksheet (and others on the course), you'll see the commands you should type into RStudio inside a grey box, followed by the output you should expect to see in one or more white boxes. Any differences

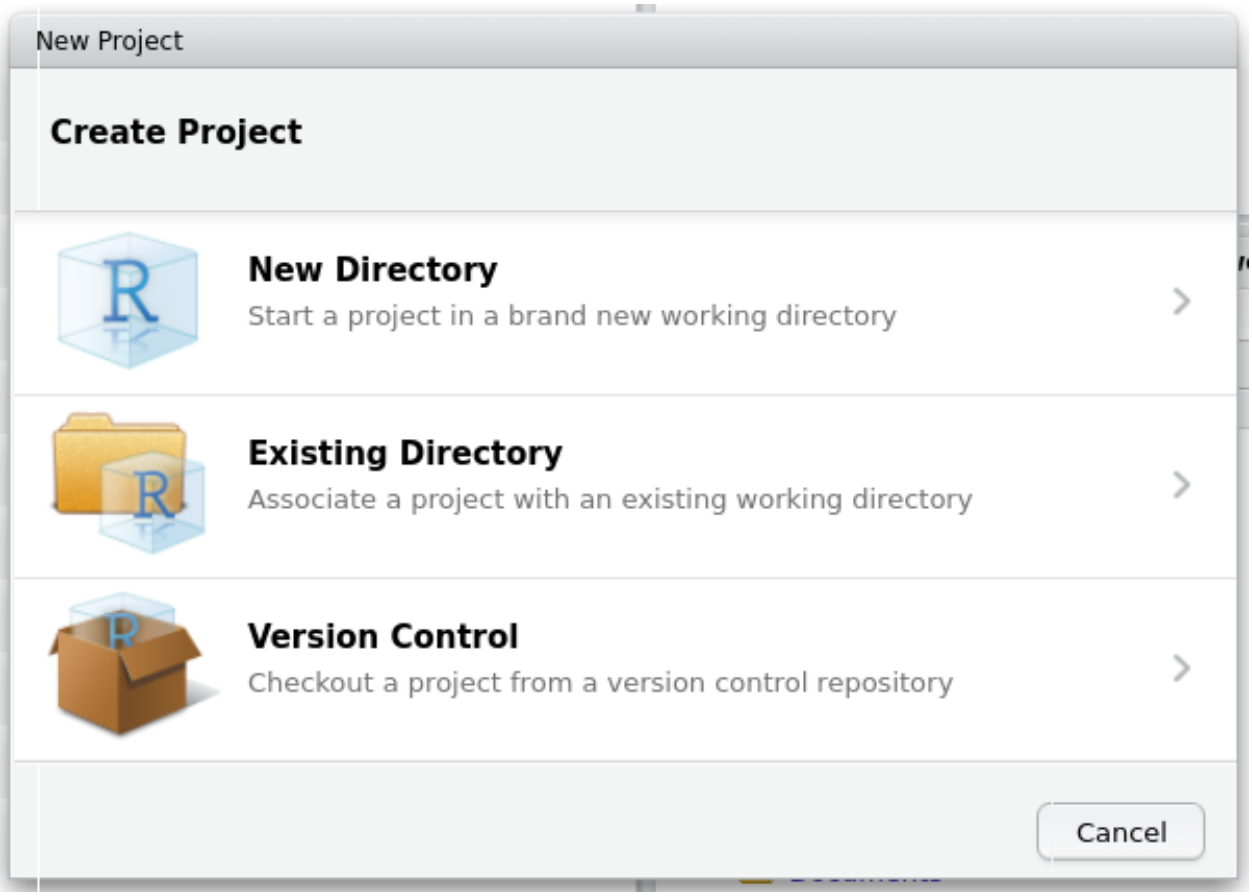



Figure 5: Project dialog #1

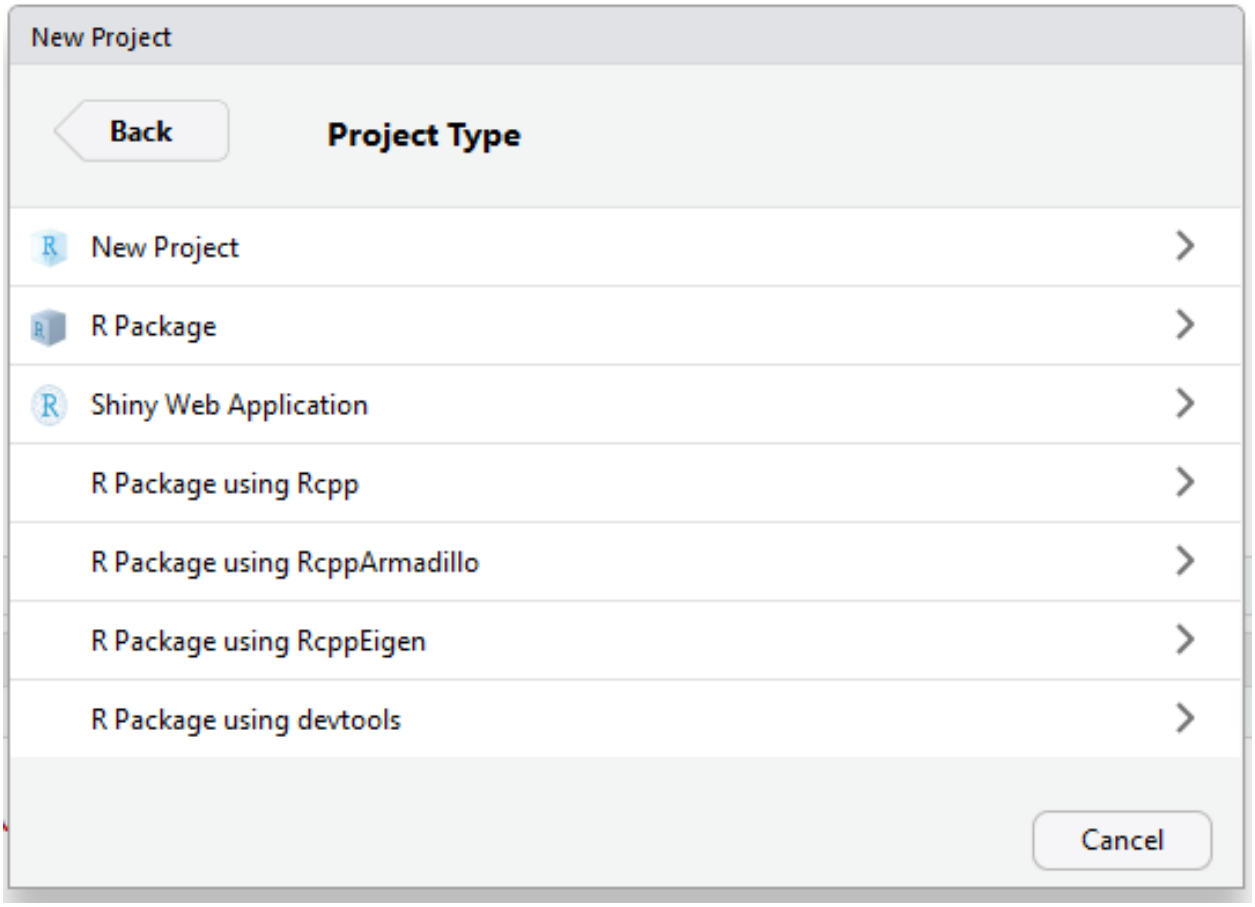



Figure 6: Project dialog #2

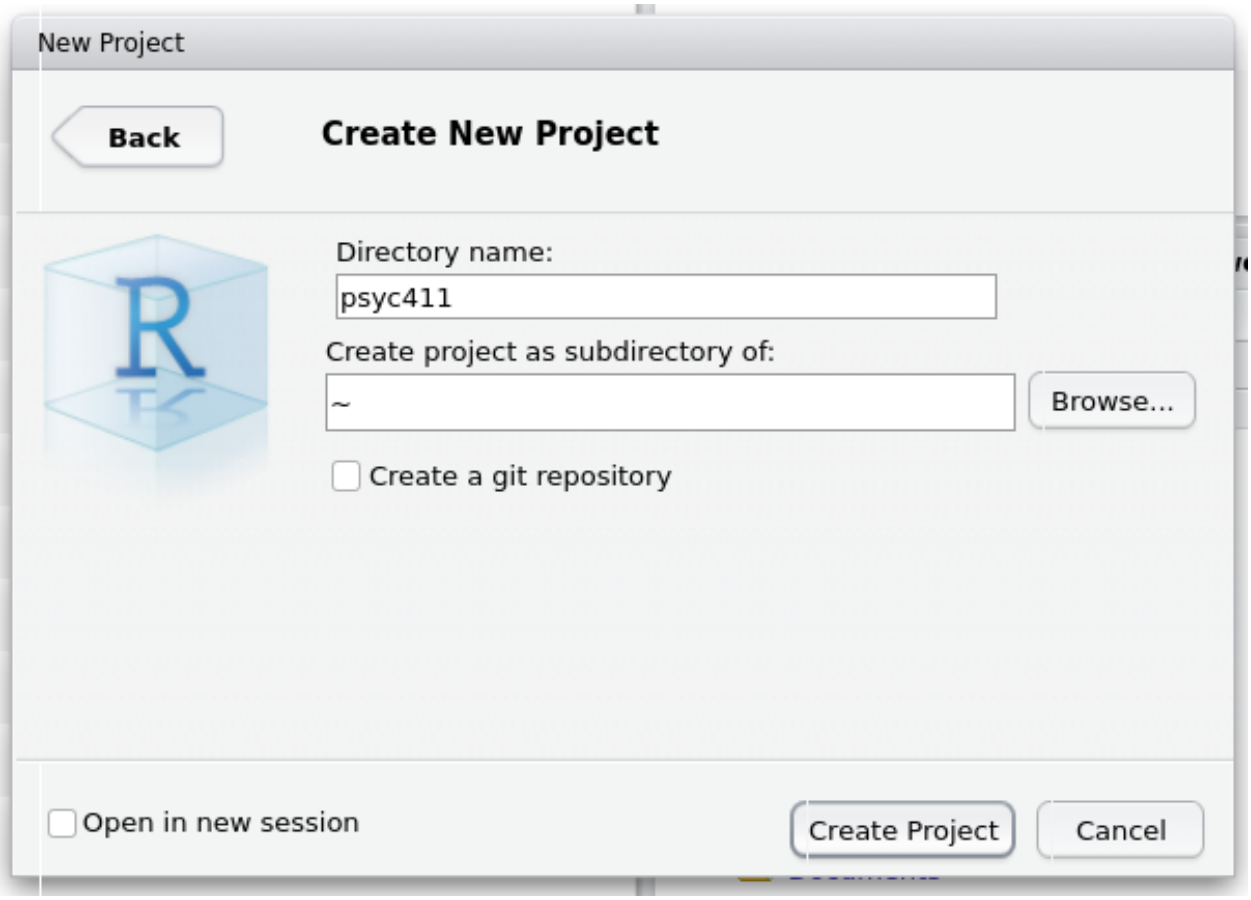



Figure 7: Project dialog #3

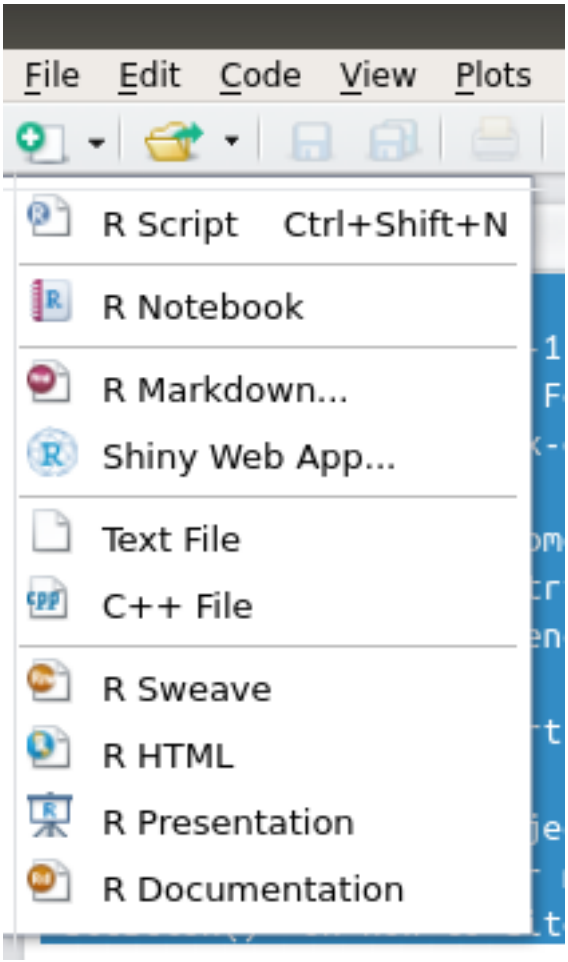



Figure 8: Script menu

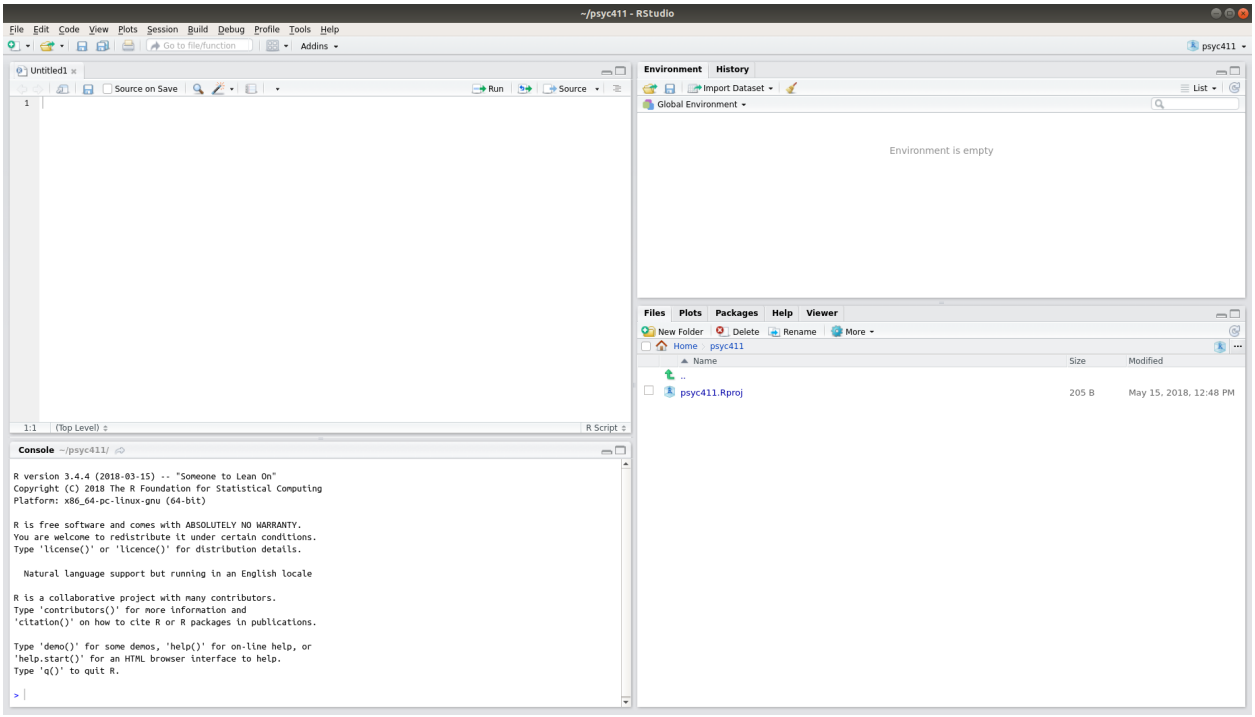



Figure 9: Project created

in the colour of the text in these boxes can be ignored.

New commands in the worksheets are followed by one or more *explanation* sections - those are there to help you understand how the commands work and how to read the output they produce.

In some cases there will also be hidden answers or additiona explanations. To see these, you can see

Click the boxes like this

And text will appear!

---

In other places you will see sections in blue, like this:

These sections are exercises for you to complete. The are the most important sections in the worksheet, and you should always work through all of these problems.

---

Finally, sometimes you will see questions with text boxes next to them. You just need to type in the right answer (42), and the box outline will go blue:

**The answer to life, the universe and everything is:**

Other questions might ask you to make calculations. For these the text box acts like a mini calculator. Try typing `1000 * 1024` into this box:

Why do we use `*` and not $\times$?).

In most computer languages (and also in Excel) the arithmetic operators are:

- `*` for multiplication
- `/` for division
- `^` for powers (e.g. `2^4` $= 2^4 = 16$)

### Loading a package

First, we need to load a package called `tidyverse`. A package is an extension to R that adds new commands. Nearly everything we'll do in this course uses the tidyverse package, so pretty much every project starts with this instruction.

**Type (or copy and paste) the command in the grey box into line 1 of the *Script* window of RStudio**. Now, with your cursor still on line 1, press CTRL+ENTER (i.e. press the key marked 'Ctrl' and the RETURN or ENTER key together).

```
library(tidyverse)
```

When you do this, line 1 is automatically copied to your *Console* window and run. Then, RStudio will print some text to the Console (shown in the white box, above). This text tells you that the *tidyverse* package has loaded ("attached") some other pacakges (e.g. *dplyr*). It also tells you that the *dplyr* package changes the way some commands in R work ("conflicts"). That's OK.

If you get an output that includes the word 'error', please see the common problems section.

### Saving your script

You should notice that the name `Untitled1` on the *Script* window has now gone red. This is to remind you that your script has changed since the last time you saved it. So, click on the "Save" icon (the little floppy disk) and save your R script with some kind of meaningful name, for example `briefguide.R`.

The `.R` indicates that it is an R script.

Re-save your script each time you change something in it; that way, you won't lose any of your work.

### Loading data

Now, we're going to load some data on the income of 10,000 people in the United States of America. I've made up this dataset for teaching purposes, but it's somewhat similar to large open data sets available on the web, such as US Current Population Survey. Here's how you get a copy of this data into RStudio so you can start looking at it.

This video runs through the steps described below:

1. Download a copy of the data, by clicking here and saving it to the Downloads folder of your computer.
2. Go to RStudio in your web browser.
3. Click on the 'Files' tab in RStudio (bottom right rectangle)
4. Click the 'Upload' button.
5. Click 'Browse…'
6. Go to your Downloads folder, and select the file you just saved there.
7. Click "OK".
8. Copy or type the following command into your RStudio script window, and run it (i.e. press CTRL+ENTER while your cursor is on that line)

```
cpsdata <- read_csv("cps2.csv")
```

**Explanation of the command** There are three parts to the command `cpsdata <- read_csv("cps2.csv")`:

1. The first part of the command is `cpsdata`. This gives a name to the data we are going to load. We'll use this name to refer to it later, so it's worth using a name that is both short and meaningful. I've called it `cpsdata` because it's somewhat similar to data from the US Current Population Survey, but you can give data pretty much any name you choose (e.g. *fart*).

2. The bit in the middle, `<-`, is an arrow and is typed by pressing `<` and then `-`, without a space. This arrow means "put the thing on the right of the arrow into the thing on the left of the arrow". In Rstudio
3. The last part of the command is `read_csv("cps2.csv")`. It loads the data file into `cpsdata`. The part inside the speech marks, `cps2.csv`, is the name of the file you just uploaded to your RStudio project. This command can also download data directly from the web, for example `read_csv("http://www.willslab.org.uk/cps2.csv")`. This would have been a quicker way to do it in this case, but of course not all data is on a web page.

**Explanation of the output** R likes to print things in red sometimes – this does not mean there's a problem. If there's a problem, it will actually say 'error'. The output here tells us that R has loaded the data, which has eight parts (columns, or `cols`). It gives us the name of the columns (`ID, sex, ...`) and tells us what sort of data each column contains: `character` means the data is words (e.g. 'female'), `double` means the data is a number (e.g. '42.78') (more about the different types of variables).

If you get an error here, please see common errors.

## Inspecting data

Make sure you have just completed the loading data worksheet and have the `cps2` dataset loaded.

Next, we'll take a peek at these data. You can do this by clicking on the data in the *Environment* tab of RStudio, see here.

We can now see the data set (also known as a *data frame*). We can see that this data frame has 8 columns and 10000 rows. Each row is one person, and each column provides some information about them. Below is a description of each of the columns. Where you see `NA` this means this piece of data is missing for this person – quite common in some real datasets.

Here's what each of the columns in the data set contains:

| Column | Description | Values |
|---|---|---|
| ID | Unique anonymous participant number | 1-10,000 |
| sex | Biological sex of participant | male, female |
| native | Participant born in the US? | foreign, native |
| blind | Participant blind? | yes, no |
| hours | Number of hours worked per week | a number |
| job | Type of job held by participant: | charity, nopay, private, public |
| income | Annual income in dollars | a number |
| education | Highest qualification obtained | grade-school, high-school, bachelor, master, doctor |

### Calculating a mean

One question we can ask of these data is "what is the average income of people in the U.S.?" (or, at least, in this sample).

In this first example, we're going to calculate the *mean* income.

As you know, you calculate a mean by adding up all the incomes and dividing by the number of incomes. Our sample has 10,000 participants, so this would be a long and tedious calculation – and we'd probably make an error.

It would also be a little bit tedious and error prone in a spreadsheet application (e.g. Excel, Libreoffice Calc). There are some very famous cases of these kinds of "Excel errors" in research, e.g. genetics, economics.

In R, we can calculate the mean instantly, and it's harder to make the sorts of errors that are common in Excel-based analysis.

To calculate mean income in R, we add the following command to our script and press CTRL+ENTER:

```
cpsdata %>%
   summarise(mean(income))
> # A tibble: 1 x 1
>   `mean(income)`
>            <dbl>
> 1         87293.
```

Your output will tell you the mean income in this sample – it's the last number on the bottom right, and it's approximately $87,000.

If you're happy with the output you've got, move on to the next section. If you would like a more detailed explanation of this output, see more on tibbles. We'll cover this later anyway.

If you get an error here, please see common errors.

**Explanation of the command** This command has three components:

1. The bit on the left, `cpsdata`, is our data frame, which we loaded and named earlier.
2. The bit in the middle, `%>%`, is called a *pipe*. Its job is to send data from one part of your command to another. It is typed by pressing `%` then `>` then `%`, without spaces. So `cpsdata %>%` sends our data frame to the next part of our command. See how to type this quickly
3. The bit on the right, `summarise(mean(income))` is itself made up of parts. The command `summarise` does as the name might suggest: it summarises a set of data (`cpsdata` in this case) into a single number, e.g. a mean. The `mean` command indicates that the type of summary we want is a mean (there are other summaries, as we will cover later). Finally, `income` is the name of the column of `cpsdata` we want to take the mean of – in this case, the income of each individual.

Make sure you are 100% clear about the difference between `<-` and `%>%`. If you're not, ask for an explanation in class now.

The main clue is to look at the direction of the arrows:

`%>%` sends data from left to right. We call this ***'piping'***.

`<-` sends results from the right hand side, to a variable named on the left. This is called ***assignment***.

---

Watch out that `->` is not the same as `%>%`. The thin arrow is always for assignment. You won't see if often, because it's normally considered bad manners to use thin right arrows like this (they get confusing).

---

**It really is worth learning the keyboard shortcuts for `<-` and `%>%` — you will be typing them a lot during the course.**

## Missing data

Make sure you have recently completed the loading data worksheet and have the `cps2` dataset is loaded.

To calculate the mean number of hours worked per week, we have to deal with the fact that there is some missing data - we don't know for all 10,000 people how many hours they work in a week, because they didn't all tell us.

To get a mean of those who did tell us, we tell R to ignore the missing data, like this:

```
cpsdata %>% summarise(mean(hours, na.rm = TRUE))
> # A tibble: 1 x 1
>   `mean(hours, na.rm = TRUE)`
>                         <dbl>
> 1                        38.9
```

**Explanation of the command** `rm` is short for 'remove', but 'ignore' would be a more accurate description, as this command doesn't delete the `NA` entries in `cpsdata`, it just ignores them. So `na.rm = TRUE` means "ignore the missing data".

If you get an error here, please see common errors.

### Patterns in missing data

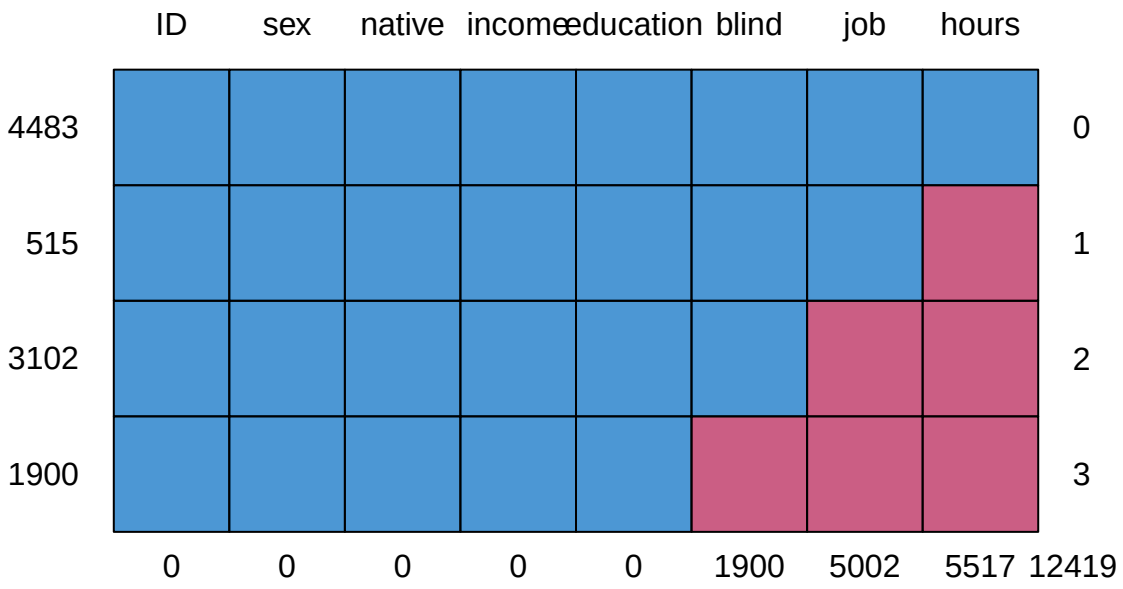
Sometime we won't only want to ignore missing data. We might also want to **count** how how many variables are missing. The `mice` package has a useful command for doing this.

First, we need to load `mice` like the did `tidyverse` above. Type (or copy and paste) the command below into your R script and run it:

```
library(mice)
```

Then we can use the `md.pattern()` function to describe the patterns of missing data:

```
cpsdata %>% mice::md.pattern()
```




```
>      ID sex native income education blind  job hours
> 4483  1   1      1      1         1     1    1     1     0
> 515   1   1      1      1         1     1    1     0     1
> 3102  1   1      1      1         1     1    0     0     2
> 1900  1   1      1      1         1     0    0     0     3
>       0   0      0      0         0  1900 5002  5517 12419
```

**Explanation of the output** `md.pattern()` produces two outputs: a plot, and a table.

In the plot we see

- The variables in the dataset listed along the top

- Squares indicating whether a variable is recorded (blue) or missing (purple/red)

Each row in the plot is a missing data ***pattern***. So:

- In the first row, the pattern is that all variables are recorded
- The pattern in the second row is that all variables bar `hours` were recorded
- The third pattern is that `job` and `hours` were missing, and so on.

The numbers on the left of the plot indicate ***how many people fit the pattern***. So:

- 4483 people had complete data (pattern 1)
- 515 people had complete data except for the `hours` variable (pattern 2)
- 3102 people were missing `hours` and `job` (pattern 3, and so on

This can be really helpful when checking whether data has been imported properly, or properly reporting missing data from our experiments (see MacPherson et al. 2010 for current guidelines for clinical trials, which would also be good practice for experimental research).

The numbers along the bottom of the plot show ***how many missing observations there were for the variable marked at the top***. So:

- There were 5517 missing observations for `hours`, across all participants
- 5002 for `job`, and so on.

The table provides the same information as the plot, but is perhaps harder to read.

#### Task: Try this on another dataset

The `mice` package includes example datasets with missing data. We can look at one of these like so:

```
mice::boys %>% glimpse
> Observations: 748
> Variables: 9
> $ age <dbl> 0.035, 0.038, 0.057, 0.060, 0.062, 0.068, 0.068, 0.071, 0....
> $ hgt <dbl> 50.1, 53.5, 50.0, 54.5, 57.5, 55.5, 52.5, 53.0, 55.1, 54.5...
> $ wgt <dbl> 3.650, 3.370, 3.140, 4.270, 5.030, 4.655, 3.810, 3.890, 3....
> $ bmi <dbl> 14.54, 11.77, 12.56, 14.37, 15.21, 15.11, 13.82, 13.84, 12...
> $ hc  <dbl> 33.7, 35.0, 35.2, 36.7, 37.3, 37.0, 34.9, 35.8, 36.8, 38.0...
> $ gen <ord> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
> $ phb <ord> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
> $ tv  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
> $ reg <fct> south, south, south, south, south, south, south, west, wes...
```

Identify the patterns of missing dataset in the `mice::boys` dataset.

1. How many participants provided complete data?

## Group differences (briefly)

#### Before you start...

Before starting this worksheet, you should have had a brief introduction to using RStudio – Using RStudio. You should also have also completed the worksheet Exploring Data. If not, take a look these earlier worksheets before continuing.

If you have completed those worksheets, then you'll have set up an R project, and you'll have a script in it that looks something like this:

```
library(tidyverse)
library(mice)
cpsdata <- read_csv("cps2.csv")
```

```
cpsdata %>% summarise(mean(income))
cpsdata %>% summarise(mean(hours, na.rm = TRUE))
```

In this worksheet, we'll add some more commands to this script.

### Contents



### Grouping data

One of the most widely discussed issues concerning income is the difference between what men and women, on average, get paid. Let's have a look at that difference in our teaching sample of 10,000 US participants.

In order to do this, we need to split our data into two groups – males and females. In R, the command `group_by` allows us to do this. In this case, we want to group the data by biological sex, so the command is `group_by(sex)`. We *pipe* (`%>%`) the data in `cpsdata` to the `group_by` command in order to group it, and then we *pipe* (`%>%`) it to `summarise` to get a summary for each group (a mean, in this case). So, the full command is:

```
cpsdata %>% group_by(sex) %>% summarise(mean(income))
> # A tibble: 2 x 2
>   sex    `mean(income)`
>   <chr>           <dbl>
> 1 female         82677.
> 2 male           92137.
```

Copy it into your script and run it (CTRL+ENTER). Women in our made-up sample get paid, on average, around 9,000 (9k) less than men. Of course, not every male gets 92k a year in the US, and not every female gets 83k. It seems very likely that the range of incomes earned by men and women overlap – meaning that if you picked one man and one woman at random, there's a reasonable chance that the woman earns more than the man. We can look at this *variation* in pay using a graph.

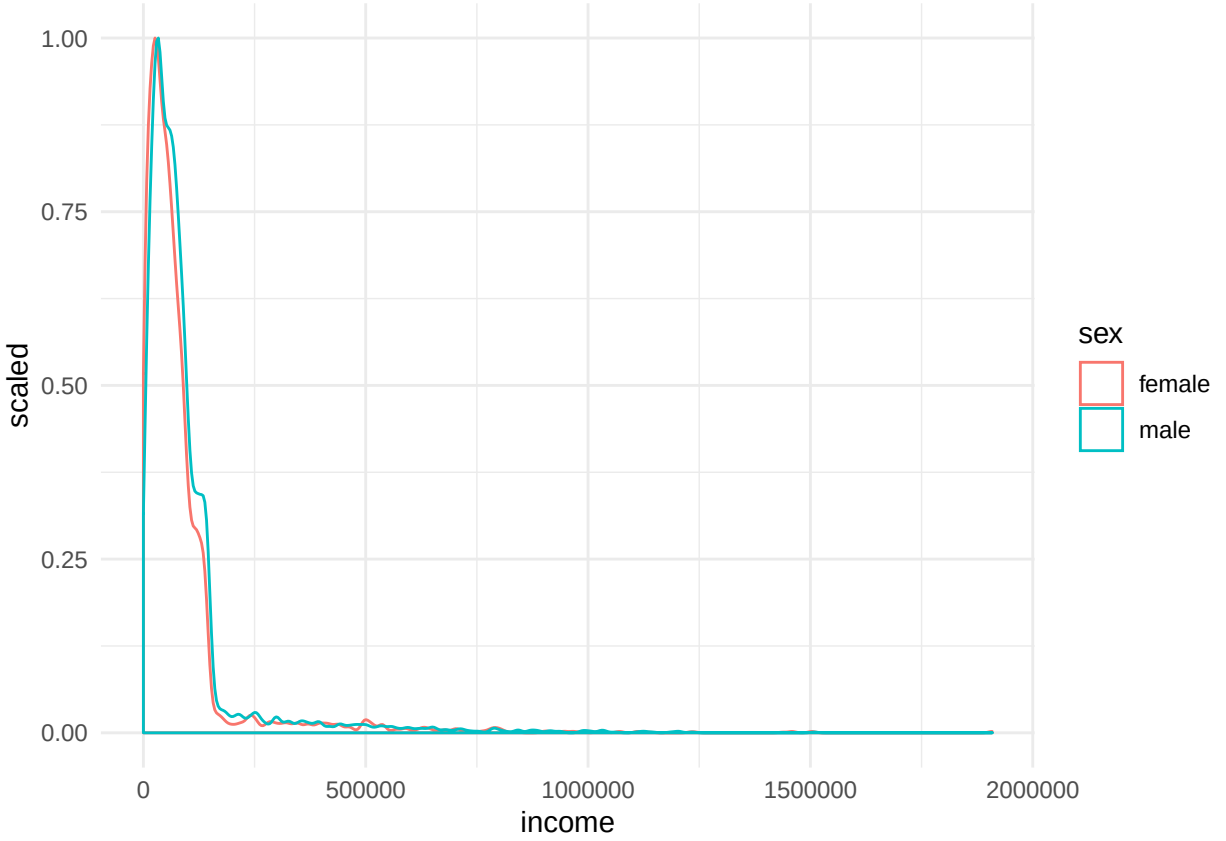
### Looking at variation using a density plot

The graph we're going to draw is a density plot. If you recall histograms from school, it's a lot like that. If not, don't worry. A density plot is a curve that shows how likely a range of incomes are. So, the higher the curve is at a particular income, the more people who have that income.

We're going to produce what's called a *scaled* density plot. The highest point on a scaled density plot is always one. This can make it easier to compare two groups, particularly if one group has fewer people in it than the other.

So here's the command to do a scaled density plot for incomes, plotting men and women separately. Copy it into your script and run it (CTRL+ENTER).

```
cpsdata %>%
   ggplot(aes(income, colour=sex)) +
   geom_density(aes(y=..scaled..))
```




**Explanation of command** Here's what each part of this command means:

- `cpsdata` - The data frame containing the data. You created this in the last worksheet.
- `%>%` - A pipe. As in the last worksheet, this pipe carries the data in `cpsdata` to the next part of the command, which does something with it.
- `ggplot()` - This means 'draw me a graph'. All graphs we use in these worksheets use the *Grammar for Graphics* (gg) plotting commands, so they'll all include the command `ggplot`.
- `aes()` - Short for *aesthetics* (what things look like). It means 'This is the sort of graph I want'.
- `income` - I want a graph of the data in the `income` column of `cpsdata`
- `color=sex` - I want you to give me two graphs on top of each other, in different colours. One colour for men, a different color for women. Use the `sex` column of `cpsdata` to work out who is male and who is female.
- `geom_density()` - I want this graph to be a *density* plot.
- `aes(y=..scaled..)` - I want this density plot to be *scaled* (see above).

**Discussion of output** Your graph will appear in the bottom-right window, and should look like the one above. You'll notice that the two lines seem basically on top of each other … but they can't be because we know the two groups differ in mean income by over nine thousand dollars! We have a problem to solve…

### Dealing with extreme data points

The problem is one of scale – there are a small number of people who earn very high salaries. In fact, both the highest-paid man, and the highest-paid woman in our sample earn considerably more than one million dollars a year.

**Filtering data** Somehow, we need to deal with the fact that a few people in our sample are very well paid, which makes the difference between men and women hard to see on our graph, despite the difference being over nine thousand dollars a year.

One of the easiest ways around this is to exclude these very high salaries from our graph.

The vast majority of people are paid less than 150k a year. So, let's restrict our plotting to just those people. We do this using the `filter` command. It's called *filter* because it works a bit like the filter paper in a chemistry lab (or in your coffee machine) – stopping some things, while letting other things pass through. We can filter our data by telling R *what data we want to keep*. Here, we want to keep all people who earn less than £150k, and filter out the rest. So the filter we need is `filter(income < 150000)`, where `<` means "less than".

We'll be using this dataset of people with <$150k incomes a few times, so we're going to give it a new name, `cpslow` (or any other name you want, e.g. *angelface* )

So, what we need to do is *pipe* (`%>%`) our `cpsdata` data to our `filter(income < 150000)`, and use an arrow, `<-`, to send this data to our new *data frame*, `cpslow`. Recall that `<-` sends the thing on its right to the thing on its left, so the full command is:

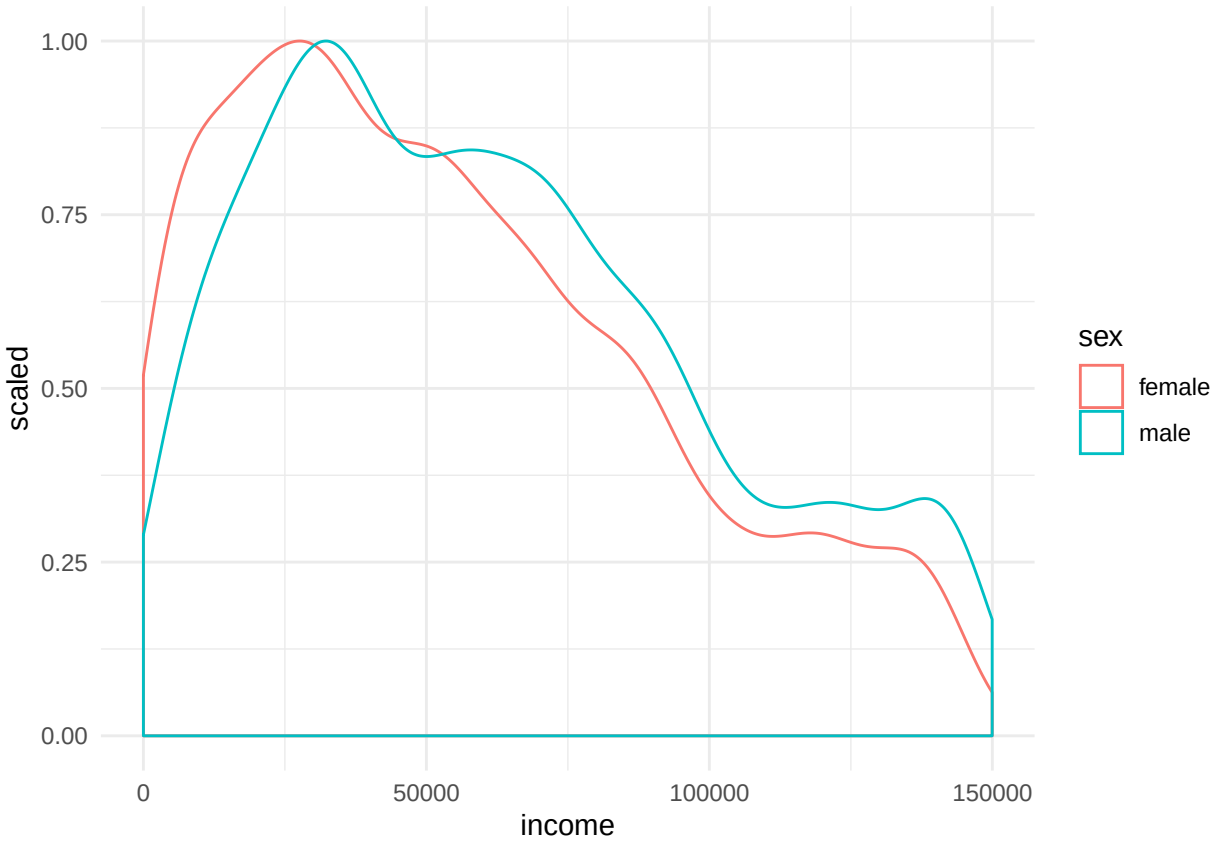
```
cpslow <- cpsdata %>% filter(income < 150000)
```

We can take a look at this new *data frame* by clicking on it in RStudio's *Environment* window (see video here if you're not sure how). By looking at the ID numbers, you can see that some people in our original sample have been taken out, because they earned at least 150k.

Now, we can plot these filtered data in the same way as before, by changing the name of the dataframe from `cpsdata` to `cpslow`.

**So start with the command** `cpsdata %>% ggplot(aes(income, colour=sex)) + geom_density(aes(y=..scaled..))`, copy it onto the next line in your script, **make that change, and press CTRL+RETURN.**

If you've got it right, your graph will look like this:




At first glance, the two distributions of incomes still look similar. For example, the *modal* income is at quite a low income, and that income is quite similar for both men and women. However, on closer inspection, you'll also see that the red line (females) is above the blue line (men) until about 25-50k, and below the blue line from then on. This means that more women than men earn less than 50k, and more men than women earn more than 50k.

So, the gender pay gap is visible in this graph. The graph also illustrates that the difference in this sample is small, relative to the range of incomes. This doesn't mean that the gender pay gap is less (or more) important than income inequality. These kinds of questions of importance are moral, philosophical, and political. Data cannot directly answer these kinds of questions, but they can provide information to inform the debate.

As we'll see later, this type of graph is also crucial to inform our choice of statistical models (like regression or Anova): Without a clear sense of what the data *look* like we can make bad decisions in our analyses.

**Exercise**

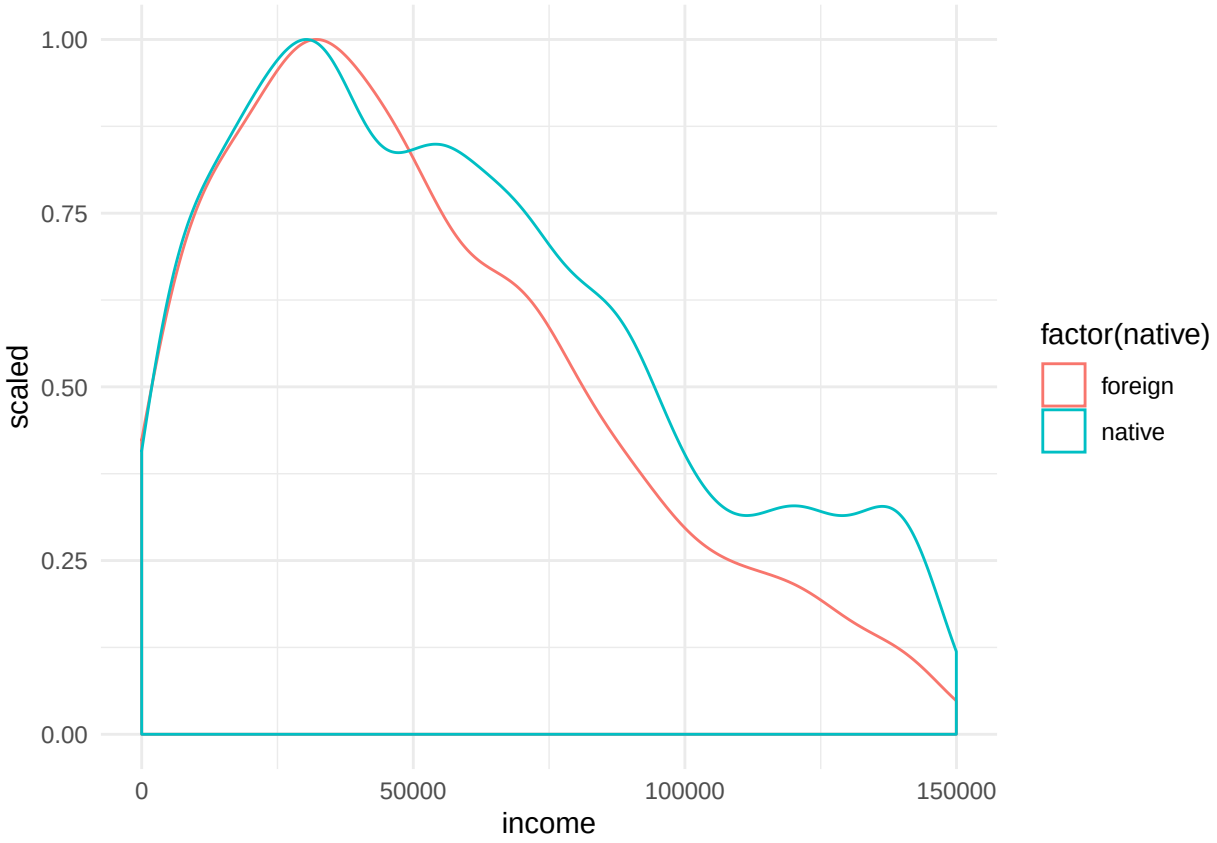
This exercise consolidates what you've learned so far.

**The task is to further examine the sub-sample of participants who are living in the US, and earning less than $150k (`cpslow`).**

Specifically, the question to answer is whether people born in the US earn more. In order to do this, you should calculate the mean income for each group, and produce a density plot with one line for each group. Below are the answers you are aiming for:

```
> # A tibble: 2 x 2
>   native  `mean(income)`
>   <chr>            <dbl>
> 1 foreign         78408.
```

```
> 2 native          88480.
```




Previously, we calculated the mean salary of men and women.

1. Why might it be a better idea to calculate the median?

Show answer

Because the data are strongly skewed, the median may be a better summary of the central tendency (the middle).

2. Adapt the commands above to calculate the median instead. What is the median salary for women: , and for men: .

Show answers

```
cpsdata %>%
  group_by(sex) %>%
  summarise(med=median(income))
> # A tibble: 2 x 2
>   sex       med
>   <chr>   <dbl>
> 1 female 52558.
> 2 male   61746.
```

**Extension exercise**

If you've some spare time and are looking for something a bit more challenging, try Exercise 2 on this slightly more advanced worksheet.

## Undergraduate stats in R

All of the statistics you will have learned at undergraduate level can be produced in R. Here we cover simple examples of:

- A t-test
- A correlation

If you have no memory of t-tests or correlations, you might want to take time to work through these expanded guides from our undergraduate course at a later date:

- Tests of group differences
- Relationships

We'll run these statistics on an example dataset which is built into R, called `mtcars`. We can look at this data using the glimpse function (this is loaded with `tidyverse`, so if you get an error make sure that is loaded too):

```
mtcars %>% glimpse()
> Observations: 32
> Variables: 11
> $ mpg  <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19....
> $ cyl  <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, ...
> $ disp <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 1...
> $ hp   <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, ...
> $ drat <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.9...
> $ wt   <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3...
> $ qsec <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 2...
> $ vs   <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, ...
> $ am   <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, ...
> $ gear <dbl> 4, 4, 4, 3, 3, 3, 3, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 4, 4, ...
> $ carb <dbl> 4, 4, 1, 1, 2, 1, 4, 2, 2, 4, 4, 3, 3, 3, 4, 4, 4, 1, 2, ...
```

**Explanation of the `glimpse` output** `glimpse` produces a list of all variables in the dataset, tells us what type they are, and lists however many obserervations from the dataset that will fit on a single line.

The type of all variables in `mtcars` is `dbl`. This is short for 'double-precision number'; for now, just know that `dbl` means a *number*.

Other types include :

- `int` — short for 'integer' variable, so only contains whole numbers (e.g. a participant id number)
- `chr` — short for 'character variable', which will contain text (e.g. an email address)
- `fct` — short for 'factor'. i.e. a categorical variable (e.g. MCQ responses)
- `ord` — short for 'ordered'. This is variant of categorical variable where the categories have a particular order (responses like "Wost" < "Better" < "Best" could be stored as an `ord`)

### Two sample t-test

`mtcars` contains a variable called `mpg`, which is the miles per gallon each car will do, and another called `am` which is encodes whether it was a manual or automatic transmission (0=automatic, 1=manual).

We can test if `mpg` differs between auto and manual cars with `t.test`:

```
t.test(mpg ~ am, data=mtcars)
>
>   Welch Two Sample t-test
>
> data:  mpg by am
```

```
> t = -3.7671, df = 18.332, p-value = 0.001374
> alternative hypothesis: true difference in means is not equal to 0
> 95 percent confidence interval:
>  -11.280194  -3.209684
> sample estimates:
> mean in group 0 mean in group 1
>        17.14737        24.39231
```

### Explanation

The command contains three parts:

- `t.test`: Says what we want to do
- `mpg ~ am`: This is a 'formula', which tells `t.test` which variables to analyse.
- `data=mtcars`: Which dataset we want to use for the analysis

The formula is split into two parts by the `~` symbol. On the left is our outcome. On the right is the grouping variable, which we hope **predicts** the outcome.

In the output you can see the test statistic, degrees of freedom and $p$ value.

The tilde symbol. Pronounced "tilder".

In R, `~` almost always means ***"is predicted by"***.

### Correlations

The `mtcars` data also contains variables for weight (`wt`) and power (`hp`, short for horsepower).

We can select just these columns and save them to a smaller dataframe like this:

```
carperformance <- mtcars %>% select(mpg, wt, hp)
```

**Explanation of the commands** On the far left we have the name of the new variable which we will create: `carperformance`.

We can tell this will be a new variable because the `<-` symbol is just to the right, pointing at it.

To work out what `carperformance` will contain, we look to the right of the `<-` There are two parts here, linked with the pipe symbol (`%>%`) which passes data from one command to the next, from left to right.

First we see the `mtcars` data. Using a pipe we pass this to the `select` command, which selects the `mpg`,`wt`, and`hp` columns.

**Explanation of the result** When running the command you won't see any output — but something **has** happened behind the scenes: A new object was created called `carperformance` which contained copies of the columns from `mtcars` we selected.

We can see the first few rows of our new smaller dataframe like this:

```
carperformance %>% head()
>                   mpg    wt  hp
> Mazda RX4         21.0 2.620 110
> Mazda RX4 Wag     21.0 2.875 110
> Datsun 710        22.8 2.320  93
> Hornet 4 Drive    21.4 3.215 110
> Hornet Sportabout 18.7 3.440 175
> Valiant           18.1 3.460 105
```

To correlate the three columns in this dataset, we can use the `cor` function and round all the results to 2 decimal places:

```
carperformance %>% cor() %>% round(2)
>       mpg    wt    hp
> mpg  1.00 -0.87 -0.78
> wt  -0.87  1.00  0.66
> hp  -0.78  0.66  1.00
```

Explain those commands…

On the left we have the `carperformance` data.

We pipe this to the `cor` function which calculates the correlation between each pair of columns and returns a special kind of table, called a matrix.

To make the output simpler, we then pass the results to the `round` function, which rounds all the results to 2 decimal places.

The `cor` function is pretty bare-bones, and doesn't produce output we could easily use in a report or article. The `apaTables` package helps us with this:

```
apaTables::apa.cor.table(carperformance, filename = "correlations.doc")
>
>
> Means, standard deviations, and correlations with confidence intervals
>
>
>   Variable M      SD    1            2
>   1. mpg   20.09  6.03
>
>   2. wt    3.22   0.98  -.87**
>                         [-.93, -.74]
>
>   3. hp    146.69 68.56 -.78**       .66**
>                         [-.89, -.59] [.40, .82]
>
>
> Note. M and SD are used to represent mean and standard deviation, respectively.
> Values in square brackets indicate the 95% confidence interval.
> The confidence interval is a plausible range of population correlations
> that could have caused the sample correlation (Cumming, 2014).
> * indicates p < .05. ** indicates p < .01.
>
```

Explain the double colons (`::`) in the code above

Sometimes we load a whole package, as we did when we wrote `library(tidyverse)` above. This is a good idea when we want to use lots of functions from that package.

When we only want to use one function from a package we can type `nameofpackage::nameoffunction` and this lets us use the function without loading the package.

This can be a good idea if the package or function is less well known, and you want to be explicit about which package it comes from—it helps 'future-you' work out what your code is doing.

**Explanation of the result** We used the `apa.cor.table` function within the `apaTables` package to create a nicely-formatted correlation table, in APA format.

We also specified a `filename`, and `apa.cor.table` created a Word document with this name containing the formatted table (click to see the result).

- Use one of the other built-in datasets in R to run a correlation between 2 variables.
- Use the built-in `sleep` data. Compute a t-test comparing the `extra` variable between groups. Describe the results of the t-test in APA format.

**Built in datasets** Some examples of built-in data are:

- `sleep`
- `iris`
- `airquality`
- `ChickWeight`
- `diamonds`

If you have loaded tidyverse you can access these by name, just like the `mtcars` data. For example::

```
diamonds %>% glimpse()
> Observations: 53,940
> Variables: 10
> $ carat   <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, ...
> $ cut     <ord> Ideal, Premium, Good, Premium, Good, Very Good, Very G...
> $ color   <ord> E, E, E, I, J, J, I, H, E, H, J, J, F, J, E, E, I, J, ...
> $ clarity <ord> SI2, SI1, VS1, VS2, SI2, VVS2, VVS1, SI1, VS2, VS1, SI...
> $ depth   <dbl> 61.5, 59.8, 56.9, 62.4, 63.3, 62.8, 62.3, 61.9, 65.1, ...
> $ table   <dbl> 55, 61, 65, 58, 58, 57, 57, 55, 61, 61, 55, 56, 61, 54...
> $ price   <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339,...
> $ x       <dbl> 3.95, 3.89, 4.05, 4.20, 4.34, 3.94, 3.95, 4.07, 3.87, ...
> $ y       <dbl> 3.98, 3.84, 4.07, 4.23, 4.35, 3.96, 3.98, 4.11, 3.78, ...
> $ z       <dbl> 2.43, 2.31, 2.31, 2.63, 2.75, 2.48, 2.47, 2.53, 2.49, ...
```

You can find out more about each of them by typing:

```
help(iris)
```

## Sequences and designs

One trick you will need later in the course is making **sequences** of numbers.

There are a few ways to do this, but the simplest is to write: `1:10`. That is, the number to start from (`1`), a colon (`:`), and then the number to end with (`10`).

Copy and paste these examples to see the output:

```
1:10
>  [1]  1  2  3  4  5  6  7  8  9 10
20:30
>  [1] 20 21 22 23 24 25 26 27 28 29 30
```

**Explanation**: The output shows that R has created a sequence of whole numbers between the start and finish number.

To get a sequence with only even numbers, we can use the `seq` function, and set the `by` argument to 2:

```
seq(from=2, to=10, by=2)
> [1]  2  4  6  8 10
```

You can set `by` to any number, including a decimal:

```
seq(0, 27, by=3)
>  [1]  0  3  6  9 12 15 18 21 24 27
seq(0, 1, by=0.2)
> [1] 0.0 0.2 0.4 0.6 0.8 1.0
```

If your sequence doesn't have a simple pattern, you can also write out the numbers by hand using the `c(..)` command:

```
c(1,40,92,188)
> [1]   1  40  92 188
```

**Explanation**: `c(...)` is short for `combine`, so this command combines the numbers 1, 40, 92, 188 into a new sequence. This is sometimes called a **vector** in R-speak.

Make some sequences which include:

- Even numbers from 10 to 20
- Numbers in the 8 times table less than 200
- 20 evenly spaced numbers between zero and 1 (including zero and 1)
- The words "Wibble", "Wobble" and "Bobble"

Show answers

We can use `seq` for numbers:

```
seq(10,20,by=2)
> [1] 10 12 14 16 18 20
seq(0,200, 8)
>  [1]   0   8  16  24  32  40  48  56  64  72  80  88  96 104 112 120 128
> [18] 136 144 152 160 168 176 184 192 200
seq(0,1, by=1/19)
>  [1] 0.00000000 0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
>  [7] 0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
> [13] 0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684
> [19] 0.94736842 1.00000000
```

But we need to use c() for lists of words:

```
c("Wibble", "Wobble", "Bobble")
> [1] "Wibble" "Wobble" "Bobble"
```

### Combinations of sequences

In designing experiments we often want to create combinations of different categories which represent conditions or stimuli.

Imagine a hypothetical study with a test phase where participants are presented with multiple words, in either red or green text, and shown at either the bottom or top of the computer screen.

The combinations look something like this:

| condition | colour | position | word |
|---|---|---|---|
| 1 | Red | Top | Nobble |
| 2 | Green | Top | Nobble |
| 3 | Red | Bottom | Nobble |
| 4 | Green | Bottom | Nobble |
| 5 | Red | Top | Wobble |
| 6 | Green | Top | Wobble |
| 7 | Red | Bottom | Wobble |

| condition | colour | position | word |
|---|---|---|---|
| 8 | Green | Bottom | Wobble |
| 9 | Red | Top | Hobble |
| 10 | Green | Top | Hobble |
| 11 | Red | Bottom | Hobble |
| 12 | Green | Bottom | Hobble |

R provides quick ways of creating combinations of variables, using a command called `expand.grid`.

First, we need to create a sequence of each of the possible values for our categories:

```
colours = c("Red", "Green")
positions = c("Top", "Bottom")
words = c("Nobble", "Wobble", "Hobble")
```

Then we can use `expand.grid` to give us all the possible combinations of these:

```
expand.grid(colour=colours, position=positions, words = words)
>    colour position  words
> 1     Red      Top Nobble
> 2   Green      Top Nobble
> 3     Red   Bottom Nobble
> 4   Green   Bottom Nobble
> 5     Red      Top Wobble
> 6   Green      Top Wobble
> 7     Red   Bottom Wobble
> 8   Green   Bottom Wobble
> 9     Red      Top Hobble
> 10  Green      Top Hobble
> 11    Red   Bottom Hobble
> 12  Green   Bottom Hobble
```

**Explanation**: The `expand.grid` function has taken the items in the three input sequences (colours, positions and words) and created a dataframe which contains all the possible combinations. We could save these to a file if we wanted to use them as part of our experiment.

### Task: create some experimental designs of your own

1. Reproduce the experiment design above by copying and pasting
2. Adapt the commands to allow for an experiment where the word position could be either top, bottom, left or right. How many different conditions would there be in this case?

As an optional stretch task:

1. How would you create a design where the order of presentation of each word is also balanced? That is, where it's equally likely to see Nobble, Wobble or Bobble in the first, second or third trial?

### Random samples

In our example, each of the experimental conditions is allocated between-participants, which (as you'll learn elsewhere in the programme) means we need recruit quite a large number of participants. In this case we can imagine we might need at least 250 participants.

Allocating participants to condition is another boring task R can help with. The trick is to combine sequences with randomness and random sampling.

For example, we might create a sequence of the numbers from 1:10:

```
1:10
>  [1]  1  2  3  4  5  6  7  8  9 10
```

The `sample` function lets us take a random sample from this sequence:

```
sample(1:10)
>  [1]  7  6  4  2  3  5  8  1 10  9
```

**Explanation** `sample` has shuffled the input sequence and gives us the original numbers (1…12) in a random order.

---

If we want to take a larger sample, longer than the input sequence, we have to add the text `replace=TRUE`:

```
sample(1:10, size = 20, replace=TRUE)
>  [1]  6  1  6  6  5  8  4  7  7  4 10  7  7  5  9 10 10  6  3  6
```

**Explanation**: Using `replace=TRUE` is like picking a number from a hat, but then putting the chosen number back into the hat so it can be picked again. This allows us to make longer random sequences where numbers in the initial sequence are repeated.

### In-use: Randomising participants to conditions/groups

If we combine our experimental design from above with this idea of randomness we can randomise participants to conditions in our experiment.

If we look again at our design we can see there are 12 possible combinations:

```
expand.grid(colour=colours, position=positions, words = words)
>    colour position  words
> 1     Red      Top Nobble
> 2   Green      Top Nobble
> 3     Red   Bottom Nobble
> 4   Green   Bottom Nobble
> 5     Red      Top Wobble
> 6   Green      Top Wobble
> 7     Red   Bottom Wobble
> 8   Green   Bottom Wobble
> 9     Red      Top Hobble
> 10  Green      Top Hobble
> 11    Red   Bottom Hobble
> 12  Green   Bottom Hobble
```

We can use `sample` to make sure we allocate participants randomly to conditions, as they are recruited:

```
sample(1:12, size=250, replace=TRUE)
>   [1]  9  4  9 12  2  9 12  3 10  3  1 11  5  2  5  9  6  9  3  2 11 11  5
>  [24]  2  5  5  4  8  7  9  5  5  9 12  4 11  2 11  1 12  7 11  4  3  7  6
>  [47]  3 12  6  8  7  2  6  1  5  8 12  4 11  1  3  4 10 12  3  3  7 10  2
>  [70] 10  8 12 11  6  6  6  8 11  8  2  4  9  5  6  2  3  6  5  4  8  1  1
>  [93]  8  2  1 12 12  8  9  7 10  1  9  2 12  2  9  7  8  7  5  4  1 11  2
> [116]  8  2 10  8  8  1 10 10  1  1  7  4 11  6 11  4 10 11  9  9  2  1  1
> [139]  3  2 10 12  5  3 10  6  4  5  9 10  6  6  6  2  3 11  9  4 12  5 12
> [162] 12 12  2 11  6  3  6  4  9 10 11  8  2  5  9  7  4  8  8  5  8 12 10
> [185]  6 10 12  2  1  5 12 10 11  4  1 11  5 12 11  1  6  4  4  4  4  9  1
> [208]  7  1 10 11  9  3  7  3  7  2  8 11 10  2  2 12 12  5 10  3  6  6  7
> [231]  9 10  5  2  7  5 11  6  5  2  4  9  8  9  9  6 11  1  9  9
```

So, we would allocate participants to their condition in this order as they are recruited.

Alternatively, we could simply run this every time a new participant showed up:

```
(new_allocation <- sample(1:12, size=1))
> [1] 11
```

**Task: Randomise your own participants**

1. Re-use the design you created above, and randomise 5 participants to conditions within it.
2. Try and create a design which might work for your own study (either this year, based on your UG project, or on any other study you might like to run).

As an optional stretch task:

1. Read this extension, which shows how to sample from datasets as well as sequences

## Reading data

So far, we have read data from CSV files. In this section we show how to import some other common types.

### Reading CSV data directly from a URL (place on the web)

When you find data on the web it can save time to read it straight into R, without worrying about downloading and uploading the files. If we do this, it's good to save a copy of the data in case the website disappears or the file is taken down.

For example, we previously downloaded data from http://www.willslab.org.uk/cps2.csv and then used `read_csv()` on this downloaded copy. We can skip a few steps by loading it directly by writing:

```
datafromtheweb <- read_csv('http://www.willslab.org.uk/cps2.csv')
```

**Explanation**: By providing `read_csv()` with a URL (a web address) R has automatically imported the data from http://www.willslab.org.uk/cps2.csv, and saved it to a new variable called `datafromtheweb`.

---

We might also want to save a copy of the data, in case the website disappears in future. In my code, I would typically write:

```
# datafromtheweb <- read_csv('http://www.willslab.org.uk/cps2.csv')
# write_csv(datafromtheweb, 'cps2.csv')
datafromtheweb <- read_csv('cps2.csv')
```

**Explanation of the code** There are three lines in the code block above:

- The first two lines start with a `#` (pronounced hash). This means R **won't** run the line of code. It is included only as a comment - for your reference in future.
- However, if we did run these lines manually (by copying the code to the console without the `#`), this would download the data from the URL, and then save your own copy in a file in your RStudio project directory.
- The third line reads data from `cps2.csv` to a variable called `datafromtheweb`.

In practice, if I was working on data available online, I would run the first two lines on one occasion (to save my own copy of the data). The third line is the only thing that is run when we execute the R script, so provided we have saved the data the script will run even when I'm not online.

### Task: Read data from the web

1. Load the CPS data from Andy Wills' website, and save it to a file in your Rstudio project directory.

Optional extension tasks:

1. Find another csv dataset on the web (hint: try searching for "Plosone .csv dataset") and read it into R.
2. Calculate the mean of a few of the variables in this file.

### Reading SPSS files

If you have old data stored in SPSS files, they are likely to have a `.sav` file extension. These can be read with the `haven` package.

For example, this paper in PlosOne provides a number of SAV files for the experiments reported: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0209900#sec039

To use these files you should:

- Download the file you want
- Save it with a `.sav` extension
- Upload it to RStudio server (use Firefox — not Edge or Explorer).

You can then load it like this:

```
library(haven)

datafromspss <- read_spss('journal.pone.0209900.s001.sav')
datafromspss %>% glimpse
> Observations: 89
> Variables: 29
> $ Subject       <dbl> 49, 37, 1, 84, 4, 85, 16, 30, 55, 22, 65, 59, 40...
> $ `filter_$`    <dbl+lbl> 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...
> $ Age           <dbl> 22, 22, 21, 20, 22, 20, 23, 22, 21, 21, 20, 19, ...
> $ Sex           <dbl> 1, 2, 2, 1, 2, 1, 1, 2, 1, 2, 2, 1, 2, 2, 2, 1, ...
> $ Education     <chr> "4", "2", "2", "2", "2", "2", "3", "2", "2", "2"...
> $ ADMC1         <dbl> 6, 6, 2, 1, 6, 5, 6, 5, 2, 6, 6, 2, 5, 4, 6, 2, ...
> $ ADMC1hap      <dbl> 1, 6, 6, 3, 2, 2, 1, 1, 1, 2, 2, 2, 1, 3, 4, 3, ...
> $ ADMC2         <dbl> 1, 1, 6, 6, 2, 2, 6, 5, 3, 5, 1, 5, 2, 2, 2, 6, ...
> $ ADMC2hap      <dbl> 1, 4, 3, 1, 2, 2, 3, 2, 2, 1, 2, 2, 1, 4, 2, 1, ...
> $ ADMC3         <dbl> 5, 1, 6, 6, 6, 6, 6, 3, 1, 2, 4, 6, 5, 5, 5, 3, ...
> $ ADMC3hap      <dbl> 2, 8, 8, 1, 1, 6, 3, 3, 1, 5, 1, 4, 3, 5, 3, 5, ...
> $ ADMC4         <dbl> 2, 4, 6, 3, 1, 1, 3, 1, 6, 5, 2, 6, 4, 3, 2, 5, ...
> $ ADMC4hap      <dbl> 3, 7, 9, 4, 2, 7, 5, 3, 7, 1, 5, 3, 1, 2, 5, 3, ...
> $ ADMC5         <dbl> 2, 6, 6, 3, 5, 1, 2, 1, 2, 1, 6, 6, 5, 3, 2, 1, ...
> $ ADMC5hap      <dbl> 1, 9, 8, 1, 3, 1, 2, 2, 2, 2, 7, 4, 3, 2, 3, 5, ...
> $ ADMC6         <dbl> 5, 6, 6, 6, 5, 5, 6, 4, 5, 2, 6, 5, 6, 6, 4, 5, ...
> $ ADMC6hap      <dbl> 2, 9, 9, 4, 5, 1, 5, 5, 5, 3, 6, 3, 6, 6, 4, 5, ...
> $ ADMC7         <dbl> 5, 5, 6, 6, 5, 1, 6, 5, 4, 6, 5, 6, 4, 3, 4, 5, ...
> $ ADMC7hap      <dbl> 4, 9, 9, 2, 4, 3, 5, 6, 7, 7, 3, 4, 6, 2, 3, 5, ...
> $ ADMC8         <dbl> 3, 1, 6, 1, 2, 2, 3, 2, 5, 1, 6, 6, 3, 2, 2, 4, ...
> $ ADMC8hap      <dbl> 3, 4, 9, 3, 3, 1, 3, 2, 3, 3, 3, 4, 2, 5, 2, 5, ...
> $ ADMC9         <dbl> 3, 5, 5, 6, 1, 6, 1, 6, 3, 1, 4, 6, 5, 5, 2, 2, ...
> $ ADMC9hap      <dbl> 1, 5, 4, 1, 4, 2, 3, 7, 1, 3, 2, 4, 6, 2, 4, 2, ...
> $ ADMC10        <dbl> 5, 6, 1, 6, 4, 6, 5, 2, 3, 6, 6, 6, 6, 5, 5, 5, ...
> $ ADMC10hap     <dbl> 1, 9, 7, 5, 2, 7, 3, 3, 5, 7, 3, 4, 6, 5, 7, 3, ...
> $ ADMC_mea      <dbl> 3.7, 4.1, 5.0, 4.4, 3.7, 3.5, 4.4, 3.4, 3.4, 3.5...
```

```
> $ ZADMC_mea      <dbl> -0.69119217, -0.06507407, 1.34369165, 0.40451450...
> $ MeaADMChappy  <dbl> 1.9, 7.0, 7.2, 2.5, 2.8, 3.2, 3.3, 3.4, 3.4, 3.4...
> $ ZMeaADMChappy <dbl> -2.7878439, 2.5067266, 2.7143568, -2.1649532, -1...
```

If your SPSS datafile has labels or other special features enabled then you can check this guide for details of how to use them. You probably won't need to though.

**Task: Reading SPSS data**

1. Load one of the other datasets from this paper: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0209900#sec039

Optional extension task:

1. Plot a density graph of one of the variables in one of the datasets. Try adding colour to distinguish another categorical variable (e.g. gender).
2. Add some other comments to your code (using `#`) to describe what the code does.
3. Read this forum discussion on how to add good comments to your code: https://www.reddit.com/r/rstats/comments/86cmj1/any_tips_on_best_practices_for_commenting_your/

**Reading from Excel**

There are two types of Excel documents, `.xls` and the newer `.xlsx` formats. You can read both with the `readxl` package.

One important thing to note about Excel spreadsheets is that **each file can contain multiple 'sheets'**. In R terms, each Excel file can contain multiple dataframes. So we need to specify which sheet we want to import.

I've included an example of an Excel file here: Excel example

To use the example:

- Download the file to your computer (use Firefox, and definitely not Edge)
- Upload to RStudio, make sure the file extension is `xslx` or `xls`

Then:

```
library(readxl)

# make sure the filename here matches yours
xldata <- read_excel('simple-excel-example.xlsx')
xldata
> # A tibble: 12 x 3
>    Participant Trial    RT
>          <dbl> <dbl> <dbl>
>  1           1     1    44
>  2           1     2    45
>  3           1     3    33
>  4           2     1    22
>  5           2     2    22
>  6           2     3    23
>  7           3     1    55
>  8           3     2    56
>  9           3     3    57
> 10           4     1    99
```

```
> 11           4     2    96
> 12           4     3  2001
```

**Explanation**: `read_excel()` has read in the FIRST sheet (called 'Experiment 1') from the Excel file. To check this, open the file in Excel and see that there are actually two experiments included in the single file.

---

If we wanted to load both experiments, we would need to write:

```
expt1 <- read_excel('simple-excel-example.xlsx', sheet="Experiment 1")
expt2 <- read_excel('simple-excel-example.xlsx', sheet="Experiment 2")
```

**Explanation**: On each line we use `read_excel()` to load one of the sheets from the original file. The names of each sheet are shown in the Excel interface like this:

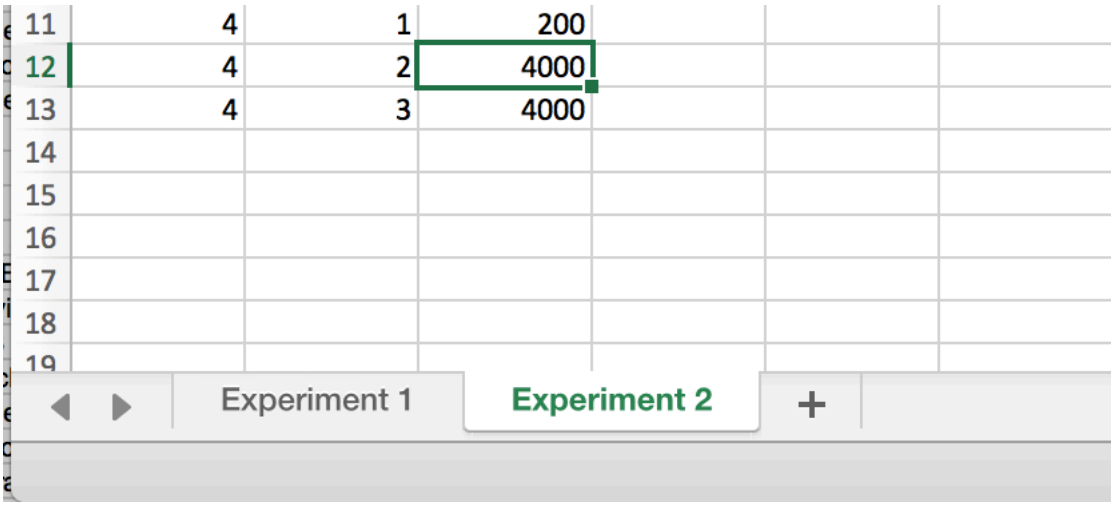



Figure 10: Sheet labels in Excel

The names of the sheets in the code which says `sheet="<SHEETNAME>"` need to match the names of the sheets in Excel exactly.

We can check the second file like this:

```
expt2  %>% glimpse
> Observations: 12
> Variables: 3
> $ Participant <dbl> 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4
> $ Trial       <dbl> 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3
> $ RT          <dbl> 22, 32, 55, 22, 21, 11, 101, 111, 122, 200, 4000, ...
```

**Task: Read Excel data**

1. Read both experiment 1 and 2 from the file simple-excel-example.xlsx
2. Use the Environment pane to check the data against the data in the Excel file.

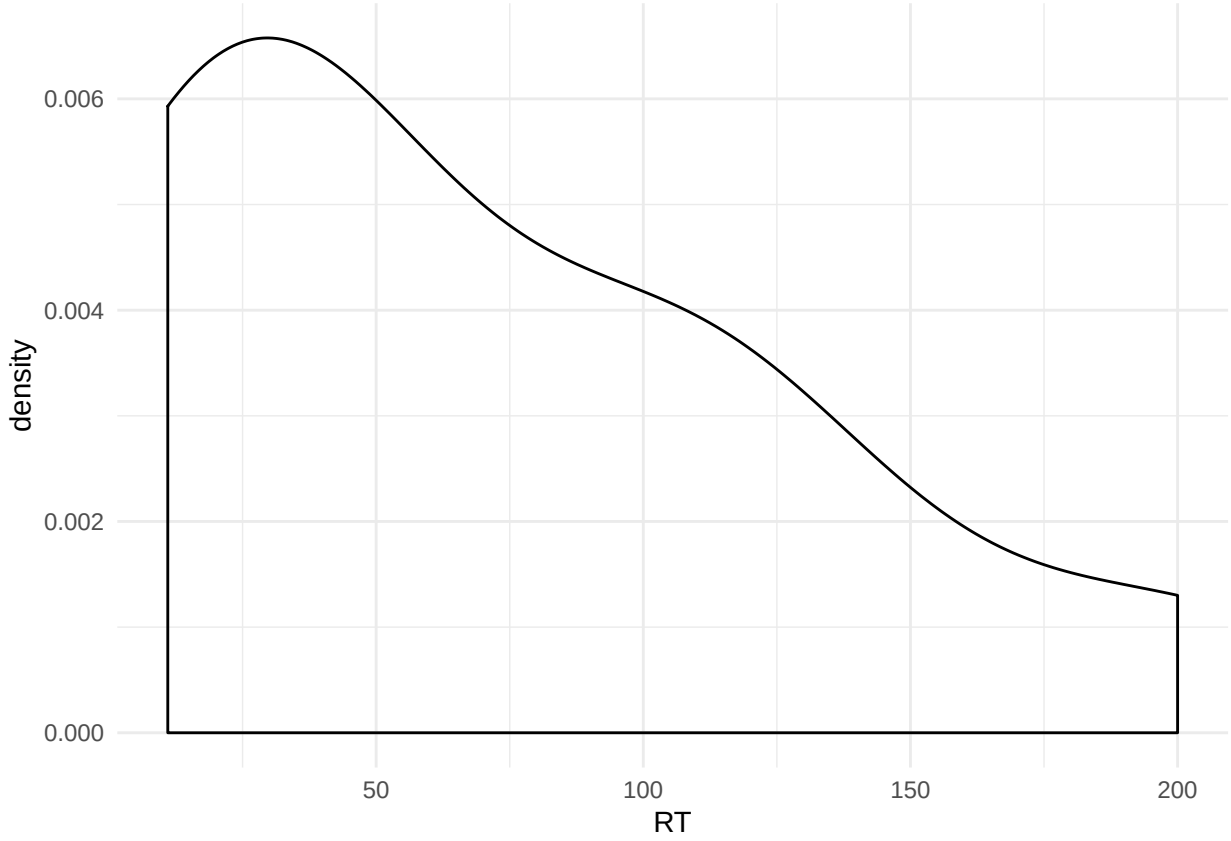
Optional extension tasks:

1. Plot the density of the RT variable in experiment 1.
2. Describe the plot
3. What pre-processing step might we want to use, before plotting (hint: in these experiments, most responses should happen within a second).

Show answer

Depending on the paradigm being used, we might want to filter very long RTs (e.g. over 1000ms):

```
expt2  %>%
  filter(RT < 1000)  %>%
  ggplot(aes(RT)) + geom_density()
```




# Visualisation and plotting

### In brief

> Visualising data is a core skill for all quantitative researchers, and is one of the most transferable skills taught on the course. Although sometimes underplayed, visualisation plays a ***much*** more important role in good science than, for example, statistical testing. Effective visualisations help scientists understand their data, spot errors, and build appropriate models.
>
> When plotting, we are often trading-off **information density** with **clarity**. A good analogy here is a map: adding detail may help us navigate; but if we add irrelevant features the map becomes less useful.

## Session 1

### 200 countries, 200 years...

If you're just getting started with data visualisation, Hans Rosling's "200 countries and 200 years in 4 minutes" is something to aspire to.

[2]Images used for educational purposes under assumption of fair use. Nautical chart screenshot from gpsnauticalcharts.com, land map from maps.google.com.

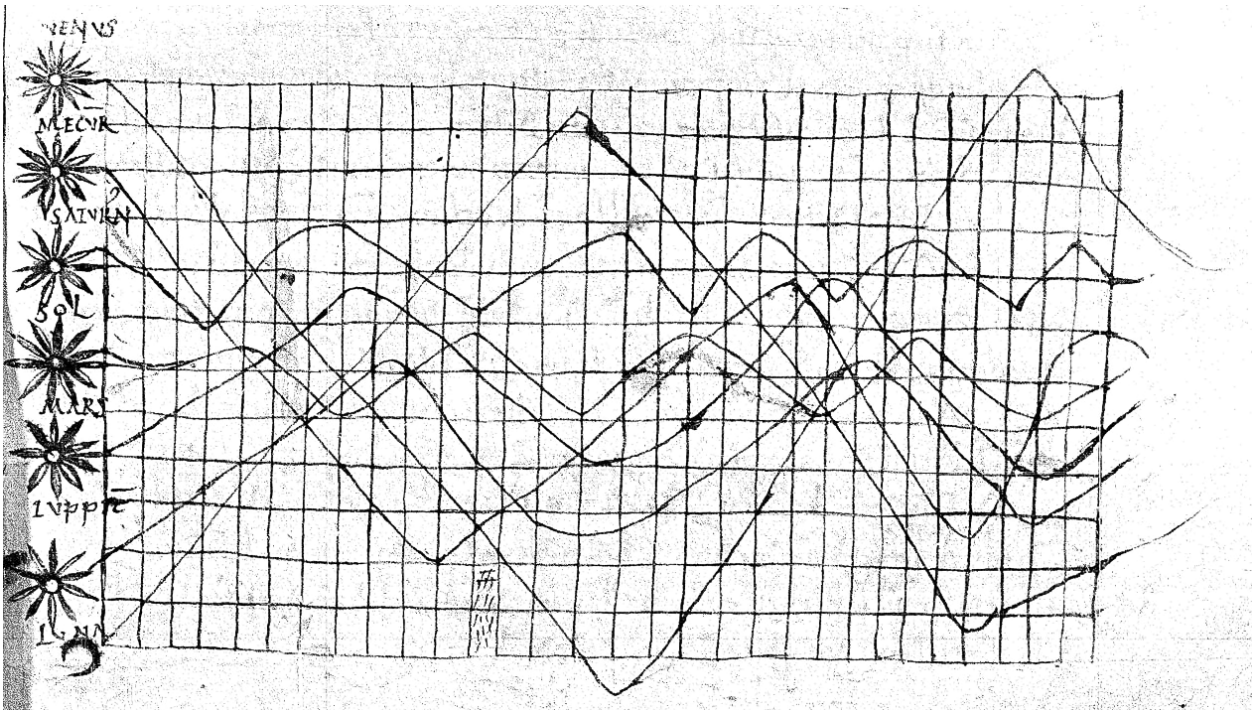



Figure 11: Nothing new under the sun: Mediaeval line plot, circa 1010. Image: wikimedia

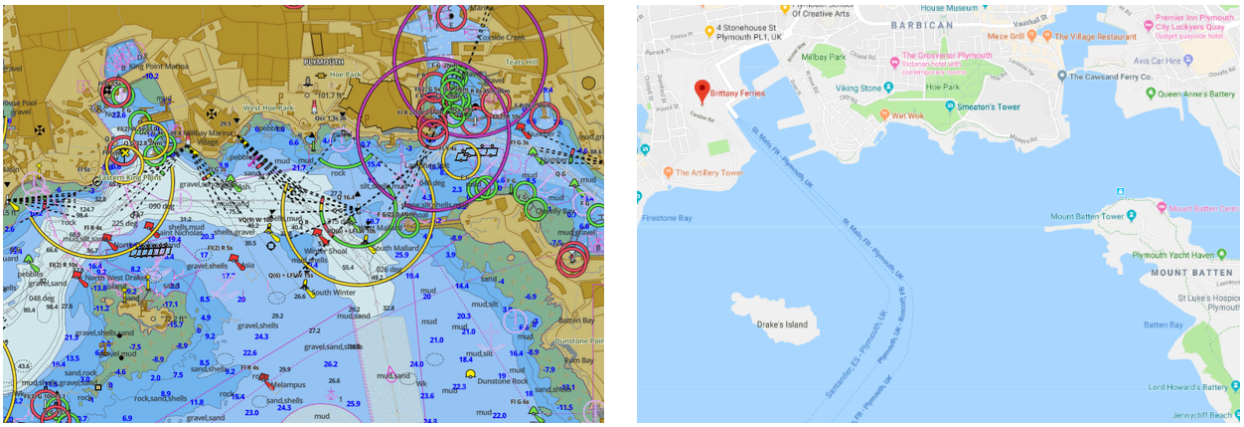



Figure 12: A land map and nautical chart of Plymouth Sound: Cartographers face similar challenges in selecting and displaying geographical information for different purposes.[2] Full size

Alongside his enthusiastic presentation, the visualisations in this clip support a clear narrative, and help us understand the data better.

His plot is interesting because it uses many different features to express features in the data:

- X and Y axes
- Size of the points
- Colour
- Time (in the animation)

These features are carefully selected to highlight important features of the data and support the story. Although we need to have integrity in our plotting (we'll see bad examples later), this narrative aspect of a plot is important: we need to consider our audience.

## Into the third (and fourth, and fifth…) dimension

We can determine how complex a plot is by how many **dimensions** it has.

For example, this plot has one dimension representing how revolting particular fruits and vegatables are:

```
Mango     Pear                  Aubergine                          Snozzcumber
|         |                     |                                  |
-------------------------------------------------------------------
```

Scatter plots are *more* complex because they have two dimensions — that is, they show two variables at once. The variables are represented by the position of each point on the X and Y axes:

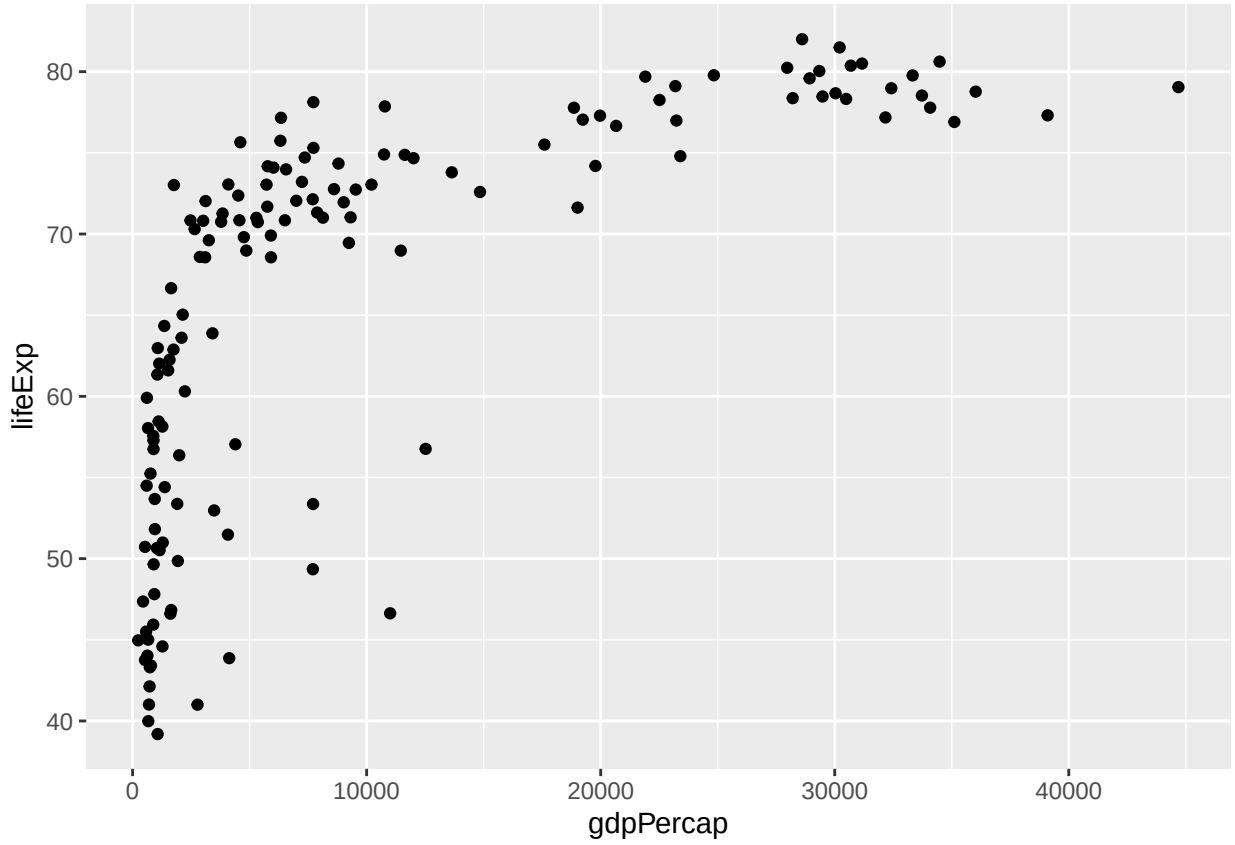
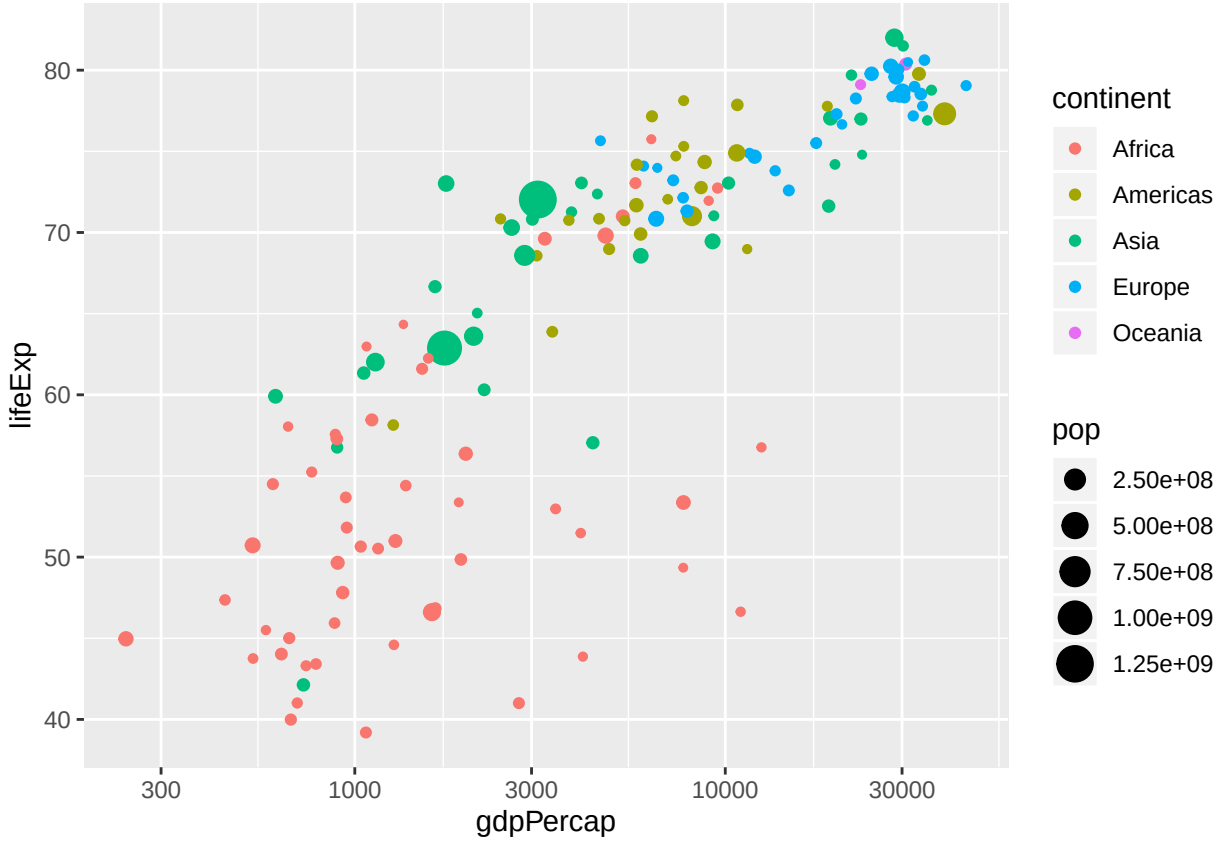



Figure 13: Life expectancy and GDP per capita in countries around the world in 2002

And Rosling's plot is more complex still because it adds dimensions of colour and size, and uses a special **logarithmic** scale for the x-axis (more on this later).




---

### Dimensions/aesthetics in `ggplot`

As you have already seen, `ggplot` uses the term **aesthetics** to refer to different dimensions of a plot. '*Aesthetics*' refers to 'what things look like', and the `aes()` command in `ggplot` creates links variables (columns in the dataset) to visual features of the plot. This is called a **mapping**.

There are x visual features (aesthetics) of plots we will use in this session:

- `x` and `y` axes
- `colour`
- `size` (of a point, or thickness of a line)
- `shape` (of points)
- `linetype` (i.e. dotted/patterned or solid)

**Task: Recreate the Rosling plot** To create a (slightly simplified) version of the plot above, the code would look something like this:

```
gapminder::gapminder %>%
  filter(BLANK==BLANK)  %>%
  ggplot(aes(x=BLANK, y=BLANK, size=BLANK, color=BLANK)) + geom_point()
```

I have removed some parts of the code. Your job is to edit the parts which say `<BLANK>` and replace them with the names of variables from the `gapminder::gapminder` dataset (you don't need to load this — it's part of the `gapminder` package).

Some hints:

- The dataset is called `gapminder::gapminder` and you need to write that in full
- Check the title of the figure above to work out which rows of the data you need to plot (and so define the filter)
- All the `BLANK`s represent variable names

**Summary of the section**

- Plots can have multiple dimensions; that is, they can display several variables at once
- Colour, shape, size and line-type are common ways of displaying Dimensions
- In ggplot, these visual features are called **aesthetics**
- The mapping between visual features and variables is created using the `aes()` command

## Layers

In visualising data, there's always more than one way to do things. As well as plotting different dimensions, different *types* of plot can highlight different features of the data. In ggplot, these different types of plots are called **geometries**, and multiple layers can be combined in the same plot by adding together commands which start with `geom_`.

As we have already seen, we can use `geom_point(...)` to create a scatter plot:

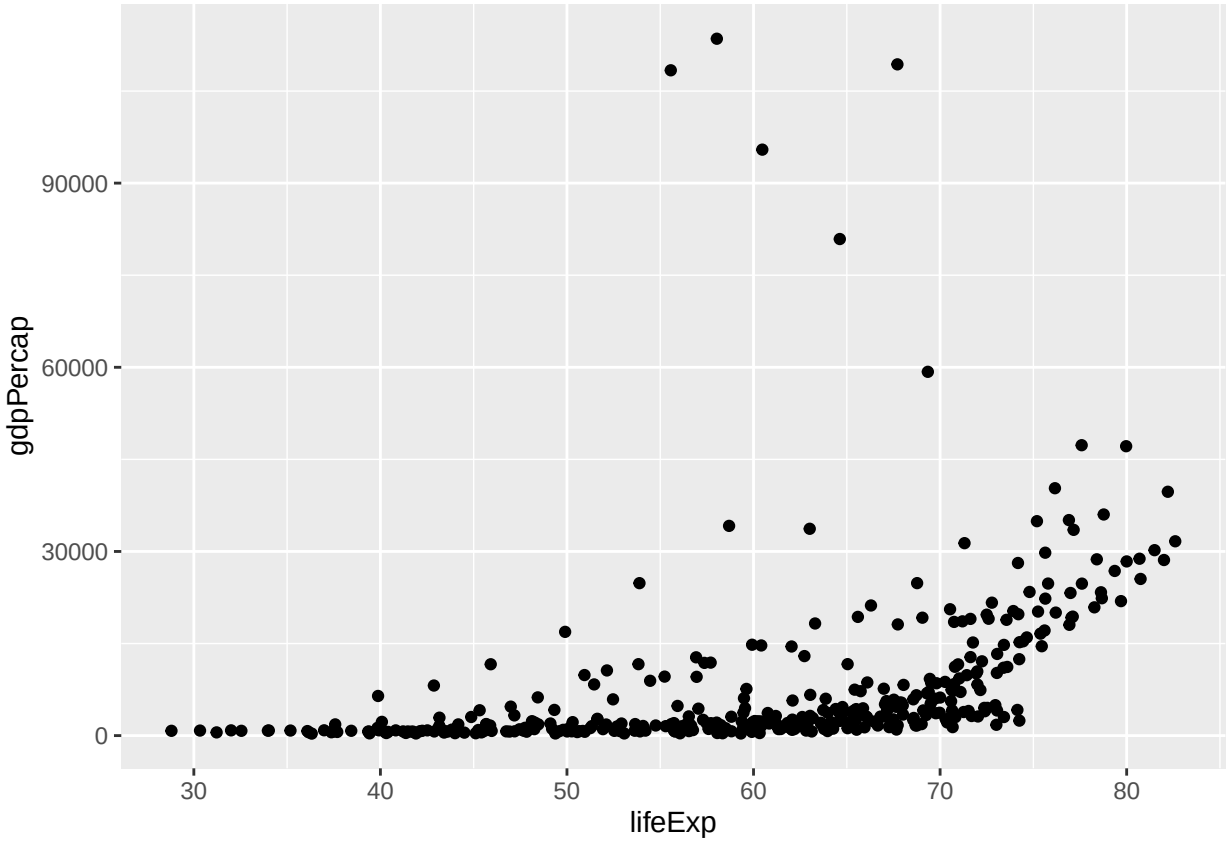
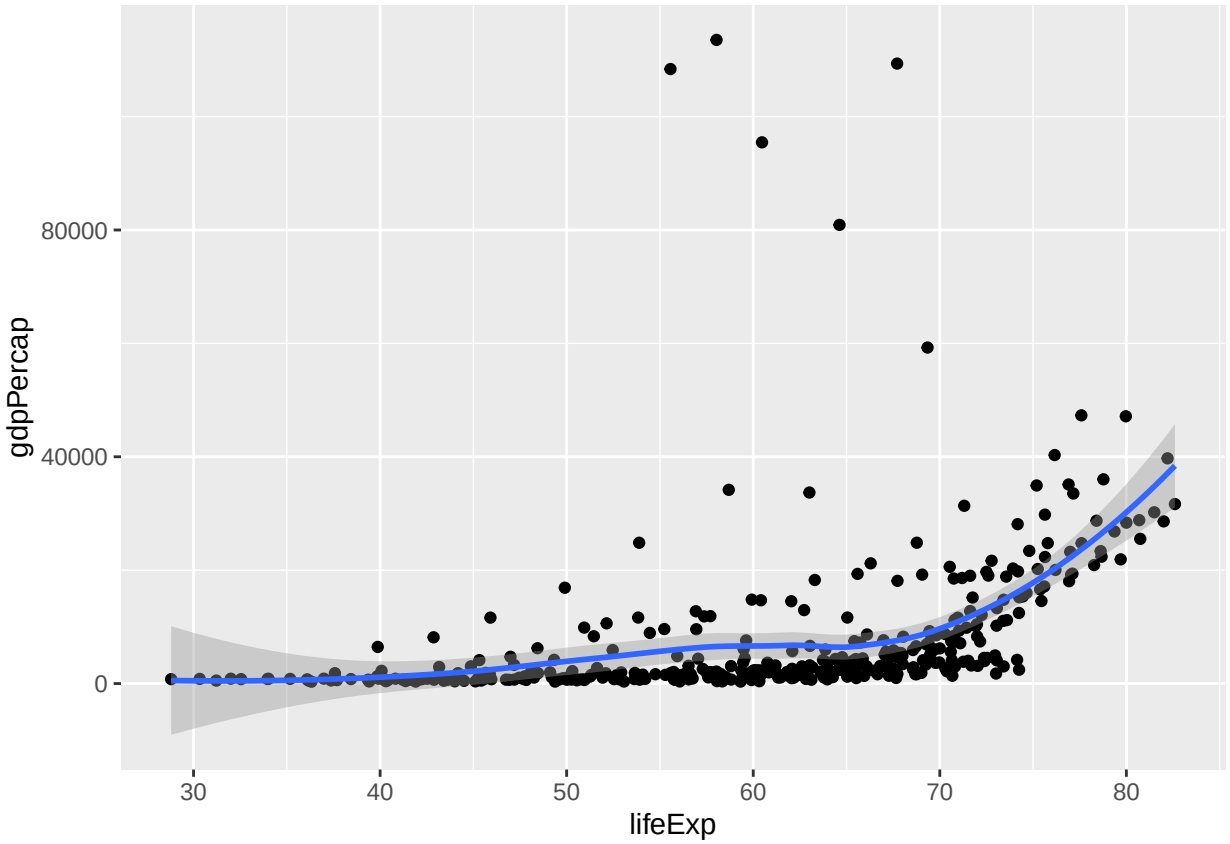



Figure 14: Life expectancy and GDP in Asia

To add additional layers to this plot, we can add extra `geom_<NAME>` functions. For example, `geom_smooth` overlays a smooth line to any x/y plot:

```
gapminder::gapminder %>%
  filter(continent=="Asia") %>%
  ggplot(aes(lifeExp, gdpPercap)) +
  geom_point() +
  geom_smooth()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




**Explanation of the command**: We added `+ geom_smooth()` to our previous plot. This means we now have two geometries added to the same plot: `geom_point` and `geom_smooth`.

**Explanation of the output**: If you run the command above you will see some warning messages which say `geom_smooth() using method = 'gam' and formula 'y ~ s(x, bs = "cs")'`. You can ignore this for the moment. The plot shown is the same as the scatterplot before, but now has a smooth blue line overlaid. This represents the local-average of GDP, for each level of `lifeExp`. There is also a grey-shaded area, which represents the standard error of the local average (again there will be more on this later).

**Task: Make a smoothed-line plot**

1. Reopen the `cps2.csv` data, or use the `mtcars` or `iris` data. Create a scatter plot of any two continuous variables.
2. Add a smoothed line to the plot using `geom_smooth`

Optional extension task:

1. Make a plot which adds colour or size aesthetics to the plot above.

**Summary of this section**

- GGplot doesn't restrict you to a single view of the data
- Plots can have multiple layers, presenting the same data different ways
- Each layer is called a 'geometry', and the functions to add layers all start with `geom_`
- Smoothed-line plots show the local average

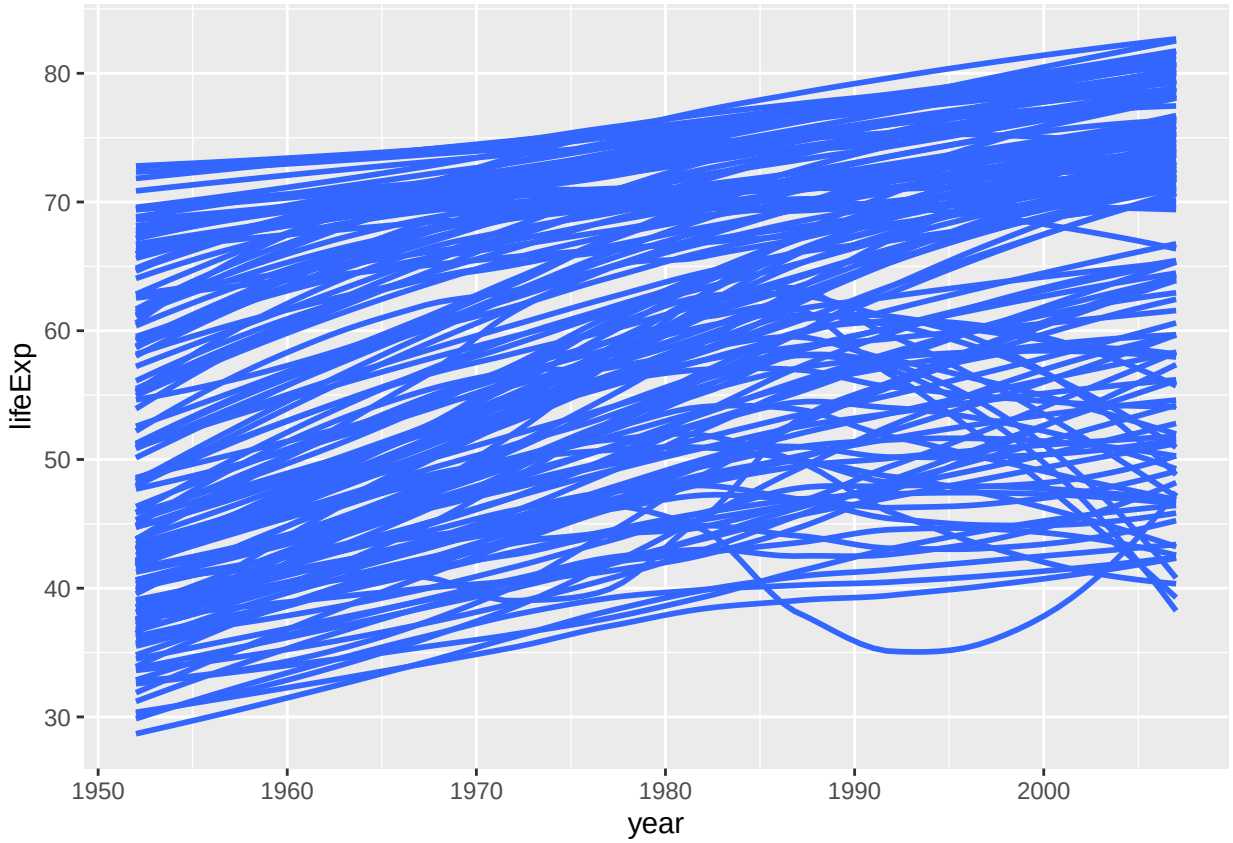
## Facets

As we add layers, plots become more complex. We run into tradeoffs between information density and clarity.

To give one example, this plot shows life expectancies for each country in the gapminder data, plotted by year:

```
gapminder::gapminder %>%
  ggplot(aes(year, lifeExp, group=country)) +
  geom_smooth(se=FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




**Explanation**: This is another x/y plot. This time though we have not added points, but rather smoothed lines (one for each country).

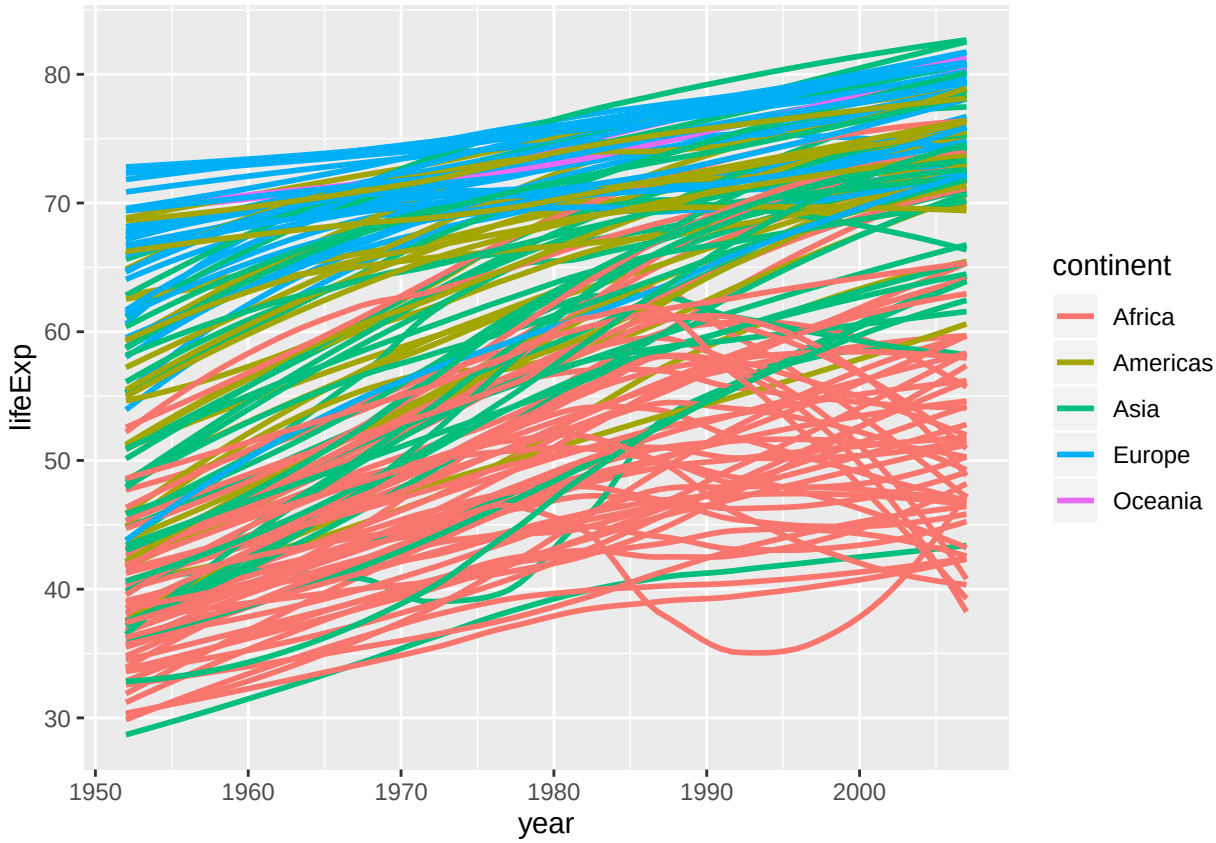
**Explanation of the code**:We have created an x/y plot as before, but this time we only added `geom_smooth` (and not `geom_point`), so we can't see the individual datapoints. We have also added the text `group=country` which means we see one line per-country in the dataset. Finally, we also added `se=FALSE` which hides the shaded area that `geom_smooth` adds by default.

**Comment on the result**: It's pretty hard to read!

To increase the information density, and explore patterns within the data, we might add another dimension and aesthetic. The next plot colours the lines by continent:

```
gapminder::gapminder %>%
  ggplot(aes(year, lifeExp, colour=continent, group=country)) +
  geom_smooth(se=FALSE)
```

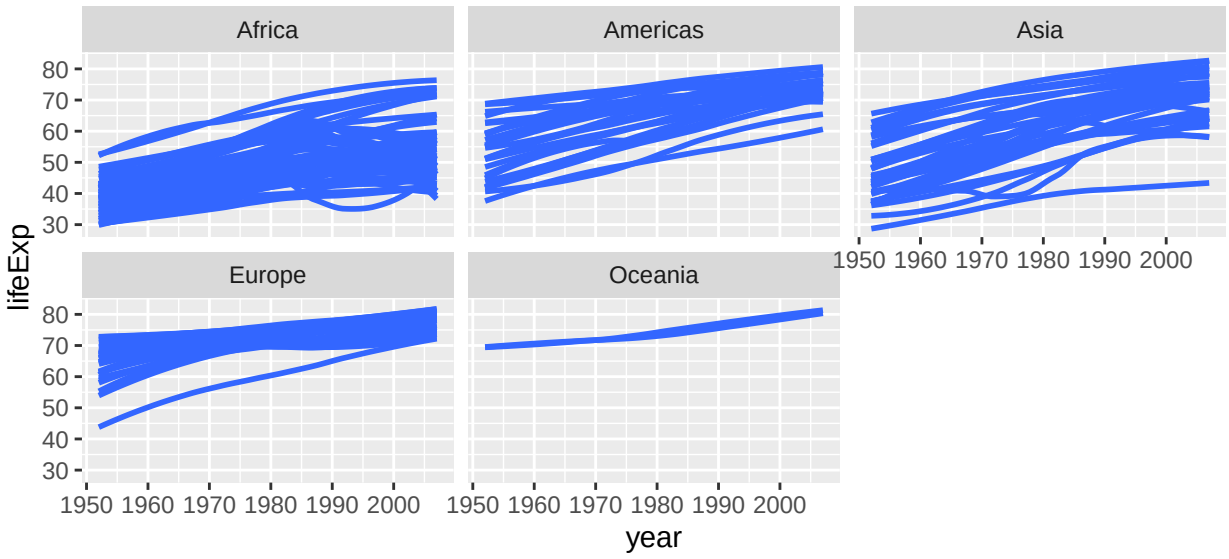
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




However, even with colours added it's still a bit of a mess. We can't see the differences between continents easily. To clean things up we can use a technique called **facetting**:

```
gapminder::gapminder %>%
  ggplot(aes(year, lifeExp, group=country)) +
  geom_smooth(se=FALSE) +
  facet_wrap(~continent)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

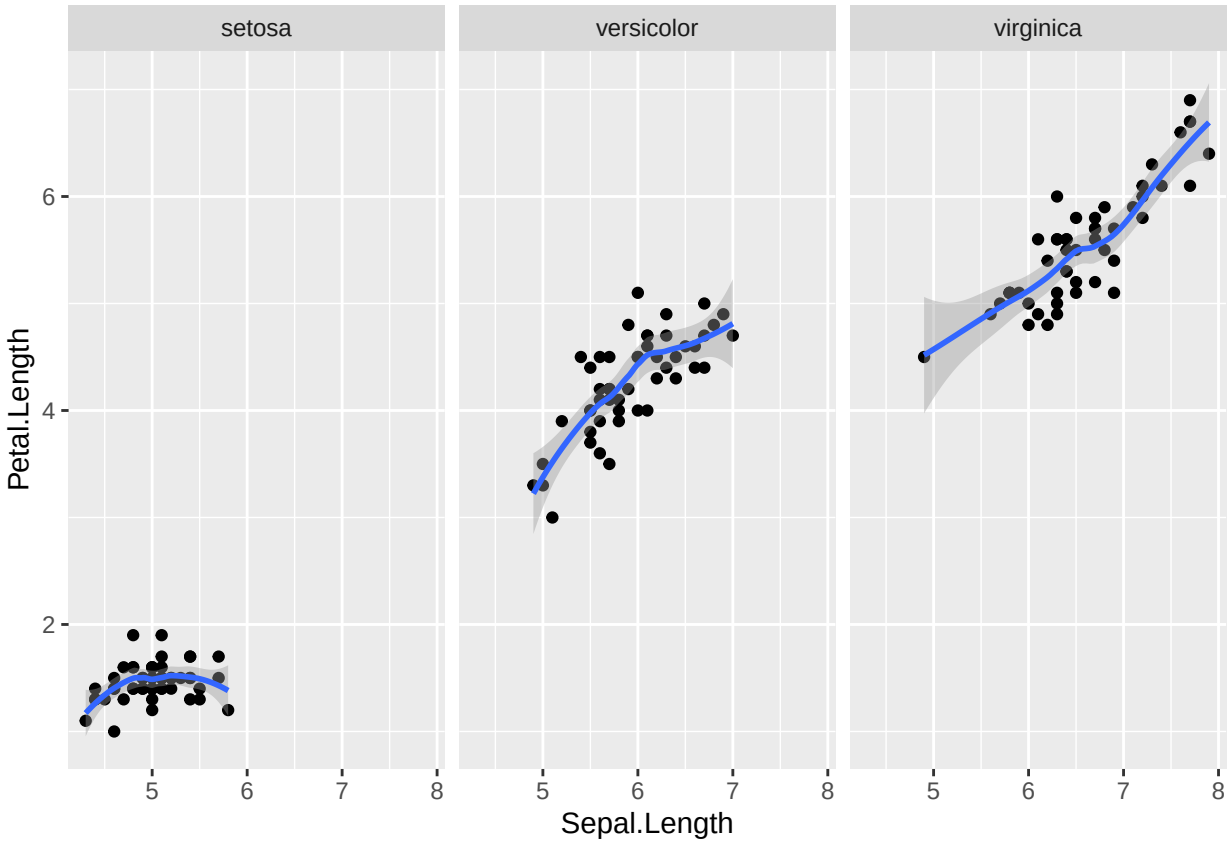



**Explanation**: We added the text `+ facet_grid(~continent)` to our earlier plot, but removed the part that said `color=continent`. This made `ggplot` create individual *panels* for each continent. Splitting the graph this way makes it somewhat easier to compare the differences *between* continents.

**Task: Use facetting** Use the `iris` dataset which is built into R.

1. Try to recreate this plot, by adapting the code from the example above:

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Optional extension tasks:

1. Create a new plot which uses colours to distinguish species and does not use Facets
2. In this example, which plot do you prefer? What influences when facets are more useful the just using colour?

Show answer

There's no right answer here, but for this example I prefer the coloured plot to the facetted one. The reason is that there are only 3 species in this dataset, and the points for each don't overlap much. This means it is easy to distinguish them, even in the combined plot. But, if there were *many* different species it might be helpful to use facets instead.

Our decisions should be driven by what we are trying to communicate with the plot. What was the research question that motivated us to draw it?

3. Try replacing `facet_grid(~continent)` with `facet_grid(continent~.)`. What happens?
4. With the `gapminder` example from above, try replacing `facet_grid` with `facet_wrap(~continent)`. What happens?
5. To see more facetting examples, see the documentation.

#### Summary of this section

- Information density is good, but can harm clarity
- Using facets can help us make comparisons between groups in the data
- It can be helpful when adding another aesthetic would clutter our plot (e.g. we have too many groups to use a different colour for each)

## Scales

As we've already seen, plots can include multiple dimensions, and these dimensions can be displayed using position (on the x/y axes) or colour, size etc.

`ggplot` does a really good job of picking good defaults when it converts the numbers in your dataset to positions, colours or other visual features of plots. However in some cases it is useful to know that you can change the default scales used.

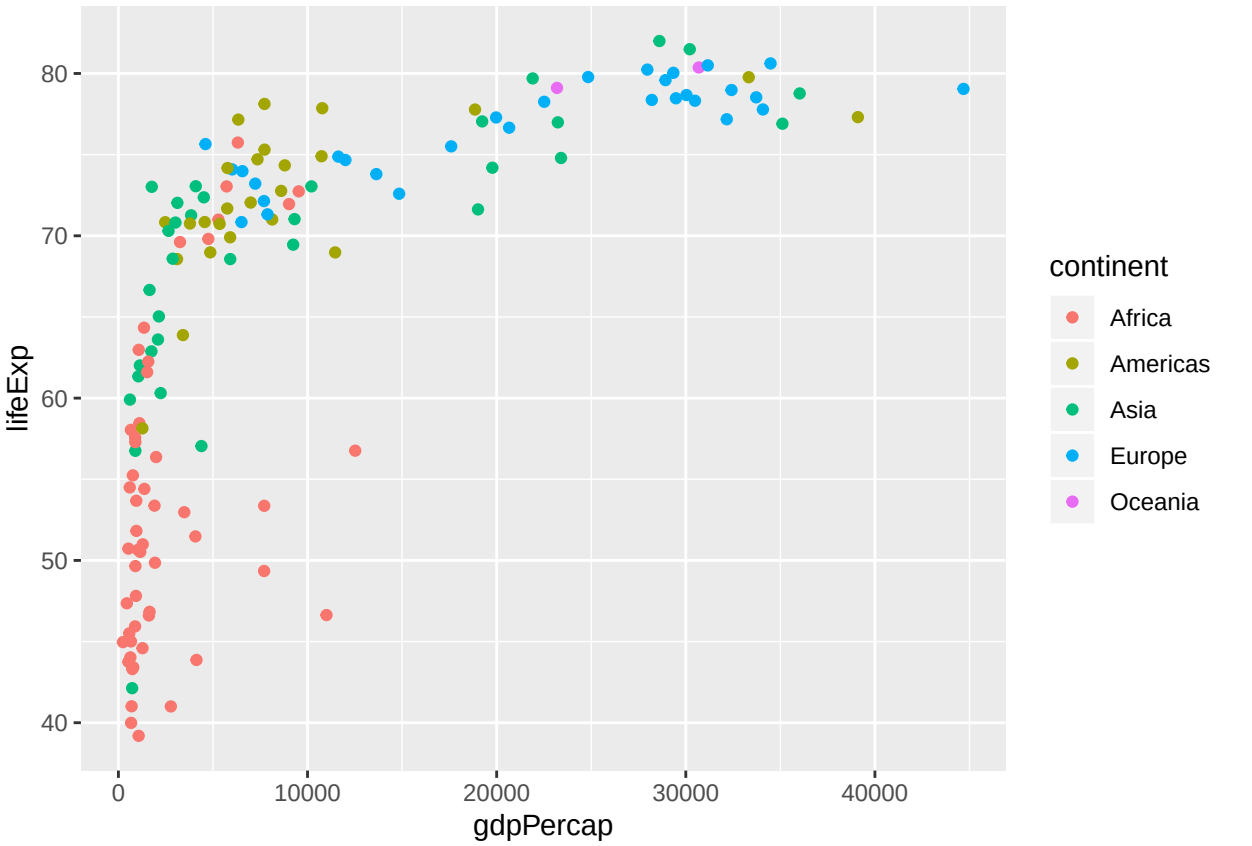
#### Continuous vs. categorical scales

So far, when we have used colour to display information we have always used **categorical** variables. This means the colour scales in our plots have looked something like this:

Set1 (qualitative)

For example, this plot shows the relationship between life expectancy and GDP in 2002, coloured by continent (using the `gapminder` data):

```
gapminder::gapminder %>%
  filter(year==2002) %>%
  ggplot(aes(gdpPercap, lifeExp, colour=continent)) +
  geom_point()
```

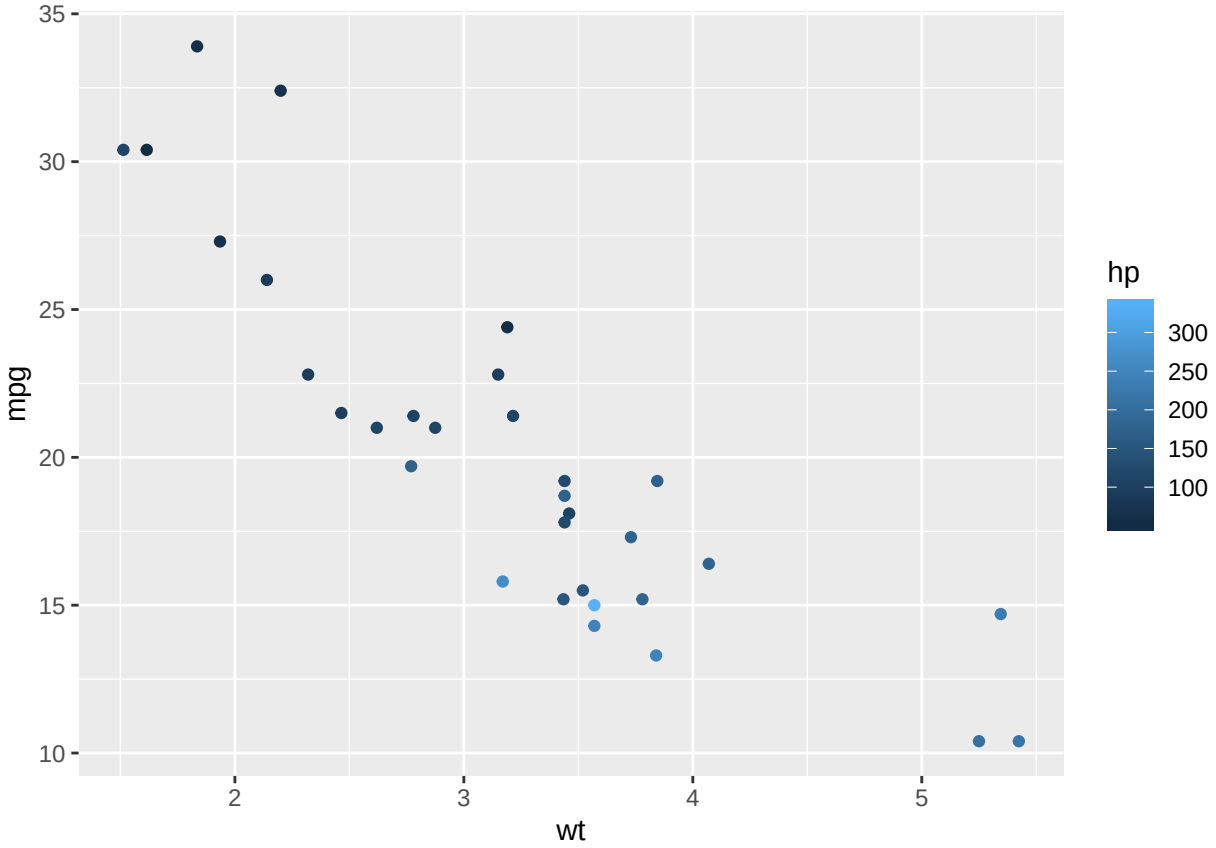



However in other cases we might want to use colour to display a continuous variable. If we want to plot continuous data in colour, we need a scale like this:

Figure 15: A continuous colour scale

In the plot below the x and y axes show the relationship between fuel economy (mpg) and weight (wt, recorded in 1000s of lbs). Colours are used to add information about how powerful (hp, short for horsepower) each car was:

```
mtcars %>%
  ggplot(aes(wt, mpg, color=hp)) +
  geom_point()
```




Did high-powered cars tend to have good or poor fuel economy?

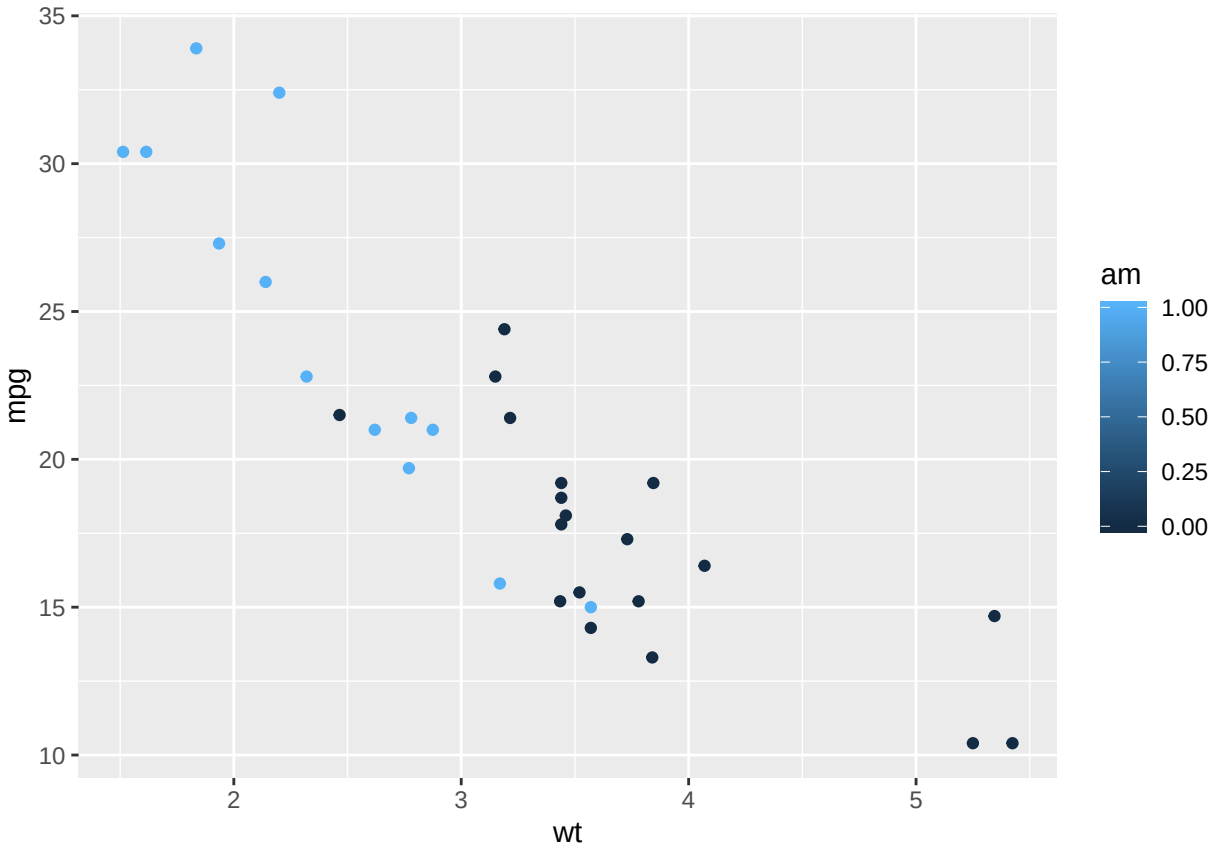
---

Sometimes variables can be stored in the 'wrong' format in R.

To give one example, the `mtcars` dataset contains a column called `am`, which indicates if a car had an automatic or manual transmission. The variable is coded as either 0 (=automatic transmission) or 1 (=manual).

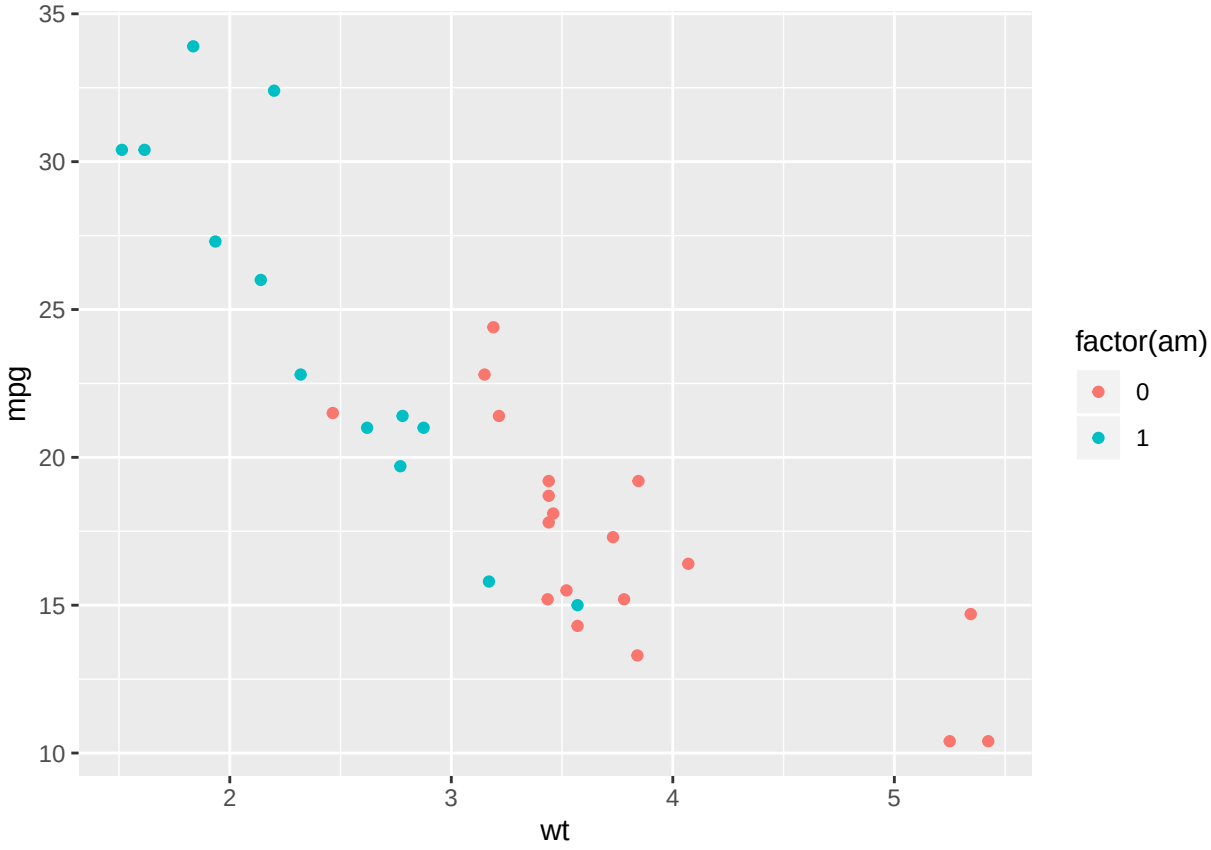
If we use the `am` variable for the colour aesthetic of a plot you will notice that `ggplot` wrongly uses a continuous colour scale, suggesting that there are values between 0 and 1:

```
mtcars %>%
  ggplot(aes(wt, mpg, color=am)) +
  geom_point()
```




To fix this, and draw `am` as a categorical variable, we can use the `factor` command:

```
mtcars %>%
  ggplot(aes(wt, mpg, color=factor(am))) +
  geom_point()
```




**Explanation**: We replaced `colour=am` in the previous plot with `color=factor(am)`. The `factor` command forces R to plot `am` as a categorical variable. This means we now see only two distinct colours in the plot for values of 0 and 1, rather than a gradation for values between 0 and 1.

The `mtcars` dataset contains another variable, `cyl`, which records how many cylinders each car had.

1. Create a scatterplot of `mpg` and `wt`, with `cyl` as the colour aesthetic, treated as a categorical variable.
2. Repeat this, but now use `cyl` as a continuous or numeric variable.

Extension task:

3. Do the same again, but using a facet rather than the colour aesthetic.

### Logarithmic scales

Another common problem with plotting data is dealing with extreme values, or skewed distributions.

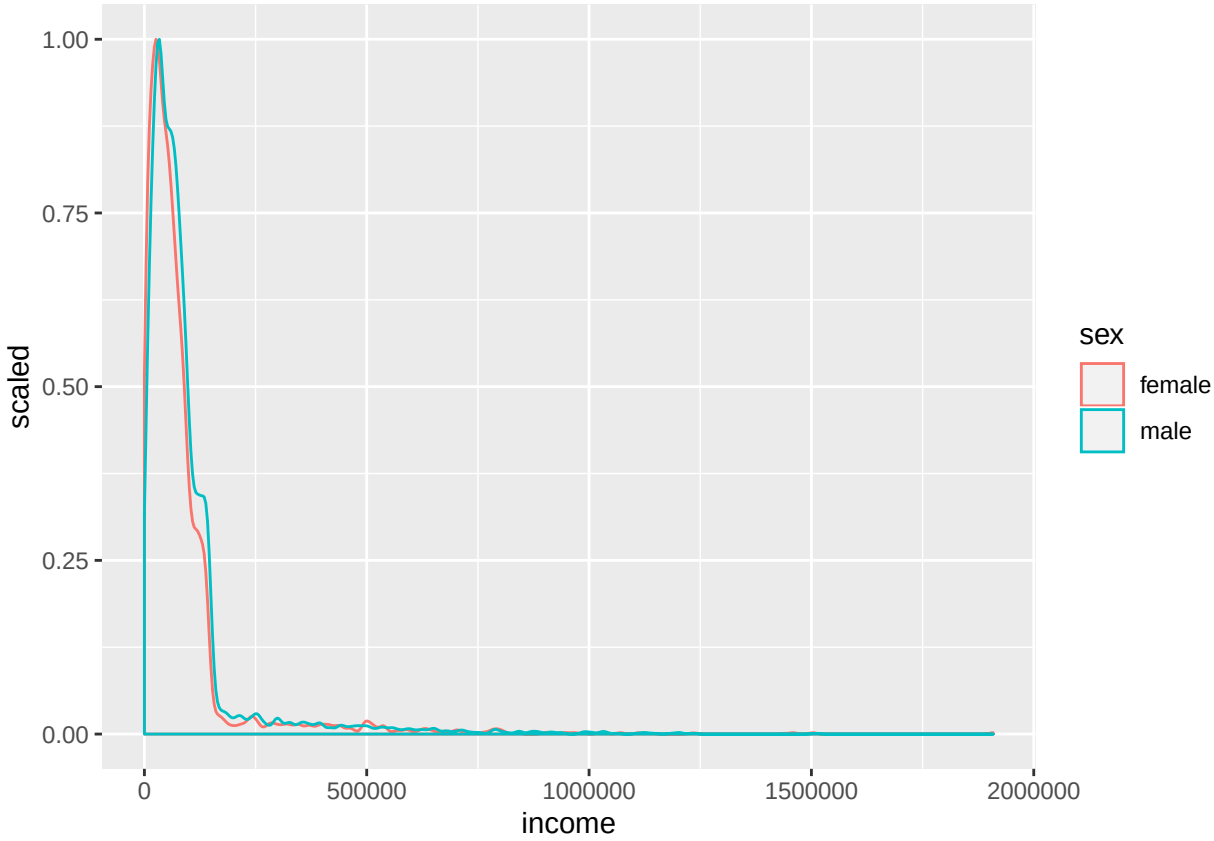
We already saw in the `cps2` dataset that income is very unevenly distributed, and a small number of people earn ***much*** more than the average: the distribution is skewed rather than normal:

```
cps2 <- read_csv('cps2.csv')
```

```
## Parsed with column specification:
## cols(
##   ID = col_double(),
##   sex = col_character(),
##   native = col_character(),
##   blind = col_character(),
##   hours = col_double(),
##   job = col_character(),
##   income = col_double(),
```

```
##   education = col_character()
## )
```

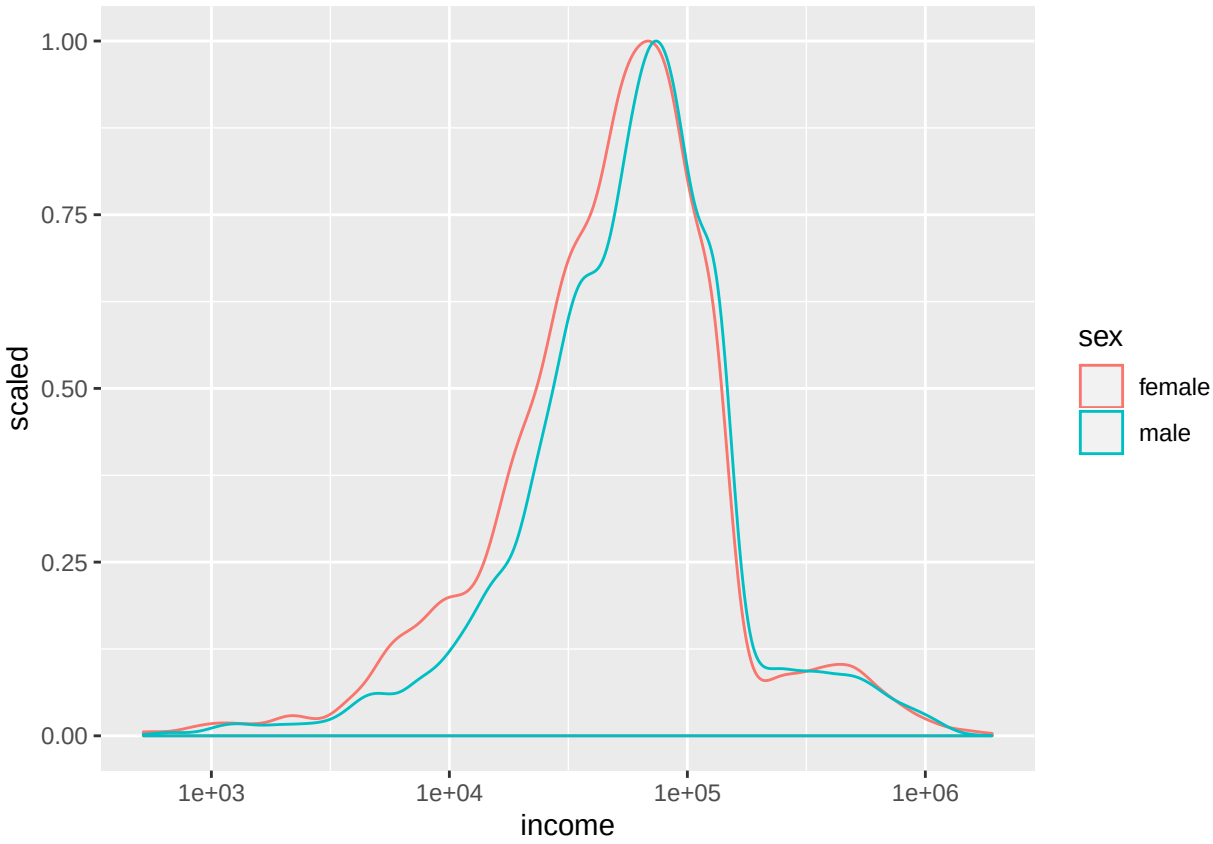
```
cps2  %>%
  ggplot(aes(income, color=sex, y=..scaled..)) + geom_density()
```




In the previous worksheet we dealt with this by using `filter` to remove cases where people earned more than $150000.

However there is another way to replot all of the data, but still see the gender pay gap: we can change the scale so that the units on the x axis are not evenly spaced: we can make it so that each marker represents an increasingly large difference:

```
cps2 %>%
  filter(income>500) %>%
  ggplot(aes(income, color=sex, y=..scaled..)) +
  geom_density() +
  scale_x_log10()
```




**Explanation of the code**: We added `+ scale_x_log10()` to our previous density plot. This command makes each unit on the x axis increase in size by a factor of 10. For this example I also filtered out individuals earning $<$ \$500 (there were very few of them, and it wasted space on the plot).

**Explanation of the output**: Two warnings are shown about 'infinite values' and 'removing non-finite values'; you can ignore these for now. The y axis of the graph has stayed the same, but the x axis has now changed. Rather than being equally-sized, the gaps in income represented by the gridlines are now uneven. Specifically, the difference between each vertical grid line is 10 times bigger than the previous one (you can find out more about what `logs and log10` means here if you are interested). R has (somewhat unhelpfully) switched to using scientific notation. This means that `1e+02` is equal to $1\times$ 10^2\$, or 100 to you an me. `1e+04` is $1\times$ 10^4\$, or 10,000, and so on. We can now see the gender pay gap much as we did before when we filtered out high earners.

**Comments on interpreting the log-scaled graph**: Although the log scale helps us see the differences between men and women, we must remember that we are interpreting a log-scaled plot. You will notice that — in contrast to the previous plot where we simply removed very high earners — the gender differences in this plot are more obvious at lower levels of income, **even though the absolute size of the difference in dollars is just as large for high as for low earners**. This is an artefact of the plotting method because the scale is unevenly spaced: For a fixed difference in income between men and women (say \$500) it will be easier to see at the bottom than at the top of the scale. Of course, the counter argument is that a \$500 difference is *more important* if you earn less than \$10,000 than if you earn $>$ \$200,000, so this extra emphasis is helpful. But there is no *correct* answer: the different plots emphasise different aspects of the data.

**Task: Use a log scale** Use the `gapminder` dataset again.

1. Filter the data so you are only using observations from a single year
2. Plot GDP per capita (x axis) against life expectancy (y axis) using a normal scale for GDP.
3. Now replot using a log scale for GDP.

4. Discuss with others working near you: what are the advantages of using a log scale in this instance?

**Optional extension task** This paper presents data on reaction times in a cross sectional sample of participants of different ages: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0189598#sec016

The data are available in Excel format from PlosOne, but I have provided a (tidied up) subset of the data here: subset of RT data.

1. Import the subset of the data, and recreate the scatterplot from Figure 1 in the paper (use `geom_smooth` for the lines, and don't worry if the lines are not exactly the same).
2. Use the `scale_y_log10` command to adjust the scale of the Y axis. The result should look something like this:

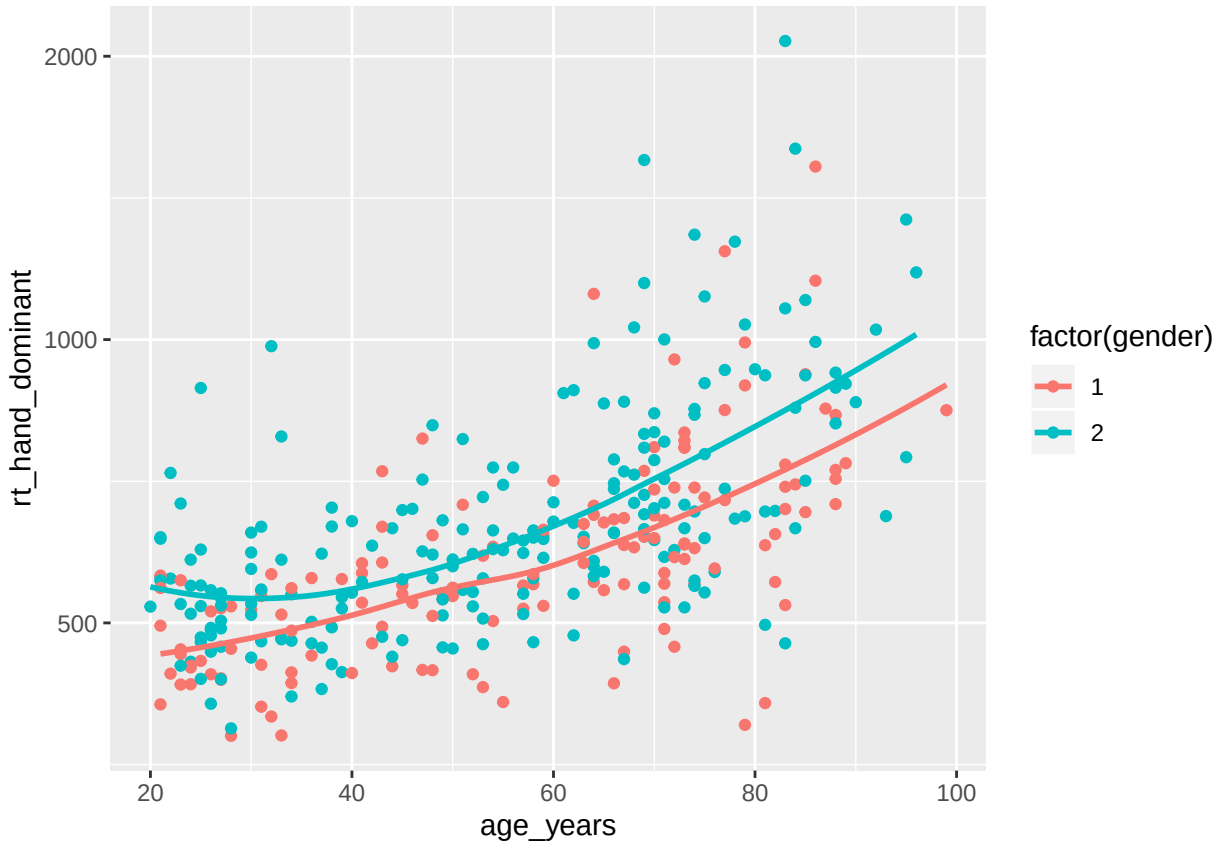
```
## Parsed with column specification:
## cols(
##   gender = col_double(),
##   rt_hand_dominant = col_double(),
##   age_years = col_double()
## )
```

```
rtsubset  %>%
  ggplot(aes(age_years, rt_hand_dominant, colour=factor(gender))) +
  geom_point() +
  geom_smooth(se=FALSE) +
  scale_y_log10()
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
## Warning: Removed 1 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```




**Summary of this section**

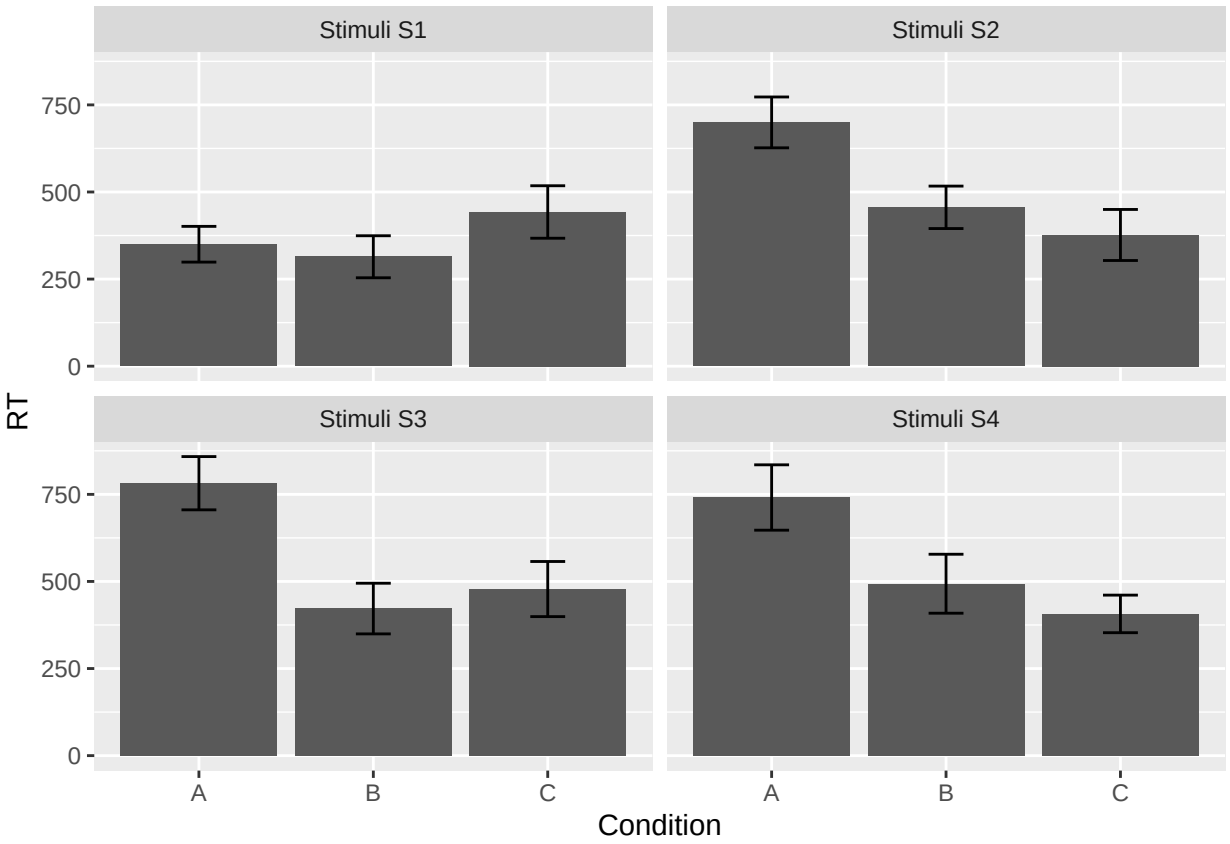
- Colour scales can be either categorical or continuous
- Sometimes data are stored in the 'wrong' format. We can use `factor(<VAR>)` to force a variable to be categorical
- Logarithmic (log) scales create uneven spacing on the axes.
- Log scales can be useful when data have a skewed distribution, but we need to be careful when interpreting them

## Comparing categories

In the examples above we have been plotting continuous variables (and adding colours etc). We've used density, scatter and smoothed line plots to do this.

Another common requirement is to use plots to compare summary statistics for different groups or categories. For example, the classic plot in a psychology study looks like this:

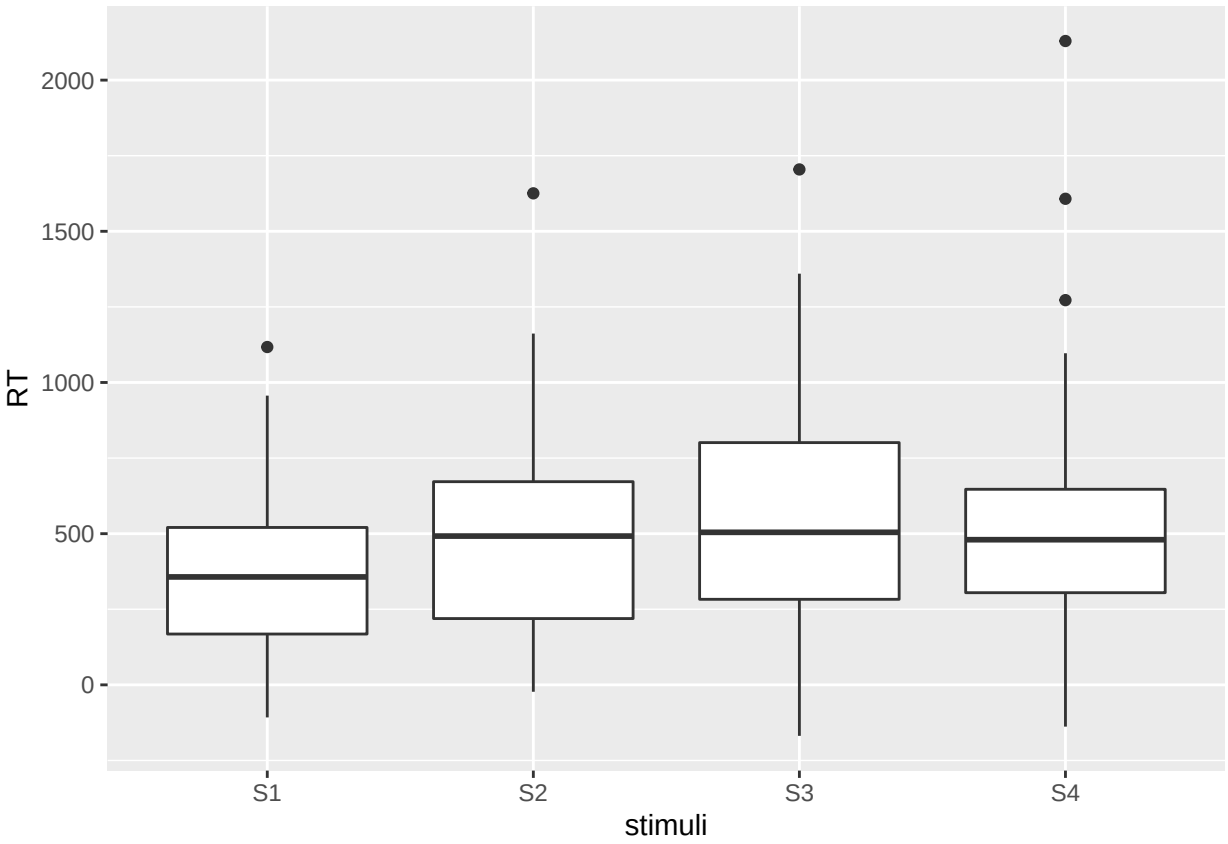
```
## Parsed with column specification:
## cols(
##   Condition = col_character(),
##   stimuli = col_character(),
##   p = col_double(),
##   RT = col_double()
## )
```




However, there is evidence that readers often misinterpret bar plots. Specifically, the problem is that we percieve values *within* the bar area as more *likely* than those just above, even though this is not in fact the case.

A better choice is (almost always) to use a boxplot:

```
expdata  %>%
  ggplot(aes(x=stimuli, y=RT)) + geom_boxplot()
```




**Explanation**: We used `Condition`, a category, as our x axis, and reaction times as the y axis. We added `geom_boxplot` to show a boxplot.

If you're not familiar with boxplots, there are more details in the help files (type `?geom_boxplot` into the console) or use the wikipedia page here

Load this (simulated) dataset here: dummy experiment data. Either download it and upload to rstudio, or read it directly from that url ([reminder of how]](#read-data-from-url)).

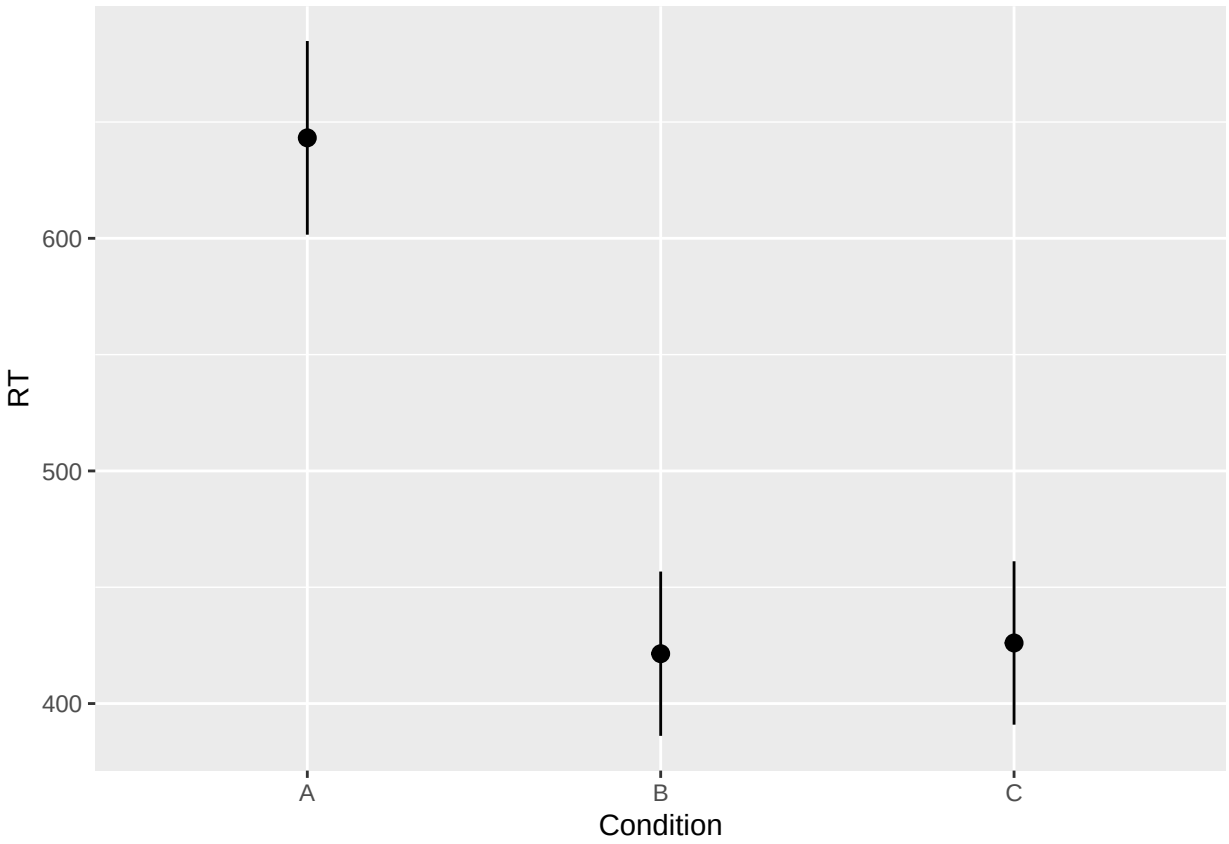
- Recreate the boxplot above
- Adjust it so that `stimuli` is the variale on the x-axis
- Use a facet to recreate the plot you saw above, combining both `Condition` and `Stimuli`

---

If you *really* need to plot the mean and standard error of different categories, ggplot has the `stat_summary` command:

```
expdata %>%
  ggplot(aes(Condition, RT)) + stat_summary()
```

```
## No summary function supplied, defaulting to `mean_se()
```




**Explanation**: We used `Condition` and `RT` as our x and y axes, as before. This time we added `stat_summary()` instead of `geom_boxplot()`. By default this plots the mean and standard error (a measure of variability) in each group, using a **point-range plot**. This is better than a bar chart because it avoids multiple a known bias in how we read them. You can ignore the warning about `No summary function supplied, defaulting to mean_se()` for now.

As an extension exercise:

- Adapt your facetted boxplot from above to show the mean and standard error instead
- Can you combine both boxplot and summary in a single plot?

## Spit and polish

Ggplot is great because it sets sensible defaults for most things (axes, colours etc). When you are exploring your data these defaults typically suffice. However for publication you will often need to polish up your plots, perhaps including:

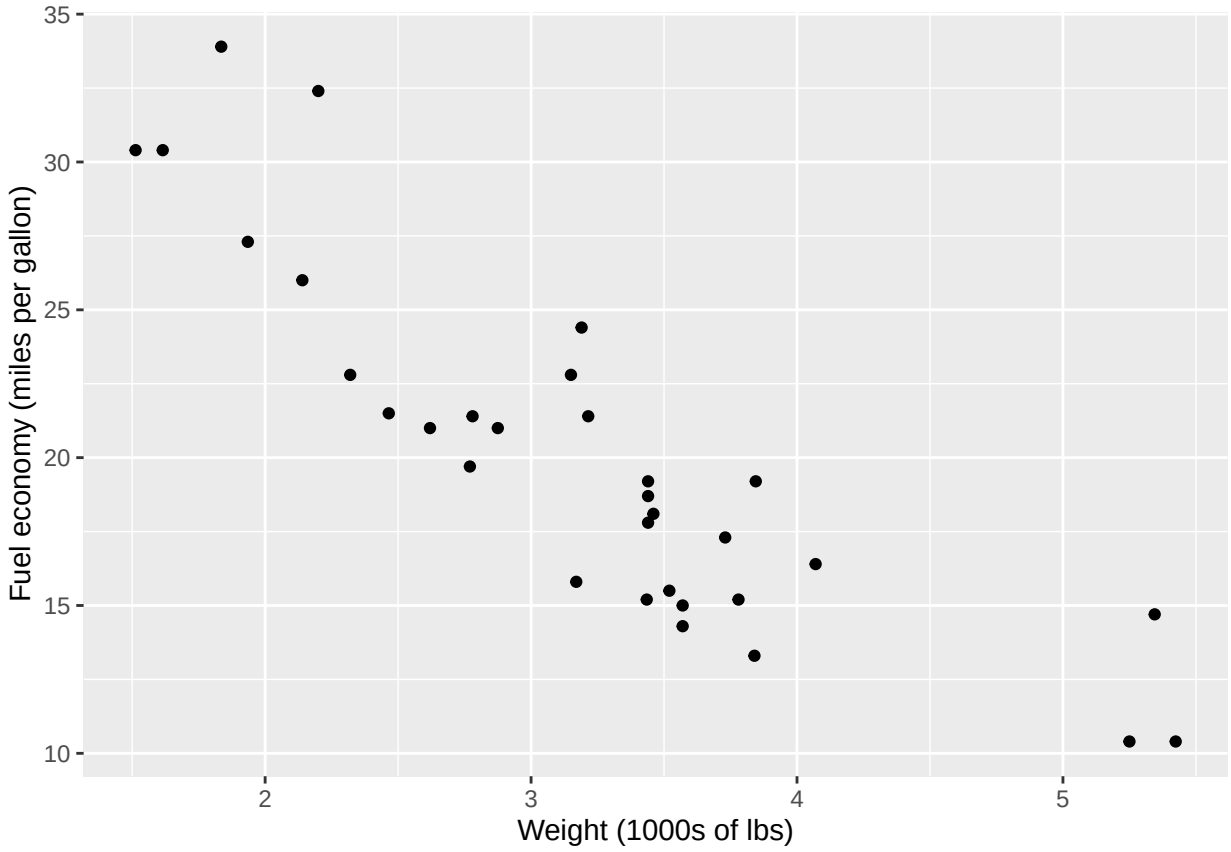
- Label your plot axes
- Add lines or text
- Change plot colours etc
- Saving to a pdf or other output format

### Labelling axes

By default, ggplot uses variable names and the values in your data to label plots. Sometimes these are abbreviations, or otherwise need changing.

To relabel axes we simply add `+ xlab("TEXT")` or `+ ylab("TEXT")` to an existing plot:

```
mtcars %>% ggplot(aes(wt, mpg)) +
  geom_point() +
  xlab("Weight (1000s of lbs)") +
  ylab("Fuel economy (miles per gallon)")
```



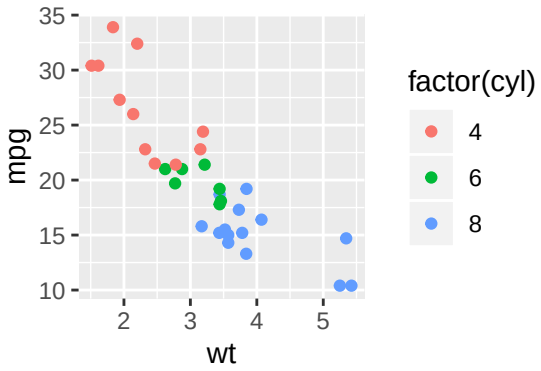

Try adding axis labels to one of your existing plots.

### Changing the label of color/shape guidelines

If you are short of time you can treat the rest of this section like an extension exercise. It might be useful for your own work, but won't form part of the assessment.

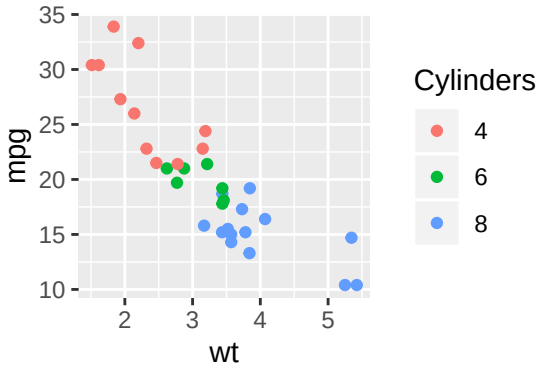
When adding the colour aesthetic, ggplot uses the variable name to label the plot legend. For example:

```
mtcars %>%
  ggplot(aes(wt, mpg, colour=factor(cyl))) +
  geom_point()
```




The generated legend label sometimes looks ugly (like above) but this is easy to fix:

```
mtcars %>%
  ggplot(aes(wt, mpg, colour=factor(cyl))) +
  geom_point() +
  labs(color="Cylinders")
```




**Explanation**: We added `labs(color="Cylinders")` to the plot to change the legend label.

Try relabelling the colour legend of one of your existing plots.

### Adding lines

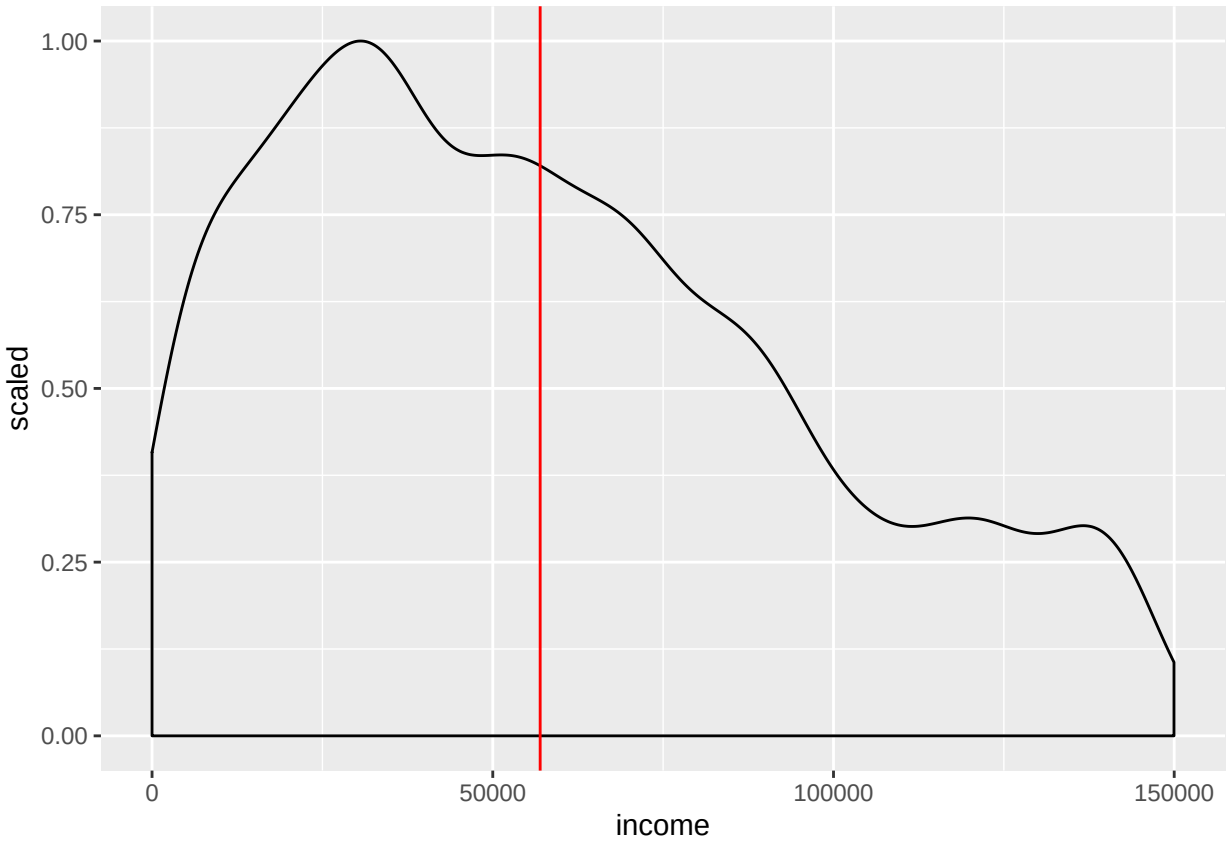
Sometimes it can be helpful to add lines to a plot: for example to show a clinically meaningful cutoff, or the mean of a sample.

For example, let's say we want to make a scatter plot of income in the `cps2` data, but adding a line showing the median income. First we calculate the median:

```
median_income = cps2 %>% summarise(median(income)) %>% pull(1)
```

**Explanation**: First, we are defining a new variable to equal the mean income in the sample. We do this by using `summarise(mean(income))`. The part which reads `pull(1)` says "take the first column". We need to do this because `summarise()` creates a new table, rather than a single value or sequence of values (which we need below).

```
cps2 %>%
  filter(income < 150000) %>%
  ggplot(aes(income, y=..scaled..)) +
    geom_density() +
    geom_vline(xintercept = median_income, color="red")
```




**Explanation**: We have regular density plot. This time we have added `geom_vline` which draws a vertical line. The `xintercept` is the place on the x axis where our line should cross.

Add a `geom_vline` to a plot you have already created. This could be either:

- A calculated value (e.g. `mean(var)`) or
- A fixed value (e.g. `xintercept = 20`)

### Saving plots to a file

So far we have created plots in the RStudio web interface. This is fine when working interactively, but sometimes you will need to send a high-quality plot to someone (perhaps a journal).

The `ggsave` function lets us do this.

The first step is to make a plot, and save it (give it a name).

```
myfunkyplot <- mtcars  %>% ggplot(aes(wt, mpg, color=factor(cyl))) + geom_point()
```

**Explanation**: We used the assignment operator `<-` to save our plot to a new name (`myfunkyplot`). This means that when we run the code RStudio won't geneate any output immediately, so we don't see the plot yet.

---

Next, we use `ggsave` to save the plot to a particular file:

```
ggsave('myfunkyplot.pdf', myfunkyplot, width=8, height=4)
```

You can see the output of the `ggsave` command by downloading the file here: myfunkyplot.pdf

#### Publication-ready plots

Some journals have specific requirements for submitting journals; common ones include submitting in particular formats (e.g. pdf or tiff), using particular fonts etc.

There are also some common types of plots which ggplot almost, but not quite, makes out of the box.

When trying to go the last mile and polish plots for publication several additional packages may be useful. If you have time, you could work through some of the examples on this page

- ggpubr: http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/78-perfect-scatter-plots-with-correlation-and-marginal-histograms/

As mentioned in the session, Edward Tufte's books have been influential in the field of data visualisation. His book 'The displayt of quantitative information' (Tufte 2001) is a great resource and guide.

- http://motioninsocial.com/tufte/ shows how to implement many of Tufte's ideas in ggplot. It would be a nice exercise to work through this, and attempt to plot some of your own data in this style.

## Session 2: Real world plots

So far we have learned about ggplot and worked through lots of examples. You might have noticed though: we focussed mostly on the technique and didn't really focus on what the data meant.

In reality, you are normally presented with a dataset need to work creatively to explore patterns of results. Your decisions will be informed by:

- Your research questions
- Prior knowledge about the domain
- Prior knowledge about the research design and the data collection process
- What your learn about the data as you work (this is an interative process)

---

In this session we are going to work through a series of examples. Each time we will start with a scenario which describes the domain and data collection, and some research questions we may have had.

You should work in groups of 3 to:

- Explore the dataset
- Develop one or two plots which illustrate key features of the data

We will then join to form larger groups to share findings, and justify decisions made.

#### Scenario 1: Secret agent

You are a MI6 agent, and have been sent a mystery dataset by one of your spies. She said it contains highly important information which will be of great interest to your superiors. Use your ggplot wizardry to recover this classified information.

The data are available to download here: data/mystery.csv

#### Scenario 2

In the 1970s the University of California, Berkley, was concerned about the fairness of their admissions procedures. They collected data from across the university for a number of years, recording the:

- Number of applicants
- The department the student applied to
- The students' gender
- Number of students accepted
- The percentage students of each gender who were accepted in each department

A summary of these data are available at this link: data/berkley.csv.

Your job is to:

- Describe the pattern of applications
- Decide if the university was fair in it's admissions procedures
- Prepare a short presentation for the university governors which includes plots

Techniques/commands you might want to use:

- `filter`
- `group_by` and `summarise`
- `stat_summary` to plot means and standard errors or deviations
- `facet_wrap(~VARNAME)` to split a plot by a categorical variable

form

# Data handling

**In brief**

> Most time in data analysis is spent 'tidying up' data: getting it into a suitable format to get started. Data scientists have a particular definition of *tidy*: Tidy datasets are "easy to manipulate, model and visualize, and have a specific structure: each variable is a column, each observation is a row" (Wickham 2014).
>
> It's often not convenient for humans to *enter* data in a tidy way, so untidy data is probably more common than tidy data in the wild. But doing good, reproducible science demands that we

document each step of our processing in a way that others can check or repeat in future. Tools like R make this easier.

## Overview

So far we have used various commands in the `tidyverse` to work with several datasets. If you want to recap these commands you can use the 'cheatsheet' here especially this section and the part on groups and summaries.

Today we will cover two additional techniques which are important when working with real datasets:

- Creating new variables/columns
- Reshaping or reformatting data from long to wide (or vice versa)
- Joining two sources of data (e.g. two spreadsheets) into a single dataframe

---

In the second part of the session we will cover RMarkdown, which is an alternative to R script files which lets us combine text, graphics and R code and is useful when sharing our work.

## Making new variables

Sometimes we need to create new columns in our dataset. For example, let's say we wanted to calculate someone's BMI from their weight and height.

There is a built in dataset called `women`, which contains heights and weights of 15 women in lbs and inches:

```
women  %>% glimpse
```

```
## Observations: 15
## Variables: 2
## $ height <dbl> 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72
## $ weight <dbl> 115, 117, 120, 123, 126, 129, 132, 135, 139, 142, 146, 150, ...
```

To calculate a BMI we first need to convert the heights to from inches to meters, and the weights from lbs to kilograms.

```
metric_women <- women  %>%
  mutate(
    height_m = height*0.0254,  # approx conversion from inches to m
    weight_kg= weight*0.45     # conversion from lbs to kg
  )
```

**Explanation**: We used mutate to covert lbs to kg and inches to m, and saved these two new columns in the `metric_women` dataset.

We can see the new columns here:

```
metric_women %>% head(3)
```

```
##   height weight height_m weight_kg
## 1     58    115   1.4732     51.75
## 2     59    117   1.4986     52.65
## 3     60    120   1.5240     54.00
```

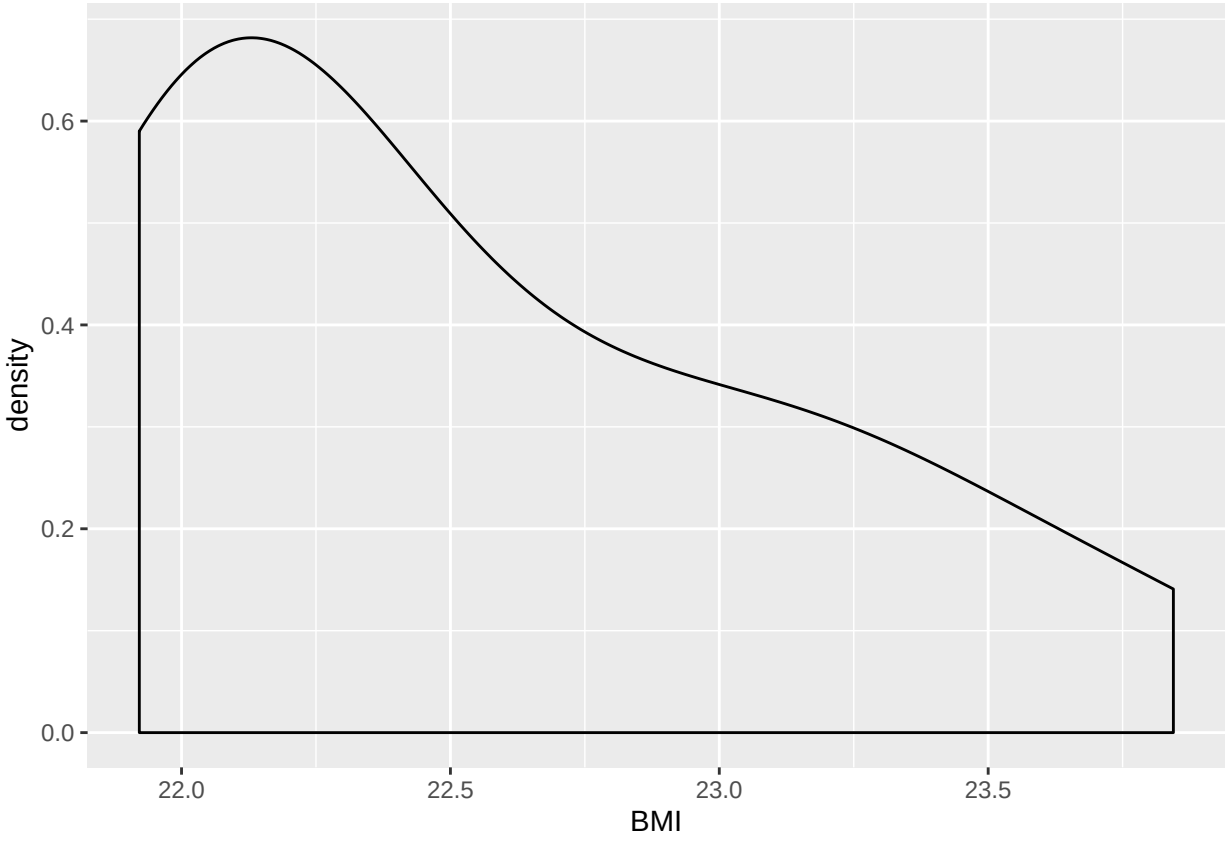
BMI is calculated as $kg/m^2$. We can use mutate again for this:

```
metric_women  %>%
  mutate(BMI = weight_kg / height_m^2) %>%
  head(3)
```

```
##   height weight height_m weight_kg      BMI
## 1     58    115   1.4732     51.75 23.84443
## 2     59    117   1.4986     52.65 23.44374
## 3     60    120   1.5240     54.00 23.25005
```

**Explanation**: We used mutate again to make a new column, `BMI`. This contains womens' weight divided by their squared height (`^2` means *to the power of 2* in R-speak).

- Create a density plot of BMI scores in the `women` dataset. It should look like this:




- What is the median BMI in the sample?

## Reshaping (melting)

Data is commonly stored in either *wide* or *long* format.

If you used SPSS to do a *t*-test or ANOVA during your undergraduate degree, you likely analysed the data in **wide** format. In wide format, each row represents the *observations from a single participant*. Each measurement for a given participant are stored in separate columns.

This is often called **row per subject** data. An example is the built in `attitude` dataset:

```
attitude %>%
  head()
```

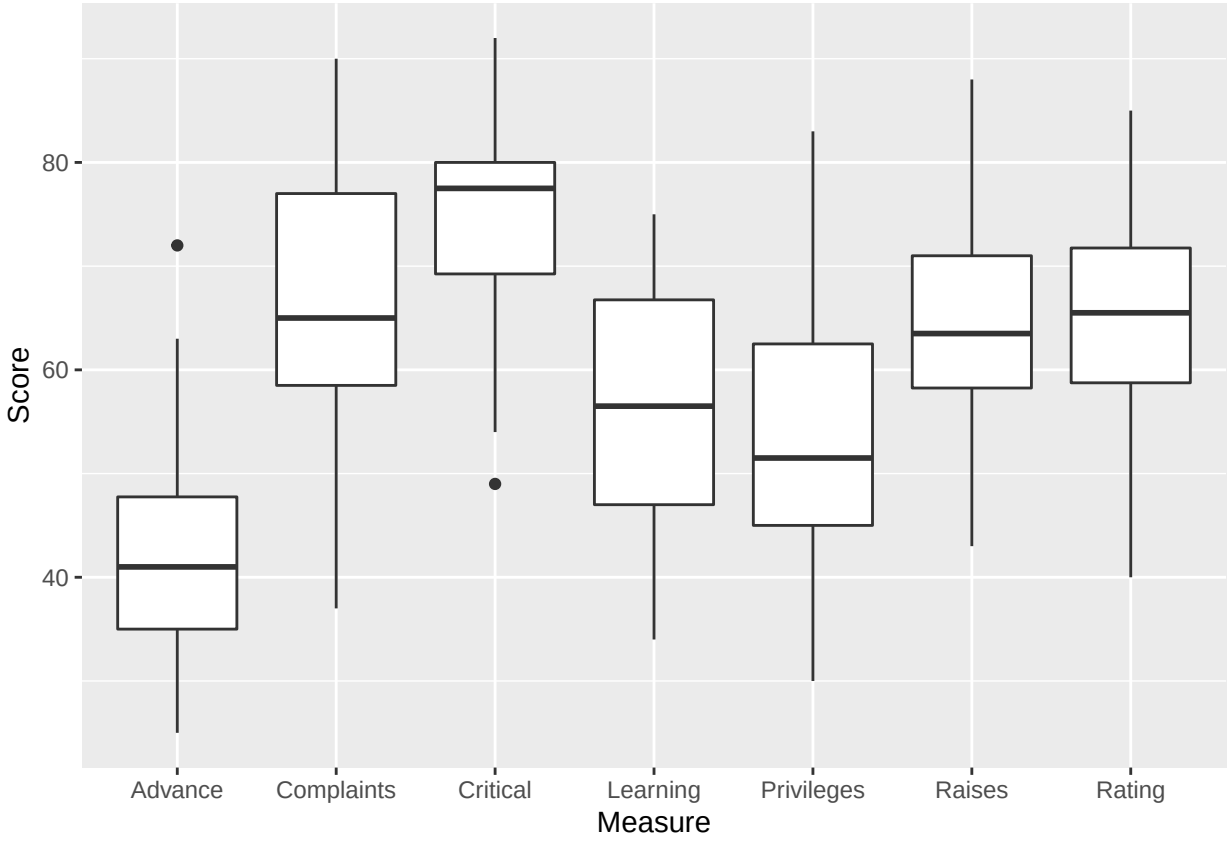
```
##   rating complaints privileges learning raises critical advance
## 1     43         51         30       39     61       92      45
## 2     63         64         51       54     63       73      47
## 3     71         70         68       69     76       86      48
## 4     61         63         45       47     54       84      35
## 5     81         78         56       66     71       83      47
```

```
## 6      43         55         49       44       54       49       34
```

**Explanation**: Each row contains scores for a particular employee on various measures. To find out more about these data you can type `?attitude` into the console.

---

Let's say we want a single plot of all these variables, something like this:




To do this we first need to convert the data to *long* format. In **long format**, each observation is saved in its own row, rather than across multiple columns.

It's often called **"row per observation"** data.

---

We can convert from wide to long using the `melt` command in the `data.table` package. First we need to load the `data.table` package:

```
library(data.table)
```

To see why the command is called 'melt', imagine taking three columns of data fo the first two rows:

```
attitude %>%
  select(rating, complaints, privileges) %>%
  head(2)
```

```
##   rating complaints privileges
## 1     43         51         30
## 2     63         64         51
```

If we use `melt` on this selection, we end up with this:

Figure 16: img: TrueWarrior

```
attitude %>%
  select(rating, complaints, privileges) %>%
  head(2) %>%
  melt()
```

```
##     variable value
## 1     rating    43
## 2     rating    63
## 3 complaints    51
## 4 complaints    64
## 5 privileges    30
## 6 privileges    51
```

The change works like like this:

In this example we don't have an explicit record of which participant was which in the `attitude` dataset, because the mapping to participants was implicit: each row was a new participant, but they were not marked.

We can make this explicit by adding a new column to the data with the `mutate` command:

```
attitude %>%
  select(rating, complaints, privileges) %>%
  mutate(person = row_number()) %>%
  head(2)
```

```
##   rating complaints privileges person
## 1     43         51         30      1
## 2     63         64         51      2
```

And now, if we repeat the melt, we can specify `id.var="person"`. This tells R which column identifies the rows:

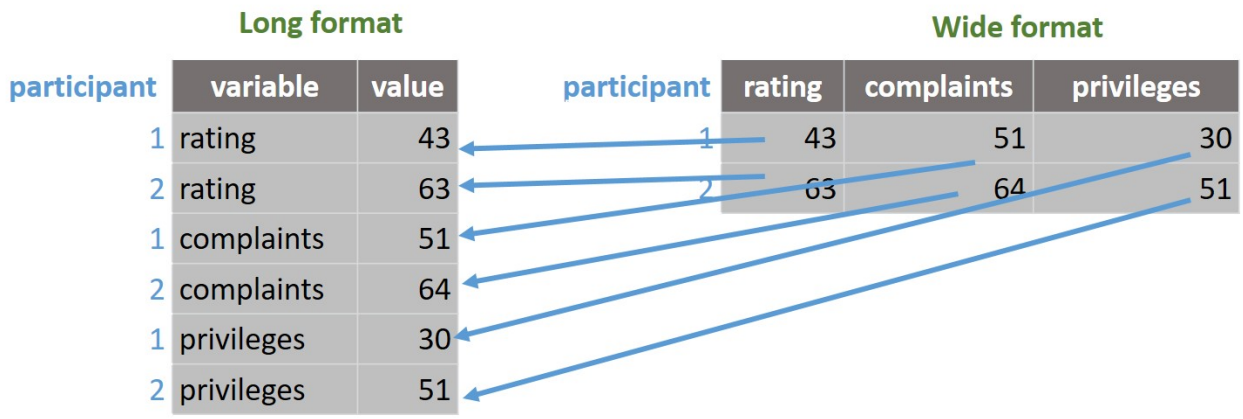


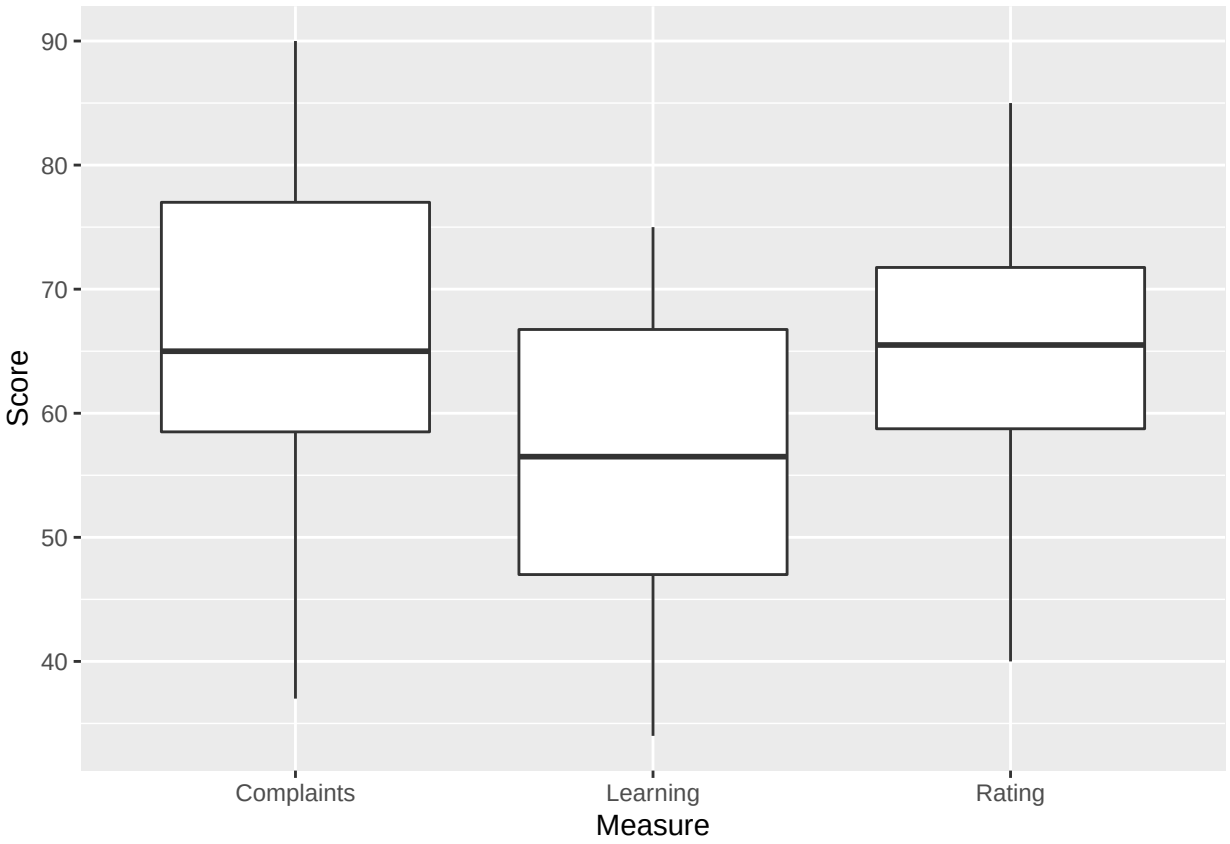

Figure 17: Converting from wide format to long format

```
attitude %>%
  select(rating, complaints, privileges) %>%
  mutate(person = row_number()) %>%
  head(2) %>%
  melt(id.var="person")
```

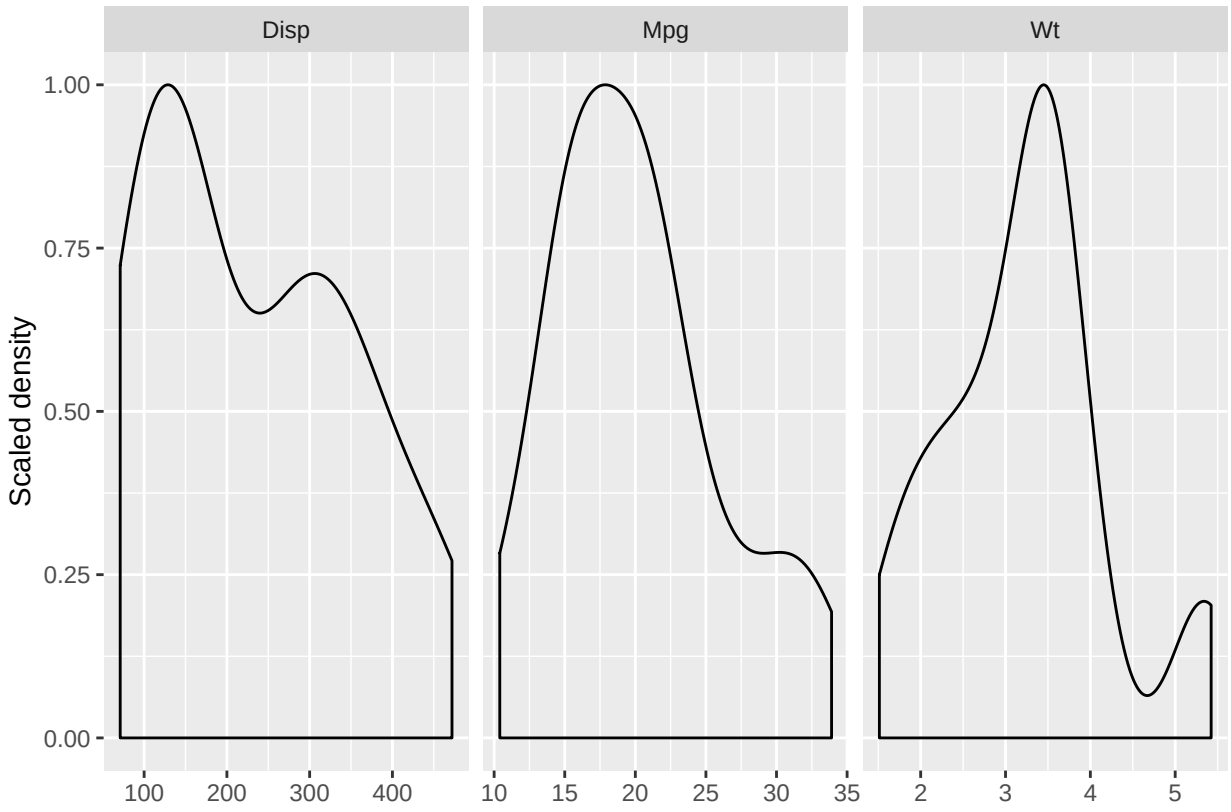
```
## Warning in melt(., id.var = "person"): The melt generic in data.table has
## been passed a data.frame and will attempt to redirect to the relevant reshape2
## method; please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
## reshape2::melt(.). In the next version, this warning will become an error.
```

```
##   person   variable value
## 1      1     rating    43
## 2      2     rating    63
## 3      1 complaints    51
## 4      2 complaints    64
## 5      1 privileges    30
## 6      2 privileges    51
```

Use the tidyverse commands you know plus `melt` to produce this plot:

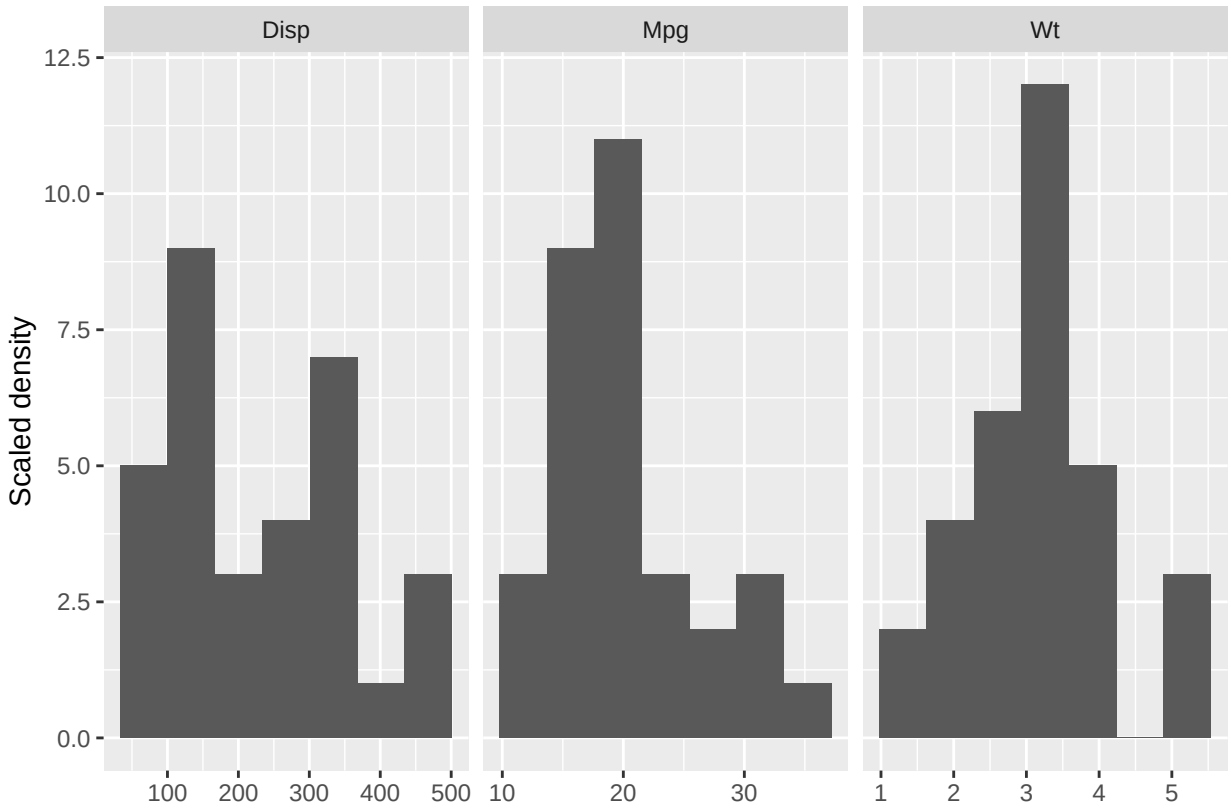



For an extension (and only if you have plenty of time), use the `mtcars` data and facetting to produce these plots:




```
## Warning in melt(.): The melt generic in data.table has been passed a data.frame
## and will attempt to redirect to the relevant reshape2 method; please note that
## reshape2 is deprecated, and this redirection is now deprecated as well. To
## continue using melt methods from reshape2 while both libraries are attached,
## e.g. melt.list, you can prepend the namespace like reshape2::melt(.). In the
## next version, this warning will become an error.

## No id variables; using all as measure variables
```




As a tip, you will need to:

- add `scales="free_x"` to the `facet_wrap` function. So, it will read: `facet_wrap(~VARNAME, scales="free_x")`
- for the histogram, replace `geom_density()` with `geom_histogram()`
- adjust the number of vertical bars in the histogram by adding `bins=7` (or some other number) to `geom_histogram()`

**Using melt to make summaries**

Imagine we want a table of the mean score on each question in the attitude dataset. This would be fiddly if we just tried to use summarise before reshaping, but if we use `melt`, `group_by` and then `summarise` (in that order) it is easy to make a table like this:

```
## Warning in melt(.): The melt generic in data.table has been passed a data.frame
## and will attempt to redirect to the relevant reshape2 method; please note that
## reshape2 is deprecated, and this redirection is now deprecated as well. To
## continue using melt methods from reshape2 while both libraries are attached,
## e.g. melt.list, you can prepend the namespace like reshape2::melt(.). In the
## next version, this warning will become an error.
```

| variable | mean | sd |
|---|---|---|
| rating | 64.63 | 12.17 |
| complaints | 66.6 | 13.31 |
| privileges | 53.13 | 12.24 |
| learning | 56.37 | 11.74 |
| raises | 64.63 | 10.4 |
| critical | 74.77 | 9.89 |
| advance | 42.93 | 10.29 |

Use the `melt`, `group_by` and `summarise` commands (in that order) to reproduce the table above.

## Reshaping wide (casting)

Sometimes we have the opposite problem: We have long data, but want it in wide format. For example, we might want a table where it's easy to compare between different years, like this:

```
## Warning in data.table::dcast(., country ~ year): The dcast generic in data.table
## has been passed a tbl_df and will attempt to redirect to the reshape2::dcast;
## please note that reshape2 is deprecated, and this redirection is now deprecated
## as well. Please do this redirection yourself like reshape2::dcast(.). In the
## next version, this warning will become an error.
```

Table 4: GDP per-capita in 3 different years from the gaminder dataset.

| country | 1997 | 2002 | 2007 |
|---|---|---|---|
| Albania | 3193 | 4604 | 5937 |
| Austria | 29096 | 32418 | 36126 |
| Belgium | 27561 | 30486 | 33693 |
| Bosnia and Herzegovina | 4766 | 6019 | 7446 |
| Bulgaria | 5970 | 7697 | 10681 |
| Croatia | 9876 | 11628 | 14619 |

---

To do this `data.table` has a command called `dcast`. The metaphor here is *casting* a fishing line or net — that is, turning something that was long/vertical into something wide/horizontal.


Figure 18: Hopefully not quite like this. Image source: gifbin

---

First make sure the package is loaded:

```
library(data.table)
```

As we saw before the `gapminder` data is a fairly long format. There are multiple rows per-country corresponding to different years.

To cast `gapminder` to a wide format to *compare* years we first need to select the data we want — country, year and GDP:

```
gapminder1990s <- gapminder::gapminder %>%
  select(country, year, gdpPercap) %>%
  filter(year > 1990)
```

Then we add the `dcast` command:

```
gapminder1990s %>%
  dcast(country~year) %>%
  head()
```

```
## Warning in dcast(., country ~ year): The dcast generic in data.table has been
## passed a tbl_df and will attempt to redirect to the reshape2::dcast; please
## note that reshape2 is deprecated, and this redirection is now deprecated as
## well. Please do this redirection yourself like reshape2::dcast(.). In the next
## version, this warning will become an error.
```

```
## Using 'gdpPercap' as value column. Use 'value.var' to override
```

```
##       country       1992       1997       2002       2007
## 1 Afghanistan   649.3414   635.3414   726.7341   974.5803
## 2     Albania  2497.4379  3193.0546  4604.2117  5937.0295
## 3     Algeria  5023.2166  4797.2951  5288.0404  6223.3675
## 4      Angola  2627.8457  2277.1409  2773.2873  4797.2313
## 5   Argentina  9308.4187 10967.2820  8797.6407 12779.3796
## 6   Australia 23424.7668 26997.9366 30687.7547 34435.3674
```

**Explanation of the output**: We started with multiple rows per country, corresponding to years. We used `dcast(country~year)` to cast the data into wide format. The result has countries listed one-per-row; data for each year are shown in a separate column. This helps us compare years within countries.

**Explanation of the command**: The `country~year` part tells dcast to create a table where countries make up the rows, and years make up the columns. That is the variables to the left of the `~` symbol will define the rows in the table, and the variables to the right of the `~` will create new columns.

The squiggly `~` symbol is called a tilde (say "tilder").

- On a Mac it's the key to the left of the `Z` (you need to press `shift` to type it).
- On a PC it's normally between the `@` symbol and the `Enter` key.

We can add multiple variables to the left or right of the `~` to create different shape tables.

For example:

```
gapminder::gapminder %>%
  select(continent, country, year, gdpPercap) %>%
  filter(year > 1990) %>%
  dcast(continent + country ~ year) %>%
  head()
```

```
##   continent      country      1992      1997       2002       2007
## 1    Africa      Algeria 5023.2166 4797.2951  5288.0404  6223.3675
## 2    Africa       Angola 2627.8457 2277.1409  2773.2873  4797.2313
```

```
## 3    Africa        Benin 1191.2077 1232.9753  1372.8779  1441.2849
## 4    Africa     Botswana 7954.1116 8647.1423 11003.6051 12569.8518
## 5    Africa Burkina Faso  931.7528  946.2950  1037.6452  1217.0330
## 6    Africa      Burundi  631.6999  463.1151   446.4035   430.0707
```

Experiment for yourself with `dcast`:

- What happens if we write `dcast(year ~ country)` instead of `dcast(country ~ year)`?
- Use more than one variable on the left or right of the `~`
- Experiment with some different datasets

As an extension:

- Think about how you might use `dcast` with the `group_by` and `summarise` approach we used above. Can you create an example where using these techniques together produces a useful table?

## RMarkdown

Markdown is a simple **text format** which can include writing, tables and images (see http://commonmark.org/help/).

An RMarkdown document mixes R code with Markdown. You can generate images (plots) and tables directly using R code.

In the language of RStudio, you **'knit'** your RMarkdown document to produce a finished document. Knitting:

1. runs your code and
2. combines the analyses, graphs, with your explanatory text to
3. make a single pdf or html file.

This knitted file documents your work and can be shared easily with others.

**Why use Rmarkdown?** Rmarkdown is becoming an important tool in the open science movement. Authors often share details of their analysis in an Rmarkdown file because journal articles are too constrained to document this properly.

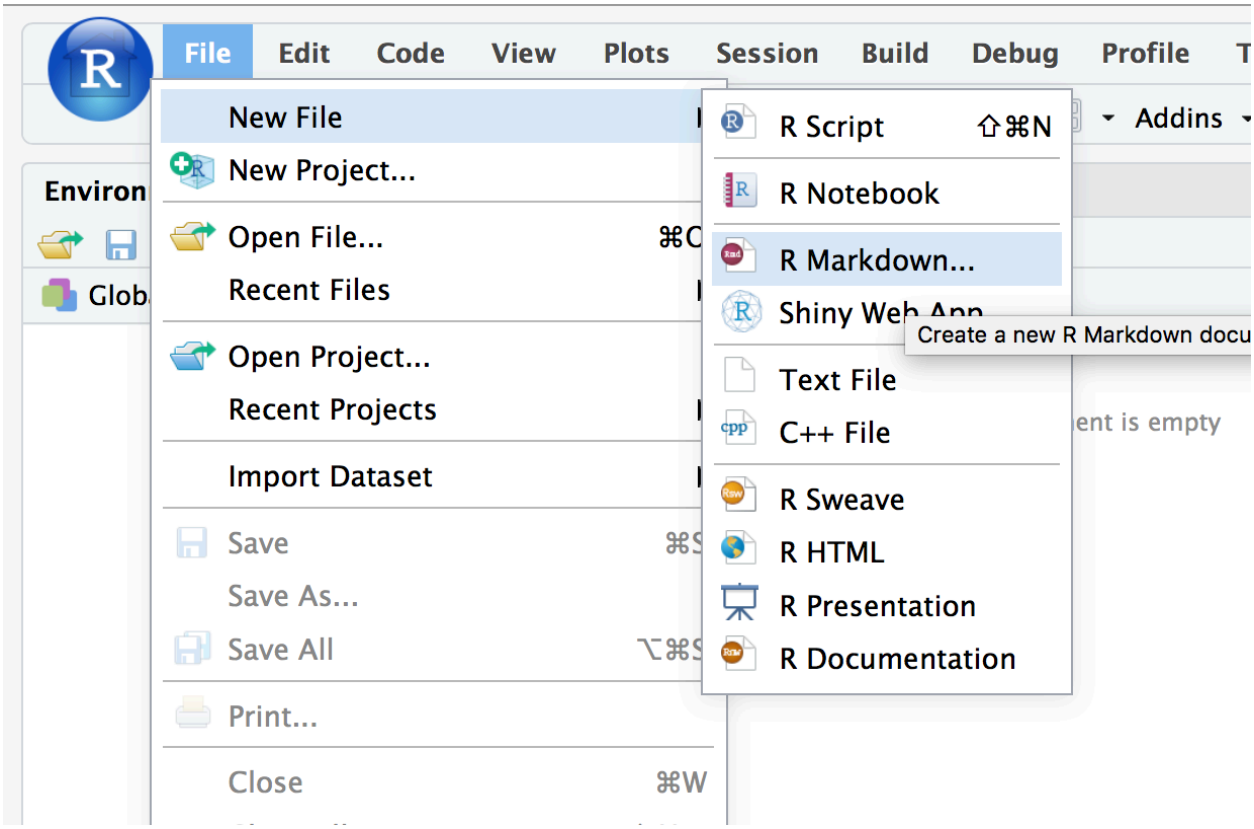
A recent example from Plymouth is shared here: https://zenodo.org/record/1120364. This repository includes:

- The datafile in CSV format
- The Rmarkdown source file
- The knitted html document (click here to have a look)

The main assessment for this module requires you to make an Rmd file like this to share a re-analysis of a published paper, so learning how to use Rmd is an important part of the course.

### Creating a fresh RMarkdown file

The easiest way to create and RMarkdown file is from the `File > New > RMarkdown` menu option:




It should look something like this when you've done it:

**Explanation** We asked RStudio to create a new Rmd file. Rstudio makes the new file from a template, and this shows off some of the main features of RMarkdown.

---

RMarkdown files have the file extensions `.rmd` and must to be saved before they can be 'knitted'.

1. Create a new Rmd file and save it somewhere in your home directory on the Rstudio server (or the same place you saved your other R Script files).
2. Have a look through the example file created in RStudio. Have a guess at what each part does.
3. Press the 'knit' button in the RStudio interface and check the results.

### Working with 'chunks' of R code

To include R code within RMarkdown we write 3 backticks followed by `{r}`.

We then write some R code, and close the chunk with 3 more backticks. This is what it looks like:

Can't find the backtick on your keyboard?

As you do this exercise, take care to get the right number of backticks, and your curly brackets (`{}`) in the right place.

1. Create a new 'chunk' of R code which includes one of the plots you have made so far today. Use one of the built in datasets for now.
2. Make sure you add `library(tidyverse)` just above your plot code.
3. Press 'knit' again and check the result.

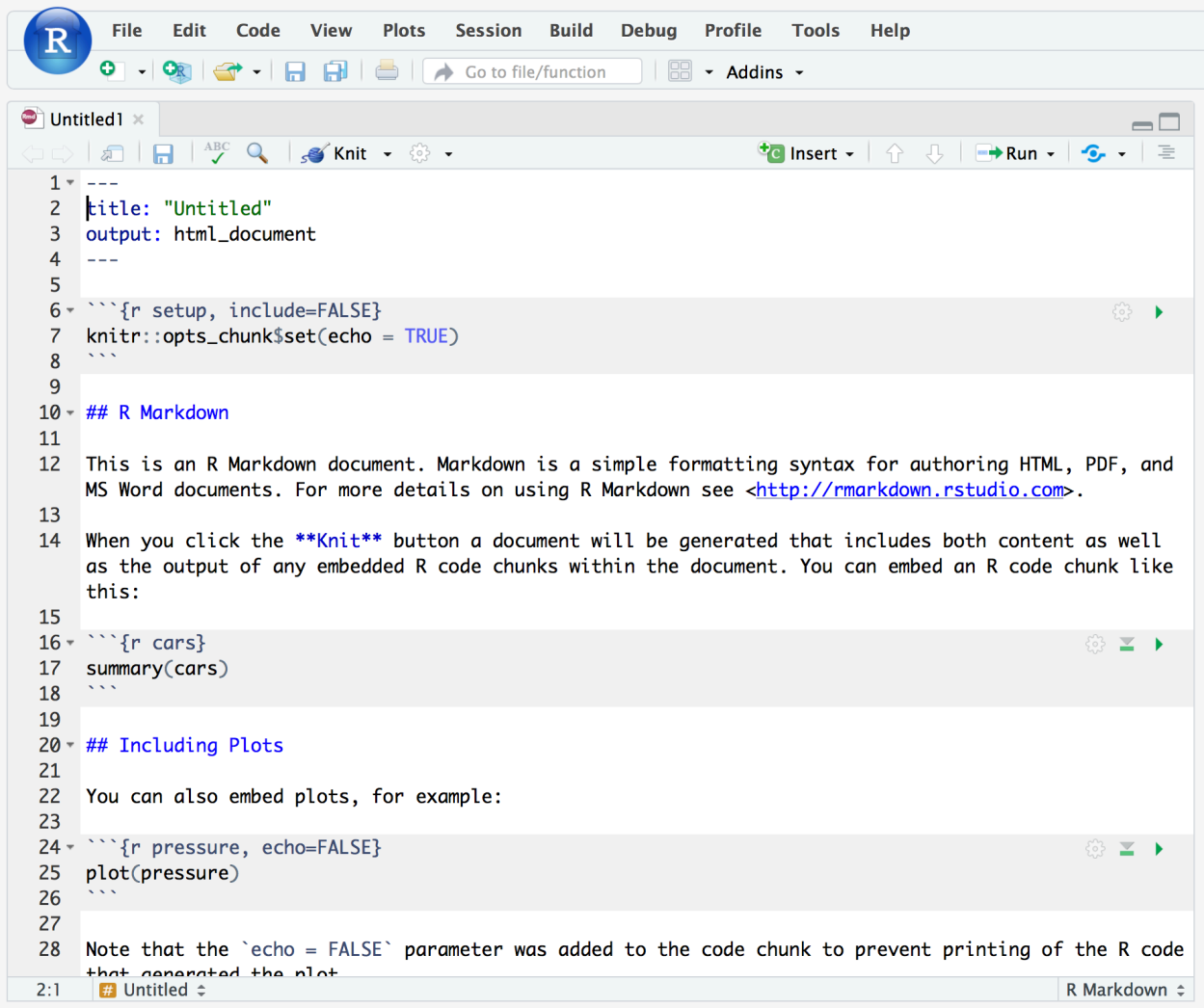



Figure 19: A new Rmd file, from the RStudio template.

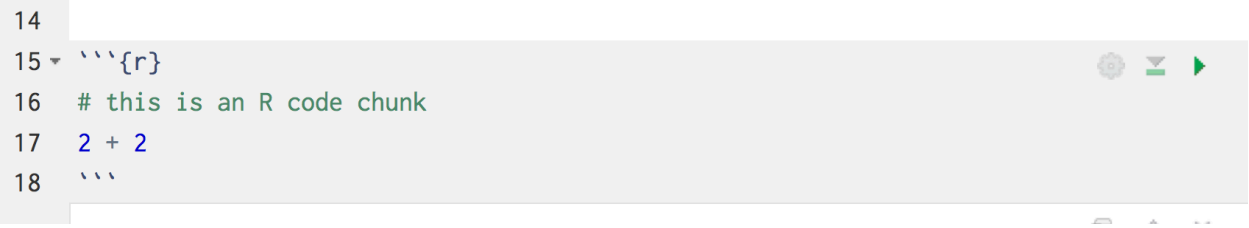



Figure 20: A code chunk in the RMarkdown editor

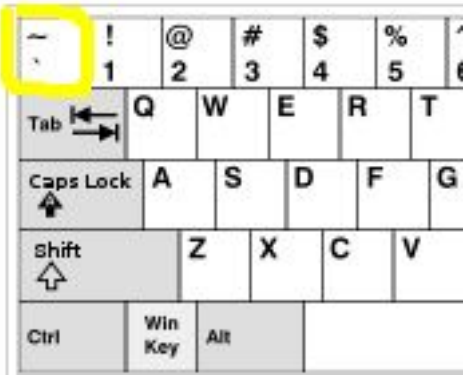



Figure 21: On windows


Figure 22: On a Mac

If you are getting errors when adding a code chunk always check you have:

- 3 backticks, then `{r}`, then **start a new line**
- Your R code comes next
- Load any packages needed at the top of the first chunk.
- 3 more backticks, **on their own line**. These final 3 backticks **must** start at the beginning of the line, have no spaces in front of them, and nothing after them.

If you are still getting errors:

- Read the error message - it can give a clue
- Simplify everything (remove R code until it works, then try again)

Using libraries in chunks:

- Load packages (e.g. `library(tidyverse)`) once at the top of the file.
- Remember that RStudio knits code from top to bottom (so load packages **before** you use them)
- R wipes it's memory each time it knits your file, so needs reminding. You only need to load the package in one chunk though (don't do it multiple times).

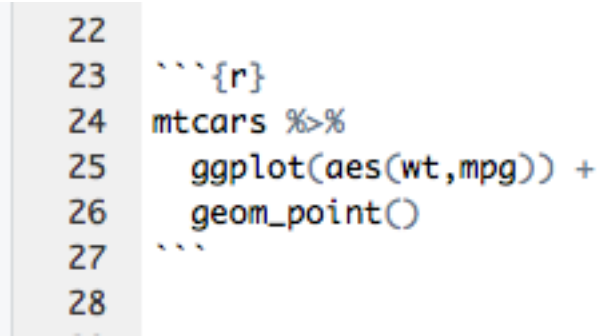
### Working interactively with Rmarkdown

Rather than typing commands in the console, **you should now run code directly from your Rmd code chunks**.

This makes it much easier to keep track of your work.

See this video for an example:

1. Put your cursor (pointer) anywhere inside the new chunk you added, containing the R code for your plot. The chunk should look something like this:




2. Press the green 'run' button (a rightward pointing triangle) in the top right corner of the chunk. Where does the result appear?
3. Change the R code in your chunk (e.g. make a different plot). Press 'run' again.

4) Write several lines of R code in your chunk. Put the cursor on one of the lines and press `ctrl+enter` or `cmd+enter` if you are on a mac. Does this run all of the code in the chunk, or just part of it?

5. Write some code which uses `summarise` inside your code chunk to produce a table

## Final task (to be completed as homework)

Work in your previous groupings.

Use Rmarkdown to present your findings from the Berkley admissions data we covered in the past session. It should:

- Load the data and packages needed

- Use section headings (i.e. use the `#` or `##` to create a 1st or 2nd level heading)
- Include 2 or 3 plots
- Include text describing your conclusions on the main question (was the admissions system 'fair')

When you are finished, knit the Rmd document to produce an html or PDF output (it's actually a good idea to knit regularly to pick up any errors).

## Extension exercises and reading

- Read this article on wide/long data formats and the tradeoffs between the two: https://www.theanalysisfactor.com/wide-and-long-data/

### Making nicer tables in RMarkdown

The `pander` function in the `pander` library can help make nicer tables when using RMarkdown.

For example, if we take a few rows from the Iris data they look like this:

```
iris %>% head
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

But if we load the `pander` package, and use the `pander` command we get a nice table, like this:

```
library(pander)
iris %>% head %>% pander
```

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |

And we can add a caption to the table like this:

```
iris %>% head %>% pander(caption="The first 6 rows from the iris dataset")
```

Table 6: The first 6 rows from the iris dataset

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |

Revisit one of the tasks that used `group_by` + `summarise`. Use pander to display the table in a nicer format, and knit your Rmarkdown file to pdf/html to see the results.

**Separating 'untidy' variables into tidy, long-form data:**

The code below generates simulated data for 100 individuals at three time points. The format is similar to the way you might record experimental data in a spreadsheet.

```
N <- 20
repeatmeasuresdata <- tibble(person = 1:N,
  time_1 = rnorm(N),
  time_2 = rnorm(N, 1),
  time_3 = rnorm(N, 3))

repeatmeasuresdata %>% head
```

```
## # A tibble: 6 x 4
##   person   time_1 time_2 time_3
##    <int>    <dbl>  <dbl>  <dbl>
## 1      1  0.641    1.36    3.52
## 2      2  0.812    1.56    4.24
## 3      3  0.317    0.499   3.30
## 4      4 -0.00605  0.497   2.83
## 5      5 -0.490    1.84    2.25
## 6      6 -0.374    0.719   1.45
```

`repeatmeasuresdata` is in **wide** format. Each row contains data for one participant, and each participant has three observations.

We can melt the data into long format like so:

```
repeatmeasuresdata %>%
  melt(id.var = "person") %>%
  arrange(person, variable) %>%
  head(7)
```

```
## Warning in melt(., id.var = "person"): The melt generic in data.table has been
## passed a tbl_df and will attempt to redirect to the relevant reshape2 method;
## please note that reshape2 is deprecated, and this redirection is now deprecated
## as well. To continue using melt methods from reshape2 while both libraries are
## attached, e.g. melt.list, you can prepend the namespace like reshape2::melt(.).
## In the next version, this warning will become an error.
```

```
##   person variable     value
## 1      1   time_1 0.6412817
## 2      1   time_2 1.3567959
## 3      1   time_3 3.5237304
## 4      2   time_1 0.8124611
## 5      2   time_2 1.5644390
## 6      2   time_3 4.2405214
## 7      3   time_1 0.3173578
```

The problem we have now is that `variable` contains text which describes at which time the observation was made. We probably want a number for each timepoint, so we can make a plot with time on the x axis.

The `separate` command separates a single character column (`variable`) into multiple columns. Rather than have a column with labels of the form 'time_1', it can create two columns, with labels 'time' and '1' in each.

```
longrepeatmeasuresdata <- repeatmeasuresdata %>%
  melt(id.var = "person") %>%
  separate(variable, into = c("_", "time"))
```

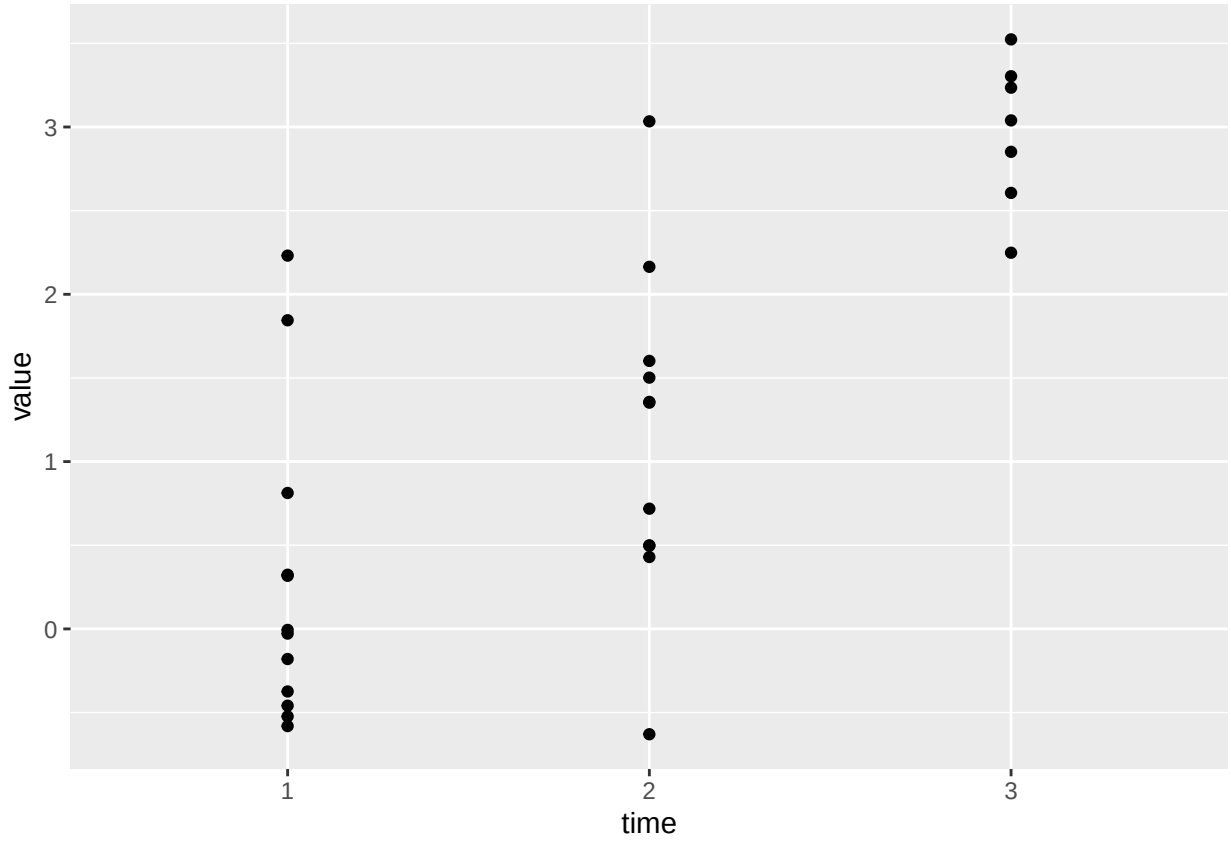
```
## Warning in melt(., id.var = "person"): The melt generic in data.table has been
## passed a tbl_df and will attempt to redirect to the relevant reshape2 method;
## please note that reshape2 is deprecated, and this redirection is now deprecated
## as well. To continue using melt methods from reshape2 while both libraries are
## attached, e.g. melt.list, you can prepend the namespace like reshape2::melt(.).
## In the next version, this warning will become an error.
```

```
longrepeatmeasuresdata %>% head
```

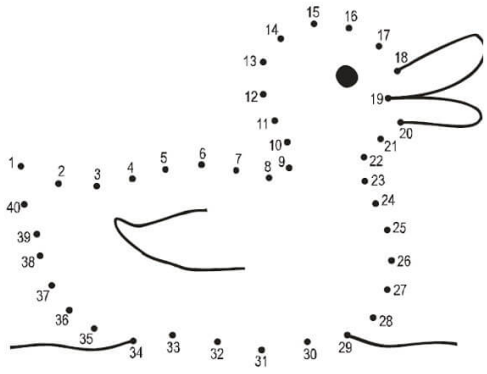
```
##   person    _ time        value
## 1      1 time    1  0.641281727
## 2      2 time    1  0.812461141
## 3      3 time    1  0.317357780
## 4      4 time    1 -0.006047964
## 5      5 time    1 -0.490141987
## 6      6 time    1 -0.373949677
```

Now the data are in long format, we can plot the points over time:

```
longrepeatmeasuresdata %>%
  sample_n(30) %>%
  ggplot(aes(x=time, y=value)) +
  geom_point()
```



# Regression



### In brief

Regression is just a fancy term for drawing the 'best-fitting' line through a scatter-plot, and summarising how well the line describes the data.

When using regression in R, the relationship between an ***outcome*** and one or more ***predictors*** is described using a ***formula***. When a model is '***fitted***' to a sample it becomes tool to make ***predictions*** for future samples. It also allows us to quantify our ***uncertainty*** about those predictions.

***Coefficients*** are numbers telling us how strong the relationships between predictors and outcomes are. But the meaning of coefficients depend on the study ***design***, and the ***assumptions*** were are prepared to make. Causal diagrams can help us choose models and interpret our results.

Multiple regression is a technique which can describe the relationship between ***one outcome*** and ***two or more predictors***. We can also use multiple regression to describe cases where two variable ***interact***. That is, when the effect of one predictor is increased or decreased by another. Multiple regression is important because it allows us to make more realistic models and better predictions.

Like any sharp tool, regression should be used carefully. If our statistical model doesn't match the underlying network of causes and effects, or if we have used a biased sample, we can be misled.

## Session 1

**Overview** In this session we will revise core concepts for selecting and interpreting linear models. Some of the material may have been covered in undergraduate courses, but we will emphasise understanding and mastery of core ideas before we develop these ideas to fit more complex models.

### Fitting lines to data

Regression (and most statistical modelling) is about 'fitting' lines to data. Our first exercise illustrates most of the key concepts without you needing to touch a computer.

In this activity you will need to:

- Work in groups
- Use some example plots
- Decide how to draw lines on these plots which represent a 'good fit'

**Study habits and academic outcomes**

For this activity I've provided some plots from a simulated dataset on study habits and academic outcomes.

(If you're doing this exercise on your own at home, the example are available here: example-plots.pdf.

The data for the example include:

- MCQ test data (i.e. academic achievement)
- Responses to a study habits questionnaire (including a question on 'hours spent studying')

In these examples, each plot only contains 10 of the data points from the larger sample (N=300).

Your task is to describe the ***relationship between hours spent working and exam grades*** with lines (hand) drawn on the plots.

1. In groups, take one of the printed graphs, and a plastic transparency.
2. Put the transparency on top of the plot.

As a group, draw 3 lines on the transparency in different colours:

1. First, draw which you think is the 'best' line. That is, the line that describes the relationship between study hours and exam grades the best.
2. Second, discuss the pros and cons of fitting a straight vs. a curved line. If you initially drew either a straight line, draw a curved line now in another colour (or vice versa).
3. Finally, draw a ***really*** curvy line with multiple bends to get as close as you can to all the data points in your sample scatter plot.

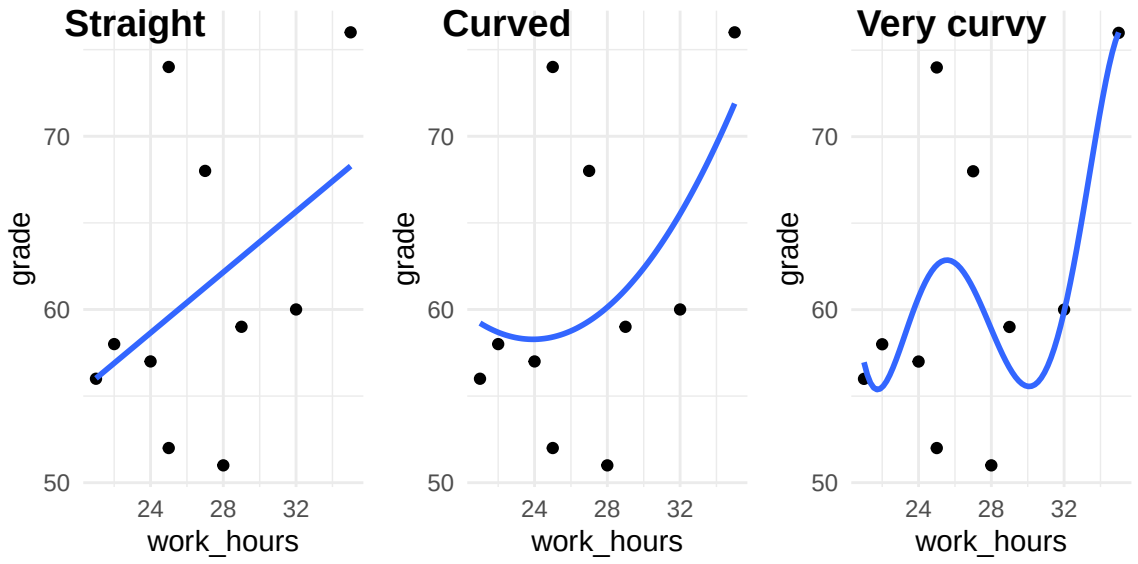



Figure 23: Examples of straight and curved lines fit to the data.

## How useful are the lines?

You can think of the lines we drew in two ways: as **maps** and as **tools**:

1. As a map, they **describe** the data we have, but they are also
2. **tools** which **predict** new data we might collect

In statistical language, the gaps between our line and the data points are called ***residuals***.

We can distinguish:

1. Residuals for the data we have now (how well does the line ***describe*** the data).
2. Residuals for new data we collect after drawing the line (how well does the line ***predict***).

Another common way to refer to residuals is as the **error** in a model. That is, the line describes a *model* (idealised) relationship between variables. And if look at the residuals we have an estimate how how much *error* there will be in our predictions when we use the model.

In your group:

- Discuss how well/badly your lines 'fit' to the sample you have (in general terms)
- Using a ruler, measure (in mm) the residual for each datapoint on your current graph. Do this separately for the straight and curvy lines. If you need to save time, only do this for the first 5 data points.
- Add up the total length of the distances (residuals) for each line. Make a note of this for later.
- Repeat the exercise at least 3 times, swapping your printout with another groups' plot. (Each plots shows a different sample).

When you have finished, discuss in your group:

- Do curved or straight lines have smaller residuals for the *original* data that were used to draw them?
- Do curved or straight lines have smaller residuals for *new* data (i.e. after swapping?)
- What do you think is going on here? What can explain the pattern you see?

---

Only when you have discussed this thoroughly, read an explanation of what is going on.

**Congratulations!**

You have successfully fit your first linear model! The next step is to formalise the process. We need a method that:

- Finds the line with the ***smallest*** residuals
- Is repeatable
- Is easy for computers to do (because we're lazy)

For this we can use R!

## Using R for regression

Before we start, the study habits data are stored at this url: https://benwhalley.github.io/rmip/data/studyhabitsandgrades.csv

Previously we have loaded data by:

- Downloading the csv to our computer
- Uploading it to RStudio server
- Opening it using `read_csv`

**A shortcut** A quicker way is to combine these 3 steps: By providing `read_csv` with the url, we can open the data in a single step:

```
studyhabits <- read_csv('https://benwhalley.github.io/rmip/data/studyhabitsandgrades.csv')
```

**Explanation**: By providing a URL to `read_csv` we can open the data over the web.

We should check the data look OK using `head` or `glimpse`:

```
studyhabits  %>% head()
studyhabits  %>% glimpse()
```

**Explanation**: `head` shows the first 6 rows of the dataset. `glimpse` provides a list of all the columns and (if your window is wide enough) will also show the first few datapoints for each.

## The first step is always plotting

**Before** we start running analyses, we should always plot the data.

Plot the `studyhabits` data in a few different ways to get a feel for the relationships between the variables. Specifically,

1. Make a density plot to see the distribution of `grade` scores
2. Add colour to this plot (or use another type of plot) to see how scores differ by gender
3. Use a scatter plot to look at the relationship between `grade` and `work_hours`.
4. Is the relationship between `grade` and `work_hours` the same for men and women?

Interpret your plots:

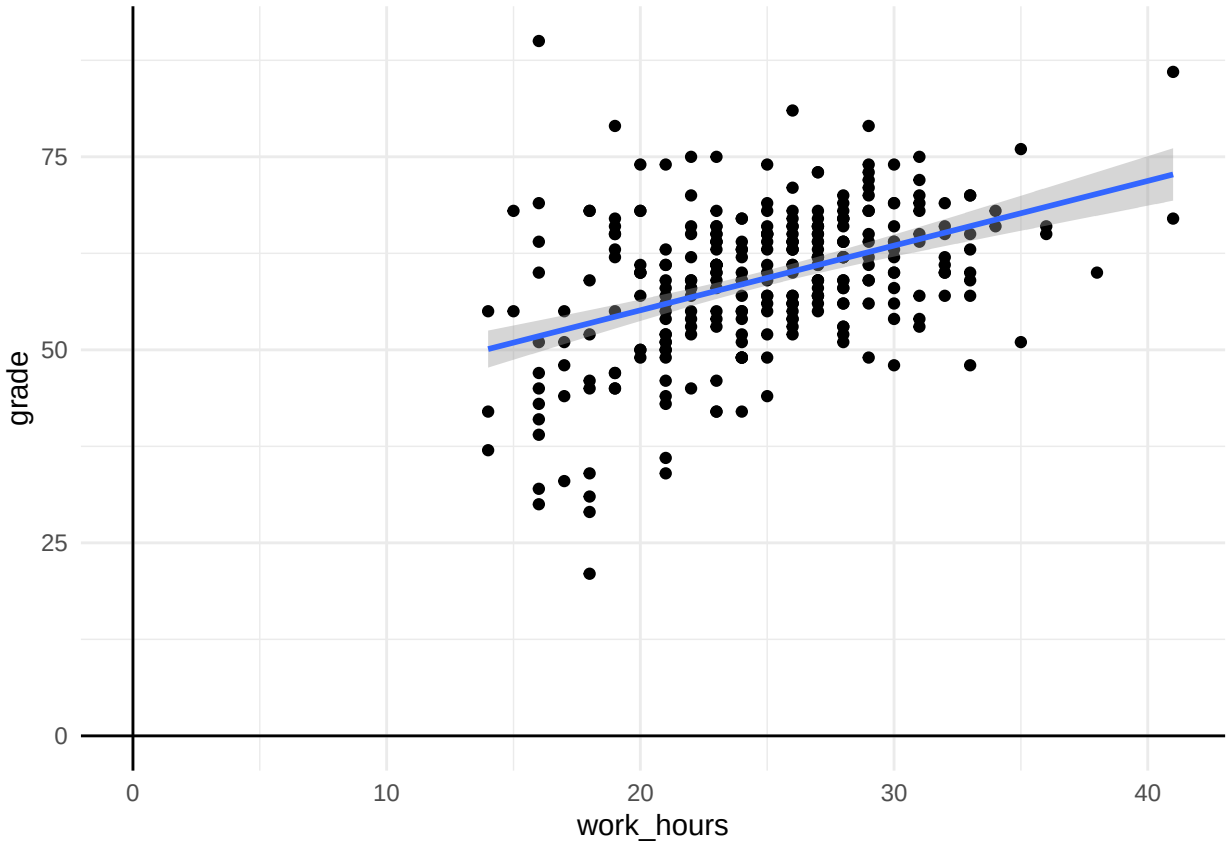
- What relationship do we see between revision and grades?
- Do you estimate this a weak, moderate or strong relationship?

## Automatic line-fitting

To get R to fit a line to these data for us we will use a new function called `lm`. The letters l and m stand for *linear* ***m****odel*.

There are lots of ways to use `lm`, but the easiest to *picture* is to get `ggplot` to do it for us:

```
studyhabits %>%
  ggplot(aes(work_hours, grade)) +
  geom_point() +
  geom_smooth(method=lm)
```




**Explanation of the code**: We used `geom_point` to create a scatterplot. Then we used a plus symbol (`+`) and added `geom_smooth(method=lm)` to add the fitted line. By default, `geom_smooth` would try to fit a curvy line through your datapoints, but adding `method=lm` makes it a straight line.

**Explanation of the resulting plot** The plot above is just like the scatter plots we drew before, but adds the blue fitted line. The blue line shows the 'line of best fit'. This is the line that **minimises the residuals** (the gaps between the line and the points). Ignore the shaded area for now (explanation here if you are keen).

Try plotting a line graph like this for yourself, with:

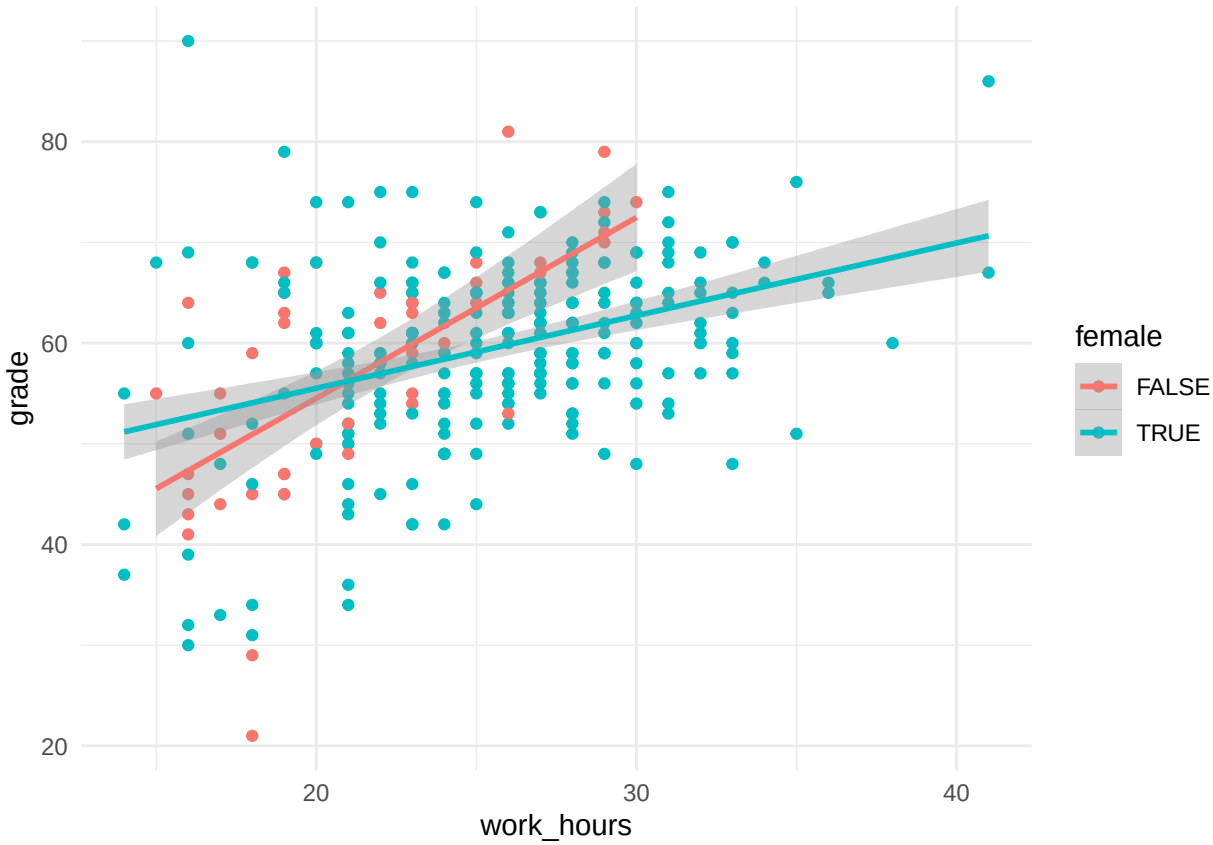
- The same variables (i.e. reproduce the plot with `work_hours` and `grade`)
- Different variables (from the `studyhabits` dataset)
- Without the `method=lm` part (to see a curvy line instead of straight)

Note whether the slope of the line is positive (upward sloping) or negative (downward sloping).

Now, add the `colour=female` inside the part which says `aes(...)`. Before you run it, predict what will happen.

show answer

It should plot different lines for men and women. Something like this:




## Putting numbers to lines

The plot we made in the previous section is helpful, because we can *see* the best-fit line.

However also want to have a *single number to say how steep the line is*. That is, *a number to say how closely related the variables are*.

To do this we can use the `lm` function directly.

Before we start make sure you have loaded the studyhabits dataset:

```
studyhabits <- read_csv('https://benwhalley.github.io/rmip/data/studyhabitsandgrades.csv')
```

In the next piece of code we fit the model. The first part is known as a model formula. The `~` symbol (it's called a 'tilde') just means "is predicted by", so you can read this part as saying "grade is predicted by work hours".

```
first.model <- lm(grade ~ work_hours, data = studyhabits)
first.model
```

```
##
## Call:
## lm(formula = grade ~ work_hours, data = studyhabits)
##
## Coefficients:
## (Intercept)   work_hours
##     38.3696       0.8377
```

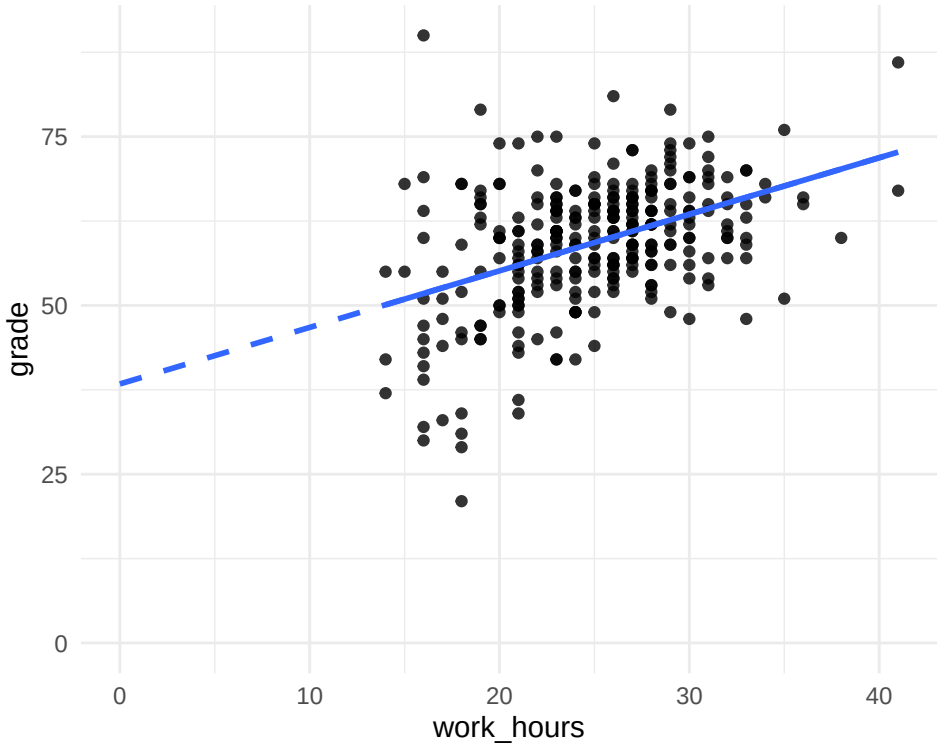
**Explanation of the code**: We used the `lm` function to estimate the relation beteen grades and work hours.

**Explanation of the output**: The output displays:

- The 'call' we made (i.e. what function we used, and what inputs we gave, so we can remember what we did later on)
- The 'coefficients'. These are the numbers which represent the line on the graph above.

**Explanation of the coefficients** In this example, we have two coefficients: the `(Intercept)` which is 38.3696 and the `work_hours` coefficient which is 0.8377.

**The best way to think about the coefficients is in relation to the plot we made**. In this version of the plot, however, I extended the line so it crosses zero on the x axis:




We can interpret the coefficients as points on the plot as follows:

- The `(Intercept)` is the point (on the y axis) where the blue dotted line crosses zero (on the x-axis).
- The `work_hours` coefficient is how **steep** the slope of the line is. Specifically, is says how many `grade` points the line will rise if we increase `work_hours` by 1.

Before you move on:

- Compare the coefficients from the `lm` output to the plot above. Can you see how they relate? If you're not 100% sure, ask now!

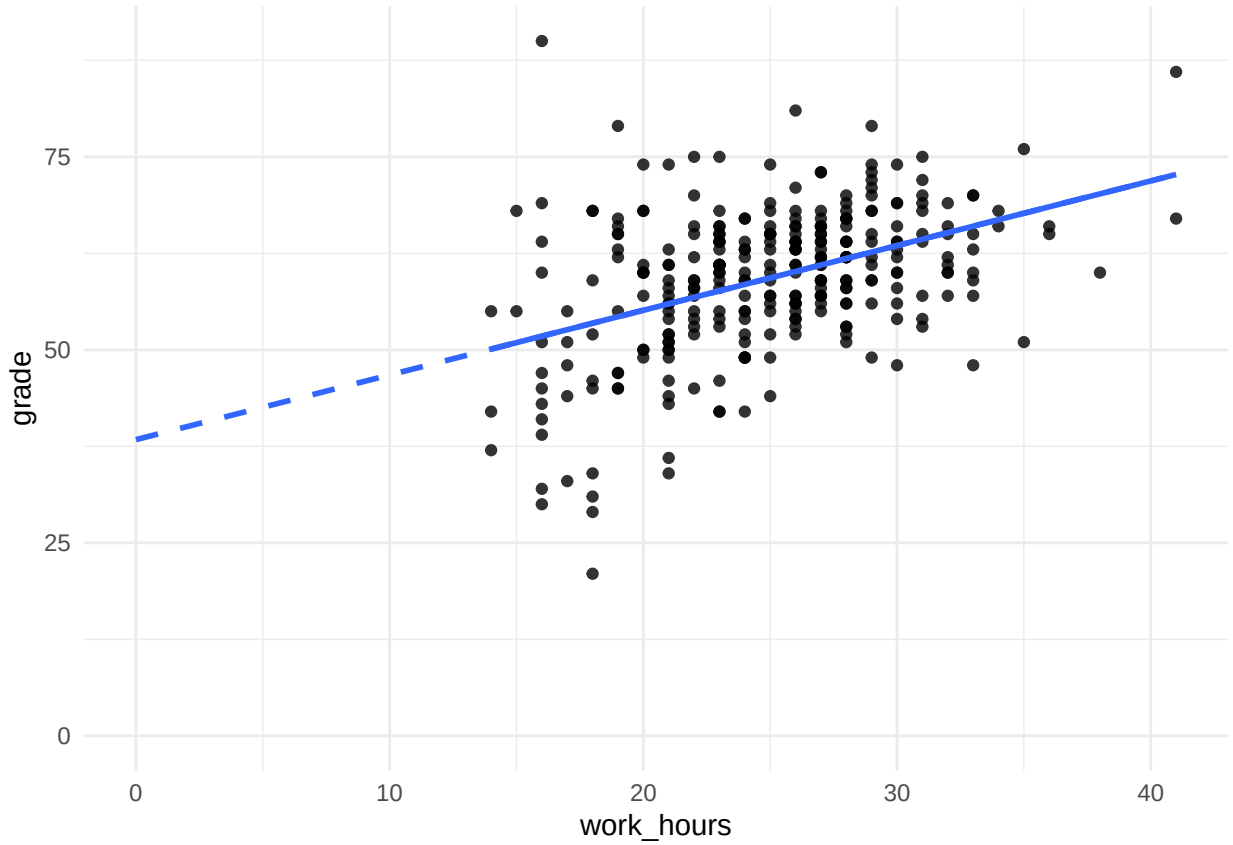
## Making predictions (by hand)

A big advantage of using the coefficients alongside the plot is that we can easily ***make predictions for future cases***.

In a pair, do this now:

Let's say we meet someone who works 30 hours per week. One way to predict would be by-eye, using the line on the plot.

What grade would you expect them to get simply by 'eyeballing' the line?




We can do the same thing using the coefficients from `lm`, because we know that:

- If someone worked for 0 hours per week then our prediction would be 38. We know this because this is the intercept value (the point on the line when it is at zero on the x-axis).
- For each extra hour we study the `work_hour`coefficient tells us that f, our `grade` will increase by 0.8377.
- So, if we study for 30 hours, our prediction is $38 + 0.8377 \times 30$

In pairs again:

- Make predictions for people who study 5, 20 or 40 hours per week
- Compare these predictions to the plot above.
- Which of the predictions (for 5, 20 or 40 study hours) should we be most confident in? Why do you think this is?

If you want to check your predictions, click here

### Making predictions using code

Rather than making predictions by hand, we save time by using the `predict()` function.

If we run a regression, we can should save the fitted model with a named variable:

```
first.model <- lm(grade ~ work_hours, data = studyhabits)
```

**Explanation**: As before, we ran a model and saved it to the named variable, `first.model`.

If we feed this model to the predict function, we get a model prediction for each row in the original dataset:

```
predict(first.model) %>% head(10)
```

```
##        1        2        3        4        5        6        7        8
```

```
## 60.15051 60.15051 51.77323 66.01460 59.31278 55.12414 65.17687 64.33915
##         9        10
## 56.79960 66.01460
```

**Explanation of the output**: We have one prediction (the point on the line) for each row in the original dataset.

---

We can also use the `augment` function in the `broom` package to do make the predictions, but return them *with* the original data. This can make it easier to use:

```
library(broom)
augment(first.model) %>%
 head()
```

```
## # A tibble: 6 x 9
##   grade work_hours .fitted .se.fit .resid    .hat .sigma  .cooksd .std.resid
##   <dbl>      <dbl>   <dbl>   <dbl>  <dbl>   <dbl>  <dbl>    <dbl>      <dbl>
## 1    65         26    60.2   0.513   4.85 0.00351   8.67 0.000554      0.561
## 2    57         26    60.2   0.513  -3.15 0.00351   8.68 0.000234     -0.364
## 3    45         16    51.8   1.04   -6.77 0.0143    8.67 0.00450      -0.788
## 4    70         33    66.0   0.970   3.99 0.0125    8.67 0.00136       0.463
## 5    55         25    59.3   0.500  -4.31 0.00334   8.67 0.000416     -0.499
## 6    50         20    55.1   0.706  -5.12 0.00664   8.67 0.00118      -0.593
```

**Explanation of the output**: `augment` has also made a prediction for each row, but returned it with the original data (`grade` and `work_hours`) that were used to fit the model. Alongside the `.fitted` value, and the `.resid` (residual) there are some other columns we can ignore for now.

---

Often though, we don't want a prediction for each row in the original dataset. Rather, we want predictions for specific values of the predictors.

To do this, we can use the `newdata` argument to `predict` or `augment`.

First, we create a new single-row dataframe which contains the new predictor values we want a prediction for:

```
newsamples <- tibble(work_hours=30)
newsamples
```

```
## # A tibble: 1 x 1
##   work_hours
##        <dbl>
## 1         30
```

(note, the `tibble` command just makes a new dataframe for us)

Then we can use this with augment:

```
augment(first.model, newdata=newsamples)
```

```
## # A tibble: 1 x 3
##   work_hours .fitted .se.fit
##        <dbl>   <dbl>   <dbl>
## 1         30    63.5   0.725
```

**Explanation of the output**: We have a new data frame with predictions for a new sample who worked 30 hours.

**Extension exercises**

If you have time, try to answer the following questions based on other datasets built into R.

Using the mtcars data:

- What do you predict the `mpg` would be for a car with 4 cylinders?
- What is the difference in `mpg` between a car with 4 and 5 cylinders?
- Is your prediction for a car with 4 cylinders the same as the *mean* mpg for cars with 4 cylinders? Can you explain why/why not?
- Run a model using `wt` to predict `mpg`
- Using `augment` with the `newdata` argument, make a ggplot (using `geom_smooth`) showing the prediction for all weight values between 1 and 6.

Using the iris dataset:

- What is your prediction for `Sepal.Length` for specimens which are 2 or 4mm wide?
- What is your prediction for a specimen which was 8mm wide? How confident are you about this prediction?

Using the CPS data saved here http://www.willslab.org.uk/cps2.csv, what you

- Is hours a good predictor of income in this dataset?
- What is your predicted income for someone who works 40 hours per week?

Show answers

The numeric answers for each question are shown below:

```
library(broom)
lm(mpg~cyl, data=mtcars)  %>%
  augment(newdata=tibble(cyl=c(3,4,5,6)))
```

```
## # A tibble: 4 x 3
##     cyl .fitted .se.fit
##   <dbl>   <dbl>   <dbl>
## 1     3    29.3   1.17
## 2     4    26.4   0.905
## 3     5    23.5   0.684
## 4     6    20.6   0.570
```

```
lm(Sepal.Length~Sepal.Width, data=iris) %>%
  augment(newdata=tibble(Sepal.Width=c(2,4)))
```

```
## # A tibble: 2 x 3
##   Sepal.Width .fitted .se.fit
##         <dbl>   <dbl>   <dbl>
## 1           2    6.08   0.177
## 2           4    5.63   0.161
```

```
cps<- read_csv('http://www.willslab.org.uk/cps2.csv')
```

```
## Parsed with column specification:
## cols(
##   ID = col_double(),
##   sex = col_character(),
##   native = col_character(),
##   blind = col_character(),
##   hours = col_double(),
```

```
##   job = col_character(),
##   income = col_double(),
##   education = col_character()
## )
```

```
lm(income~hours, data=cps) %>%
  augment(newdata=tibble(hours=40))
```

```
## # A tibble: 1 x 3
##   hours .fitted .se.fit
##   <dbl>   <dbl>   <dbl>
## 1    40 103410.   2063.
```

#### Summary of today's session

So far we have:

- Seen that fitting straight lines to data can be a useful technique to:
  - **Describe** existing data we have and
  - **Predict** new data we collect
- We used `lm` to add a fitted line to a ggplot. We used this to make 'eyeball' predictions for new data.
- Then we used `lm` directly, using one variable (i.e. a column in a dataframe) to predict another variable.
- We saw how the coefficients from `lm` can be interpreted in relation to the fitted-line plot.
- We used the coefficients to make numeric predictions for new data.

If you're not clear on any of these points it would be worth going over today's materials again, and attend the catchup session before the next workshop.

In the next session we extend our use of `lm`, building more complex models, and using new R functions to make and visualise predictions from these models.

## Thinking about causes

#### In brief

> Scientists develop **theoretical models** which aim to describe the true relationships between variables. **Statistical models** are used to link their theories with data. They allow us to make **predictions** about future events, and our confidence in these predictions. But accurate predictions alone aren't enough: We also want to ***understand*** our data and the process that generated it.
>
> **Causal diagrams** are a useful tool for thinking about causes and effects. For psychological phenomena the diagrams can become complicated. But good research simplifies: We focus on small parts of a large network of causes and effects to make incremental progress.
>
> The quality of evidence for different parts of a causal diagram can vary a lot. We are much more sure about some links in the network than others. Causal diagrams can represent both our **knowledge** and our **hypotheses** explicitly. This is useful as we build statistical models to check how well our theories perform in the real world.

#### Drawing causal models

- Additional ppt slides drawn from the materials presented on this website are here

In this section we step back from learning specific techniques in R and think about *why* we want to run statistical models at all.

Our first job, as quantitative researchers, is to try and describe the network of *causes and effects* between the phenomena we're interested in. At this point we can also notice any ambiguities or uncertainties in our thinking.

To represent our model we can use a special type of diagram, called a *directed acyclic graph*. That's a fancy name for a boxes-and-arrows diagram, with a few special rules (we can come to those later).

For the moment, make sure you understand the following diagrams.

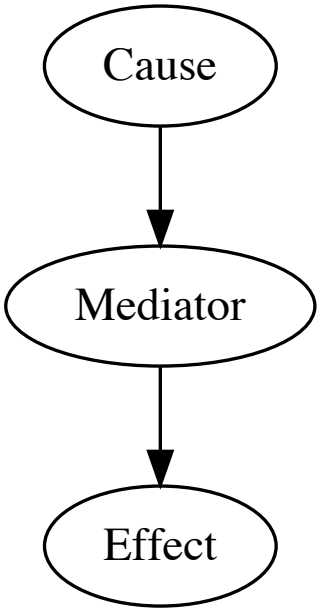
**Simple cause and effect**

**No causation** This diagram says footwear and exam grades AREN'T related AT ALL, because we didn't draw a line between them.

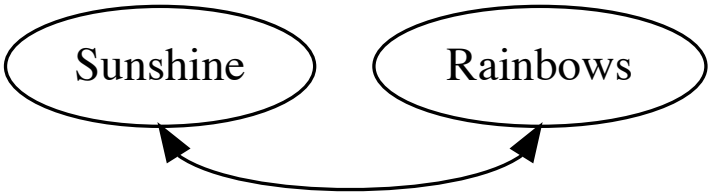

**Causal sequences** And can also describe how variables are related in a particular *causal sequence*. For example, we might ask:

- does childhood poverty reduce academic achievement by delaying brain development?
- does weaker childhood attachment reduce academic performance by reducing the motivation to study?

This pattern – where variables are linked in a series – is called **'mediation'**:




**Correlation (as distinct from causation)** Finally, if you don't know which **direction** the arrow should point — that is, you don't know which is the cause and which is the effect — we can (temporarily) draw an arrowhead at both ends like this:



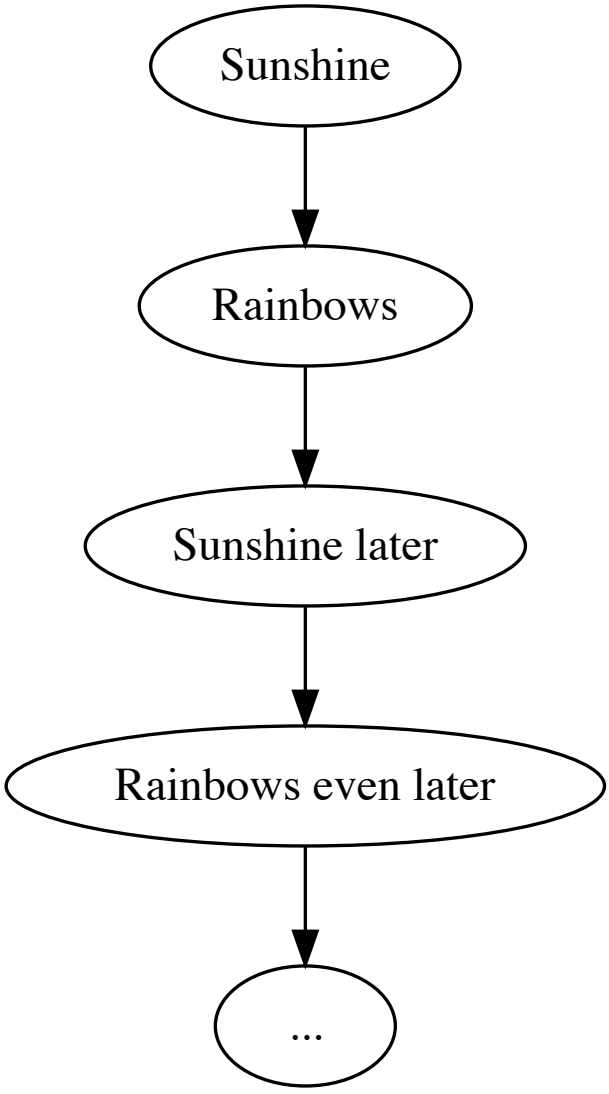

This represents a correlation (see the stage 1 notes on correlation and relationships).

Our hope is that — as we learn more, by collecting data or running experiments — we can decide *which way* the arrow should point.

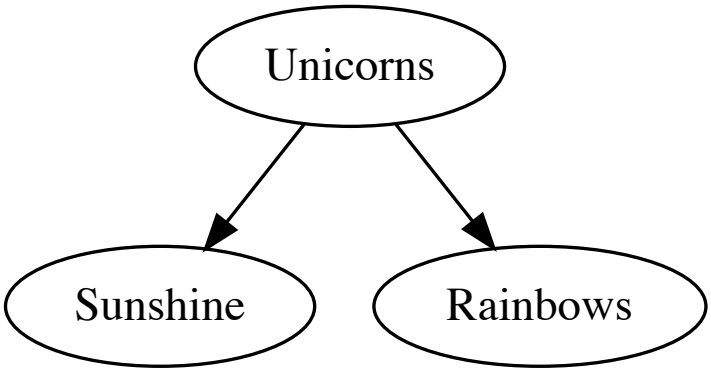
---

**An aside: Why does the arrow *have* to point in a particular direction?**

When we draw a double headed arrow we are essentially expressing our ignorance about about the relationship. What we probably mean is one of two things:




Or, perhaps:




That is, a correlation either implies:

- An unmapped sequence of reciprocal causes
- A common cause that is not explicitly in the model yet

---

In groups:

1. Pick one of the topics listed below. Try to think of at least 4 or 5 behaviours or psychological constructs to include. Sketch this out with pen and paper.
    - Effectiveness of psychotherapy
    - Coping with chronic ill health
    - Student satisfaction
2. Discuss how strong you think each of the relationships (lines) are. What kinds of evidence do you have (or know of) that make you think the diagram is correct?
3. If there are some boxes which don't have links between them, discuss if you think there is really **no** relationship between these constructs.
4. Can you find examples of *mediation* in your diagram? If you can't can you look more closely at one of the links and think about whether it is a *direct* effect, or if something else could link these constructs.

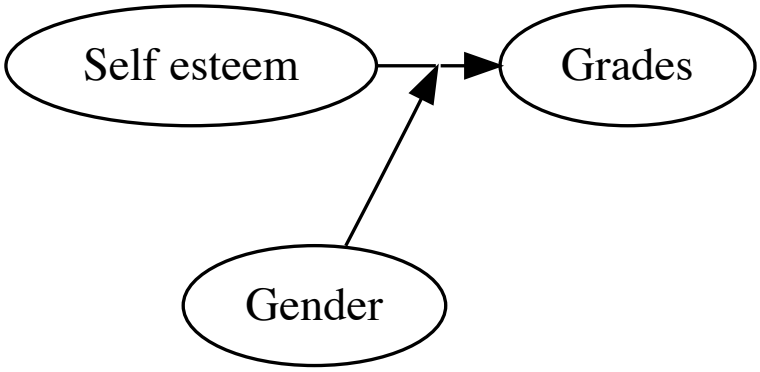
## 'Effect modification'

Another common question for researchers is whether relationships between variables are *true all the time*, or if they *vary depending on the context, individual or some other factor.*

In concrete terms, we might ask questions like:

- does low self esteem hurt academic performance more for women than for men?
- are older therapists more effective than younger therapists?
- does social media use cause less anxiety for people with high emotional intelligence (EQ)?

Relations like this can be represented in different ways in the diagram. For the moment, you should draw it like this:




The arrow from gender points at the relationship between self esteem and grades. We mean that the *effect* of self esteem on grades depends on whether you are a woman.

This pattern is called ***moderation*** or ***effect modification***. Checking to see if a relationship is the same for different groups is also called ***stratification***.

In groups: Consider your diagram from the previous task:

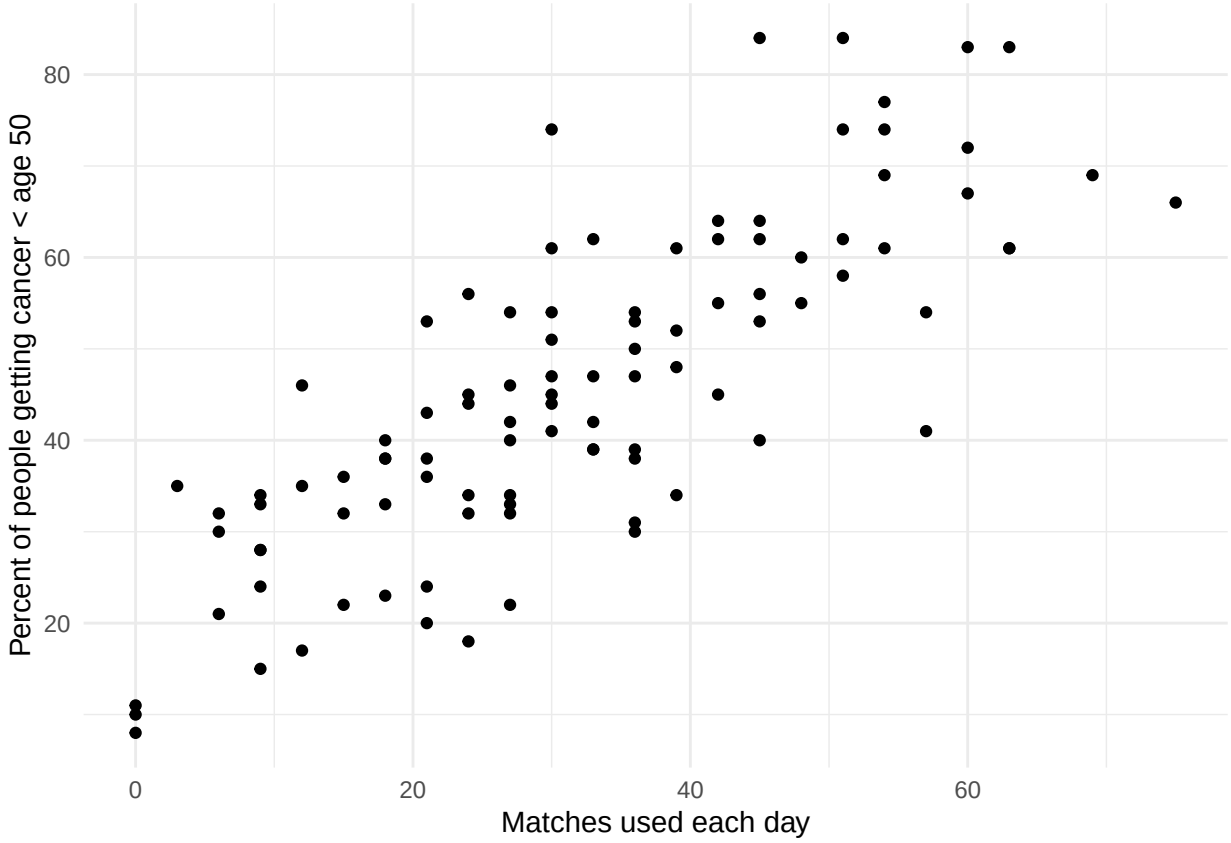
- If gender is not already included, add a new box for it.
- Discuss in groups: could gender *moderate* one of the other relationships in the diagram? If so, draw this in now.
- Are there other examples in your model where one variable could affect the relationships between two others?

### 'Tricky' relationships


---

**Using diagrams to think about models** (In the previous section we said that to represent a correlation in a causal diagram we use a double headed arrow like this: $\longleftrightarrow$)

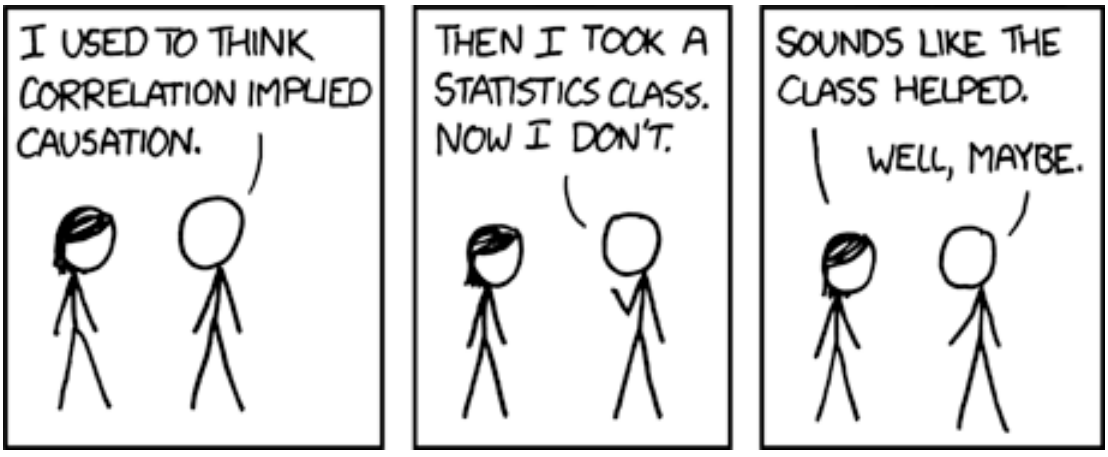
Consider this scatter plot:




1. Draw the best causal diagram you can based ONLY on the data in this plot. Your diagram should have 2 boxes, and either zero or one arrow.
2. Do you think the model is a good description of how the world works?
3. Redraw the diagram to make it **more plausible**, adding at least one extra variable (box) to your diagram, and converting any double-headed arrows ($\leftrightarrow$) to single-headed arrows.
4. Discuss in your group what you think is happening here. Have you come across this idea before?

Read an explanation

**Correlations, causation and experiments**



How can we be sure we haven't missed anything and stop worrying about confounding? There are three main ways:

1. Use experiments to make confounding *impossible*.
2. Design our studies carefully, and use multiple sources of evidence, to convince ourselves that confounding is *improbable* in this case (see notes on why smoking is a good example of this).
3. Account for *all* the possible confounders (this is virtually impossible, but sometimes trying this is the best we can hope for).

#### Accounting for confounders

In your groups, consider the original causal diagram you drew:

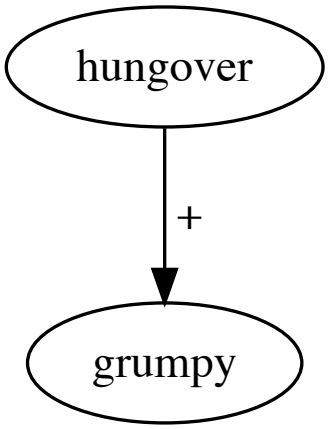
1. Look at each of the boxes which has an arrow pointing away from it. Could you run an experiment which *randomises* people to have higher or lower scores on these variables? If not, why not?
2. Is it possible that you missed any variables when you drew your diagram? Could confounding be taking place? If so, update your diagram to make this possibility explicit.

Read more explanation/discussion on this point

## Diagrams and models

Researchers choose statistical models (e.g. t-tests, anova, regression) to test implications from causal models. These causal models may be implicit in their work (i.e. not drawn out as diagrams) but are there nonetheless.

For example, if we use a t-test we are implying a model like this:




In this case we would choose a t-test where 'hungover' is recorded as a binary variable. If 'hungover' was recorded as categorical (e.g. not at all/a little/very) we might use Anova instead (which is a special type of regression), or if 'hungover' was recorded as continuous score (e.g. from 1 to 100) we might use regression.

Different types of model are needed to test **moderation** (interactions in Anova or regression, as we will see in future sessions), and **causal sequences** (mediation analysis, or path analysis; more on this here).

> These are really 'implementation details' though; the important part is the causal model itself. We should choose statistical models which are the most appropriate for our model, and for the data we sampled.

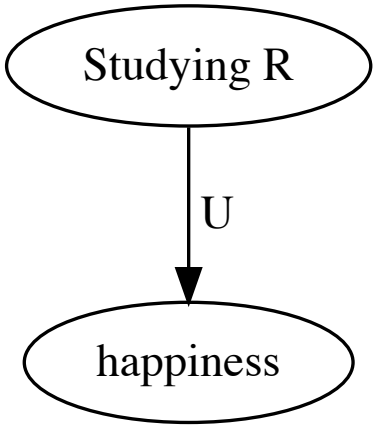
#### Non-linear relationships

One final point: In some cases you will want to indicate that a line on the graph implies a non-linear (e.g. curved relationship).

There's no commonly agreed way to represent this in causal diagrams, but I like to mark the arrow with either a +, a −, a ∪ or a ∩.

- + is a positive linear relationship
- − is a negative linear relationship
- ∩ is a relationship that starts positive but reverses as the level of the predictor increases
- ∪ is a relationship that starts negative but reverses as the level of the predictor increases

For example:




**Application to real examples**

These papers present data and make inferences on the links between diet or alcohol consumption and risk of death:

- Doll et al. (1994)
- Seidelmann et al. (2018)

Choose one of the papers above and:

- Draw out a causal diagram of all the variables mentioned in the paper
- Be sure to add possible confounders or unobserved variables — even if they are not measured or considered by the authors.
- How has drawing out the diagram affected your understanding of the results?

```
grVizPng('
digraph mary {
  {node [shape=box];}
  wt -> mpg [label=" U "]
  }
')
```

```
## [1] "media/gv/f3b8720ece679f66aa23b624f5674711.pdf"
```

–>

# Multiple regression

**In brief**

> Multiple regression is a technique which can describe the relationship between ***one outcome*** and ***two or more predictors***.

We can also use multiple regression to describe cases where two variable ***interact***. That is, when the effect of one predictor is increased or decreased by another (this is called moderation, as we saw in the session on causal models).

Multiple regression is important because it allows us to make more realistic models and better predictions. But, like any sharp tool, it should be used carefully. If our statistical model doesn't match the underlying network of causes and effects, or if we have used a biased sample, we can be misled.

When evaluating our models we can ask two useful questions: First, *how much of the variability in the outcome do my predictors explain?* The $R^2$ statistic answers this. Secondly: *does the model make better predictions than just taking the* ***average*** *outcome* (or using a simpler model with fewer predictors)? For this we can compute a BayesFactor.

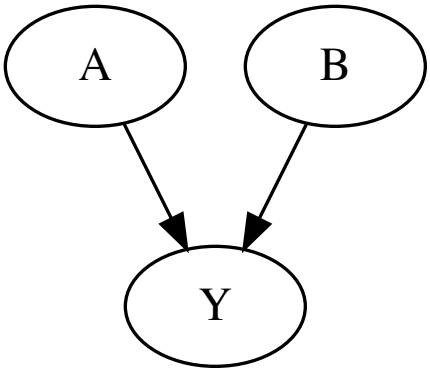
## Why use multiple regression?

- Slides here

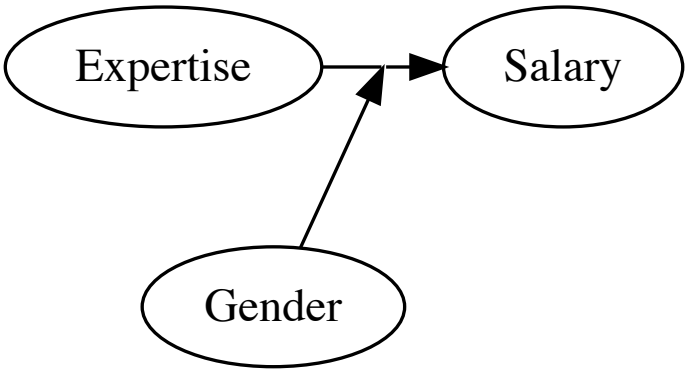
Think back to our last session on causes and effects. When we drew causal diagrams of our research question we found cases where there were:

- Multiple causes of a single outcome, and where
- One variable might alter the effect of another

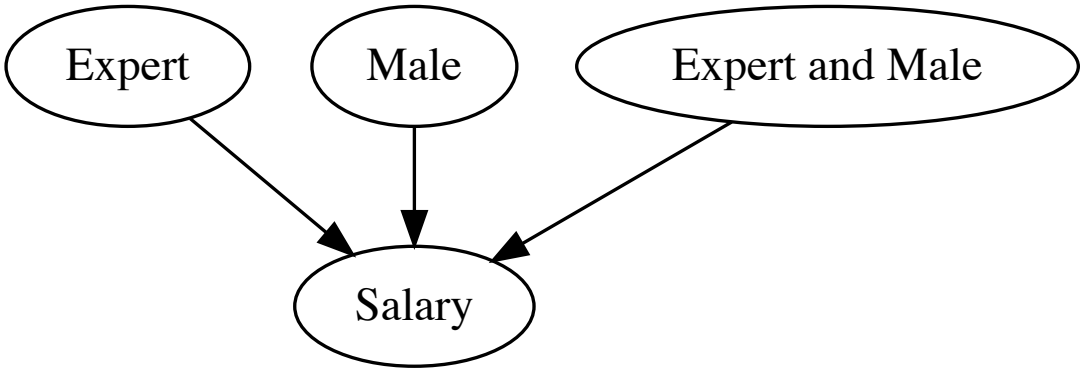
We drew diagrams like this for those cases:



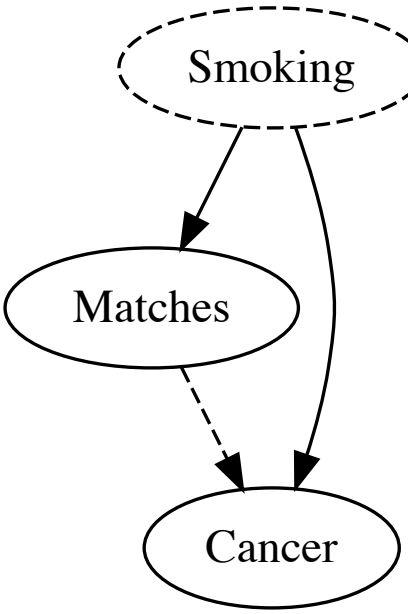

And




Another way to think about the diagram we say that effect modification is taking place is to draw it like this:




We also came across the idea of confounding. This is where we see a pattern like this:




As we discussed, the problem is that if smoking causes us to use matches *and* it causes cancer then if we look at correlations of match-use and cancer we might get mislead. This would be an example of a spurious correlation.

**Some benefits of using multiple regression**

There are a number of benefits to using multiple regression:

1. If we think that the relationship between two variables might be changed by another (for example, if a relationship between expertise and earnings were different for men and women), we can **test** if that is the case. I.e. we can test if moderation is occuring.
2. If we include extra variables (e.g. smoking as well as matches-used) we can reduce the effect of confounding, and make better inferences about cause-effect relationships (although this isn't guaranteed and we need to be careful).
3. From a practical perspective, including extra variables can also reduce noise in our predictions and increase statistical power.
4. Multiple regression can also be used to fit curved lines to data and avoid the assumption that all relationships can be described by a straight line (Chris will cover this later in the course).

---

**For now we are going to focus on the first example.** If you are interested in the second case, you can read more here.

A warning! You will sometimes see people claim that multiple regression provides a way of choosing between different possible predictors of an outcome. This is basically untrue; see here for why.

## Different relationships?

If you have a hypothesis that a relationship might differ for two different groups, **the first thing you should do is plot the data**.

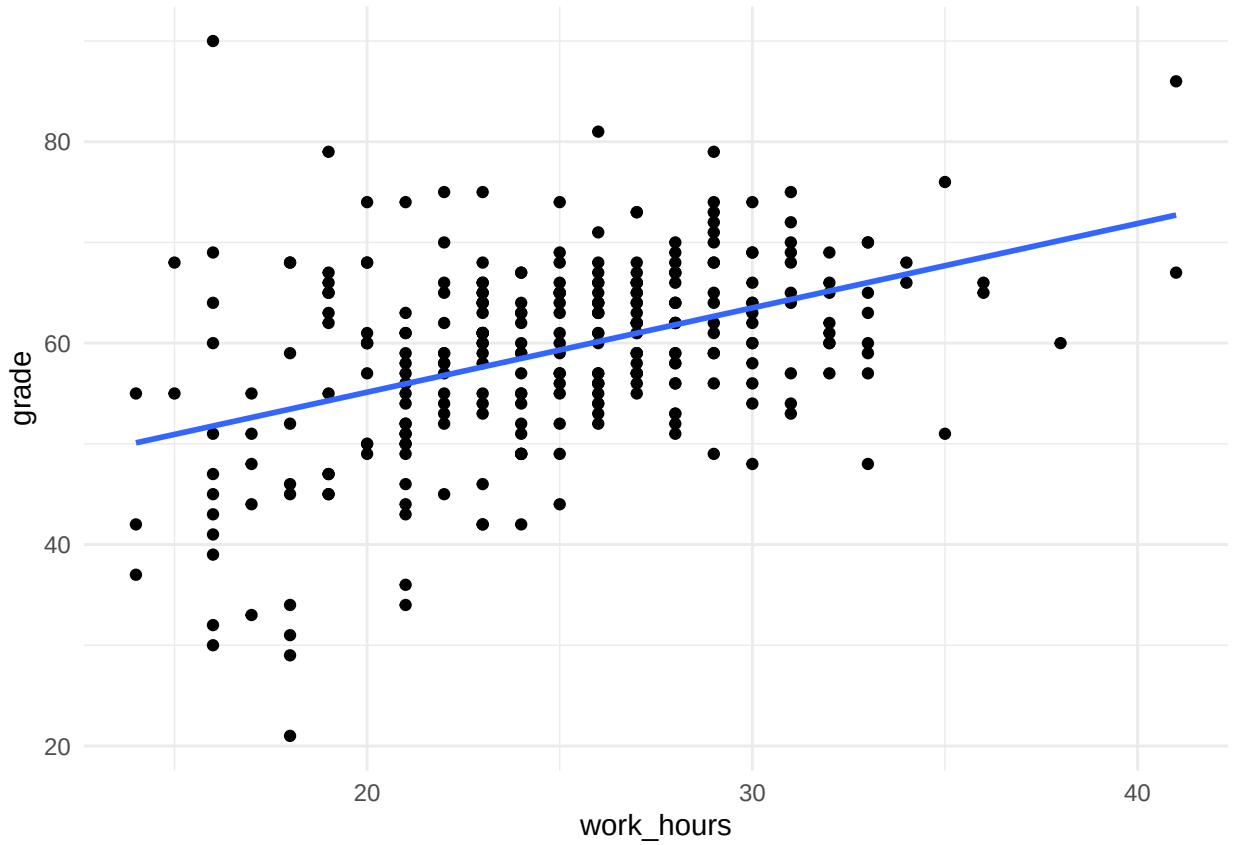
First let's reload the example dataset on student grades and study habits:

```
library(tidyverse)
studyhabits <- read_csv('https://benwhalley.github.io/rmip/data/studyhabitsandgrades.csv')
```

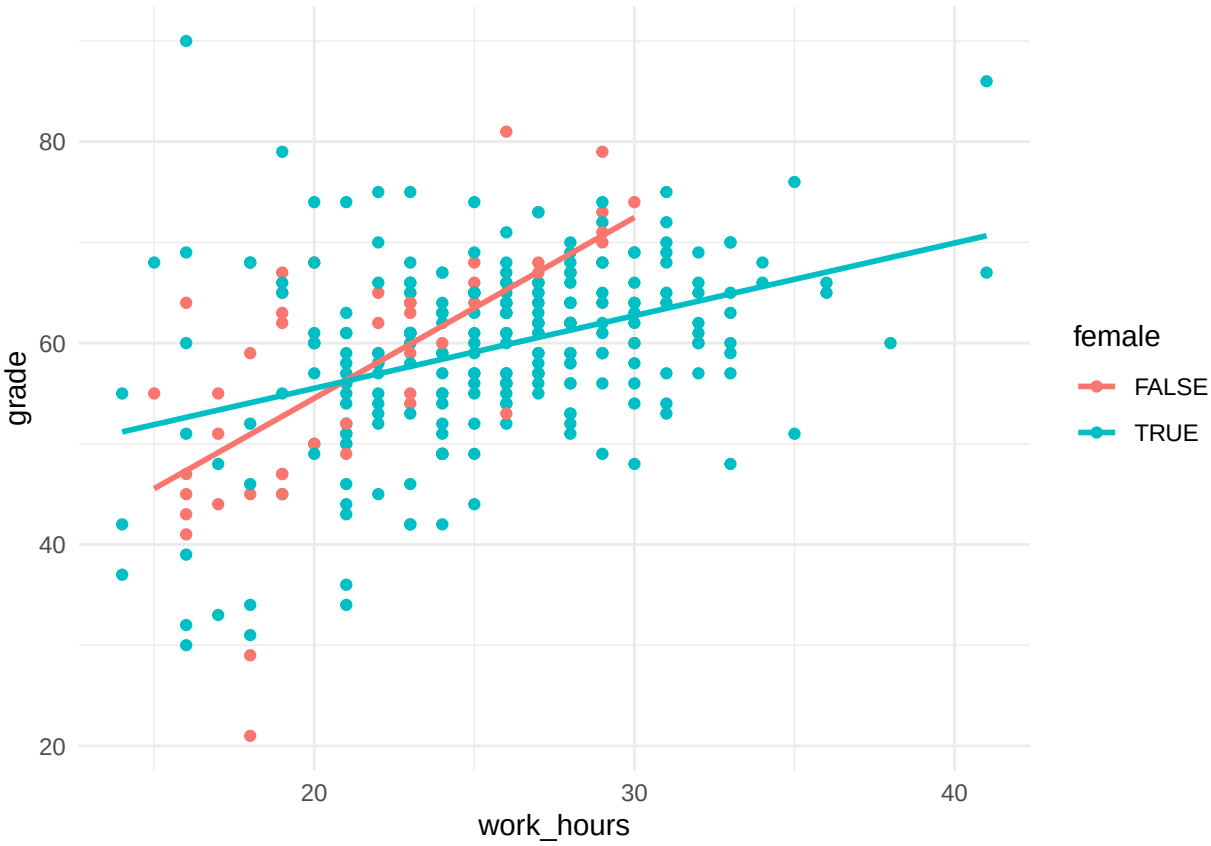
```
## Parsed with column specification:
## cols(
##   work_consistently = col_double(),
##   revision_before = col_double(),
##   focus_deadline = col_double(),
##   progress_everyday = col_double(),
##   work_hours = col_double(),
##   msc_student = col_logical(),
##   female = col_logical(),
##   unique_id = col_double(),
##   grade = col_double()
## )
```

We know there is a link (in these data) between study hours and grades because we can see it in the plot, and we modelled it using `lm` in a previous session:

```
studyhabits %>%
  ggplot(aes(work_hours, grade)) +
  geom_point() +
  geom_smooth(se=F, method=lm)
```




We could also ask, is this relationship the same for men and women? To show the differences, we can use a coloured plot:




**What is the main pattern in the data?**

First load the data and reproduce the coloured plot from above.

Second, agree within your groups:

- What is the overall pattern of these (imagined) results?
- Does extra time spent working benefit men and women equally?

## Using `lm` for multiple regression

If you don't already have it loaded in RStudio, load the example dataset:

```
studyhabits <- read_csv('https://benwhalley.github.io/rmip/data/studyhabitsandgrades.csv')
```

As we did for a single predictor regression, we can use `lm` to get numbers to describe the slopes of the lines.

```
second.model <- lm(grade ~ work_hours * female, data = studyhabits)
second.model
```

```
##
## Call:
## lm(formula = grade ~ work_hours * female, data = studyhabits)
##
## Coefficients:
##           (Intercept)             work_hours             femaleTRUE
##                18.623                  1.795                 22.463
## work_hours:femaleTRUE
##                -1.074
```

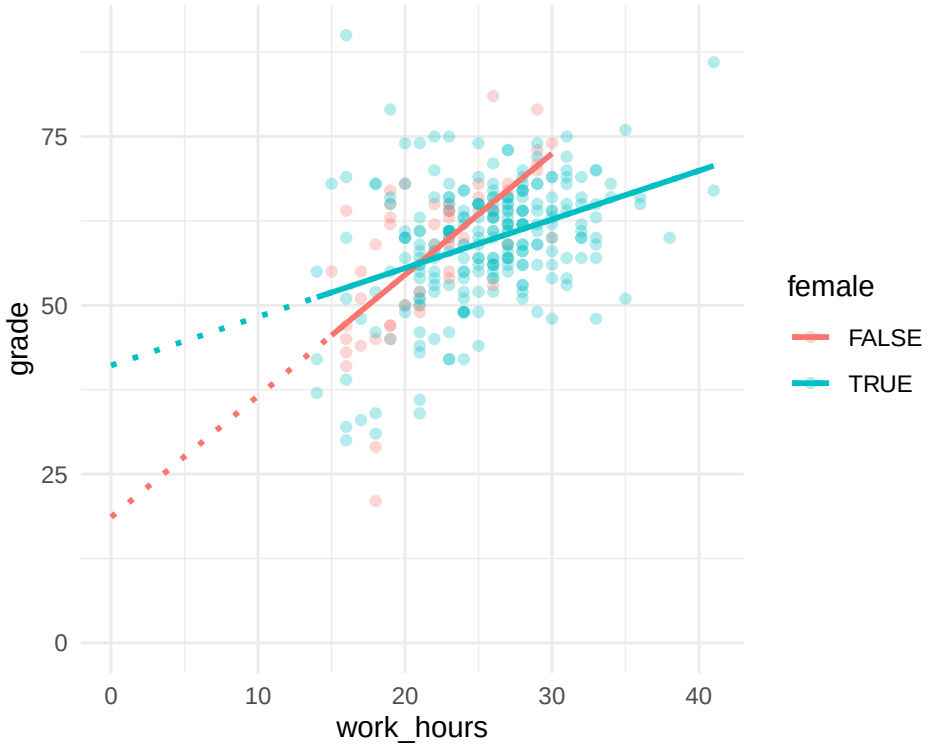
**Explanation of the `lm` code above** This time we have changed the formula and:

- Added `female` as a second predictor
- Used the `*` symbol between them, which allows the slope for `work_hours` to be *different* for men and women.

**Explanation of the `lm` output** The output looks similar, but this time, we have 4 coefficients:

- `(Intercept)`
- `work_hours`
- `femaleTRUE`
- `work_hours:femaleTRUE`

**Interpreting the `lm` coefficients** The coefficients have changed their meaning from model 1. But we can still think of them as either **points** or **slopes** on the graph with fitted lines. Again, I have extended the lines to the left to make things easier:




- `(Intercept)` is the point for men, where `work_hours` = 0 (where the red line crosses zero on the x axis).
- `femaleTRUE` is the difference between men and women, when `work_hours` = 0 (the difference between the blue and red lines, at the zero point on the x axis)
- `work_hours` is the slope (relationship) between `work_hours` and `grade` *for men* (the steepness of the red line)
- `work_hours:femaleTRUE` is the *difference in slopes* for work hours, for women. So this is the slope for women *minus* the slope for men (that is, the difference in steepness between the red and blue lines. It's NOT the slope of the blue line).

Double check you understand how to interpret the `work_hours:femaleTRUE` coefficient. It's very common for regression coefficients to represent **differences** in this way. But in this example it does mean we have to know both `work_hours` (the slope for men) and `work_hours:femaleTRUE` (the difference in slopes for men and women) to be able to work out the slope for women.

To test your knowledge:

- What is the slope for women in `second.model` above?

Show answer

To get the answer we need to add the slope for `work_hours` to the coefficient `work_hours:femaleTRUE`.

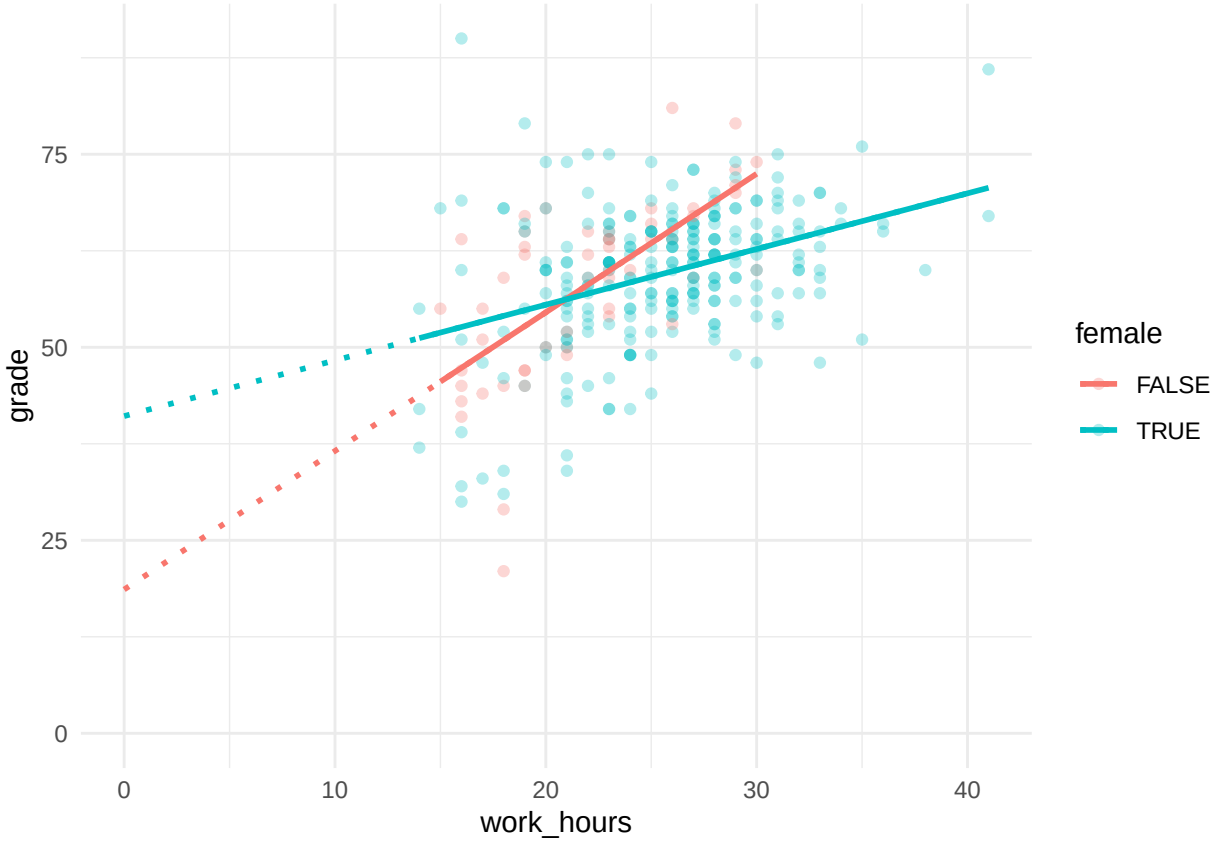
- `work_hours` represents the slope for men
- `work_hours:femaleTRUE` represents the difference in slopes between men and women

So the slope for women = $1.7948984 + -1.0735653 = 0.7213331$ (you can round this to 0.7).

### Linking coefficients with plots

Compare the model output below with the plot:

```
##
## Call:
## lm(formula = grade ~ work_hours * female, data = studyhabits)
##
## Coefficients:
##           (Intercept)             work_hours             femaleTRUE
##                18.623                  1.795                 22.463
## work_hours:femaleTRUE
##                -1.074
```




As a group:

1. For each of the 4 coefficients, agree if it represents a point or a slope
2. Find each of the points on the plot (i.e. which coefficient is it)

3. Compare the slope coefficients to the lines on the plot - can you explain which coefficient describes which slope?
4. What would happen if the sign of each coefficient was reversed? E.g. if one of the coefficients was now a negative number rather than positive? What would this mean for the plot?

### Making predictions

As before, we can use `augment` from the `broom` package to make predictions.

The steps are the same:

0. Fit the model we want
1. Load the `broom` package
2. Create a new dataframe with a small number of rows, including only the values of the predictor variables we want predictions for
3. Use `augment` with the model and new dataframe

Optionally, we can then plot the results.

---

We have already fit the model we want to use, which was:

```
second.model$call
```

```
## lm(formula = grade ~ work_hours * female, data = studyhabits)
```

Next we should load the broom package:

```
library(broom)
```

And make a dataframe (a tibble is a kind of dataframe) with values of the predictor variables that would be of interest, or would provide good exemplars.

For example, lets say we want predictions for men and women, who work either 20 or 40 hours each. We can write this out by hand:

```
newdatatopredict = tibble(
  female=c(TRUE,TRUE, FALSE,FALSE),
  work_hours=c(20,40, 20,40)
)

newdatatopredict
```

```
## # A tibble: 4 x 2
##   female work_hours
##   <lgl>       <dbl>
## 1 TRUE           20
## 2 TRUE           40
## 3 FALSE          20
## 4 FALSE          40
```

The last step is to pass the model and the new dataframe to `augment`:

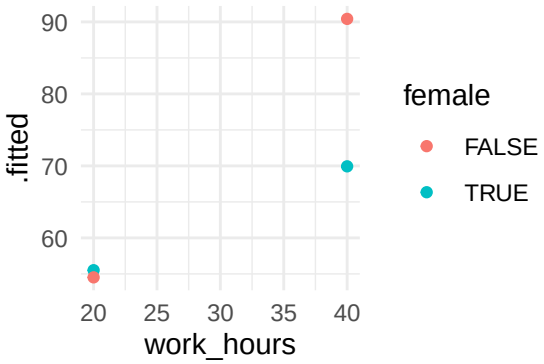
```
second.model.predictions <- augment(second.model, newdata=newdatatopredict)
second.model.predictions
```

```
## # A tibble: 4 x 4
##   female work_hours .fitted .se.fit
##   <lgl>       <dbl>   <dbl>   <dbl>
## 1 TRUE           20    55.5   0.821
```

```
## 2 TRUE           40    69.9   1.72
## 3 FALSE          20    54.5   1.29
## 4 FALSE          40    90.4   5.19
```

And we can plot these new predictions using ggplot:

```
second.model.predictions %>%
  ggplot(aes(work_hours, .fitted, color=female)) +
  geom_point()
```



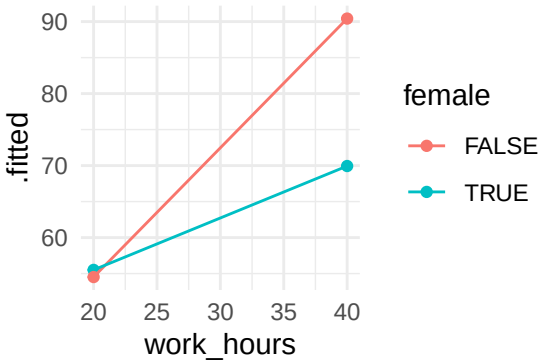

---

This basic plot is OK, but we can improve it by:

- Adding lines to emphasise the difference in slopes for men and women.
- Adding error bars.
- Tidying the axis labels.

To add lines to the plot we can use `geom_line()`. We have to add an additional argument called `group` to the `aes()` section of the plot. This tells `ggplot` which points should be connected by the lines:

```
second.model.predictions %>%
  ggplot(aes(work_hours, .fitted, color=female, group=female)) +
  geom_point() +
  geom_line()
```




Next we can add error bars. If we look at the datafram that `augment` produced, there is a column called `.se.fit`. This is short for **standard error of the predicted value**:

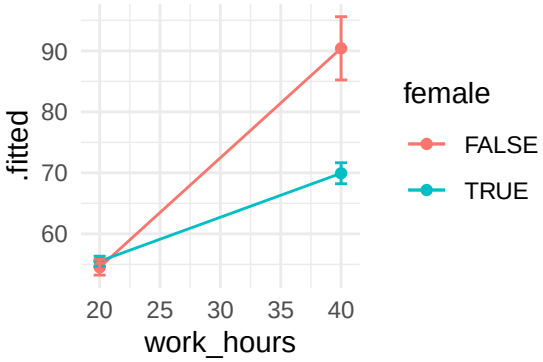
```
second.model.predictions
```

```
## # A tibble: 4 x 4
##   female work_hours .fitted .se.fit
##   <lgl>       <dbl>   <dbl>   <dbl>
## 1 TRUE           20    55.5   0.821
```

```
## 2 TRUE           40    69.9   1.72
## 3 FALSE          20    54.5   1.29
## 4 FALSE          40    90.4   5.19
```

We can use a new `geom_` function with this column to add error bars to the plot. The `geom_errorbar` needs two additional bits of information inside the `aes()` section. These are `ymin` and `ymax`, which represent the bottom and top of the error bars, respectively:

```
second.model.predictions %>%
  ggplot(aes(
    x=work_hours,
    y=.fitted,
    ymin =.fitted - .se.fit,
    ymax =.fitted + .se.fit,
    color=female,
    group=female)) +
  geom_point() +
  geom_line() +
  geom_errorbar(width=1)
```




**Explanation of the code**: We added the `geom_errorbar` function to our existing plot. We also added two new arguments to the `aes()` section: `ymin` and `ymax`. We set the `ymin` value to the fitted value, ***minus*** the standard error of the fitted value (and the same for `ymax`, except we added on the SE).

**Explanation of the resulting plot**: The plot now includes error bars which represent the standard error of the fitted values. We will cover more on intervals, including standard errors, in a later workshop.

### Extension exercises

1. Tidy up the plot above by adding axis labels.
2. In the example above we created a dataframe by hand to tell `augment` what predictions we wanted. Now try using `expand.grid` to make the new dataframe instead (we first used `expand.grid` in the first session). For example, try making predictions for men and women who work 20, 25, 30, 35, or 40 hours per week.

Data from a clinical trial of Functional Imagery Training (Solbrig et al. 2019, FIT) are available at https://zenodo.org/record/1120364/files/blind_data.csv. In this file, `group` represents the treatment group (FIT=2, motivational interviewing=1). The `kg1` and `kg3` variables represent the patients' weights in kilograms before and after treatment, respectively. Load these data and complete the following tasks:

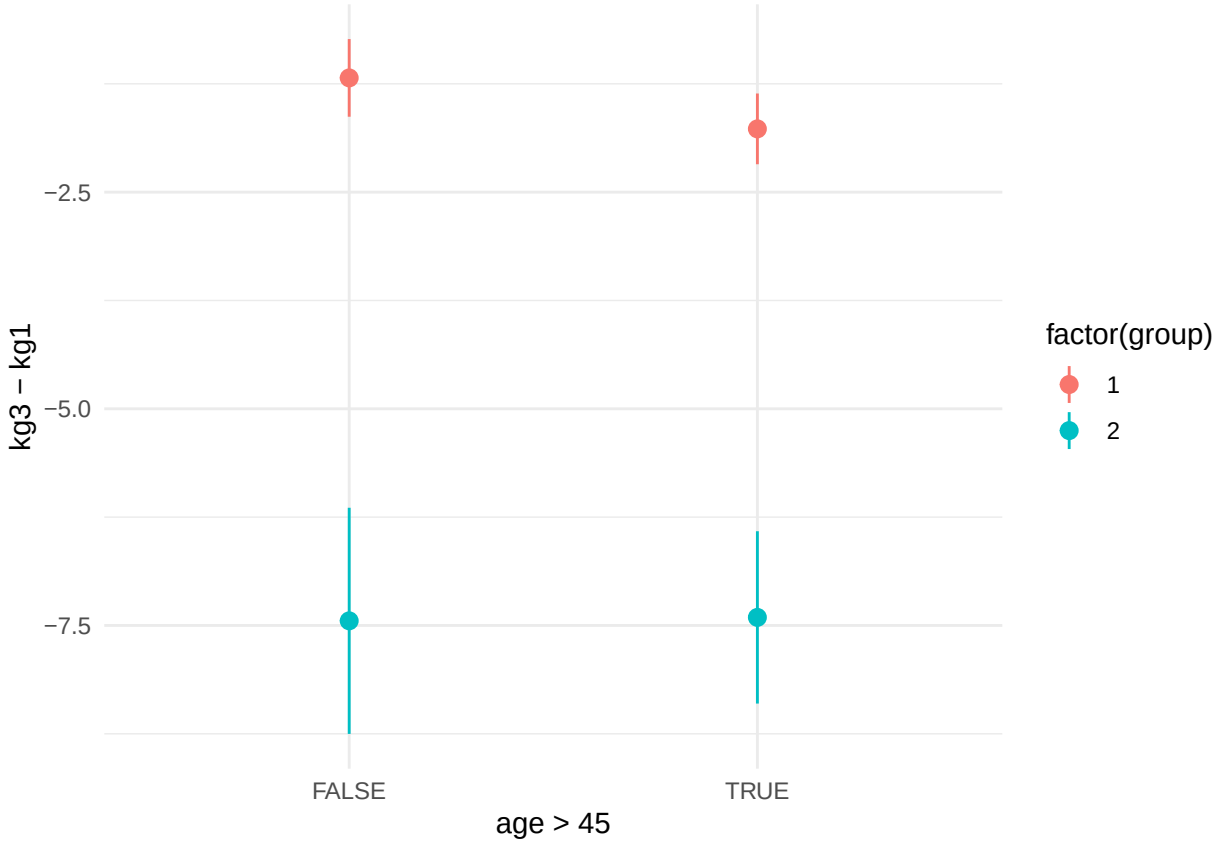
1. Plot the difference in weight between treatment groups at followup (`kg3`)
2. Create a plot to show whether men or women benefitted most from the treatment (this will require some thinking about what goes on the y-axis, and perhaps some pre-processing of the data).

3. Create a plot to show whether older participants benefitted more or less than younger ones (again this will require some thinking, and there are quite a number of different plot types which could be used, each with different pros and cons).

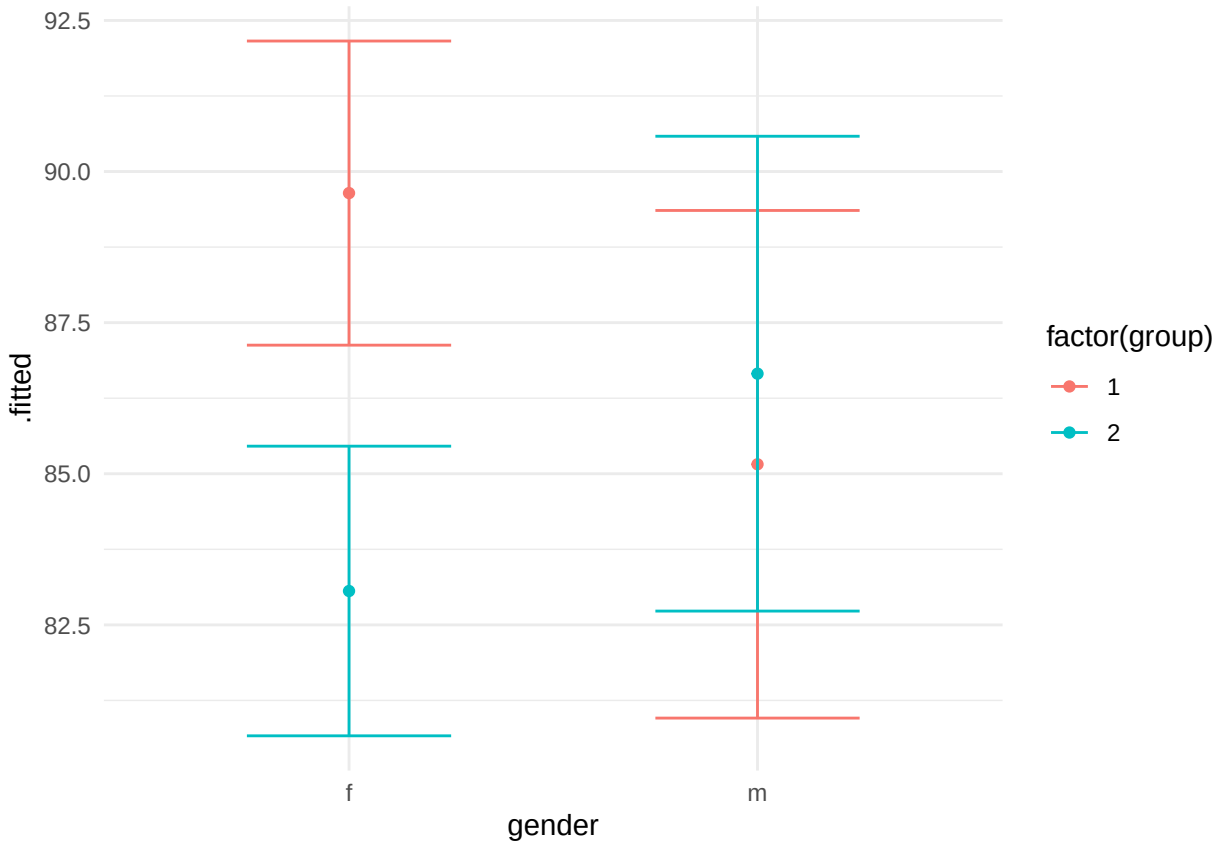
```
fit %>% ggplot(aes(age>45, kg3-kg1, color=factor(group))) + stat_summary()
```

```
## Warning: Removed 9 rows containing non-finite values (stat_summary).
```

```
## No summary function supplied, defaulting to `mean_se()
```

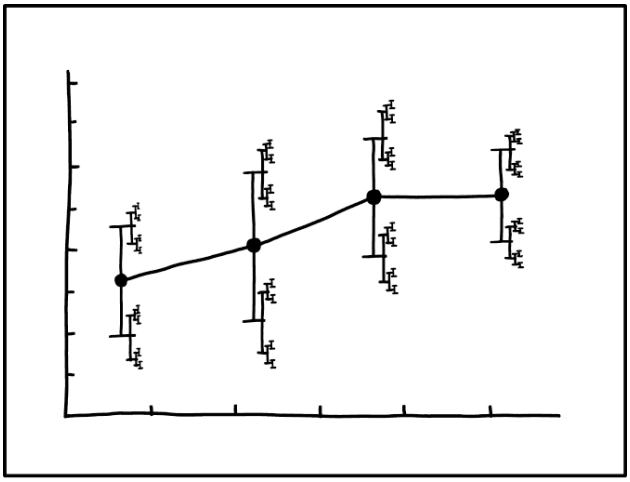



```
m1 <- lm(kg3 ~ group*gender, data=fit)
augment(m1, newdata=expand.grid(group=c(1,2), gender=c("f", "m"))) %>%
  ggplot(aes(gender, .fitted, ymin=.fitted-.se.fit, ymax=.fitted+.se.fit, colour=factor(group))) +
    geom_point() +
    geom_errorbar(width=.5)
```




->

## Uncertainty




Statistical require us to be explicit about our assumptions and help us make numerical predictions for new data. They also let us **quantify our uncertainty** about these predictions. Quantifying uncertainty is important for both practical and theoretical purposes. Practically, we might want

to put 'bounds' on our predictions, allowing policy-makers or individuals to make better decisions. Theoretically, we can use estimates of uncertainty to attach probabilities to our experimental hypotheses and this forms part of the process of evaluating existing and new theory.

## Overview

- Slides from today

So far we've seen that regression lines imply **predictions**, and made some predictions of our own. Statistical models like regression are used to:

- Making *explicit* our assumptions
- Making predictions
- Quantifying *uncertainty* about predictions

### Intervals

If we make an verbal estimate then it's common to add quantifying adjectives to indicate our uncertainty. We might say: "he's *about* 6 feet tall" or "assistant psychologists earn *roughly* £21,000 per year".

When using regression we can be specific about what we mean by 'about' or 'roughly': We use *intervals* to quantify this uncertainty. The interval represents our uncertainty in terms of *probabilities*. That is, the probability of 'some event' happening or not happening.

There are a number of different types of interval though and each refers to *the probability of a different event happening*. The main types of interval are:

- Confidence intervals (which are related to $p$ values)
- Prediction intervals
- Credible intervals

It's important to remember what events these probabilities refer to.

### Confidence intervals

You might have learnt about $p$ values and confidence intervals during your undergraduate study. However it's likely you're still not clear on how to interpret them. You are in good company: despite extensive training in research methods many researchers are also confused, and frequently misinterpret p values and confidence intervals in peer reviewed journals.

The confusion arises because:

- confidence intervals and $p$ values **don't** tell us anything particularly useful but
- *they sound like they do!*

With very few exceptions the only reasons they are still used are convention, inertia, and ignorance of the alternatives (which can sometimes be more fiddly to calculate).

**Confidence intervals: The probability of *what*?** As you might guess, confidence intervals (and $p$ values) relate to probabilities.

But they *aren't* related to the probability of your experimental hypothesis. Instead, they relate to ***properties of the statistical test procedure itself***. If that sounds a bit obscure and not very helpful, you'd be right.

*Your* hypothesis might be that there is a "experimental participants will have longer reaction times", or that "there is a positive relationship between mother and child IQ scores".

But confidence intervals *do not* relate to the probability these relationships are true. Instead, they relate to the probability the test procedure will mislead you if you used it over many repeats of the experiments. And that's not the same thing at all.

Despite frequently misinterpreting them, psychologists often use confidence intervals to report uncertainty. Because of this it is worth learning about and understanding confidence intervals - even if there are normally better alternatives.

**Summary** $p$ values are ***not*** the probability your hypothesis is right or wrong. Confidence intervals are ***not*** the range within which we have 95% confidence the true answer will fall.

If you want to read more about this see this additional explanation of confidence intervals

---

### Prediction intervals

In many applications of psychology we want to make predictions *for an individual*. For example, when a doctor meets a patient with depression they want to be able to give them an idea of their prognosis if they undergo a new treatment.

Alternatively, a psychiatrist might want to decide whether switching antidepressant medication will be helpful for a patient or not (this is a prediction about alternative futures).

As we have seen before, we can use data to make predictions. Data from an RCT might let us predict that, on average, patients taking CBT will report that symptoms reduce by about 3 points on a standard scale.

But ***nobody is the average*** patient. The hypothetical doctor wants to give an idea of the range of outcomes that are possible or likely. This is the predictive interval.

Classical prediction intervals are one attempt to define the predictive interval. We can describe them like this:

> Imagine we replicate our study many times. Each time we run a regression and calculate the classical prediction interval. If we then collect a new observation, we would expect it to fall within the prediction interval a certain percent of the time.

Like confidence intervals, classical prediction intervals are tricky. Unfortunately—although it seems like it *should*—we *can't* say that 95% of all new observations will fall within any single prediction interval calculated using a single sample.

To calculate the range in which we think 95% of new observations would fall we must use the Bayesian posterior predictive interval described below.

### Bayesian intervals

Scientists (and humans) aim to collect and use data to update their beliefs to fit the facts. When we generate hypotheses, we'd often like to make statements about how ***probable*** they are. That is, we often want to know:

- $P(hypothesis|data)$ (probability of the hypothesis given the data)

This isn't possible with classical statistics, which instead tells us:

- $P(data|!hypothesis)$ (probability of the data, if the hypothesis were *not* true)

Giorgio covered the basics of Bayesian inference in PSYC752, but for a nice expansion of this, and an example involving bananas and exhaust pipes, see: the evidence section of the undergraduate teaching materials.

There are two types of Bayesian intervals that are analogous to – but easier to interpret than – classical confidence/prediction intervals.

- Credible intervals: The range we are 95% sure the true mean is within.
- Bayesian predictive intervals: The range we expect 95% of new observations to fall within.

In the following section you'll learn how to calculate all of these different kinds of interval.

## Calculating intervals

At the end of this session, you should be able to:

- Make predictions with confidence or prediction intervals from a simple linear model.
- Re-fit the same model and calculate a Bayesian credible interval.
- Correctly interpret confidence, prediction and credible intervals.

---

To give us an example to work with, let's load some data which records IQ test scores for 434 children along with their mothers' IQ scores, ages, and whether they completed high school.

```
kidiq <- read_csv("http://bit.ly/arm-kids-iq")
```

```
## Parsed with column specification:
## cols(
##   kid_score = col_double(),
##   mom_hs = col_character(),
##   mom_iq = col_double(),
##   mom_work = col_double(),
##   mom_age = col_double()
## )
```

```
kidiq %>% glimpse()
```

```
## Observations: 434
## Variables: 5
## $ kid_score <dbl> 65, 98, 85, 83, 115, 98, 69, 106, 102, 95, 91, 58, 84, 78...
## $ mom_hs    <chr> "Completed", "Completed", "Completed", "Completed", "Comp...
## $ mom_iq    <dbl> 121.118, 89.362, 115.443, 99.450, 92.746, 107.902, 138.89...
## $ mom_work  <dbl> 4, 4, 4, 3, 4, 1, 4, 3, 1, 1, 1, 4, 4, 4, 2, 1, 3, 3, 4, ...
## $ mom_age   <dbl> 27, 25, 27, 25, 27, 18, 20, 23, 24, 19, 23, 24, 27, 26, 2...
```

As we did in the regression section, we can create a model. Here we predict childrens' scores from mothers' IQ and schooling:

```
iqmodel <- lm(kid_score ~ mom_iq + mom_hs, data=kidiq)
iqmodel
```

```
##
## Call:
## lm(formula = kid_score ~ mom_iq + mom_hs, data = kidiq)
##
## Coefficients:
##            (Intercept)                  mom_iq  mom_hsDid not complete
##                31.6815                  0.5639                 -5.9501
```

Load the data and run the code for `iqmodel`. Then move on to the next section.

### Confidence intervals

To calculate intervals for a model prediction there are 2 steps:

1. Create a dataframe which contains values of the *predictor variables*, at the levels we want to make predictions for (like we did with augment).
2. Make the predictions, and request the type of interval we want.

This code makes a new dataframe containing values of `mom_iq` and `mom_hs` which we'd like a prediction for; in this case, just one mother who completed high school:

```
new.mother <- tibble(mom_iq=97, mom_hs="Completed")
```

We make predictions like we have before, but now adding the argument: `interval = "confidence"`:

```
predict(iqmodel, newdata = new.mother, interval = "confidence")
```

```
##        fit      lwr      upr
## 1 86.38052 84.35293 88.40812
```

The output is the prediction (`fit`) and the lower and upper bounds of the confidence interval. The 95% interval is used by default.

To be clear, ***the confidence interval is range which, if we calculated it 100 times from different samples, we think the true value would would be within 95% of the time.***

Use the `predict` function with `iqmodel` to answer the following questions:

- What is the predicted `kid_score` for mothers with IQ = 97 who completed high school?
- Make a new tibble, and then `predict` values for 3 different mothers, with different scores on `mom_iq` and `mom_hs`
- 95% of mothers with IQ = 97 who completed high school will have a child with an IQ < 88.4: True False
- Fewer than 2.5% of mothers with IQ = 97 who completed high school will have a child with an IQ > 88.4 True False

Explain the answers

86 is the prediction from the output shown above.

To create a new tibble and predict scores for different mothers, repeat the code above but changing the values for `mom_iq` or `mom_hs`.

Both the 2nd and 3rd answers are false because the confidence interval is the range within which we think the true average for mothers with IQ = 97 who completed high school would fall, if we repeated the study and calculated the CI many times.

We can't make probability statements about the quality of the predictions we actually made using classical statistics: We can only say how we think the method used to make them will perform, if we repeated it many times.

### Prediction intervals

Prediction intervals allow us to express uncertainty about future **individuals** we might sample, rather than the average value.

We can repeat the process above to make the classical *prediction* interval instead. We just change `interval = "confidence"` to `interval = "prediction"`

```
predict(iqmodel, newdata = new.mother, interval = "prediction")
```

```
##        fit      lwr      upr
## 1 86.38052 50.67755 122.0835
```

So this is the prediction (`fit`) and the lower and upper bounds of the prediction interval (again, the 95% interval is used by default).

To be clear, ***this is the range within which we would expect new children to fall if their mother had IQ-97 and had completed high school, if we repeated our sampling and testing many times.***

- Is the prediction itself (the `fit`) the same or different when we request the prediction interval (as compared with the confidence interval)? Same Higher Lower
- If we met a mother of IQ = 97 who had completed high school, and we predicted their child would have an IQ between 50.7 and 122.1, we would be right 95% of the time : True False

Explain the answers

The prediction is the same, because it's the point on the regression line for these values of `mom_iq` in both cases, and both models had near-identical parameter estimates.

The second answer is false because the prediction interval says that:

- if we repeated our sample many times *and*
- calculated the prediction interval each time *then*
- the true `kid_score` of the new mother we meet will fall within 95% of these intervals

Like with confidence intervals, can't make statements about the quality of the prediction we made: We can only say how we think the method used to make the predictions will perform, if repeated many times.

### Bayesian intervals

Bayesian analysis requires making explicit our assumptions about what is most likely to happen *before you analyse the data*. If you have taken statistics courses before, this might seem counterintuitive but it's actually something humans seem to do all the time in everyday life: For example, our perception of visual stimili is strongly influenced by how probable we think they are.

We call this our ***prior***; that is, our prediction *prior* to seeing the data.

To begin, though, you won't have to do this for yourself. In these examples we use the `rstanarm` package which tries to look as much like regular linear models as possible, and sets sensible defaults to make it easy to use Bayesian methods; `rstanarm` works in three steps:

1. It makes some assumptions that small effects are quite likely, but very large effects (e.g. large regression coefficients) are much less likely. This is our ***prior***. We can adjust it if we like, but the defaults are sensible.
2. It then runs the model, calculating the most likely parameter values (much like `lm` would).
3. It combines these estimates with our prior assumptions using Bayes rule. The result is the posterior probability distribution, or ***posterior***.

As it works, `rstanarm` repeats steps 2 & 3 many thousands of times as part of a simulation. The variation between runs of the simulation allow us to quantify how uncertain we are about our predictions.

---

Doing this is simpler than describing it. First we need to load `rstanarm` and the `tidybayes` package:

Then we just replace the `lm` function with `stan_glm`. Everything else remains the same apart from lots of extra output as the model fits, which you can ignore for now:

```
library(rstanarm) # load this first to make stan_glm available
iqmodel.bayes <- stan_glm(kid_score ~ mom_iq + mom_hs, data=kidiq)
```

The `rstanarm` package is a bit 'chatty' in it's output, but if we just want to see the model coefficients we can use the `tidy` function in the `broom` package:

```
library(broom)
tidy(iqmodel.bayes)
```

```
## # A tibble: 3 x 3
##   term                  estimate std.error
```

```
##   <chr>                     <dbl>     <dbl>
## 1 (Intercept)              31.7      6.33
## 2 mom_iq                    0.564    0.0613
## 3 mom_hsDid not complete   -5.96     2.16
```

Try re-running our `iqmodel` using `stan_glm()` in place of `lm()`. Use the formula: `kid_score ~ mom_iq + mom_hs`.

Use `summary` or `tidy` to see the coefficients and standard errors.

Compare your results with the results using `lm`

Why don't my result match my friends' exactly?

Because of the way Bayesian models are fit, results can vary very slightly each time you run them. There are ways of minimising this variation, but don't be worried about it; trying to estimate regression coefficients to more than a few decimal places is probably an example of false precision.

**Bayesian credible intervals** When using `stan_glm`, rather than using `predict` or augment we can use the `add_fitted_samples` and `add_predicted_samples` functions to calculate Bayesian *credible* or *predictive* intervals.

These samples can be for individuals (like prediction intervals) or for the predicted mean (like confidence intervals):

We calculate the credible interval in two steps:

1. Make predictions (samples) from the simulation used to fit the model; usually we make many thousands of simulated predictions (the computer does it for us).
2. We calculate the 2.5th and 97.5th percentiles of all of the samples to give the interval of interest.

The `tidybayes` package makes this almost automatic. First, make sure it's loaded:

```
library(tidybayes)
```

If we want the **credible interval** for the mean of a sample we use the `add_fitted_draws` to make our thousands of simulations. Then we use the `mean_qi` function to summarise them. By default it gives the 95% credible interval:

```
add_fitted_draws(iqmodel.bayes, newdata=new.mother) %>%
  mean_qi()
```

```
## # A tibble: 1 x 9
## # Groups:   mom_iq, mom_hs [1]
##   mom_iq mom_hs     .row .value .lower .upper .width .point .interval
##    <dbl> <chr>     <int>  <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1     97 Completed     1   86.4   84.4   88.4   0.95 mean   qi
```

If we want to calculate the Bayesian **prediction interval**, we replace `add_fitted_draws` with `add_predicted_draws`

```
add_predicted_draws(iqmodel.bayes, newdata=new.mother) %>%
  mean_qi()
```

```
## # A tibble: 1 x 9
## # Groups:   mom_iq, mom_hs [1]
##   mom_iq mom_hs     .row .prediction .lower .upper .width .point .interval
##    <dbl> <chr>     <int>       <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1     97 Completed     1        86.4   50.7   122.   0.95 mean   qi
```

Compare the Bayesian credible/prediction intervals with the classical confidence/prediction intervals. Hopefully they are not too different in this example.

How does this all work?

Although interpreting Bayesian models is simpler, fitting them is computationally quite complex. Thankfully various R packages do the heavy lifting for us, but you can read more about some of the background here.

## Summary of key points

The table below shows the links between the different types of intervals we have covered:

| Predict for… | Classical | Bayesian |
|---|---|---|
| An individual | Prediction interval | Posterior predictive interval |
| The mean | Confidence interval | Credible interval |

Interpretations for 95% intervals of each type are as follows:

- Prediction interval: complicated, see above
- Confidence interval: complicated, see above
- Posterior predictive interval: the range in which we expect 95% of future observations to fall.
- Credible interval: the range in which we are 95% sure the mean is.

Use the (new) model formula `kid_score ~ mom_iq + mom_hs + mom_age`, and assume a mother aged 30 with IQ=100, who completed high school.

Calculate:

- Classical confidence intervals
- Bayesian credible and posterior predictive intervals.

Try using expand grid to plot a range of predictions, for mothers of different ages.

Tell me how to do it!

For the Classical intervals, use `predict(model, newdata=..., interval="...")`

For the Bayesian intervals you need to use `add_predicted_samples` or `add_fitted_samples` to get the simulations, and then use `mean_qi` to get the 95% interval.

I give up, show me exactly how to do it…

Classical:

```
model <- lm(kid_score ~ mom_iq + mom_hs + mom_age, data=kidiq)
new.mother <- tibble(mom_iq=100, mom_hs="Completed", mom_age=30)

# prediction interval
predict(model, newdata=new.mother, interval="prediction")
```

```
##       fit      lwr      upr
## 1 89.6287 53.62501 125.6324
```

```
# confidence interval
predict(model, newdata=new.mother, interval="confidence")
```

```
##       fit      lwr     upr
## 1 89.6287 84.72269 94.5347
```

Bayesian:

```
new.mother = tibble(mom_iq=100, mom_hs="Completed", mom_age=30)

add_predicted_draws(model, newdata=new.mother) %>% mean_qi()
```

```
## # A tibble: 1 x 10
## # Groups:   mom_iq, mom_hs, mom_age [1]
##   mom_iq mom_hs  mom_age  .row .prediction .lower .upper .width .point .interval
##    <dbl> <chr>     <dbl> <int>       <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1    100 Comple~      30     1        89.6   53.1   126.   0.95 mean   qi
```

```
add_fitted_draws(model, newdata=new.mother) %>% mean_qi() #
```

```
## # A tibble: 1 x 10
## # Groups:   mom_iq, mom_hs, mom_age [1]
##   mom_iq mom_hs    mom_age  .row .value .lower .upper .width .point .interval
##    <dbl> <chr>       <dbl> <int>  <dbl>  <dbl>  <dbl>  <dbl> <chr>  <chr>
## 1    100 Completed      30     1   89.6   85.0   94.4   0.95 mean   qi
```

## Beyond intervals (extension exercises)

One of the nice things about modern Bayesian methods is that we don't have to pick just a single number to represent our estimate, or a single interval to represent our uncertainty.

As mentioned described in this additional explanation, to fit the model the computer runs a simulation thousands of time. On each run, it can make a prediction. Because of the way the method works, predictions from the simulation are made *in proportion* to their probability.

This means that:

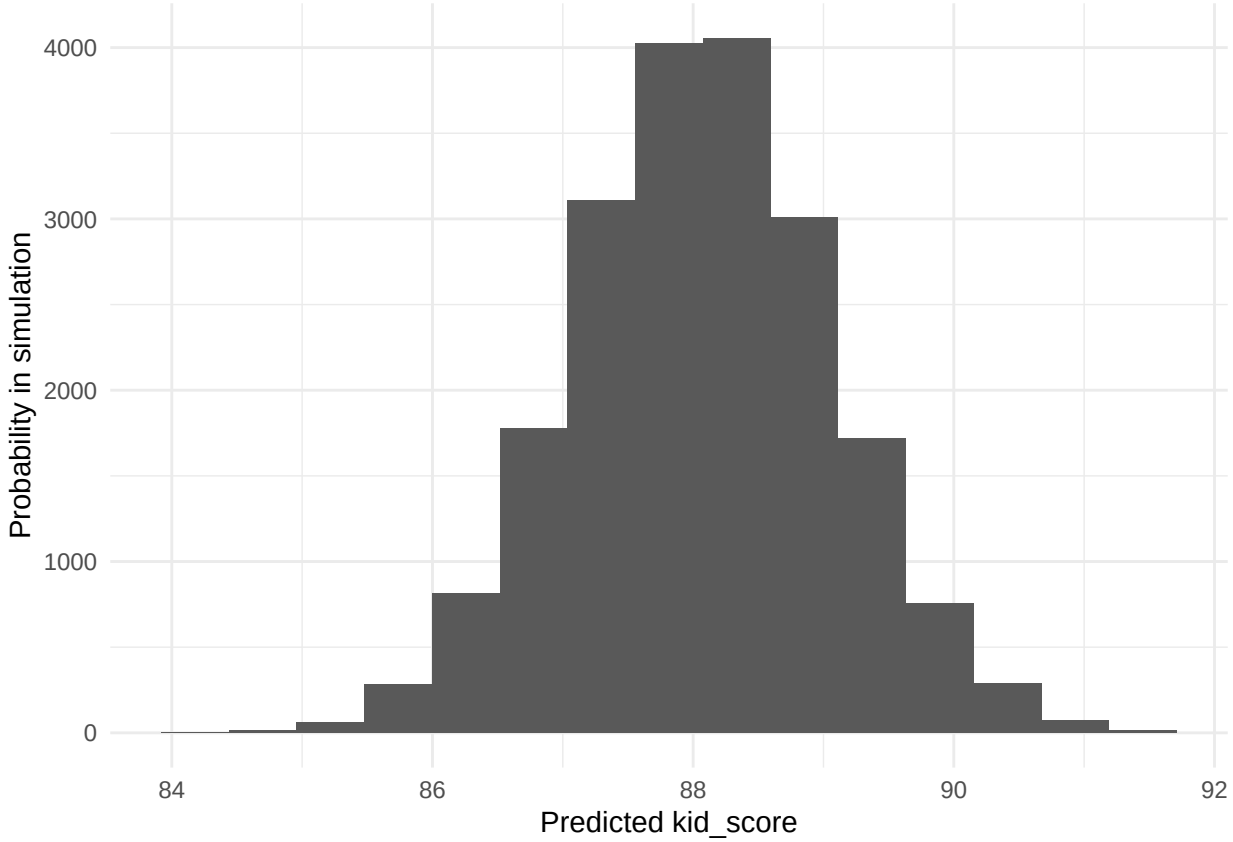
1. If we plot the simulation results, we can see what is the most likely *distribution* of outcomes.
2. By counting different outcomes within the simulation, and doing simple arithmetic, we can calculate the *probability* of different outcomes occurring.

### Visualising the posterior distribution

Many experts in Bayesian methods recommend avoiding single-number summaries rather than trying to summarise the simulation results as a single value or range (e.g. see Gabry et al, although this covers many more advanced topics too).

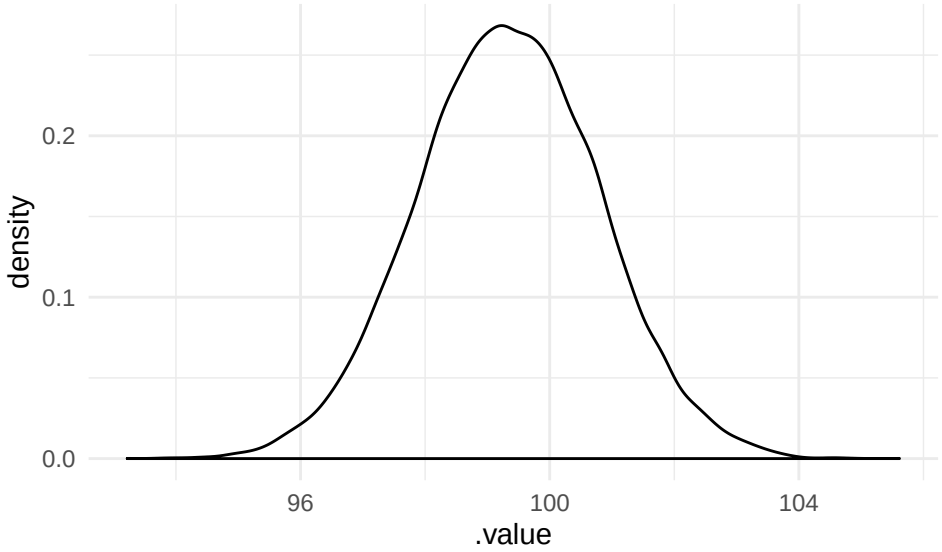
In the code above we used `mean_qi` to calculate the mean and percentile intervals of the predictions from the simulations. But we can also use `ggplot` to look at the *distribution* of simulated predictions directly:

```
add_fitted_draws(iqmodel.bayes, newdata=new.mother) %>%
  ggplot(aes(.value)) +
  geom_histogram(bins=15) +
  ylab("Probability in simulation") +
  xlab("Predicted kid_score")
```

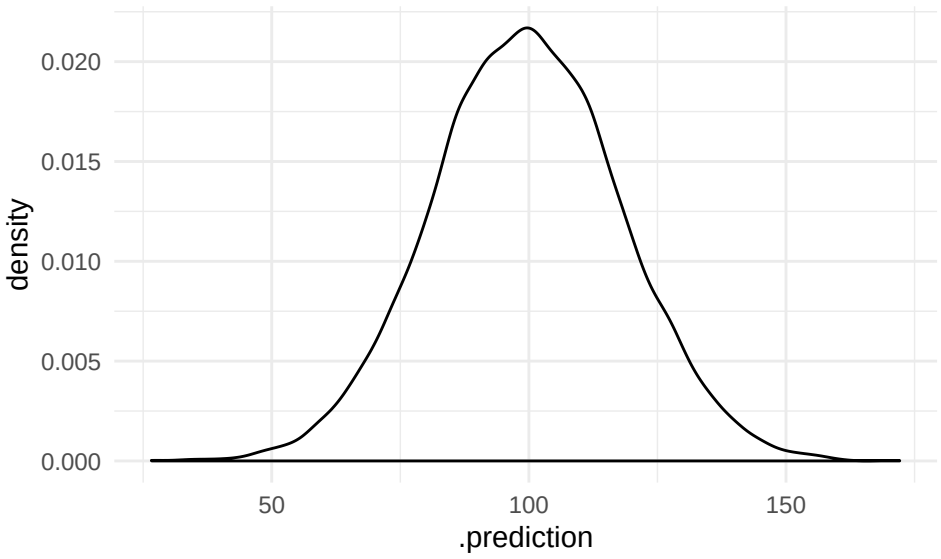



Make new predictions (using `add_fitted_samples`) for the *average* `kid_score` for a mother with an IQ of 120 who completed school.

Visualise these predictions using `ggplot`. It should look something like this if you use a density plot (`geom_density`) instead of a histogram:




Now make predictions for new *individuals*, who have the same IQ and who completed school (i.e. use the `add_predicted_samples` function). Plot the predictions; they should look something like this:

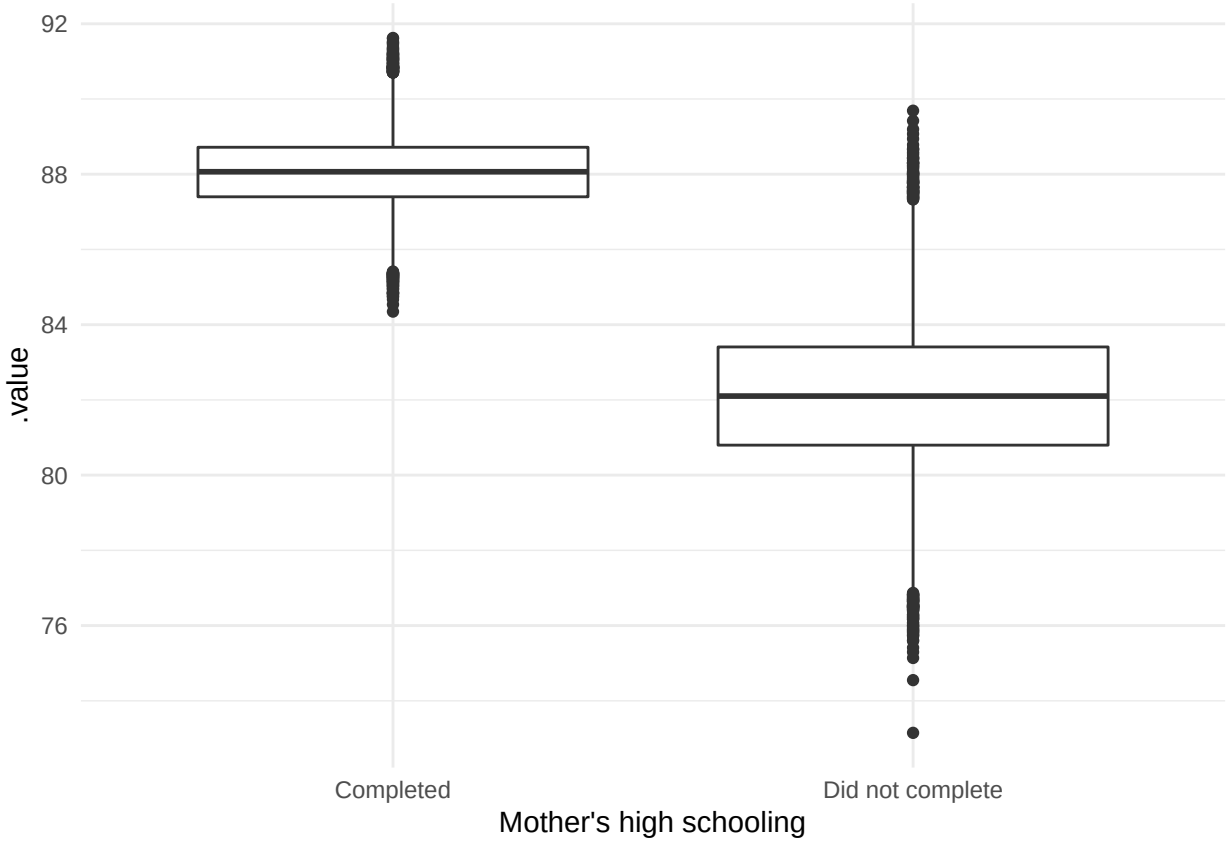



Are you more certain about the future *average* prediction, or the future *individual* prediction? How is this shown in your plots?

**More predictions**

Using `iqmodel.bayes`, predict *average* IQs for children of different mothers with an IQ of 100, with or without high school education.

Create a boxplot to compare *average* outcomes. It should look something like this:



### Being a Bookie

In applied work (or if you really did want to set up a betting shop) we might also want to make probability statements about particular examples or cases, based on our data.

For example clinicians or policy makers might want to hear statements like:

- "There's a 65% chance CBT will reduce your depression by more than 10 points."
- "If you have an IQ of 100, there's a 5% chance your child's IQ will be $< 90$."

To do this we just ***count*** the number of times each outcome happened within the simulation. For example:

```
new.mother = tibble(mom_iq=97, mom_hs="Completed")
add_predicted_draws(iqmodel.bayes, newdata=new.mother) %>%
  summarise(mean(.prediction < 60))
```

```
## # A tibble: 1 x 4
## # Groups:   mom_iq, mom_hs [1]
##   mom_iq mom_hs     .row `mean(.prediction < 60)`
##    <dbl> <chr>     <int>                    <dbl>
## 1     97 Completed     1                   0.0714
```

Explain that code in detail…

In the code above, `add_predicted_draw` makes many predictions for `new.mother`… that is a new mother which an IQ = 100, and who completed high school.

Rather than just one prediction, this makes many (thousands) of predictions for this new mother, one for each run of the model simulation.

It returns a dataframe, with a `.prediction` column, which is the predicted value in each simulation run.

We then use `summarise` to count how many times `.prediction < 60`. We do this within the `mean(...)` function; this converts the `TRUE` and `FALSE` values to `0` and `1` first, so the average of `c(TRUE, FALSE)` is 0.5.

In short, `mean(pred < 60)` gives us the ***probability*** that `.prediction` was less than 60 across all of the simulation runs.

So that's about 8% of cases where the `mom_iq` was 97 and `kid_score` is predicted to be less than 60 (your numbers might vary a little because of the way the software works).

If you were running a bookmaker, what odds should we offer a mother with an IQ of 70 who wants to wager her child will be highly intelligent and have an IQ $> 120$?

That is, if the mother is prepared to bet £1, how much would you offer in return?

To make the calculation, use `iqmodel.bayes`, make predictions (`add_predicted_draws`) for a new mother who has an IQ of 70 and who completed high school. What is the probability that her child will have an IQ $> 120$?

Remember, the odds (as used in betting shops) are $p/(1-p)$.

–>

# Building models 1

### In brief

> Models need to be *appropriately complex*. That is, we want to make models that represent our theories for the underlying causes of our data. Often this means adding many variables to a regression model. But we won't always be sure which variables to add. Adding multiple variables also brings challenges. Where predictors are highly correlated (termed **multicollinearity**) then model results can be confusing.

## Multiple regression with several continuous predictors

- Slides from the session

### Overview

So far, you have used regression to predict an outcome variable from a predictor variable. For example, can we predict *academic performance* from *hours of study*?

You've also used it to determine whether the relation between two variables differs according to a categorical variable. Does the relation between academic performance and hours of study, for example, differ for *men* and *women*?

We often want to determine the extent to which an outcome variable is predicted by **several continuous predictors**.

For example, in addition to hours of study, a person's *IQ* or *academic interest* might also predict their academic performance. We may want to add these predictors to a model because it may serve to *improve* the prediction of academic performance.

Today, we will:

- learn how to conduct a multiple regression with several continuous predictor variables
- evaluate the regression model with statistics ($R^2$, $F$-statistic, $t$-values)
- use Venn diagrams to help conceptualise the contribution of predictors to a model

### Simple vs. multiple regression

- **Simple regression** is a linear model of the relationship between *one outcome variable and one predictor variable*. For example, can we predict `exam performance` on the basis of `IQ` scores?
- **Multiple regression** is a linear model of the relationship between *one outcome variable and more than one predictor variable*. For example, can we predict `exam performance`based on `IQ` scores *and* `attendance` at lectures?

## Analysing the model

Suppose we want to construct a model to predict final university exam scores. This is the task faced by some admissions tutors! We'll start off with a simple regression model, then work up to multiple regression.

Load the `ExamData` dataset from https://bit.ly/37GkvJg. This contains exam scores for students taking a university course. (Make sure `tidyverse` is loaded first!)

### Learning tip

Try typing out the code today if you usually cut and paste it to R!

```
ExamData <- read_csv('https://bit.ly/37GkvJg')

ExamData %>% head()
> # A tibble: 6 x 7
>   finalex entrex   age project    iq proposal attendance
>     <dbl>  <dbl> <dbl>   <dbl> <dbl>    <dbl>      <dbl>
> 1      38     44  21.9      50   110       44          0
> 2      49     40  22.6      75   120       70          0
> 3      61     43  21.8      54   119       54          0
> 4      65     42  22.5      60   125       53          0
```

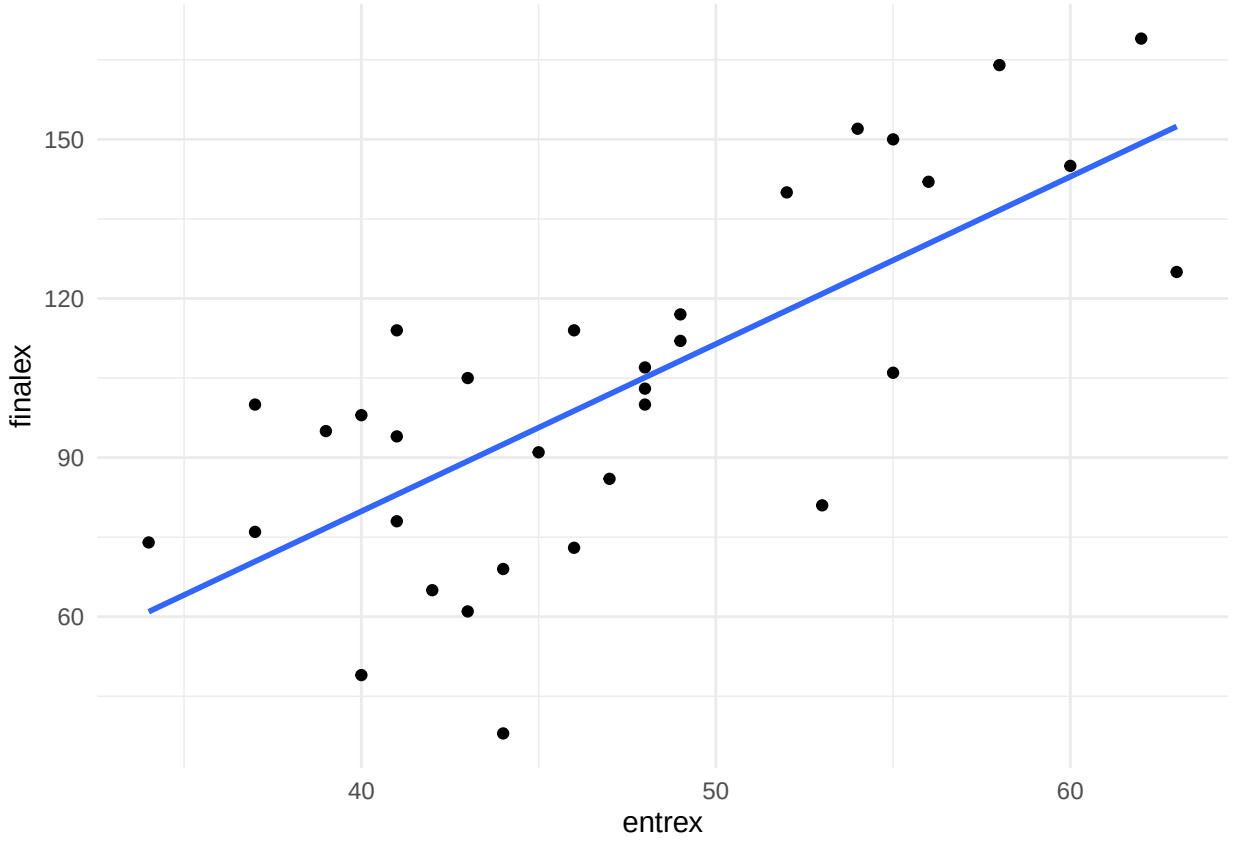
```
> 5      69     44  21.9      82   121        73          0
> 6      73     46  21.8      65   140        62          0
```

These are the variables in `ExamData`:

- `finalex`: final examination marks
- `entrex`: entrance examination marks
- `age`: age in years
- `project`: dissertation project marks
- `iq`: IQ score
- `proposal`: dissertation proposal grade
- `attendance`: 1 = high attendance; 0 = low attendance

First, let's ask whether `finalex` is predicted by `entrex`. Plot these variables:

```
ExamData %>%
  ggplot(aes(x = entrex, y = finalex)) +
  geom_point() +
  geom_smooth(se=F, method=lm)
```




There looks to be a positive association - students with higher entrance exam scores tend to have higher final exam scores. A good start!

To conduct the simple regression with `finalex` as the outcome variable, and `entrex` as the predictor variable, use `lm`:

```
m1 <- lm(finalex ~ entrex, data = ExamData)
```

**Explanation**: `finalex ~ entrex` can be read as "`finalex` is predicted by `entrex`". The model is stored in `m1`.

View the intercept of the regression line and the coefficient for `entrex`:

```
m1
>
> Call:
> lm(formula = finalex ~ entrex, data = ExamData)
>
> Coefficients:
> (Intercept)       entrex
>     -46.305        3.155
```

We can therefore write the regression equation:

$Predicted\ final\ exam\ score = -46.305 + 3.155(entrance\ exam)$

Use `summary(m1)` to display statistical analysis of the model:

```
summary(m1)
>
> Call:
> lm(formula = finalex ~ entrex, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -54.494 -21.185   3.733  18.124  30.969
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) -46.3045    25.4773  -1.817   0.0788 .
> entrex        3.1545     0.5324   5.925 1.52e-06 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 22.7 on 31 degrees of freedom
> Multiple R-squared:  0.531,   Adjusted R-squared:  0.5159
> F-statistic:  35.1 on 1 and 31 DF,  p-value: 1.52e-06
```

**Explanation of the output**:

**`Residuals:`** provides an indication of the discrepancy between the values of `finalex` predicted by the model (i.e., the regression equation) and the actual values of `finalex`. If the model does a good job in predicting `finalex`, the residuals should be relatively small.

- The difference between `Min` and `Max` gives us some idea of the range of error in the prediction of `finalex` scores. The difference in `3Q` and `1Q` is the interquartile range. The `median` of the residuals is 3.73.

**`Coefficients:`** contains tests of statistical significance for each of the coefficients. The values in the column headed `Pr(>|t|)` are the $p$-values associated with the $t$-values for the coefficients for each predictor. The $t$-values test a null hypothesis that the coefficients are equal to zero. A $p$-value less than .05 indicates that a predictor is statistically significant.

- The row for the `(intercept)` reports a $t$-test for whether the value of the intercept differs from zero. We're not usually interested in this test (so don't report it).
- The row for `entrex` tests whether the value of its coefficient (3.15) differs from zero. A coefficient of

zero would be expected if the predictor explained no variance in the outcome variable. The coefficient for `entrex` (3.15) is clearly greater than zero. We can report this by saying that `extrex` is a statistically significant predictor of `finalex`, $b = 3.15$, $t(31) = 5.92$, $p < .001$.

**`Multiple R-squared:`** This is $R^2$ - the **proportion of variance in `finalex` explained by `entrex`**. Here, $R^2 = 0.531$. So approximately half of the variance in `finalex` is explained by `entrex`. It's usually referred to simply as "R-squared" or $R^2$.

- $R^2$ is often reported as a percentage. To get this, simply multiply the value by 100. i.e., 0.531 x 100 = 53.10%.

**`Adjusted R-squared:`** is an estimate of $R^2$, but adjusted for the population. Despite the usefulness of this statistic, most studies still tend to report only the (unadjusted) $R^2$ value. If reporting the `Adjusted R-squared` value, be sure to label it clearly as such. Here, Adjusted R-squared = 0.52.

**`F-statistic:`** This compares the variance in `finalex` explained by the model with the variance that it does not explain (i.e., explained variance divided by unexplained variance). Higher values of $F$ indicate that the model explains greater variance in an outcome variable. If the $p$-value associated with the $F$-statistic is less than .05, we can say that the model significantly predicts the outcome variable.

Hence, we can say that a model consisting of `entrex` alone is a significant predictor of `finalex`, $F(1, 31) = 35.10$, $p < .001$. Higher `entrex` scores tend to be associated with higher `finalex` scores. If our model did not explain any variance in `finalex`, we wouldn't expect this to be statistically significant.

- In simple regression, the null hypothesis being tested on the $F$-statistic is that the slope of the regression line in the population is equal to zero. This is actually equivalent to the $t$-test on the `entrex` coefficient. So in simple regression, report the $F$-statistic for the overall regression or the $t$-test on the coefficient (not both). This equivalence between $F$ and $t$ does not hold true for multiple regression, as we shall see later.

### Now you have a go

Run another simple regression:

- set `finalex` as the outcome variable and `project` as the predictor variable
- store the output in a variable with a different name (`m2`)
- then display the output of `m2` using `summary()`.

Try yourself first before clicking to show the code

```
m2 <- lm(finalex ~ project, data= ExamData)

summary(m2)
>
> Call:
> lm(formula = finalex ~ project, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -64.015 -21.686  -0.573  21.758  70.427
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept)   4.6968    40.1677   0.117   0.9077
> project       1.4442     0.5861   2.464   0.0195 *
```

```
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 30.32 on 31 degrees of freedom
> Multiple R-squared:  0.1638,  Adjusted R-squared:  0.1368
> F-statistic: 6.072 on 1 and 31 DF,  p-value: 0.01948
```

Answer the following: (report statistics to 2 decimal places)

- What is the value of the coefficient for `project`?
- What proportion of the variance in `finalex` is explained by `project`?: $R^2 =$ (or %).
- Write down the regression equation (on a bit of paper).

Show me

- $Predicted\ final\ exam\ score = 4.70 + 1.44(project)$
- Is `project` alone a statistically significant predictor of `finalex`, as indicated by the $F$-statistic? no yes
- Report the $F$-ratio in APA style, that is, in the form

  **$F$(df1, df2) = $F$-statistic, $p$ = $p$-value**:

Show me

$F(1, 31) = 6.07$, $p = .02$

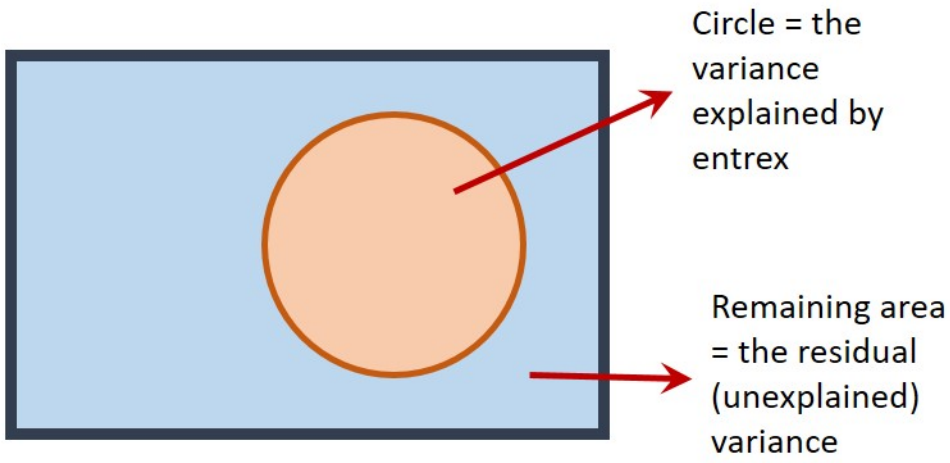
- Individuals with lower higher project scores tended to have higher final exam scores.

## Conceptualising the variance explained by predictors

Venn diagrams are useful for understanding the variance that predictors explain in the outcome variable. They are especially useful for understanding what's going on in multiple regression.

Suppose the rectangle below represents all of the *variance* in `finalex` to be explained.

The area of the circle below represents the variance in `finalex` explained by `entrex` in the first simple regression we did. If this diagram were drawn to scale (it's not), the area of the circle would be equal to the value of $R^2$ (i.e., 53.1% of the rectangle).




The part of the rectangle not inside the circle represents the variance in `finalex` that is *not* explained by the model (i.e., the unexplained or *residual* variance).

To *improve* the model, we can explore whether adding in other predictors to the model explains additional variance, thereby increasing the total $R^2$ of the model.

You might think that we can simply add in variables (circles, above) to the model as we wish, until all the residual variance has been explained. This seems fine to do until we learn that if we were to add as many predictors to the model as there are rows in our data (33 individuals in our `ExamData`), then we'd perfectly predict the outcome variable, and have an $R^2$ of 100%! This would be true even if the predictors consisted of random values. Our model would clearly be meaningless though. We ideally want to explain the outcome variable with relatively few predictors.

## Adding predictor variables to the model

An issue that can arise when adding variables to a model is that predictors are usually correlated to some extent. This can make interpretation of multiple regressions tricky. For example, a predictor that is statistically significant in a simple regression may become non-significant in a multiple regression. Let's see a demonstration of this!

We'll now add `project` to the model with `entrex`. First, check the correlation between predictors:

```
ExamData %>%
  select(entrex,project) %>%
  cor()
>             entrex   project
> entrex  1.0000000 0.2908253
> project 0.2908253 1.0000000
```

The correlation between `entrex` and `project` is $r =$

Our predictor variables are weakly correlated. We should keep this in mind going forward.

Now run a *multiple regression* to predict `finalex` from both `entrex` and `project`. Again, use `lm` but use the `+` symbol to add predictors to the model:

```
m3 <- lm(finalex ~ entrex + project, data = ExamData)

summary(m3)
>
> Call:
> lm(formula = finalex ~ entrex + project, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -41.880 -16.617   4.636  15.562  35.273
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) -84.8289    33.6846  -2.518   0.0174 *
> entrex        2.8894     0.5406   5.344 8.81e-06 ***
> project       0.7515     0.4457   1.686   0.1021
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 22.06 on 30 degrees of freedom
> Multiple R-squared:  0.5716,  Adjusted R-squared:  0.5431
> F-statistic: 20.02 on 2 and 30 DF,  p-value: 3e-06
```

In this model with `entrex` and `project`as predictors:

What is the value of $R^2$ (as a percentage): %

By how much has $R^2$ *increased* in this model, relative to the model with `entrex` alone (where $R^2$ was 53.10%)? (as a percentage) (you will need to calculate this) %

Is the overall regression model predicting `finalex` on the basis of `entrex` and `project` statistically significant? yes no

- Is `entrex` a statistically significant predictor of `finalex`? yes no
- We can report this in the following way: the $t$-test on the coefficient for `entrex` is statistically significant, $b = 2.89$, $t(30) = 5.34$, $p < .001$.
- Is `project` a statistically significant predictor of `finalex` in this model? yes no
- What is the value of the coefficient for `project`? $b =$
- Report the $t$-statistic in APA style:

Show me

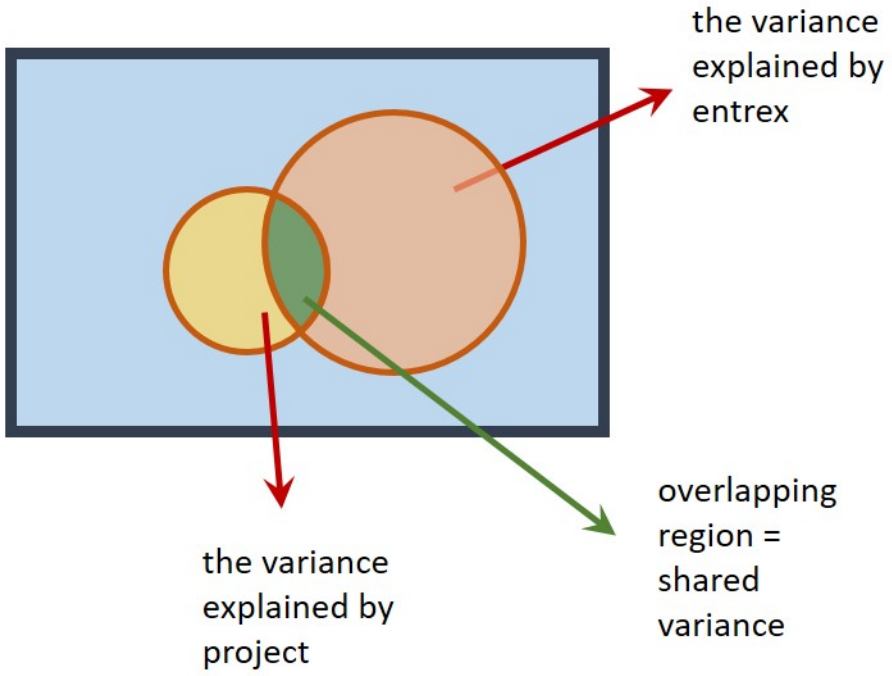
Project mark was not a statistically significant predictor of final examination in this model, $b = 0.75$, $t(30) = 1.69$, $p = .10$

Looking across the analyses we've performed, we can see that `project` is a (weak) but statistically significant predictor of `finalex` in a simple regression. However, when it is included in a model that also includes `entrex` it is not a significant predictor! What's going on?

- The model containing only `project` explains 16.38% of the variability in `finalex`.
- The model containing only `entrex` explains 53.10% of the variability in `finalex`.
- However, a model containing *both* `project` and `entrex` only explains 57.16% of the variability in `finalex`, not 16.38 + 53.10 = 69.48%, as we might expect.

This is because the predictors are *correlated* ($r = .29$) and so the variance they explain in `finalex` is *shared*.

We could represent this on a Venn diagram as follows:




The correlation is represented as an overlap in the circles. Their total area (57.16%) is therefore *less* than the area they'd explain if there were no overlap (69.48%) (i.e., if there was no correlation).

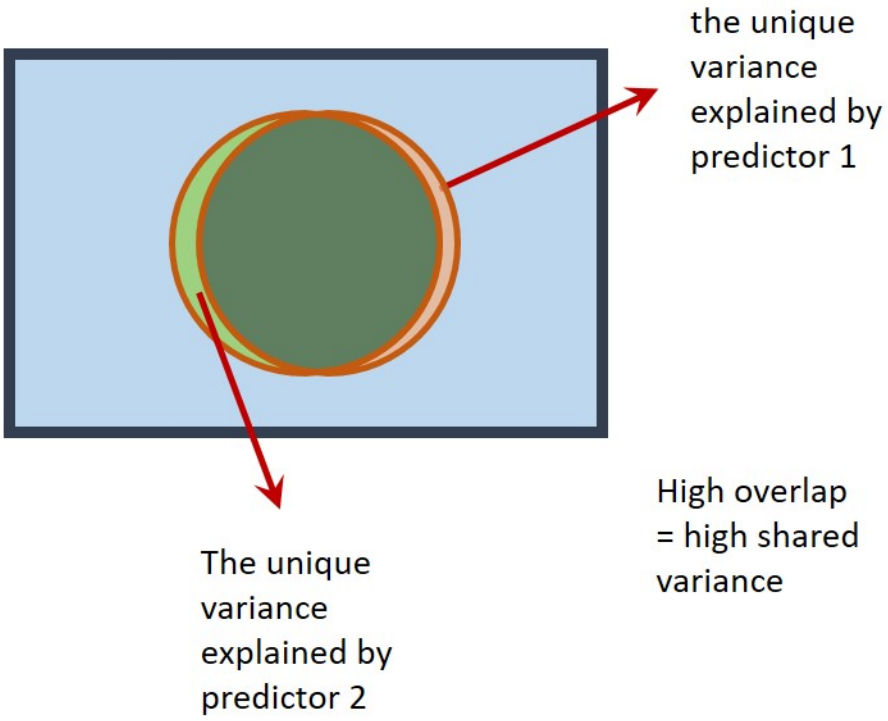
**This demonstrates an important point**: The $t$-tests on the coefficients in a multiple regression assess the **unique** contribution of each predictor in the model. That is, they test the variance a predictor explains in an outcome variable, **after** the variance explained by the other predictors has been taken into account. This is why `project` is not statistically significant in the multiple regression model – it only explains a small amount of variance once `entrex` has been taken into account.

It is possible to think of the $F$-statistic and $t$-value in multiple regression in terms of the Venn diagram:

- The ***F*-statistic** compares the explained variance with the unexplained variance. The explained variance is represented by the ***outline*** of the two circles in the Venn diagram above. The unexplained variance is the remaining blue area of the rectangle.
- The ***t*-value** compares the unique variance a predictor explains with the remaining unexplained variance. For example, for `project` in the Venn diagram above, this would be the area in the orange ***crescent***, relative to the remaining blue area in the rectangle.

## Multicollinearity

If the correlation between predictors is very high (greater than $r = 0.9$), this is known as **multicollinearity**. On a Venn diagram, the circles representing the predictors would almost completely overlap. Multicollinearity can be a problem in multiple regression. Predictors may explain a large amount of variance in the outcome variable, but their 'unique' contribution in a multiple regression may be small. A situation can arise where *neither* predictor may be statistically significant even though the overall regression is significant!




An example of multicollinearity in the `ExamData` dataset can be seen with the variables `project` and `proposal`.

Obtain the correlation between `project` and `proposal`:

Show me

```
ExamData %>%
  select(project, proposal) %>%
  cor()
>             project  proposal
> project  1.0000000 0.9371487
> proposal 0.9371487 1.0000000
```

The correlation between `project` and `proposal` is $r =$ .

To see the effects of multicollinearity, conduct a regression with `finalex` as the outcome variable and `project` and `proposal` as the predictor variables.

Show me

```
multi1 <- lm(finalex ~ project + proposal, data = ExamData)

summary(multi1)
>
> Call:
> lm(formula = finalex ~ project + proposal, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
```

```
> -64.287 -22.590  -0.346  22.395  70.289
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept)   4.8784    40.8601   0.119    0.906
> project       1.2751     1.7072   0.747    0.461
> proposal      0.1826     1.7263   0.106    0.916
>
> Residual standard error: 30.81 on 30 degrees of freedom
> Multiple R-squared:  0.1641,  Adjusted R-squared:  0.1084
> F-statistic: 2.945 on 2 and 30 DF,  p-value: 0.06797
```

- How much variance in `finalex` is explained by the model: $R^2 =$ %.
- Is the overall regression statistically significant? yes no
- Is the coefficient for `project` statistically significant? yes no
- Is the coefficient for `proposal` statistically significant? yes no

Now run two simple regressions to determine whether `project` and `proposal` explain variance in `finalex` and are statistically significant predictors when in models on their own.

Show me

```
multi2 <- lm(finalex ~ project, data = ExamData)
summary(multi2)
>
> Call:
> lm(formula = finalex ~ project, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -64.015 -21.686  -0.573  21.758  70.427
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept)   4.6968    40.1677   0.117   0.9077
> project       1.4442     0.5861   2.464   0.0195 *
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 30.32 on 31 degrees of freedom
> Multiple R-squared:  0.1638,  Adjusted R-squared:  0.1368
> F-statistic: 6.072 on 1 and 31 DF,  p-value: 0.01948

multi3 <- lm(finalex ~ proposal, data = ExamData)
summary(multi3)
>
> Call:
> lm(formula = finalex ~ proposal, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -64.987 -22.987  -1.378  24.059  68.921
>
> Coefficients:
```

```
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept)   16.628     37.441   0.444   0.6601
> proposal       1.391      0.598   2.326   0.0267 *
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 30.59 on 31 degrees of freedom
> Multiple R-squared:  0.1486,  Adjusted R-squared:  0.1211
> F-statistic: 5.409 on 1 and 31 DF,  p-value: 0.02675
```

- In a simple regression with `finalex` as the outcome variable, and `project` as the predictor variable, $R^2 =$ %.
- Is the overall regression statistically significant? yes no
- In a simple regression with `finalex` as the outcome variable, and `proposal` as the predictor variable, $R^2 =$ %.
- Is the overall regression statistically significant? yes no
- Try to explain what's going on here in your own words. Click below or ask if you get stuck.

Explain

*Interpretation*

- Because `proposal` and `project` are highly correlated ($r = 0.94$), this gives rise to the situation where the simple regressions indicate that they explain variance in `finalex`, but when both are included as predictors in a multiple regression, it appears as if neither are significant predictors of `finalex` !
- If this were a real scenario, we'd consider dropping `project` or `proposal` from the model. Because the correlation is so high, having one predictor is as good as having the other (more or less).
- It seems intuitive that a person's final project mark would be highly correlated with their proposal mark.
- The take-home message here is to check for high correlations between your predictor variables before including them in a multiple regression.

## Final exercise

As a final exercise, run a multiple regression to predict `finalex` from **three** predictors: `entrex`, `project`, and `iq`.

Show me how

```
multi4 <- lm(finalex ~ entrex + project + iq, data = ExamData)

summary(multi4)
>
> Call:
> lm(formula = finalex ~ entrex + project + iq, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -40.444 -16.174   5.509  14.312  33.338
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) -130.3803    54.7288  -2.382 0.023981 *
```

```
> entrex          2.6180     0.5978   4.379 0.000142 ***
> project         0.6874     0.4490   1.531 0.136620
> iq              0.4862     0.4610   1.055 0.300214
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 22.02 on 29 degrees of freedom
> Multiple R-squared:  0.5875,  Adjusted R-squared:  0.5448
> F-statistic: 13.77 on 3 and 29 DF,  p-value: 9.168e-06
```

Which variables are statistically significant predictors of `finalex`?

- `entrex` yes no
- `project` yes no
- `iq` yes no

On the basis of all the models conducted so far (with `entrex`, `project`, and `iq`), which model would you choose to best predict `finalex`?

Tell me which model seems best

The model containing `entrex` alone, as this seems to provide the simplest and most effective model of the `finalex`.

A general goal of regression is to identify the fewest predictor variables necessary to predict an outcome variable, where each predictor variable predicts a substantial and independent segment of the variability in the outcome variable.

### Summary of key points

- Predictors can be added to a model in `lm` using the `+` symbol
- e.g., `lm(finalex ~ entrex + project + iq)`
- Predictor variables are often correlated to some extent. This can affect the interpretation of individual predictor variables. Venn diagrams help to understand the results.
- The ***F*-statistic** tells us whether the model *as a whole* significantly predicts the outcome variable.
- The ***t*-values** tell us whether individual predictors in the model are statistically significant.
- In multiple regression, it's important to understand that the statistical significance of individual predictors only holds **after taking into account the other predictors in the model**.
- **Multicollinearity** exists when predictors are very highly correlated ($r$ above 0.9) and should be avoided.

## Building models 2

#### In brief

> In this session we discuss model selection in the context of ANOVA and the use of Bayes Factors to choose between theoretically interesting models.

### Using ANOVA and Bayes Factors to compare models

- **Slides from the session** (available 25-01-2020)

**Overview**

In the previous session, we saw that we can construct a linear model to predict an outcome variable (e.g., *final exam score* from *entrance exam score*). We also investigated how we can *improve* a model by adding several continuous predictors to it.

How do we know if one model is *better* or should be *preferred* over another model? We touched on a common sense approach in the last session - we ideally want models that explain the variance in an outcome variable but each predictor in the model should make a sizable and relatively independent contribution to the model.

Today we will cover a more formal approach to model comparison using:

- **ANOVA (Analysis of Variance)** and
- **Bayes Factors**

It's important that you are comfortable with the material from the first Building Models 1 session before proceeding today.

## Comparing models using ANOVA

We can use ANOVA to determine whether the addition of variables into a model leads to a statistically significant improvement in the variance it explains *overall*. We may want to do this, for example, when building on existing theories or models.

We'll start by comparing a model with *one* predictor vs. a model with *three* predictors.

Using the `ExamData` from the previous session, we'll run:

- a linear model with `finalex` as the outcome variable, and `entrex` as the predictor.
- a linear model with `finalex` as the outcome variable, and `entrex`,`age`, and `project` as the predictors.

```
ExamData <- read_csv('https://bit.ly/37GkvJg')
model1   <- lm(finalex ~ entrex, data = ExamData)
model2   <- lm(finalex ~ entrex + age + project, data = ExamData)
```

**Explanation of the code**: first the data is loaded into `ExamData`. The results of the simple regression are stored in `model1`. Those of the multiple regression are stored in `model2`.

Use `summary()` to display the results of each regression:

**Model 1:**

```
summary(model1)
>
> Call:
> lm(formula = finalex ~ entrex, data = ExamData)
```

```
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -54.494 -21.185   3.733  18.124  30.969
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) -46.3045    25.4773  -1.817   0.0788 .
> entrex        3.1545     0.5324   5.925 1.52e-06 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 22.7 on 31 degrees of freedom
> Multiple R-squared:  0.531,   Adjusted R-squared:  0.5159
> F-statistic:  35.1 on 1 and 31 DF,  p-value: 1.52e-06
```

**Model 2:**

```
summary(model2)
>
> Call:
> lm(formula = finalex ~ entrex + age + project, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -42.563 -16.519   4.901  16.991  36.424
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) -117.9159    46.4211  -2.540   0.0167 *
> entrex         3.0889     0.5734   5.387 8.66e-06 ***
> age            1.4231     1.3756   1.035   0.3094
> project        0.6280     0.4609   1.363   0.1835
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 22.03 on 29 degrees of freedom
> Multiple R-squared:  0.5869,  Adjusted R-squared:  0.5442
> F-statistic: 13.73 on 3 and 29 DF,  p-value: 9.353e-06
```

(If you are not sure what it means by "e-06" in the output above then see the FAQs here)

Make note of the variance explained by each model ($R^2$), i.e., `Multiple R-squared`: (report as a percentage, to 2 decimal places)

- Model 1: $R^2 =$ %
- Model 2: $R^2 =$ %

Which model explains a greater proportion of variance in `finalex`? extrex alone entrex, age, project

- Calculate the difference in $R^2$ between the models. `model2` improves the prediction of `finalex` by %

To compare the variance explained by each model, use `anova()`:

```
anova(model1, model2)
> Analysis of Variance Table
>
> Model 1: finalex ~ entrex
> Model 2: finalex ~ entrex + age + project
>   Res.Df   RSS Df Sum of Sq      F Pr(>F)
> 1     31 15981
> 2     29 14078  2      1903 1.9601 0.1591
```

**Explanation of the output:**

- `anova()` compares the variance that `model1` and `model2` explain with an $F$-statistic.
- `Pr(>F)` gives the $p$-value for this statistic. If the $p$-value is less than .05, then we can reject the null hypothesis that there is no difference in the variance explained by each model, and we can say that the variance that `model2` explains in `finalex` is significantly greater than that of `model1`.
- We can report the $F$-statistic in APA style as $F(2, 29) = 1.96$, $p = .16$. We can say that the additional 5.59% variance that `model2` explains relative to `model1` does not represent a statistically significant increase in $R^2$, and so `model2` should **not** be preferred over `model1`.

Comparing models in steps as we've done is sometimes called **hierarchical regression** or **sequential regression**. This type of regression is usually used for logical or theoretical reasons, when we want to know the contribution of a predictor (or a set of predictors) **over and above** an existing one.

**Now, you try using `anova` to compare models.**

The variable `attendance` in `ExamData` scores individuals according to whether their class attendance was low (0) or high (1). A researcher suspects that `attendance` may explain additional variance in `finalex` over and above `entrex`.

As an exercise, compare the following two models using the `anova()` approach above:

1. a model with `entrex` as a sole predictor of `finalex` (i.e., `model1`), and
2. a model where `finalex` is predicted by `extrex` and `attendance` (call this `model3`).

Is there sufficient evidence that a model with `entrex` *and* `attendance` explains more variance than a model with `entrex` alone?

Try yourself first, then click to see the code

```
# model1 was created earlier
summary(model1)
>
> Call:
> lm(formula = finalex ~ entrex, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -54.494 -21.185   3.733  18.124  30.969
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) -46.3045    25.4773  -1.817   0.0788 .
> entrex        3.1545     0.5324   5.925 1.52e-06 ***
```

```
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 22.7 on 31 degrees of freedom
> Multiple R-squared:  0.531,   Adjusted R-squared:  0.5159
> F-statistic:  35.1 on 1 and 31 DF,  p-value: 1.52e-06

# specify model3
model3 <- lm(finalex ~ entrex + attendance, data = ExamData)

# show model3
summary(model3)
>
> Call:
> lm(formula = finalex ~ entrex + attendance, data = ExamData)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -42.750 -11.750   1.801   9.689  30.347
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) -63.3108    20.2768  -3.122  0.00395 **
> entrex        3.2741     0.4173   7.846 9.35e-09 ***
> attendance   28.8202     6.3398   4.546 8.37e-05 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 17.76 on 30 degrees of freedom
> Multiple R-squared:  0.7223,  Adjusted R-squared:  0.7038
> F-statistic: 39.02 on 2 and 30 DF,  p-value: 4.499e-09

#compare model1 and model3
anova(model1, model3)
> Analysis of Variance Table
>
> Model 1: finalex ~ entrex
> Model 2: finalex ~ entrex + attendance
>   Res.Df     RSS Df Sum of Sq      F   Pr(>F)
> 1     31 15980.6
> 2     30  9462.4  1    6518.1 20.665 8.37e-05 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The variance explained by a model with `entrex` alone is $R^2 =$ %
- The $R^2$ for the model that also included `attendance` was $R^2 =$ %
- The increase in $R^2$ was %
- The ANOVA comparing models can be reported as: $F(,) =$ , $p < .001$.
- The increase in $R^2$ was statistically significant not significant.
- As indicated by the estimates of the coefficients for `entrex` and `attendance`, both negatively positively predict `finalex`.

- A higher `entrex` score and greater `attendance` is associated with a higher lower `finalex` score.

## Comparing models using Bayes Factors

An alternative approach to using ANOVA to compare models is to use **Bayes Factors**.

A **Bayes Factor** is the **probability of obtaining the data under one model compared to another**.

For example, a Bayes Factor equal to 2 would tell us that the data are *twice* as likely under one model than another. A Bayes Factor equal to 0.5 would tell us that the data are *half* as likely under one model than another.

Unlike classical tests of statistical significance (with *p*-values), Bayes Factors also allow us to *quantify* evidence for the null hypothesis. Very handy!

To compute a Bayes Factor for a specific linear model, we use `lmBF` in the `BayesFactor` package (where `lm` stands for *linear model* and `BF` stands for *Bayes Factor*).

First, we need to load the `BayesFactor` package:

```
library('BayesFactor')
```

We can use the `lmBF` function in the same way we use `lm`. The function will return a **Bayes Factor** for the model we specify.

Let's determine the Bayes Factor for `model1`

```
model1.BF <- lmBF(finalex ~ entrex, data = as.data.frame(ExamData) )
```

**Explanation of the code**: The model is specified in exactly the same way as with `lm`. Due to a limitation of the package, however, we must convert `ExamData` from a tibble to a data frame using `as.data.frame`. Otherwise, the command works in the same way. The results are stored in `model1.BF`.

To look at what's stored in `model1.BF`:

```
model1.BF
> Bayes factor analysis
> --------------
> [1] entrex : 8310.846 ±0.01%
>
> Against denominator:
>   Intercept only
> ---
> Bayes factor type: BFlinearModel, JZS
```

**Explanation of the output**:

- The Bayes Factor provided for the model with `entrex` is equal to **8310.85**.
- The `Against denominator: Intercept only` means that the model with `entrex` is being compared with a model that contains an **intercept only**. In an intercept-only model, the coefficient for `entrex` is equal to zero; that is, the regression line is a flat line (equal to the *mean* of `entrex`).

- The value of our Bayes Factor indicates that the model with `entrex` in is much more likely than a model that contains only an intercept (8310.85 times more likely, to be precise). We can therefore be confident that a model with `entrex` is preferable to the intercept only model (just as with our classical analysis). Happy days!

Now let's do the same for `model2`:

```
# specify the model
model2.BF <- lmBF(finalex ~ entrex + age + project, data = as.data.frame(ExamData) )

# show the Bayes Factor
model2.BF
> Bayes factor analysis
> --------------
> [1] entrex + age + project : 2427.676 ±0%
>
> Against denominator:
>   Intercept only
> ---
> Bayes factor type: BFlinearModel, JZS
```

**Explanation:** The Bayes Factor is equal to **2427.68**. Again, this indicates that the model with `entrex` and `age` is much more likely than a model with only the intercept in (this is not that surprising given the result for `model1.BF` above).

But, what we want to know is whether `model2` (containing `entrex` and `age`) is **more** likely than `model1` (containing only `entrex`). We can determine this by *dividing* the Bayes Factor for `model2` by the Bayes Factor for `model1`:

```
model2.BF / model1.BF
> Bayes factor analysis
> --------------
> [1] entrex + age + project : 0.2921093 ±0.01%
>
> Against denominator:
>   finalex ~ entrex
> ---
> Bayes factor type: BFlinearModel, JZS
```

**Explanation:** The Bayes Factor for this comparison is 0.29. This means that `model2` is ***less than a third as likely*** than `model1`. So, `model2` is much *less* likely than `model1`. Not good news for `model2`!

**Interpreting the Bayes Factor**

- A Bayes Factor **equal to 1** tells us that probability of each model is the same.
- A Bayes Factor **greater than 1** means that `model2` is more likely than `model1`.
- A Bayes Factor **less than 1** means that `model1` is more likely than `model2`.

**Thus, our Bayes Factor of 0.29 indicates that `model1` is more likely than `model2`.**

**Reporting Bayes Factors**

**Notation**

We usually write the Bayes Factor in reports as $BF_{10}$ where:

- the subscript **1** in $BF_{10}$ denotes the less-constrained model (the alternative hypothesis). This is the model with **more predictors** (our `model2`).
- the subscript **0** in $BF_{10}$ denotes the more constrained or simpler model (i.e., the null hypothesis). This is the model with **fewer predictors** (our `model1`).

(You can just write BF10 if you prefer.)

**The Size of the Bayes Factor**

- If the Bayes Factor is **greater than 3** (i.e., $BF_{10} > 3$), we say that there is **substantial evidence for `model2`** (the less constrained model).
- If the Bayes Factor is **less than 0.33** (i.e., $BF_{10} < 0.33$), we usually say that there is **substantial evidence for `model1`** (the more constrained model).
- We say that intermediate values for the Bayes Factor (between 0.33 and 3) don't offer strong evidence for either model.

Thus, because our Bayes Factor of 0.29 is less than 1, this indicates greater evidence for `model1` than `model2`. Furthermore, because the Bayes Factor is less than 0.33, we have *substantial* evidence for `model1` over `model2`.

It's becoming increasingly common to report the Bayes Factor alongside the results of a classical analysis. Thus, we could report our results as follows: "There was insufficient evidence that the addition of age and project to the model containing entrance exam resulted in an increase in $R^2$, $F(2, 29) = 1.96$, $p = .16$; BF10 $= 0.29$."

**Now you try using Bayes Factors to compare models**

To supplement the comparison of `model3` and `model1` that you did with `anova`, now compute the Bayes Factor for `model3` vs. `model1`.

You'll need the following steps:

- Model 1: Obtain the Bayes Factor for a model with `extrex` as a sole predictor of `finalex` (we did this already above; it's stored in `model1.BF`)
- Model 2: Obtain the Bayes Factor for a model where `finalex` is predicted by `extrex` *and* `attendance` and store this in `model3.BF`.
- Compare the Bayes Factors in `model3.BF` and `model1.BF`.

Try yourself first, then click here for the code

```
# 1. show the BF for model1 vs. intercept only
model1.BF
> Bayes factor analysis
> --------------
> [1] entrex : 8310.846 ±0.01%
>
> Against denominator:
>   Intercept only
> ---
> Bayes factor type: BFlinearModel, JZS

# 2. Obtain the BF for model3 vs. intercept only, then show it
```

```
model3.BF <- lmBF(finalex ~ entrex + attendance, data = as.data.frame(ExamData) )

model3.BF
> Bayes factor analysis
> --------------
> [1] entrex + attendance : 2351114 ±0%
>
> Against denominator:
>   Intercept only
> ---
> Bayes factor type: BFlinearModel, JZS

# 3. Compare the BFs for model3 vs model1
model3.BF / model1.BF
> Bayes factor analysis
> --------------
> [1] entrex + attendance : 282.897 ±0.01%
>
> Against denominator:
>   finalex ~ entrex
> ---
> Bayes factor type: BFlinearModel, JZS
```

**Answer the following questions from the output:**

How much more likely is a model with`entrex` than an intercept only model?

- times more likely.

How much more likely is a model with `entrex` and `attendance` than an intercept only model?

- times more likely.

How much more likely is a model with `entrex` and `attendance` as predictors than a model with `entrex` alone?

- times more likely.

There is insufficient strong evidence that a model with `entrex` and `attendance` should be preferred over a model with `entrex` alone, given the data.

## Exercise

Now you will practise using ANOVA and Bayes Factors to compare models with a new dataset.

**Scenario:** A researcher would like to construct a model to predict scores in a memory task from several different variables. The data from 234 individuals are stored in the `memory_data` dataset, which are located at https://bit.ly/37pOTrC.

Use `read_csv` to load in the data at the link above to the variable `memory_data` and preview it with `head()`.

Try this yourself first. Click to show code

```
memory_data <- read_csv('https://bit.ly/37pOTrC')
memory_data %>% head()
> # A tibble: 6 x 7
>   attention   sex blueberries    iq   age sleep memory_score
```

```
>       <dbl> <dbl>       <dbl> <dbl> <dbl> <dbl>       <dbl>
> 1      95.8     1         308  99.9  44.9  9.94        128.
> 2      66.7     1         270 137.   29.4  8.04        127.
> 3     102.      1         442 110.   31.9 11.0         118.
> 4      36.9     1         219 110.   27.9  5.28         95.5
> 5      91.7     0         450 119.   36.7  9.30        122.
> 6     146.      1         255  85.6  23.9  7.05        102.
```

**About the data:**

- **attention**: sustained attention score (higher = better attention)
- **sex**: 0 = female, 1 = male
- **blueberries**: average number of blueberries consumed per year
- **iq**: the individual's IQ
- **age**: age of person in years
- **sleep**: average hours of sleep per night
- **memory_score**: memory test score

The researcher wants to test whether `attention` and `sleep` predict `memory_score`. She is already aware of a well-established finding in the literature, which is that `iq` and `age` predict `memory_score`.

She therefore wants to use a hierarchical regression approach to determine whether `attention` and `sleep` explain additional variance in `memory_score` *over and above* `iq` and `age`.

1. First, fit a linear model to determine the extent to which `memory_score` is predicted by `iq` and `age`. Store the results in `memory1`.

Try first, then click to see the code

```
# specify the baseline model
memory1 <- lm(memory_score ~ iq + age, data = memory_data)

# see the model results
summary(memory1)
>
> Call:
> lm(formula = memory_score ~ iq + age, data = memory_data)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -44.154 -11.754   0.732  11.608  40.790
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept)  71.1669     9.0796   7.838 1.67e-13 ***
> iq            0.1073     0.0699   1.534    0.126
> age           0.8220     0.1461   5.627 5.27e-08 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 16.1 on 231 degrees of freedom
```

```
> Multiple R-squared:  0.1303,  Adjusted R-squared:  0.1228
> F-statistic: 17.31 on 2 and 231 DF,  p-value: 9.875e-08
```

2. Next, add `attention` and `sleep` to the model, storing your results in `memory2`.

Try first, then click to see the code

```
# specify the next model
memory2 <- lm(memory_score ~ iq + age + attention + sleep, data = memory_data)

# show the results
summary(memory2)
>
> Call:
> lm(formula = memory_score ~ iq + age + attention + sleep, data = memory_data)
>
> Residuals:
>     Min      1Q  Median      3Q     Max
> -28.935  -8.555   1.713   8.450  31.384
>
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept)  9.60112    8.57889   1.119 0.264246
> iq           0.18673    0.05451   3.426 0.000726 ***
> age          0.86579    0.11308   7.656 5.32e-13 ***
> attention    0.22894    0.02757   8.302 8.88e-15 ***
> sleep        3.68609    0.39328   9.373  < 2e-16 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 12.46 on 229 degrees of freedom
> Multiple R-squared:  0.4839,  Adjusted R-squared:  0.4749
> F-statistic: 53.68 on 4 and 229 DF,  p-value: < 2.2e-16
```

3. Now, compare the `memory1` and `memory2` models using `anova()`

Try first, then click to see the code

```
anova(memory1, memory2)
> Analysis of Variance Table
>
> Model 1: memory_score ~ iq + age
> Model 2: memory_score ~ iq + age + attention + sleep
>   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
> 1    231 59912
> 2    229 35554  2     24359 78.447 < 2.2e-16 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

**Answer the following questions:**

- A model with `iq` and `age` as predictors explains  % of the variance in `memory_scores`

- A model with `iq`, `age`, `attention` and `sleep` as predictors explains  % of the variance in `memory_scores`
- Calculate the additional variance explained by the second model: Change in $R^2 =$  %
- The ANOVA comparing models can be reported as: $F(,) = , p < .001$.
- Is there a statistically significant improvement in the prediction of `memory_scores` as a result of adding `attention` and `sleep` to the model? no yes

**Now use Bayes Factors to determine how much more likely the `memory2` model is than the `memory1` model .**

Try first, click here for a reminder of the steps

- Determine the Bayes Factor for `memory1`
- Determine the Bayes Factor for `memory2`
- Compare the Bayes Factors for `memory2` and `memory1`

Try first, click here to see the code

```
# Store the Bayes Factor for the first model in memory1.BF
memory1.BF <- lmBF(memory_score ~ iq + age, data = as.data.frame(memory_data) )

# Store the Bayes Factor for the second model in memory2.BF
memory2.BF <- lmBF(memory_score ~ iq + age + attention + sleep, data = as.data.frame(memory_data) )

# Compute the Bayes Factors for memory2.BF vs memory1.BF
memory2.BF / memory1.BF
> Bayes factor analysis
> --------------
> [1] iq + age + attention + sleep : 4.168455e+23 ±0%
>
> Against denominator:
>   memory_score ~ iq + age
> ---
> Bayes factor type: BFlinearModel, JZS
```

**Answer the following questions:**

- The Bayes Factor comparing `memory2` and `memory1` to (2 decimal places) is  e+ .
- Does the Bayes Factor support the conclusions from the ANOVA? no yes

Click for answer

Yes! The Bayes Factor is equal to $4.17 \times 10^{23}$, and this therefore strongly supports the inclusion of `attention` and `sleep` in the model already containing `iq` and `age`.

**Extra exercises, if there's time**

**1.**

The researcher wishes to predict the `memory_score` for a new individual with `iq` = 105, `age` = 27, `attention` = 90, `sleep` = 8. Determine the prediction.

Hint: in a previous session, you have previously used the `predict()` function to do this.

- The predicted `memory_score` is

Try first, then click to show the code for the answer

```
# create tibble for the new data
new_data <- tibble(iq = 105, age = 27, attention = 90, sleep = 8)

# use predict to derive prediction from new data
predict(memory2, new_data)
>        1
> 102.6768
```

**2.**

The researcher is interested to know whether annual consumption of blueberries has any bearing on `memory_scores`, and so wants to add `blueberries` to the model in `memory2`.

Determine the Bayes Factor comparing `memory2` with a model that additionally contains `blueberries`.

- The Bayes Factor for the model comparison is  (to 2 decimal places)
- The Bayes Factor indicates that the model with `blueberries` is more likely less likely than the model without it.
- Should the researcher add `blueberries` to the model? no yes if it tastes good

Try yourself first, then click for the code

```
# add blueberries to memory2; store in memory3.BF
memory3.BF <- lmBF(memory_score ~ iq + age + attention + sleep + blueberries, data = as.data.frame(memo

# calculate the BF for memory3 vs memory2
memory3.BF / memory2.BF
> Bayes factor analysis
> --------------
> [1] iq + age + attention + sleep + blueberries : 0.1663574 ±0%
>
> Against denominator:
>   memory_score ~ iq + age + attention + sleep
> ---
> Bayes factor type: BFlinearModel, JZS
```

### Summary of key points

- We can compare a model with one that has more predictors by using `anova(model1, model2)`.
- We can compare models using Bayes Factors with `lmBF` in the `BayesFactor` package.
- A **Bayes Factor** is probability of one model relative to another, *given the data*.
- To compare Bayes Factors of models:
  - First obtain the Bayes Factors for `model1` and `model2`.
  - Then use `model2 / model1` to get the Bayes Factor, indicating how much more likely `model2` is.
- Bayes Factors less than 1 indicate evidence for `model1`
- Bayes Factors greater than 1 indicate evidence for `model2`
- We can report Bayes Factors as $BF_{10} = 2.23$ (or BF10 = 2.23)

Next week's session will build on what was done in this session, so make sure you understand what was covered and ask the instructor to clarify anything you're unsure of.

## Fitting curves

#### In brief

> So far all our regression models have assumed that our variables have ***linear relationships***. That isn't always the case, and sometimes we need to fit curved lines to describe the relationship of predictors and outcomes. As we saw before, fitting curved lines has costs as well as benefits: A curved line is more likely to **overfit** the data, and may be less good at predicting new data. But for some models curved lines are essential to describe the world as it really is.

### Using polynomials to fit curves

- **Slides from the session** (available 02-02-2020)

#### Overview

In this session we will:

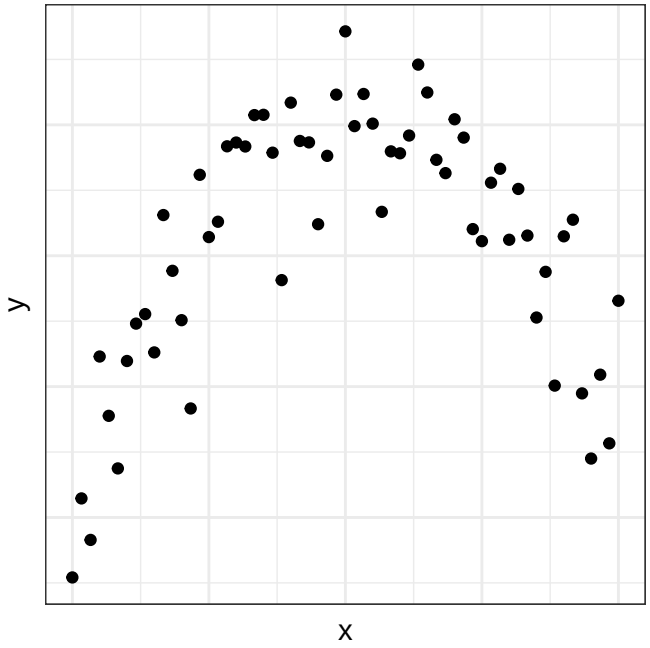
- See how we can add *polynomial terms* such as $x^2$, $x^3$ to a regression model to capture non-linear relationships.
- Use ANOVA and Bayes Factors to determine whether these terms improve the model.

You should be comfortable with what we did in the previous **Building Models 1** and **Building Models 2** sessions before attempting this one.

## Polynomials

The regression models we have been fitting assume a **linear** (i.e., straight line) relationship between variables. However, variables may not always be related in a linear fashion.

Suppose variables x and y showed the following trend:




It is clear that this relationship would not be explained well by a straight line. We'd lose important information about the relationship if we only fit a straight line. A curve would be better.

We can fit a curve to the data by adding **polynomial** terms to the regression equation.

**Polynomial** means that a variable is raised to a particular power. For example:

- $x^2$ means x-squared, which is x-multiplied-by-x, or "x to the power of two"
- $x^3$ means x-cubed, which is x-multiplied-by-x-multiplied-by-x, or "x to the power of three"

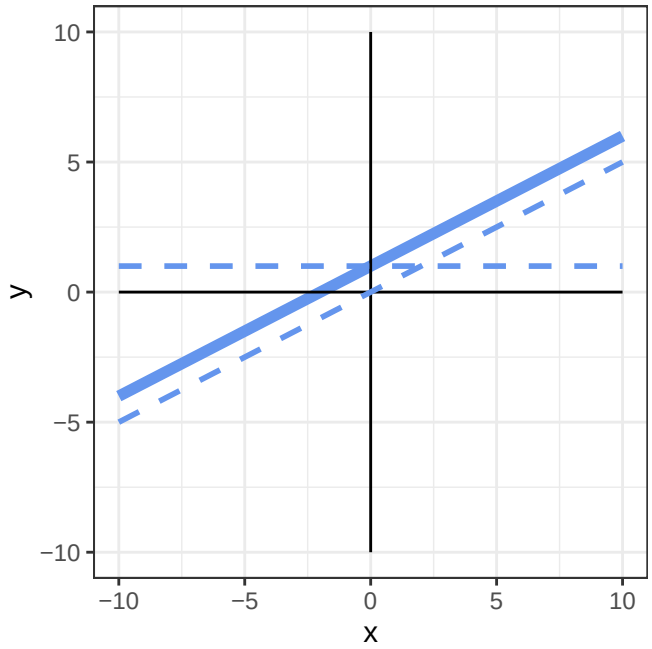
If a model has a **quadratic component** it means it has an $x^2$ term in the equation.

If a model has a **cubic component** it means it has an $x^3$ term in the equation.

To see why this approach works, recall that lines can be represented by equations.

### Components of a regression line

The equation $y = 1 + 0.5x$ would be represented as follows:

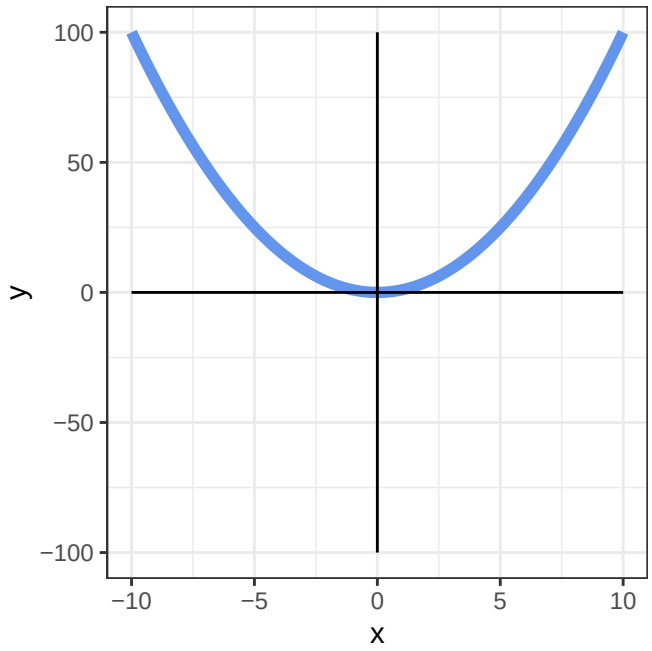



We can think of this line as being made up of the **constant** and a **linear** component.

- The **constant** in this equation is indicated by the dashed horizontal line.
- The **linear** component to this equation 0.5x is indicated by the dashed slope line.
- The solid blue line is a *combination* of these two components.

### Quadratic

The equation $y = x^2$ would be represented as follows:

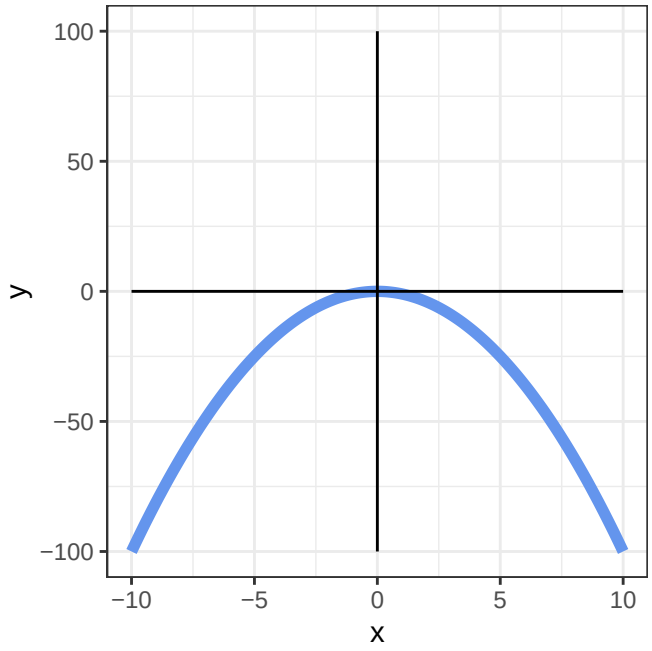



To get each value of y, we square the value of x.

So, when x = -5, y is 25.

And if x = -4, y = 16, and so on…

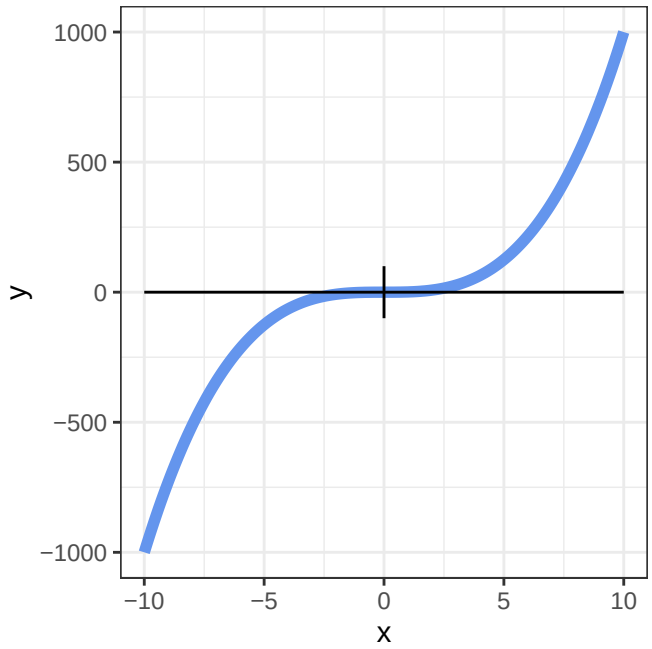
$y = -x^2$, would look as follows:




Curves with quadratic components have **one bend**.

**Cubic**

The equation $y = x^3$ would be represented as follows:

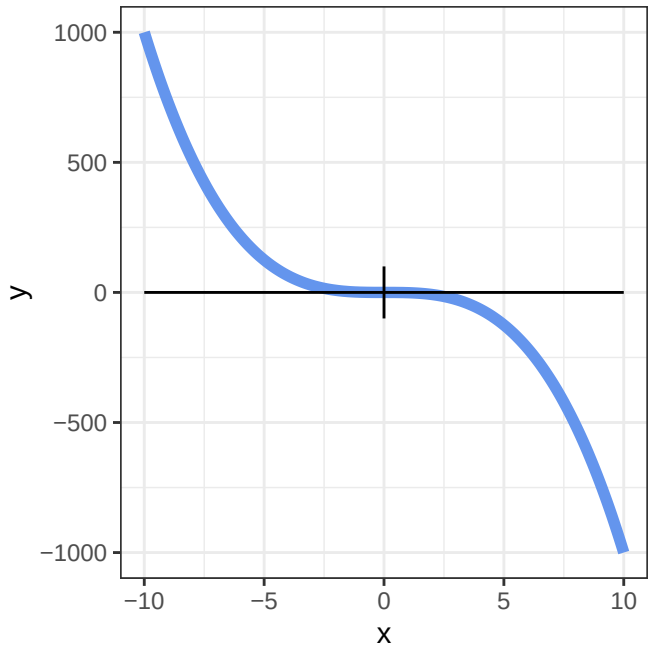



To get each value of y, we *cube* the value of x.

So, when x = -5, y is -125.

And if x = 10, y = 1000, and so on…

$y = -x^3$, would look as follows:

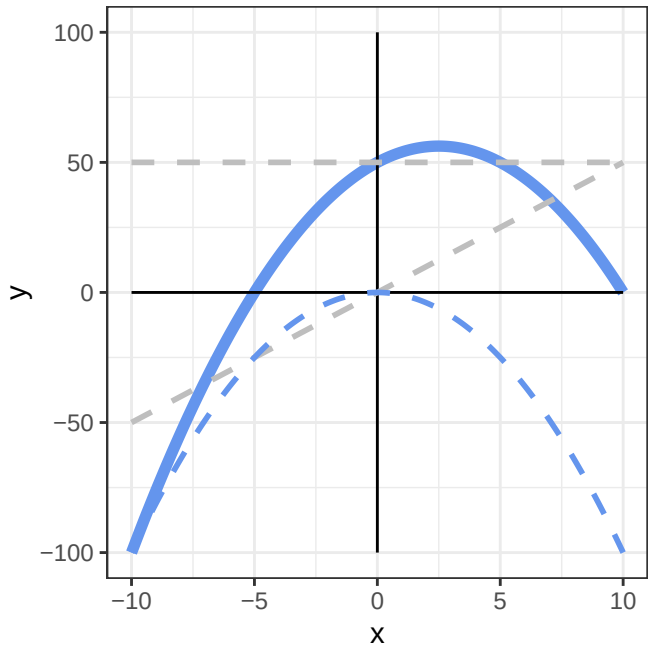



Curves with cubic components have **two bends**.

### Linear plus quadratic components

The equation $y = 50 + 5x - x^2$ has

- a constant equal to **50**
- a linear component **5x**
- a quadratic component $-x^2$:



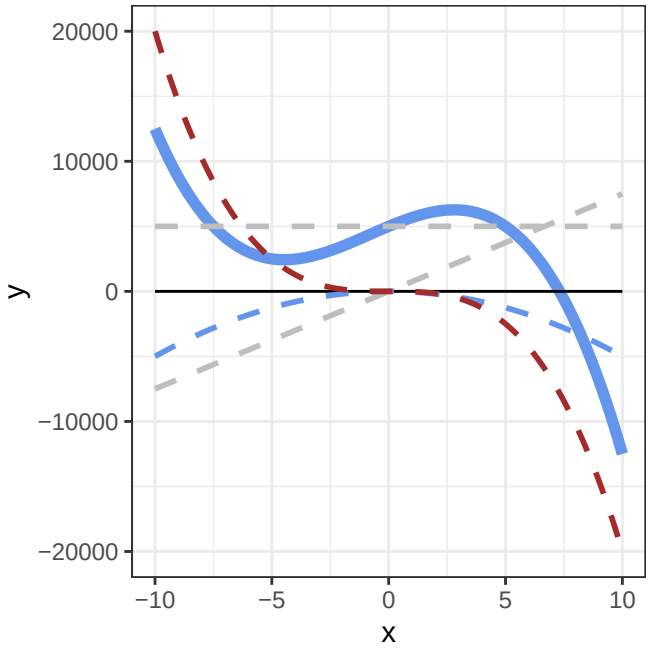

The dashed lines on the plot indicate the *intercept*, *linear component*, and *quadratic component* of the

equation. The solid line represents the equation.

**Linear + quadratic + cubic components**

The equation $y = 5000 + 750x - 50x^2 - 20x^3$ has

- a constant equal to **5000**
- a linear component **750x**
- a negative quadratic component $-50x^2$
- a negative cubic component $-20x^3$




The dashed lines in the plot indicate the different components of the curve indicated by the solid blue line.

When we see any curve, it is possible to think of it as being composed of components like this. In theory, we can keep adding components, but it's rare to see even higher-order components (e.g., $x^4$) being added. Issues regarding overfitting and generalisability can also arise (mentioned in the slides).

## Identifying polynomial components

To determine whether a model should have quadratic, cubic, or higher order components, we can use the **sequential regression** approach covered in the previous session. We take the following steps, and look at the change in $R^2$ associated with each step.

- First fit the **linear** model
- then test for the addition of the **quadratic** ($x^2$) component
- then test for the addition of the **cubic** ($x^3$) component

Let's see this in action!

**Learning tip**

Try storing all your code in an **R Markdown** file today if you are not doing so already! You can use code chunks and write text to describe each chunk as was described in the slides.

We'll use a dataset inspired by a 2016 survey of the National Office for Statistics. They investigated happiness across the life span. Approximately 300,000 individuals of all ages answered questions related to well-being.

Each participant answered the following question regarding their *happiness*:

**"Overall, how happy did you feel yesterday?**
**Where 0 is 'not at all happy' and 10 is 'completely happy'."**

This `happy` dataset are located at "https://bit.ly/2uIxM5K".

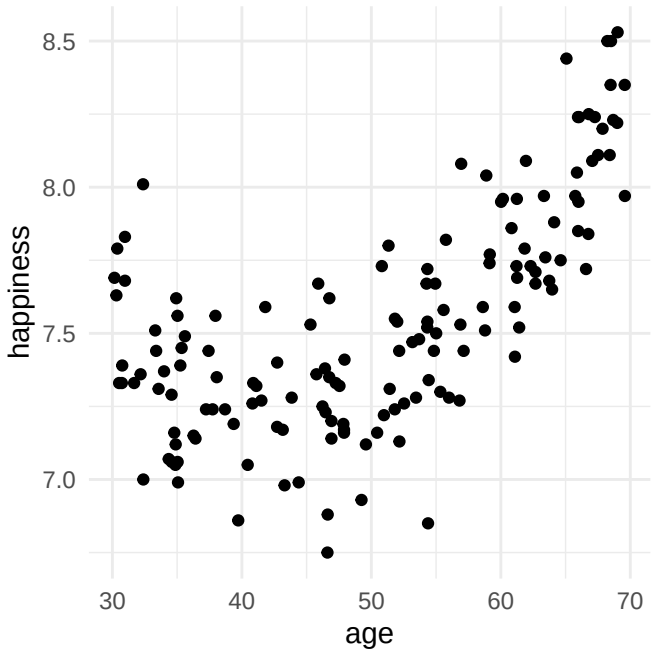
Let's load the data into R, and preview the data using `head`:

```
# load the data
SurveyData <- read_csv("https://bit.ly/2uIxM5K")

# preview it
SurveyData %>% head()
> # A tibble: 6 x 3
>     age happiness anxiety
>   <dbl>     <dbl>   <dbl>
> 1  66.0      7.85    2.33
> 2  35.0      7.56    2.58
> 3  58.6      7.59    3.43
> 4  35.0      7.06    1.67
> 5  60.2      7.96    2.13
> 6  67.5      8.11    1.09
```

Plot the relationship between `age` and `happiness`:

```
SurveyData %>%
  ggplot(aes(x=age, y=happiness)) +
  geom_point()
```




If you had to guess from the plot, which components seem to be present in the relationship between `happiness` and `age`?

Linear: no possibly

Quadratic: no yes

Cubic: no yes

Try to describe the relationship between `happiness` and `age`.

Try first, click here for a description

Happiness of individuals appears to decline from 30 years to the late forties. Happiness then increases beyond the late forties, reaching its peak at 70 years, at which age people reported the highest levels of happiness - higher even than levels shown in early thirties.

### Linear component

To determine whether there is a linear component, run a simple regression with `happiness` as the outcome variable and `age` as the predictor:

```
polynomial1 <- lm(happiness ~ age, data = SurveyData)
summary(polynomial1)
>
> Call:
> lm(formula = happiness ~ age, data = SurveyData)
>
> Residuals:
>      Min       1Q   Median       3Q      Max
> -0.78019 -0.16858 -0.04762  0.19811  0.84368
>
```

```
> Coefficients:
>             Estimate Std. Error t value Pr(>|t|)
> (Intercept) 6.484101   0.102340   63.36   <2e-16 ***
> age         0.021076   0.001979   10.65   <2e-16 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 0.2912 on 148 degrees of freedom
> Multiple R-squared:  0.4339,  Adjusted R-squared:  0.4301
> F-statistic: 113.5 on 1 and 148 DF,  p-value: < 2.2e-16
```

**Explanation:** The linear model is stored in `polynomial1`. `summary` displays the results.

What percentage of the variance in `happiness` scores is explained by `age`? 0.44 43.39 29.12 %

Is `age` a statistically significant predictor of `happiness` no yes

The linear model does okay, but remember it is only fitting a straight line through our data, which appear to show a curved relationship!

### Adding a quadratic component

We can add a quadratic component to the regression model using `poly()`. If we type `poly(age, 2)` when specifying the model, the '2' in the `poly()` function tells R that we want to fit a model with **both** linear and quadratic components of `age`. This is the model it'll fit:

$$predicted\ happiness = a + b_1(age) + b_2(age^2)$$

where $a$ is the intercept, and $b_1$ and $b_2$ are the coefficients for the linear and quadratic components, respectively.

```
polynomial2 <- lm(happiness ~ poly(age,2), data = SurveyData)
summary(polynomial2)
>
> Call:
> lm(formula = happiness ~ poly(age, 2), data = SurveyData)
>
> Residuals:
>      Min       1Q   Median       3Q      Max
> -0.58896 -0.12752 -0.02333  0.13274  0.59724
>
> Coefficients:
>              Estimate Std. Error t value Pr(>|t|)
> (Intercept)    7.54433    0.01779  424.06   <2e-16 ***
> poly(age, 2)1  3.10223    0.21789   14.24   <2e-16 ***
> poly(age, 2)2  2.36118    0.21789   10.84   <2e-16 ***
```

```
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 0.2179 on 147 degrees of freedom
> Multiple R-squared:  0.6853,  Adjusted R-squared:  0.681
> F-statistic: 160.1 on 2 and 147 DF,  p-value: < 2.2e-16
```

**Explanation of the code:** We've told R we want to add a quadratic component to the model by using `happiness ~ poly(age, 2)`.

**Explanation of the output:** You will see in the output separate coefficient estimates for `poly(age, 2)1` and `poly(age, 2)`. These are the estimates of the coefficients for the linear and quadratic components of `age` (i.e., $b_1$ and $b_2$ in the equation above).

What percentage of the variance in `happiness` does a model with a **quadratic component** of `age` explain? ____ %

Compare the value of $R^2$ in `polynomial1` and `polynomial2`.

- Does the addition of a quadratic component result in a numerical increase in $R^2$ in `polynomial2`? yes no
- What is the change in $R^2$? ____ % (to 2 decimal places)

Click to see how the answer is calculated

$R^2$ change from `polynomial1` to `polynomial2` = 68.53 - 43.39 = 25.14%

Therefore, the model with the quadratic component of `age` accounts for **25.14% more variance** in `happiness` than the model with only a linear component.

We can test whether the ***increase* in $R^2$** in `polynomial2` represents a statistically significant increase by comparing `polynomial1` and `polynomial2` using `anova`:

```
anova(polynomial1, polynomial2)
> Analysis of Variance Table
>
> Model 1: happiness ~ age
> Model 2: happiness ~ poly(age, 2)
>   Res.Df     RSS Df Sum of Sq      F    Pr(>F)
> 1    148 12.5542
> 2    147  6.9791  1    5.5752 117.43 < 2.2e-16 ***
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Is the increase in $R^2$ associated with the addition of the quadratic component statistically significant? yes no

Answer

Yes. We can report the improvement in fit as follows:

A model with a quadratic component of `age` accounted for a statistically significantly greater proportion of variance in `happiness` than a model with only a linear component, $F(1, 147) = 117.43$, $p < .001$.

### Adding a cubic component

Now we'll test for a cubic component.

```
polynomial3 <- lm(happiness ~ poly(age,3), data = SurveyData)
summary(polynomial3)
>
> Call:
> lm(formula = happiness ~ poly(age, 3), data = SurveyData)
>
> Residuals:
>      Min       1Q   Median       3Q      Max
> -0.60468 -0.14165 -0.01844  0.13839  0.58176
>
> Coefficients:
>               Estimate Std. Error t value Pr(>|t|)
> (Intercept)    7.54433    0.01777 424.447   <2e-16 ***
> poly(age, 3)1  3.10223    0.21769  14.251   <2e-16 ***
> poly(age, 3)2  2.36118    0.21769  10.846   <2e-16 ***
> poly(age, 3)3 -0.24530    0.21769  -1.127    0.262
> ---
> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
>
> Residual standard error: 0.2177 on 146 degrees of freedom
> Multiple R-squared:  0.688,   Adjusted R-squared:  0.6816
> F-statistic: 107.3 on 3 and 146 DF,  p-value: < 2.2e-16
```

The '3' in `poly(age,3)` tells R that we want to specify a model with linear, quadratic *and* cubic components, of the form:

$$happiness = a + b_1(age) + b_2(age^2) + b_3(age^3)$$

What percentage of the variance in `happiness` does a model with a **cubic component** of `age` explain? %

Compare the value of $R^2$ in `polynomial3` and `polynomial2`.

- Does the addition of a cubic component result in a numerical increase in $R^2$ in `polynomial3`? yes no
- What is the increase in $R^2$ as a result of adding in the cubic component? (Compare $R^2$ between `polynomial3` and `polynomial2`).

  The increase in $R^2$ is %

To determine if the increase in $R^2$ is statistically significant, we can again use `anova`:

```
anova(polynomial2, polynomial3)
> Analysis of Variance Table
>
> Model 1: happiness ~ poly(age, 2)
> Model 2: happiness ~ poly(age, 3)
>   Res.Df    RSS Df Sum of Sq      F Pr(>F)
> 1    147 6.9791
> 2    146 6.9189  1  0.060171 1.2697 0.2617
```

Is the increase in $R^2$ associated with the addition of a cubic component statistically significant? no yes

Description of the answer

The `anova` comparing `polynomial3` and `polynomial2` is not statistically significant, $F(1, 146) = 1.27$, $p = .26$, indicating that the addition of the cubic component of `age` into the regression model does not increase the variance in `happiness` explained.

On the basis of the tests conducted so far, which model should be preferred? One with:

a linear component of age only linear and quadratic components of age linear, quadratic, and cubic components of age

Explain

Our analyses suggest that a model with a quadratic component of `age` (i.e., the model in `polynomial2`) is sufficient to explain the data.

#### A note about `poly()`

`poly` automatically creates polynomial terms for us. The polynomials it creates are actually a special type, called **orthogonal** polynomials. This means that the polynomials are not correlated with one another. For example, the correlation between the *age* and $age^2$ components created by `poly` is zero . Likewise, the correlation betweem $age^2$ and $age^3$ components created by `poly` is also zero.

This is desirable because if the components were not **orthogonalised**, they'd be highly correlated with each other. That is, the raw scores for *age* and $age \times age$ are likely to be highly correlated. As we covered in the first Building Models 1 session, high correlations between our predictors is undesirable as it can lead to **multicolinearity**.

## Bayesian approach

As we did in the previous session, we can use Bayes Factors to compare the models with different polynomial components.

#### Preparations

Unfortunately, `poly()` does not work seamlessly with `lmBF`, as it did with `lm`. Instead, we need to create separate variables in `SurveyData` for the quadratic and cubic components before we work out the Bayes Factors with `lmBF`.

To add the quadratic component to `SurveyData`:

```
SurveyData <-
  SurveyData %>% mutate( age2 = poly(age,2)[,"2"] )
```

**Explanation of the code:** The code takes `SurveyData`, then uses `mutate` to add a new variable `age2` to the dataset. `age2` contains the quadratic component of `age`, created by `poly(age,2)[,"2"]`.

We can see the new variable `age2` when we look at `SurveyData` again:

```
SurveyData %>% head()
> # A tibble: 6 x 4
>     age happiness anxiety    age2
>   <dbl>     <dbl>   <dbl>   <dbl>
> 1  66.0      7.85    2.33  0.0761
> 2  35.0      7.56    2.58  0.0483
> 3  58.6      7.59    3.43 -0.0440
> 4  35.0      7.06    1.67  0.0481
> 5  60.2      7.96    2.13 -0.0246
> 6  67.5      8.11    1.09  0.110
```

Now create the variable for the **cubic component**:

```
SurveyData <-
  SurveyData %>% mutate( age3 = poly(age,3)[,"3"] )
```

**Explanation of the code:** As before, the code takes `SurveyData`, then uses `mutate` to add a new variable `age3` to the dataset. `age3` contains the quadratic component of `age`, created by `poly(age,3)[,"3"]`.

Again, we can see the new variable `age3` when we look at `SurveyData`:

```
SurveyData %>%  head()
> # A tibble: 6 x 5
>     age happiness anxiety    age2     age3
>   <dbl>     <dbl>   <dbl>   <dbl>    <dbl>
> 1  66.0      7.85    2.33  0.0761  0.00888
> 2  35.0      7.56    2.58  0.0483  0.0353
> 3  58.6      7.59    3.43 -0.0440 -0.0988
> 4  35.0      7.06    1.67  0.0481  0.0356
> 5  60.2      7.96    2.13 -0.0246 -0.0974
> 6  67.5      8.11    1.09  0.110   0.0707
```

### Derive the Bayes Factors

First, make sure the BayesFactor package is loaded (`library(BayesFactor)`). We can use `lmBF` to get the

Bayes Factor for each model, as we did in the previous session.

To derive the Bayes Factor for `polynomial1`:

```
polynomial1BF <- lmBF(happiness ~ age, data = as.data.frame(SurveyData) )
```

To derive the Bayes Factor for `polynomial2`:

```
# store the Bayes Factor for polynomial2
polynomial2BF <- lmBF(happiness ~ age + age2, data = as.data.frame(SurveyData) )
```

**Explanation:** With `lmBF` we need to specify the polynomial equation with both linear and quadratic components separately, hence `happiness ~ age + age2`.

The Bayes Factor comparing `polynomial2` and `polynomial1` is then:

```
polynomial2BF / polynomial1BF
> Bayes factor analysis
> --------------
> [1] age + age2 : 2.618277e+17 ±0.01%
>
> Against denominator:
>   happiness ~ age
> ---
> Bayes factor type: BFlinearModel, JZS
```

How many more times likely is a model with a **quadratic component** of `age` than one with only a **linear component**? 2.62 2.62e-17 2.62e+17

Does this constitute strong evidence for the addition of a quadratic component? yes no

Explain why

The Bayes Factor tells us that a model with a quadratic component of `age` is 2.62e+17 or $2.62 \times 10^{17}$ times more likely than one that simply contains a linear component. This is very strong evidence for the inclusion of a quadratic component of `age` in the model.

Next, determine the Bayes Factor for `polynomial3`:

```
polynomial3BF <- lmBF(happiness ~ age + age2 + age3, data = as.data.frame(SurveyData) )
```

**Explanation:** Again, we need to specify the polynomial equation with linear, quadratic, and cubic components separately, hence `happiness ~ age + age2 + age3`.

Compare `polynomial3BF` and `polynomial2BF`:

```
polynomial3BF / polynomial2BF
> Bayes factor analysis
```

```
> --------------
> [1] age + age2 + age3 : 0.1631281 ±0%
>
> Against denominator:
>   happiness ~ age + age2
> ---
> Bayes factor type: BFlinearModel, JZS
```

How many more times likely is a model with a **cubic component** than one with only **linear and quadratic components**?

Does this constitute strong evidence for the inclusion of a cubic component in the model? yes no

Explain why

The Bayes Factor tells us that a model with a cubic component of `age` is only 0.16 times more likely than one that contains both linear and quadratic components. Because the Bayes Factor is less than 0.33, this constitutes strong evidence for the simpler model with only linear and quadratic components.

On the basis of the model comparison with Bayes Factors, which model should be preferred? One with:

a linear component of age only linear and quadratic components of age linear, quadratic, cubic components of age

A comparison of Bayes Factors agrees with the comparison of the models with `anova`. There's strong evidence that the relationship between `age` and `happiness` contains both linear and quadratic components of `age`. There was no evidence for a cubic component.

## Exercise

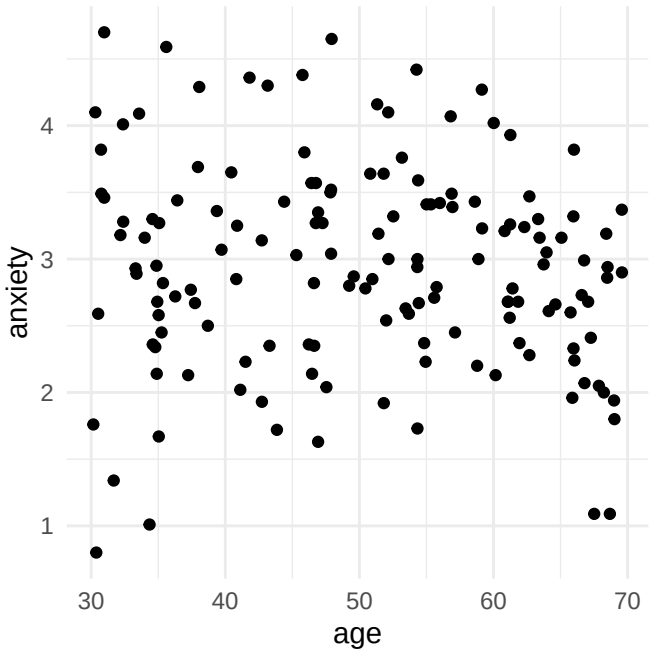
**Now you try incorporating polynomials to a regression, and do so by investigating the relationship between `age` and `anxiety` in `SurveyData`.**

The column `anxiety` in `SurveyData` contains responses to the question:

**"Overall, how anxious did you feel yesterday? Where 0 is 'not at all anxious' and 10 is 'completely anxious'."**

- Create a scatterplot of `age` vs. `anxiety`. Does there appear to be a linear or non-linear relationship?

Try to create the plot yourself first. Click to show the code




What kind of relationship between `age` and `anxiety` do you think is present?

Try yourself first, then click to see answer

A slight **bow** is evident in the plot such that `age` and `anxiety` seem to follow an inverted U-shaped relationship.
Reported `anxiety` levels increase from 30 years to middle age (approx. 50 years) and then declines from 50 to 70 years. This mirrors the relationship with `age` and `happiness`. `anxiety` is greatest when `happiness` seems lowest.

**Answer the following questions:**

**Linear component**

- What percentage of the variance in `anxiety` is explained by a model with `age` as predictor? %

**Quadratic component**

- What percentage of the variance in `anxiety` is explained by a model containing both linear and quadratic components of `age` as predictors? %
- What is the *increase* in $R^2$ if a quadratic component of `age` is added to the model? %
- Does this increase represent a statistically significant increase? yes no
- What is the $F$-statistic comparing the model with a linear component vs. one with linear and quadratic components? $F(1, 147) =$ , $p = .009$

**Cubic component**

- What percentage of the variance in `anxiety` is explained by a model containing both linear, quadratic and cubic components of `age` as predictors? %
- What is the *increase* in $R^2$ if a cubic component of `age` is added to the model? %
- Does this increase represent a statistically significant increase? yes no
- What is the $F$-statistic and $p$-value for the test of the model with a linear + quadratic component vs. linear + quadratic + cubic components? $F(1, 146) =$ , $p =$

**Decision** - On the basis of the model comparison with ANOVA, which model should be preferred? linear component of age only linear and quadratic components of age linear, quadratic, cubic components of age

Show me the code to do all of this

```
# fit a linear model, show results
anx1 <- lm(anxiety ~ age, data = SurveyData)
summary(anx1)

# fit a quadratic component, show results
anx2 <- lm(anxiety ~ poly(age,2), data=SurveyData)
summary(anx2)

# compare linear and linear+quadratic models
anova(anx1, anx2)

# fit a cubic component
anx3 <- lm(anxiety ~ poly(age,3), data=SurveyData)
summary(anx3)

# compare (linear + quadratic) and (linear + quadratic + cubic) models
anova(anx2, anx3)
```

**Now use Bayes Factors:**

(Note: you do not need to re-add the quadratic and cubic components of age to `SurveyData`, as we did this before. These should still be in `SurveyData` as `age2` and `age3`.)

- The Bayes Factor comparing a model with linear and quadratic components vs. one with a linear component only is
- This indicates that there is more evidence for which model? linear component only linear plus quadratic components

- The Bayes Factor comparing a model with linear, quadratic and cubic components vs. one with linear and quadratic components only is
- This indicates that there is more evidence for which model? linear plus quadratic component linear, quadratic, and cubic components

Do the comparisons of models with Bayes Factors agree with the conclusions made with anova? no yes

Show me the code to determine the Bayes Factors

```
library(BayesFactor)

# BF for model anx1
anx1BF <- lmBF(anxiety ~ age, data = as.data.frame(SurveyData) )

# BF for model anx2
anx2BF <- lmBF(anxiety ~ age + age2, data = as.data.frame(SurveyData) )

# BF for model anx3
anx3BF <- lmBF(anxiety~ age + age2 + age3, data = as.data.frame(SurveyData) )

# compare BFs for anx2 and anx1
anx2BF / anx1BF
> Bayes factor analysis
> --------------
> [1] age + age2 : 5.476443 ±0%
>
> Against denominator:
>   anxiety ~ age
> ---
> Bayes factor type: BFlinearModel, JZS

# compare BFs for anx3 and anx2
anx3BF / anx2BF
> Bayes factor analysis
> --------------
> [1] age + age2 + age3 : 0.7556007 ±0%
>
> Against denominator:
>   anxiety ~ age + age2
> ---
> Bayes factor type: BFlinearModel, JZS
```

## Summary of key points

- Polynomial terms (e.g., $x^2$, $x^3$) can be added to regression models to fit curves in our data.

- `poly(predictor name, X)` can be used with `lm` to specify models with polynomial terms of the Xth order.
  - The improvement in fit ($R^2$) as a result of adding in a polynomial term can be tested using `anova(polynomial1, polynomial2)`.

- Bayes Factors can also be used to compare models with polynomial terms using `lmBF`.
  - You must store the polynomial components in the dataset first before using `lmBF`. Use `poly(predictor name, X)[,"X"]`, where X is the order of the polynomial you will test (e.g., `poly(age, 3)[,"3"]`).

- **A note of caution**: Although curves of any complexity can be fit, it may not always be meaningful or parsimonious to do so.
  - Complex models may *overfit* the data and may not necessarily generalise to new datasets well.
  - It is also important not to extrapolate beyond the range of data used to generate the model when making predictions from the model, as the same relationship may not be present.

# Testing

### In brief

A single statistical model can test many different hypotheses. Sometimes these hypotheses can, superficially, sound similar—but selecting the relevant test and reporting it correctly can sometimes be a challange. R packages exist to make specifying and reporting tests easier, but none can automate the process: Testing hypotheses always requires thought about both the research question in hand alongside statistical issues. The replication crisis has brought renewed focus on the pitfalls of multiple testing and researcher degrees of freedom. Both technical strategies and research policies can mitigate this risk to some degree, but research integrity is crucial.

## Session 1

Types of test to cover:

### Multiple parameters

- Interactions with F
- Interactions with BF model comparison
- Multiple (blocked) predictors

### Group/cell comparisons

- Pairwise comparisons
- Cell vs mean

### Specific predictions

- Ordered differences with BF

### Planned contrasts Using BF

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(BayesFactor)
```

```
## Loading required package: coda
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## ************
## Welcome to BayesFactor 0.9.12-4.2. If you have questions, please contact Richard Morey (richarddmorey
##
## Type BFManual() to open the manual.
## ************
```

```
disgust_data = read.table(url('http://www.learnbayes.org/disgust_example.txt'),header=TRUE)
bf1 = anovaBF(score ~ condition, data = disgust_data)

bf_for_null <- as.vector(1 / bf1)

samples.df = posterior(bf1, iterations = 100000) %>% as_tibble() %>%
  rename(Con = `condition-control`,
         Lem = `condition-lemon`,
         Sul = `condition-sulfur`)
```

```
## Warning: Calling `as_tibble()` on a vector is discouraged, because the behavior is likely to change i
## This warning is displayed once per session.
```

```
samples.df %>%
  mutate(rightorder = (Con > Lem) & (Sul > Con)) %>%
  summarise( mean(rightorder) / (1/6) )
```

```
## # A tibble: 1 x 1
##   `mean(rightorder)/(1/6)`
##                      <dbl>
## 1                     4.33
```

```
4.33 / bf_for_null
```

```
## Intercept only
##       3.350625
```

### With emmeans

In this session we primarily cover the **emmeans** library and ways to test planned contrasts in an experimental design.

# Communicating results

**In brief**

> Communicating research findings is an art and a science, but professional standards and integrity demands that we are open, honest and sufficiently skeptical in our reporting of our own data. In academic publishing there are a number of common patterns which can helpful in structuring results sections, breaking-down a complex set of findings and easing interpretation.

## Session 1: Communicating Anova and Regression

In this session we practice analysing and writing up real-world examples, compare our work with published findings, and reflect on how our communication might be improved.

# Real world data

**In brief**

> In the real world datasets are never neat and tidy, and findings rarely cut-and-dried. Dealing with complexity and ambiguity is part of the role of the scientist. In these final sessions and the assessment we apply the skills we have learnt to real world examples.

## Session 1: Reproducing a real paper

In this session we dissect a specific published paper, attempt to replicate the authors findings, and reflect on the challenges of reproducing findings given the current state of scientific publishing.

**The assessment** Read the assessment. Discuss in groups what you think is necessary for each part of the assessment. What sort of work is going to be required for each part? Make notes on any points of disagreement or different interpretations and collate these.

**Your paper** Work in groups of 3 and give a short summary of each of your papers to the others in the group. If there are people in your group without their own paper, discuss the process of finding it and what worked for you.

Reflect on the process of reading statistical methods sections in detail. Is this something you do regularly? What did you take away from the experience this time?

### Questions

- What are the primary research questions of the paper?
- What predictions (if any) do the authors make?

Complete the relevant columns in the xls file here: https://liveplymouthac-my.sharepoint.com/:x:/g/personal/ben_whalley_plymouth_ac_uk/EamnhLiSjGNJqtQa-ZKOwrsBqzFlvSqG25RnHT-wQTj6vw?e=GS6OIV

### Manipulation and measurement

- What measurements does the study make to answer these questions? What instruments/scales/other outcomes are there?
- What variables are used as predictors?
- What experimental manipulations are used? How many groupings are formed by group allocations?

- Can these measurements be used directly, or do they need to be aggregated summarised or processed in some way before they are informative?

Complete the relevant columns in the xls file.

**Statistical tests**

- What statistical techniques does your paper use to address the primary research questions?
- Have all of the techniques been covered in PSY753? If not, make specific notes of terms you are unfamiliar with, and/or would like to learn.
- Confer with your group. Sometimes the same technique can be given different names. See if you can collaborate to work out what techniques/models are being used.

Complete the relevant columns in the xls file.

**For reference,**

For reference, the fields in the spreadsheet are:

- First name
- Surname
- Email
- Programme
- Title of your chosen paper
- DOI or URL
- Brief summary of the paper
- Describe the primary outcome
- Was there an experimental manipulation? If so describe it briefly here.
- Describe any other predictor variables here.
- Describe/record predictions made in the introduction. Note if they are explicit or implicit.
- What statistical techniques are used in this paper? If you can describe them in terms of an `lm` model in R that is great, but don't worry if not - just describe what is done.
- Specifically, note any techniques you are not familiar with, or don't think have been covered so far in PSYC753. Include specific quotes from the methods section to illustrate.

I want everyone to have had a first-pass at completing the xls sheet before our next session.

## Session 2: Developing skills

In this session we will do additional work to build the skills you need to develop your own replication (responding to the gaps you have identified). You will build on skills learnt in previous weeks to wrangle the published data into a suitable format, plot the data, and decide which analyses to run for inferential tests.

**What is Ancova?**

Note: what follows are notes to support the talk I gave in-class. I don't expect you to work through these materials like a worksheet because the content is broadly similar to what has gone before — it's really the explanation that differs. There are specific exercises to complete at the end.

Anova is just a regular linear model (`lm`) with one or more categorical variables as predictors.

The example dataset I described was as follows:

> A student dissertation project investigated the analgesic quality of music during an experimental pain stimulus. Music was selected to be either *liked* (or disliked) by participants and was either *familiar* or unfamiliar to them. Pain was rated without music (`no.music`) and with music (`with.music`) using a 10cm visual analog scale anchored with the labels "no pain" and "worst pain ever".

I suggested we might run a linear model to predict pain scores when music was playing, including the two experimental variables:

```
# this library is useful to provide a summary of lm models, see below
library(pander)

# read the data in
painmusic <-  read_csv('data/painmusic.csv')
```

```
## Parsed with column specification:
## cols(
##   liked = col_character(),
##   familiar = col_character(),
##   no.music = col_double(),
##   with.music = col_double()
## )
```

```
# run a linear model
m.withmusic <- lm(with.music ~ liked * familiar, data=painmusic)

# show the output
m.withmusic %>%
  pander()
```

Table 8: Fitting linear model: with.music ~ liked * familiar

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **(Intercept)** | 5.5 | 0.3851 | 14.28 | 1.243e-26 |
| **likedLiked** | -0.75 | 0.5446 | -1.377 | 0.1713 |
| **familiarUnfamiliar** | 0.3571 | 0.5446 | 0.6558 | 0.5133 |
| **likedLiked:familiarUnfamiliar** | 0.1071 | 0.7701 | 0.1391 | 0.8896 |

The issue with the output is that we have 4 coefficients which don't map onto the questions we have. We want to know:

- Did familiarity make a difference
- Did liking make a difference
- Was there an *interaction* between familiarity and liking

To test this using frequentist methods we can use the `car::Anova` function:

```
car::Anova(m.withmusic, type=3)
```

```
## Anova Table (Type III tests)
##
## Response: with.music
##                Sum Sq  Df  F value Pr(>F)
## (Intercept)    847.00   1 204.0086 <2e-16 ***
## liked            7.88   1   1.8968 0.1713
## familiar         1.79   1   0.4301 0.5133
## liked:familiar   0.08   1   0.0194 0.8896
## Residuals      448.39 108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The `car::Anova` function is really just a special way of **displaying** results from a linear model. The

important part is still the model formula, `with.music ~ liked * familiar`.

### Why use `car::Anova()` and not `anova`

For many years SPSS made things simple: it largely hid the fact that when reporting Anova tests you need to choose the way the sums of squares (and so F ratios) are calculated when interactions are included in a model.

- Note: this ONLY matters if your model has an interaction term (e.g. `y ~ A*B`).
- Sometimes people don't report which they use.
- Because so many people used SPSS for so many years, most papers will report ***Type 3*** sums of squares. This is a good first guess.
- There are good reasons to use Type 2 instead though.
- The default `anova` function in R gives Type 1 sums of squares. This doesn't matter for models without an interaction, but is probbaly not a good choice if you do have interactions.
- The easist way to chose which type is to use `car::Anova(savedmodel, type=3)` instead of the built in `anova` function.

If you want more explanation of why/how to choose, see: https://mcfromnz.wordpress.com/2011/03/02/anova-type-iiiiii-ss-explained/ or for a more technical reference see Venables (1998).

### Modelling change scores

The model aove only uses the `with.music` pain ratings - i.e. scores when music was playing. To estimate the effect of the intervention properly we want to see the difference between scores with and without music:

```
m.effectofmusic <- lm((no.music - with.music) ~ liked * familiar, data=painmusic)
m.effectofmusic %>%
  pander()
```

Table 9: Fitting linear model: (no.music - with.music) ~ liked * familiar

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **(Intercept)** | -0.5357 | 0.2928 | -1.829 | 0.0701 |
| **likedLiked** | 1.071 | 0.4141 | 2.587 | 0.01101 |
| **familiarUnfamiliar** | 0.07143 | 0.4141 | 0.1725 | 0.8634 |
| **likedLiked:familiarUnfamiliar** | -0.9643 | 0.5857 | -1.646 | 0.1026 |

Notice that we didn't create a new variable using mutate: we just used `(no.music - with.music)` on the left hand side of the `~` sign. R knows what we mean and creates the extra column for us behind the scenes.

After running the model we can use the `anova` function again to get the Anova table and F tests:

```
car::Anova(m.effectofmusic, type=3)
```

```
## Anova Table (Type III tests)
##
## Response: (no.music - with.music)
##                 Sum Sq  Df F value  Pr(>F)
## (Intercept)      8.036   1  3.3466 0.07010 .
## liked           16.071   1  6.6933 0.01101 *
## familiar         0.071   1  0.0297 0.86339
## liked:familiar   6.509   1  2.7108 0.10258
```

```
## Residuals      259.321 108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Because we are now modelling **differences** in pain between when music is/isn't playing we have a much more powerful test. We can see that the F test for `liked` is $p < .05$ (although that isn't a very high bar).

### Using Ancova to 'control for baseline'

A common alternative is to put the `no.music` score on the right of the ~ sign, as a covariate:

```
music.ancova <- lm(with.music ~ no.music + liked * familiar, data=painmusic)
music.ancova %>%
  pander()
```

Table 10: Fitting linear model: with.music ~ no.music + liked * familiar

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **(Intercept)** | 1.847 | 0.4585 | 4.029 | 0.0001052 |
| **no.music** | 0.7359 | 0.07345 | 10.02 | 4.63e-17 |
| **likedLiked** | -0.9865 | 0.3937 | -2.506 | 0.01372 |
| **familiarUnfamiliar** | 0.04177 | 0.3943 | 0.1059 | 0.9158 |
| **likedLiked:familiarUnfamiliar** | 0.7379 | 0.5593 | 1.319 | 0.1899 |

This means we have run a multiple regression with two categorical variables and their interaction PLUS a continuous predictor.

If we use the `anova` function on this model, the results are (perhaps confusingly) known as an **Ancova** or analysis of co-variance.

```
music.ancova %>% car::Anova(type=3)
```

```
## Anova Table (Type III tests)
##
## Response: with.music
##                 Sum Sq  Df  F value    Pr(>F)
## (Intercept)     35.092   1  16.2292 0.0001052 ***
## no.music       217.031   1 100.3721 < 2.2e-16 ***
## liked           13.576   1   6.2788 0.0137250 *
## familiar         0.024   1   0.0112 0.9158285
## liked:familiar   3.763   1   1.7404 0.1899089
## Residuals      231.362 107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As with the change score model above, we have more power to detect effects of the intervention than with the very first model.

By including no-music scores as a covariate we have 'controlled-for' the variation between individuals. Each participant acts as their own control.

### Why this isn't a repeated measures model?

In this experiment we took two measures of pain: with and without music. However, because we are interested in the difference between these two, we can either use change scores as our outcome, or include no-music as a covariate (which amounts to much the same thing).

We couldn't have done this if we had taken 3 measures… e.g. if we had measures before, during and after music was played.

The key difference is that in these models, the outcome an be represented by a single column of numbers, and where each participant has only 1 row in the datafile. Any time you have more data than that we need to use a proper repeat-measures model.

### Other uses of Ancova

The other example I gave in the workshop was from `gapminder`.

This is the case where we have some outcome, `y`, and a categorical predictor, `x`, but we want to 'control for' some other variable, `z`.

Technically, this is just the same as the Ancova example above. Only the ***interpretation*** differs.

I said, in R, we just write: `y ~ z + x` to estimate group differences, controlling for `z` (and where `x` is a categorical variable).

The models I showed were:

```
m.without <- lm(lifeExp~continent, data=gapminder::gapminder)
m.without %>% pander
```

Table 11: Fitting linear model: lifeExp ~ continent

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **(Intercept)** | 48.87 | 0.3696 | 132.2 | 0 |
| **continentAmericas** | 15.79 | 0.6486 | 24.35 | 1.38e-112 |
| **continentAsia** | 11.2 | 0.5931 | 18.88 | 2.438e-72 |
| **continentEurope** | 23.04 | 0.611 | 37.71 | 1.383e-226 |
| **continentOceania** | 25.46 | 1.92 | 13.26 | 2.99e-38 |

And then including `pop` and `gdpPercap` as covariates:

```
m.with <-  lm(lifeExp~gdpPercap+pop+continent, data=gapminder::gapminder)
m.with %>% pander()
```

Table 12: Fitting linear model: lifeExp ~ gdpPercap + pop + continent

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **(Intercept)** | 47.81 | 0.3395 | 140.8 | 0 |
| **gdpPercap** | 0.0004495 | 2.346e-05 | 19.16 | 3.239e-74 |
| **pop** | 6.57e-09 | 1.975e-09 | 3.326 | 0.0009007 |
| **continentAmericas** | 13.48 | 0.6 | 22.46 | 5.188e-98 |
| **continentAsia** | 8.193 | 0.5712 | 14.34 | 4.064e-44 |
| **continentEurope** | 17.47 | 0.6246 | 27.97 | 6.335e-142 |
| **continentOceania** | 18.08 | 1.782 | 10.15 | 1.59e-23 |

I then used `broom::augment` (which we previously encountered in the section on predictions from regression models).

I created a new dataframe with one row per-continent, and where the values for `pop` and `gdpPercap` were set to the mean of all countries:

```
nd = tibble(continent=unique(gapminder::gapminder$continent),
            gdpPercap=mean(gapminder::gapminder$gdpPercap),
            pop=mean(gapminder::gapminder$pop),
                          )
```

Then I made predictions from both models (with and without covariates) as a kind of ***sensitivity analysis***:

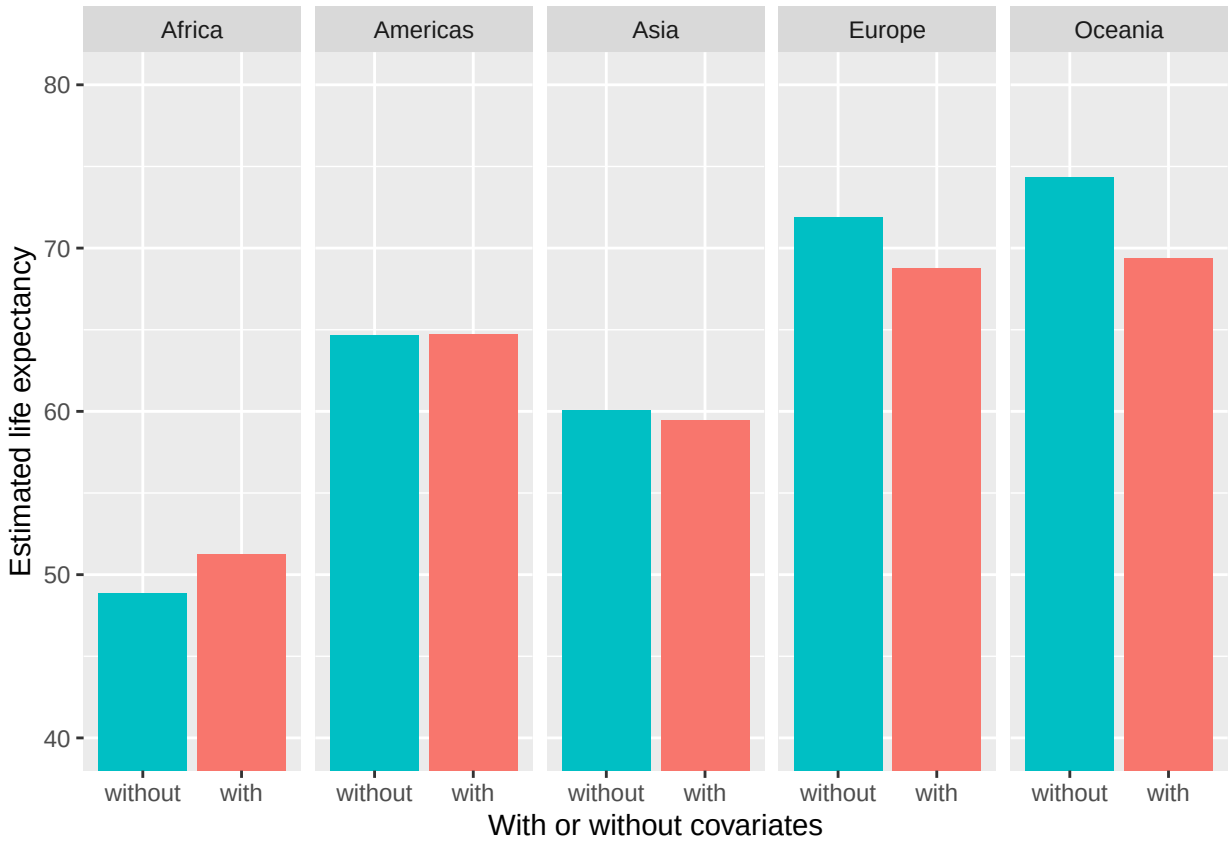
```
without <- lm(lifeExp~continent, data=gapminder::gapminder) %>%
  broom::augment(newdata=nd) %>%
  mutate(model="without")

with <- lm(lifeExp~gdpPercap+pop+continent, data=gapminder::gapminder) %>%
  broom::augment(newdata=nd) %>%
  mutate(model="with")
```

Then I used `bind_rows` from `dplyr` to combine the two dataframes and make a single plot:

```
bind_rows(without, with) %>%
  ggplot(aes(forcats::fct_rev(model), .fitted, fill=model)) +
  stat_summary(geom="bar") +
  coord_cartesian(ylim=c(40,80) )+
  facet_grid(.~continent) +
  xlab("With or without covariates") + ylab("Estimated life expectancy") +
  scale_fill_discrete(guide=F)
```

```
## No summary function supplied, defaulting to `mean_se()
## No summary function supplied, defaulting to `mean_se()
## No summary function supplied, defaulting to `mean_se()
## No summary function supplied, defaulting to `mean_se()
## No summary function supplied, defaulting to `mean_se()
```




At this point, I probably droned on and on about how:

- Anova is a nickname for one way of presenting the unadjusted model.
- An Ancova is a nickname for the same thing for the adjusted model.
- Sensitivity analyses are great and we should do more of them

We can look at both 'anova' tables here:

```
anova(m.without)
```

```
## Analysis of Variance Table
##
## Response: lifeExp
##             Df Sum Sq Mean Sq F value    Pr(>F)
## continent    4 139343   34836  408.73 < 2.2e-16 ***
## Residuals 1699 144805      85
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(m.with) # this is actually an Ancova tale
```

```
## Analysis of Variance Table
##
## Response: lifeExp
##             Df Sum Sq Mean Sq  F value    Pr(>F)
## gdpPercap    1  96813   96813 1383.457 < 2.2e-16 ***
## pop          1   1815    1815   25.938 3.919e-07 ***
## continent    4  66766   16691  238.521 < 2.2e-16 ***
## Residuals 1697 118754      70
```

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

We saw that the F value for `continent` is smaller in the adjusted model, but we still see small $p$ values for the group comparison so we would reject the null hypothesis that all the continents are the same.

Note also that in this example — because we don't have an interaction term — the type of sums of squares doesn't make a difference:

```
car::Anova(m.with, type=2)
```

```
## Anova Table (Type II tests)
##
## Response: lifeExp
##           Sum Sq   Df F value    Pr(>F)
## gdpPercap  25684    1  367.02 < 2.2e-16 ***
## pop          774    1   11.06 0.0009007 ***
## continent  66766    4  238.52 < 2.2e-16 ***
## Residuals 118754 1697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
car::Anova(m.with, type=3)
```

```
## Anova Table (Type III tests)
##
## Response: lifeExp
##              Sum Sq   Df  F value    Pr(>F)
## (Intercept) 1387694    1 19830.13 < 2.2e-16 ***
## gdpPercap     25684    1   367.02 < 2.2e-16 ***
## pop             774    1    11.06 0.0009007 ***
## continent     66766    4   238.52 < 2.2e-16 ***
## Residuals    118754 1697
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The tables differ a little bit, but the F values are identical.

**Is using Ancova a good idea?** In many papers you will see authors 'controlling' for all sorts of variables. This is typically reported as the natural thing to do, and unproblematic.

However, in addition to concerns about researcher degrees of freedom, it has been known for many years that including covariates can introduce biases. In my talk I gave a few examples of this and the bias that can be generated.

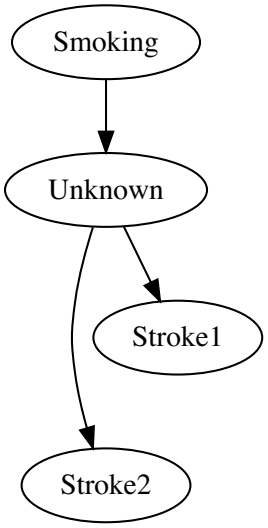
The nature of the possible bias is dependent on the true causal pathways responsible for generating the data. It's important to note that for real datasets we can never know for sure if our results are biased: we only know it's a possiblity because simulation studies show this is the case.

For more on new approaches to deciding when to include covariates see Pearl and Mackenzie (2018), chapter 4.

The main example I gave was: Imagine smoking causes some underlying change in our cardiovascular system, which then causes strokes. This causal diagram represents that (quite plausible) situation:

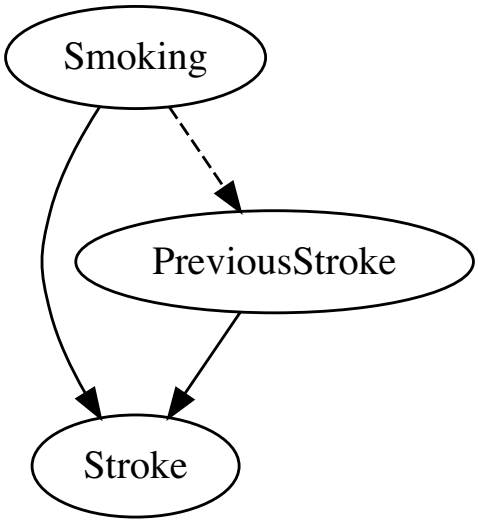
```
grViz("
      digraph ab {
      Smoking -> Unknown -> Stroke2
      Unknown -> Stroke1
```

```
      Stroke1 -> Stroke2 [style=invis]
      }
      ", height=200)
```




If we wanted to know the effect of smoking on stroke we might be tempted to 'control for' the number of previous strokes, but that would be a disaster:

```
grViz("
      digraph ab {
      Smoking -> Stroke
      PreviousStroke -> Stroke
      Smoking -> PreviousStroke [style=dashed]
      }
      ", height=200)
```




The reason is that the effect of smoking is now being 'shared', and we might well underestimate the true causall effect of smoking on stroke, misattributing is to some individual/dispositional cause.

**Should you use covariates then?** There is a fairly simple rule of thumb: If you have an experiment, controlling for covariates is probably fine and is often a good idea.

If you have observational data it's really not so simple and you should do some more thinking before assuming it will improve the quality of the inferences you draw.

(Note - for the purposes of the assessment you should reproduce the model faithfully, but you might like to consider the use of covariates when interpreting your findings or thinking about a sensitivity analysis).

### Exercises

#### Exercise 1

1. Re-run both the models from above which use the gapminder data and calculate the Bayes Factor
2. How much evidence do we have that these covariates improve a continent-only model?

**Exercise 2** Read in the following data like this:

```
fitdata <- read_csv('https://zenodo.org/record/1120364/files/blind_data.csv?download=1')
```

```
## Parsed with column specification:
## cols(
##   kg1 = col_double(),
##   kg2 = col_double(),
##   kg3 = col_double(),
##   cm1 = col_double(),
##   cm2 = col_double(),
##   cm3 = col_double(),
##   group = col_double(),
##   gender = col_character(),
##   age = col_double(),
##   height = col_double(),
##   gqol1 = col_double(),
##   gqol2 = col_double(),
##   ceq_logical = col_double(),
##   ceq_successfull = col_double(),
##   ceq_recommend = col_double()
## )
```

This is data from a trial of Functional Imagery Training (FIT) for weight loss which you have seen before. The variables starting `kg` relate to participants weights. The `group` variable indicates whether they were allocated to FIT or motivational interviewing (the control condition).

- Run an Anova predicting `kg3` from `group`.
- Now run an Ancova predicting `kg3` from group, controlling for `kg1`
- What are the $F$ and $p$ values for the `group` effect in each model?
- Which model is better, and why?
- Calculate the BF for the effect of FIT.

Try for yourself first, but click here to see the code and results

```
m.fit <- lm(kg3~kg1 + group, data=fitdata)
m.fit.without <- lm(kg3~ group, data=fitdata)

anova(m.fit)
```

```
## Analysis of Variance Table
##
## Response: kg3
##            Df  Sum Sq Mean Sq  F value    Pr(>F)
## kg1         1 24046.5 24046.5 1037.292 < 2.2e-16 ***
```

```
## group       1   990.3   990.3   42.719 2.094e-09 ***
## Residuals 109  2526.8    23.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(m.fit.without)
```

```
## Analysis of Variance Table
##
## Response: kg3
##            Df  Sum Sq Mean Sq F value Pr(>F)
## group       1   546.2  546.16  2.2237 0.1388
## Residuals 110 27017.5  245.61
```

The $F$ for the Anova is much smaller, and the p larger. We have less power in this model to detect a group difference.

In the Ancova each participant acts as their own control, increasing statistical power.

**Graphs** If you didn't complete this in the previous workshop the following would be good preparation for the assessment:

- For each of (up to 3) question/predictions, draw out a graph, by hand, which would make it easy to *see* the answer to the question.
- Create alternative graphs of the same information which emphasise different aspects of the data, or use different visual aesthetics to make distinctions. Some points to consider when designing different graphs:
  - Is there a tradeoff between showing group differences vs. relationships vs. distributional information?
  - Is it possible to show both raw data and summaries at the same time? What are the tradeoffs involved?
  - Who is your graph for? Are you designing for other interested experts, or busy managers?

Please note for this final question you don't have to limit yourself to graphs you (personally) can actually plot. If you can describe or draw the graph we can discuss how/whether it can be made and provide support.

**Optional: Causal diagram**

Draw out a causal diagram for the study you are trying to replicate (it might be quite simple or rather complex depending on the study). What other variables might influence the outcome which are not being captured by the authors? Would Ancova make sense in this case?

**Optional: Chi-squared** If your study requires a chi-squared test, you can work through the examples here:

- https://benwhalley.github.io/just-enough-r/crosstabs.html#crosstabs

**Optional: Correlations** If your study reports correlations, see this page for details of how to create and format a correlation table:

- https://benwhalley.github.io/just-enough-r/correlations.html

If you want to run a Bayesian test on a correlation, see these examples:

- https://ajwills72.github.io/rminr/vbg_corr.html

If your study reports non-parametric correlations ask me during a workshop and we can figure out the best way of calculating the correct numbers.

**Optional extension** One thing we have not covered in details is the way in which outcomes and predictors can be scaled to ease interpretation of regression coefficients.

Read this page: https://benwhalley.github.io/just-enough-r/scaling-predictors.html and also Gelman (2008).

Run models using the gapminder data to predict `lifeExp` which include scaled or unscaled predictors. Compare the regression coefficients from each and decide which you find easiest to interpret and why.

### Session 3: Reproducing a real paper

We will finalise work on your replication, building towards the individual assessment.

# Extras

## Keyboard shortcuts

There are a two very useful keyboard shortcuts it is definitely worth learning:

#### To insert a pipe: `%>%`

Type `CTRL + SHIFT + M`

#### To add an assignment arrow: `<-`

Type `ALT + -` (that is, ALT + the hyphen/minus key)

## Common problems

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Some of this material was adapted from Andy Wills' RminR.

#### Try these things first...

**Check your typing** - The command has to be exactly as it appears on the worksheet. In particular, check your brackets `( )`, and check that you haven't missed off the speech marks `" "`. Check you haven't missed out any commas `,`. Check you have typed captial letters as shown, e.g. `bayesfactor` is not the same as `BayesFactor`. Check whether your command uses `=` or `==`, these are not the same thing (see below for explanation).

**Restart R** - If the output you get is very different to what is shown in this worksheet, go to "Session" on the RStudio menu, and select "Restart R". If that doesn't fix your problem, see below.

### Errors when loading a package

If you've installed R on your own machine, and you get a message like:

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

Have you installed this package? You need to install it before you can use it, see the cheat sheet.

### Errors when loading data

If you get a message like:

`Error in open.connection(con, "rb") : Could not resolve host: www.willslab.org.uk`

Check your internet connection (e.g. by using the web browser to look at your Twitter feed). If you have an internet connection, try the command again.

### Is it = or == ?

In R, `=` means the same as `<-`. Computers don't cope well with situations were the same symbol means two different things, so we use `==` to mean "equal to". For example `filter(education == "master")` in our income data set means keep the people whose education level equal to "master".

### When trying to knit Rmd files

- Check you have the right number of backticks?
- Have you used the `View()` function in your code? This only works when inside RStudio, not when knitting documents.
- Have you loaded your packages at the top of the file? If you try to call functions from tidyverse or other packages without loading them first you will get errors. Sometimes this is not apparent when working in an interactive session, but creates an error when a package is loaded after it is used in the code.

### My variables have spaces of other special characters in their names!!!

Try to avoid spaces or punctuation in your variable names if possible. If you do end up with spaces in your column names, you can still access them by putting 'backticks' around the name.

Our annoying dataframe might be like this:

```
annoying_dataframe  %>% head
```

```
## # A tibble: 3 x 1
##   `What is your favourite colour?`
##   <chr>
## 1 Red
## 2 Blue
## 3 Green
```

We can rename the column:

```
annoying_dataframe %>%
  rename('favourite_colour' = `What is your favourite colour?`) %>%
  head
```

```
## # A tibble: 3 x 1
##   favourite_colour
##   <chr>
## 1 Red
```

```
## 2 Blue
## 3 Green
```

## Cheatsheet

This covers everything we have done to-date (or will soon), in abbreviated form. You will need to load tidyverse for most of these examples:

```
library(tidyverse)
```

### Basics

Assign a value to a name (variable):

```
meaningoflife <- 43
```

Simple arithmetic

```
2+2
```

```
## [1] 4
```

```
2*2
```

```
## [1] 4
```

```
2^3 # 2 cubed
```

```
## [1] 8
```

Compare values:

```
2 == 2  # note the doubled = sign
```

```
## [1] TRUE
```

```
2 != 4  # 2 is not equal to 4
```

```
## [1] TRUE
```

```
2 > 2
```

```
## [1] FALSE
```

```
2 >= 2
```

```
## [1] TRUE
```

### Sequences

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1,10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1,100, by=7) # 7 times table
```

```
##  [1]  1  8 15 22 29 36 43 50 57 64 71 78 85 92 99
```

For strings use `c()`

```
c("Wibble", "Wobble", "Nobble")
```

```
## [1] "Wibble" "Wobble" "Nobble"
```

Combinations of sequences:

```
expand.grid(
   colour=c("Red", "Green"),
   position=c("Top", "Bottom"))
```

```
##   colour position
## 1    Red      Top
## 2  Green      Top
## 3    Red   Bottom
## 4  Green   Bottom
```

### Randomness

A random sample:

```
sample(1:20)
```

```
##  [1]  7 19  9 10  8  6 17 20  4  2 15 14 16 13 11 12  1 18  3  5
```

Random-normal numbers:

```
rnorm(n=20, mean=0, sd=5)
```

```
##  [1]  4.4832469 -0.3608593  7.3977207 -4.3818071  3.8509301 -2.8043747
##  [7]  2.7987554  5.4842756  3.4258839 -8.8188081  4.4380099  3.9718106
## [13] -1.3609724  4.2579277 -9.2809916  2.3759649 -3.0197646 -3.6575429
## [19] -0.1709411 -3.5309354
```

```
rnorm(20, 0, 5)  # equivalent to line above(but this is a different sample!)
```

```
##  [1]  3.7524876  7.1677582 -4.0138850 -5.0519508  4.0769638  6.3728209
##  [7]  3.1055055  3.2647324  6.3410550 -0.7167005 -3.9325464 -1.4452360
## [13] -6.3224102 -4.5711502 -6.6243061  2.3307275  3.5124875  5.6475370
## [19] -2.3076180  0.1887434
```

Other types of random numbers:

```
runif(10)
```

```
##  [1] 0.77696682 0.83072583 0.51916177 0.32288403 0.01330045 0.56139464
##  [7] 0.13467411 0.53170066 0.83642125 0.40127429
```

```
rbinom(10, size=1, prob=.5) # like a coin toss where 1=heads
```

```
##  [1] 1 1 1 0 0 0 1 1 1 0
```

```
rbinom(10, size=5, prob=.5) # like best of 5 coin tosses
```

```
##  [1] 1 1 3 1 3 2 1 2 3 3
```

### Loading data

CSV:

```
df <- read_csv('filename.csv') # from a file on your server account
df <- read_csv('<URL>') # direct from the web
```

SPSS:

```
library(haven)  # load haven first
df_from_spss <- read_spss('filename.sav')
```

Excel (note different sheets are loaded from same file):

```
expt1 <- read_excel('simple-excel-example.xlsx', sheet="Experiment 1")
expt2 <- read_excel('simple-excel-example.xlsx', sheet="Experiment 2")
```

### 'Looking at' datasets

If a dataset is large we don't want to look at it all at once.

Show the first 3 rows for all variables:

```
iris %>% head(3)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
```

Show a list of all the ***columns*** with as many datapoints as fit in the window:

```
iris %>% glimpse
```

```
## Observations: 150
## Variables: 5
## $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.0, 4.4, 4.9, 5.4,...
## $ Sepal.Width  <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.4, 2.9, 3.1, 3.7,...
## $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5,...
## $ Petal.Width  <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.2, 0.2, 0.1, 0.2,...
## $ Species      <fct> setosa, setosa, setosa, setosa, setosa, setosa, setosa...
```

Calculate some useful summaries of all variables in the dataset:

```
library(skimr)
iris %>% skim()
```

Table 13: Data summary

| | |
|---|---|
| Name | Piped data |
| Number of rows | 150 |
| Number of columns | 5 |
| | |
| Column type frequency: | |
| factor | 1 |
| numeric | 4 |
| | |
| Group variables | None |

**Variable type: factor**

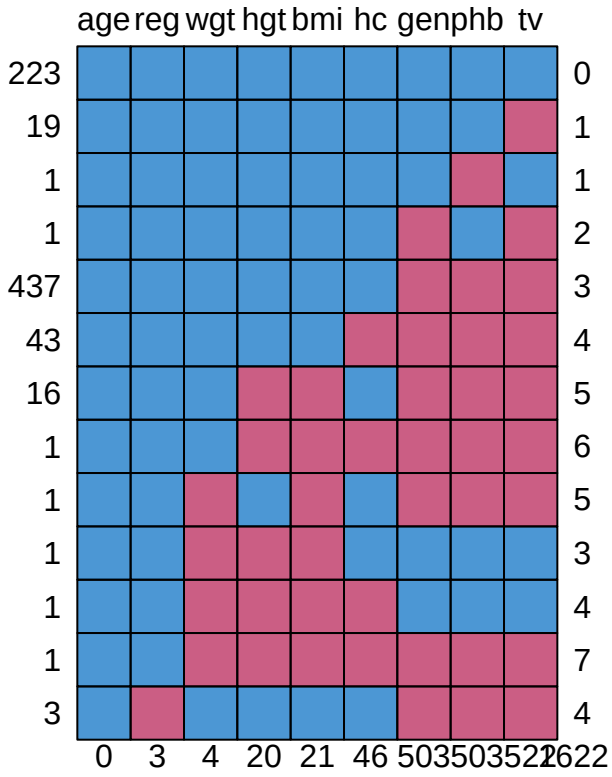
| skim_variable | n_missing | complete_rate | ordered | n_unique | top_counts |
|---|---|---|---|---|---|
| Species | 0 | 1 | FALSE | 3 | set: 50, ver: 50, vir: 50 |

**Variable type: numeric**

| skim_variable | n_missing | complete_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
|---|---|---|---|---|---|---|---|---|---|---|
| Sepal.Length | 0 | 1 | 5.84 | 0.83 | 4.3 | 5.1 | 5.80 | 6.4 | 7.9 | |
| Sepal.Width | 0 | 1 | 3.06 | 0.44 | 2.0 | 2.8 | 3.00 | 3.3 | 4.4 | |
| Petal.Length | 0 | 1 | 3.76 | 1.77 | 1.0 | 1.6 | 4.35 | 5.1 | 6.9 | |
| Petal.Width | 0 | 1 | 1.20 | 0.76 | 0.1 | 0.3 | 1.30 | 1.8 | 2.5 | |

Check for patterns of missing data:

```
library(mice)
boys %>% md.pattern()
```




```
##     age reg wgt hgt bmi hc gen phb  tv
## 223   1   1   1   1   1  1   1   1   1    0
## 19    1   1   1   1   1  1   1   1   0    1
## 1     1   1   1   1   1  1   1   0   1    1
## 1     1   1   1   1   1  1   0   1   0    2
## 437   1   1   1   1   1  1   0   0   0    3
## 43    1   1   1   1   1  0   0   0   0    4
## 16    1   1   1   0   0  1   0   0   0    5
## 1     1   1   1   0   0  0   0   0   0    6
## 1     1   1   0   1   0  1   0   0   0    5
## 1     1   1   0   0   0  1   1   1   1    3
## 1     1   1   0   0   0  0   1   1   1    4
## 1     1   1   0   0   0  0   0   0   0    7
## 3     1   0   1   1   1  1   0   0   0    4
##       0   3   4  20  21 46 503 503 522 1622
```

### Choosing columns and rows

```
mtcars %>% select(mpg, wt) %>% head()
```

### Selecting columns

```
##                    mpg    wt
## Mazda RX4         21.0 2.620
## Mazda RX4 Wag     21.0 2.875
## Datsun 710        22.8 2.320
## Hornet 4 Drive    21.4 3.215
## Hornet Sportabout 18.7 3.440
## Valiant           18.1 3.460
```

Selecting columns which start with a particular string:

```
iris %>% select(starts_with('Sepal')) %>% head
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
## 4          4.6         3.1
## 5          5.0         3.6
## 6          5.4         3.9
```

Selecting columns which match a name:

```
iris %>% select(matches('Width')) %>% head
```

```
##   Sepal.Width Petal.Width
## 1         3.5         0.2
## 2         3.0         0.2
## 3         3.2         0.2
## 4         3.1         0.2
## 5         3.6         0.2
## 6         3.9         0.4
```

```
mtcars %>% select(wt) %>% rename(weight=wt) %>% head
```

### Renaming columns

```
##                   weight
## Mazda RX4          2.620
## Mazda RX4 Wag      2.875
## Datsun 710         2.320
## Hornet 4 Drive     3.215
## Hornet Sportabout  3.440
## Valiant            3.460
```

Try to avoid spaces or punctuation in your variable names if possible.

If you do end up with spaces in your column names, you can still access them by putting 'backticks' around the name.

Our annoying dataframe is like this:

```
annoying_dataframe  %>% head
```

```
## # A tibble: 3 x 1
##   `What is your favourite colour?`
##   <chr>
## 1 Red
## 2 Blue
## 3 Green
```

Rename the column:

```
annoying_dataframe %>%
  rename('favourite_colour' = `What is your favourite colour?`) %>%
  head
```

```
## # A tibble: 3 x 1
##   favourite_colour
##   <chr>
## 1 Red
## 2 Blue
## 3 Green
```

**Filtering rows** Select rows where a variable matches a particular number:

```
gapminder::gapminder %>% filter(year==2002) %>% head
```

```
## # A tibble: 6 x 6
##   country     continent  year lifeExp      pop gdpPercap
##   <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
## 1 Afghanistan Asia       2002    42.1 25268405      727.
## 2 Albania     Europe     2002    75.7  3508512     4604.
## 3 Algeria     Africa     2002    71.0 31287142     5288.
## 4 Angola      Africa     2002    41.0 10866106     2773.
## 5 Argentina   Americas   2002    74.3 38331121     8798.
## 6 Australia   Oceania    2002    80.4 19546792    30688.
```

Select rows where a variable matches a particular string value (letters/words):

```
iris %>% filter(Species=="setosa") %>% head
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Select rows meeting (both of) 2 criteria:

```
mtcars %>% filter(wt > 3 & cyl == 4)
```

```
##    mpg cyl  disp hp drat   wt qsec vs am gear carb
## 1 24.4   4 146.7 62 3.69 3.19 20.0  1  0    4    2
## 2 22.8   4 140.8 95 3.92 3.15 22.9  1  0    4    2
```

Select rows meeting either one criteria OR another:

```
mtcars %>% filter(wt > 3 | cyl == 4) %>% head
```

```
##    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## 1 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## 2 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## 3 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## 4 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## 5 14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## 6 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
```

### Groups and summaries

Calculate statistics on a dataset:

```
iris %>% summarise(mean(Sepal.Length), mean(Sepal.Width))
```

```
##   mean(Sepal.Length) mean(Sepal.Width)
## 1           5.843333          3.057333
```

Give our summary variables specific names (these are bad names, don't do this!):

```
iris %>% summarise(wibble=mean(Sepal.Length), wobble=mean(Sepal.Width))
```

```
##     wibble   wobble
## 1 5.843333 3.057333
```

Calculate summaries for groups in the a dataset:

```
iris %>% group_by(Species) %>% summarise(mean(Sepal.Length), sd(Sepal.Length))
```

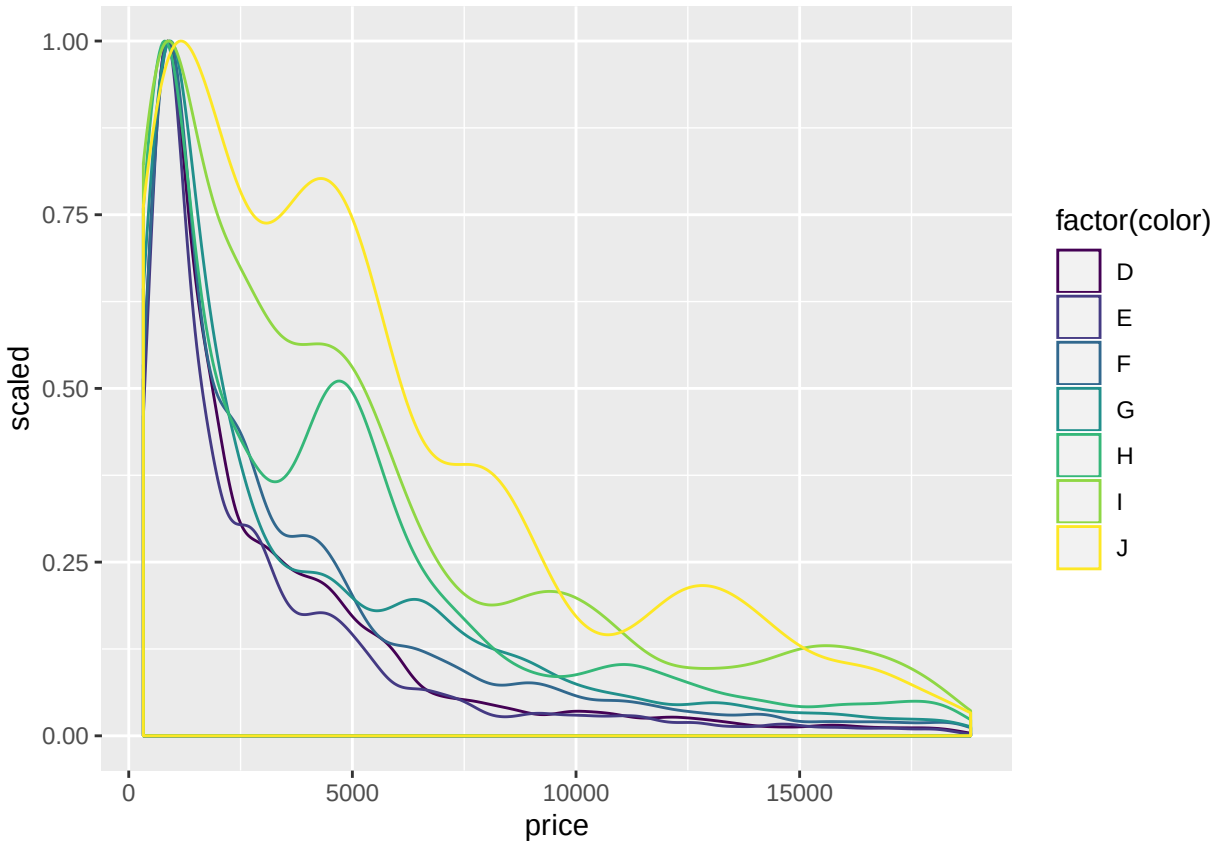
```
## # A tibble: 3 x 3
##   Species    `mean(Sepal.Length)` `sd(Sepal.Length)`
##   <fct>                     <dbl>              <dbl>
## 1 setosa                     5.01              0.352
## 2 versicolor                 5.94              0.516
## 3 virginica                  6.59              0.636
```

### Plotting

Also see the section on visualisation.

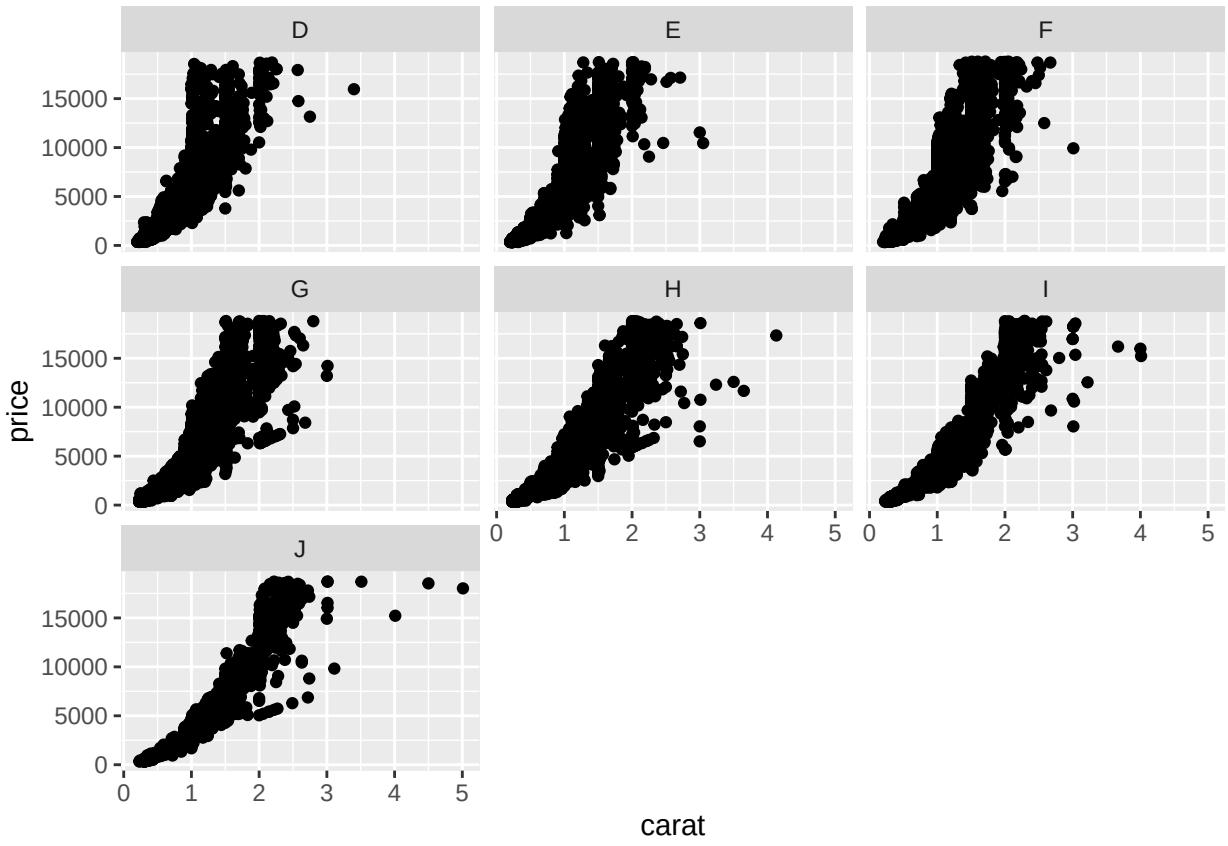
Density plot with colours:

```
diamonds  %>%
  ggplot(aes(x=price, y=..scaled.., color=factor(color))) +
  geom_density()
```



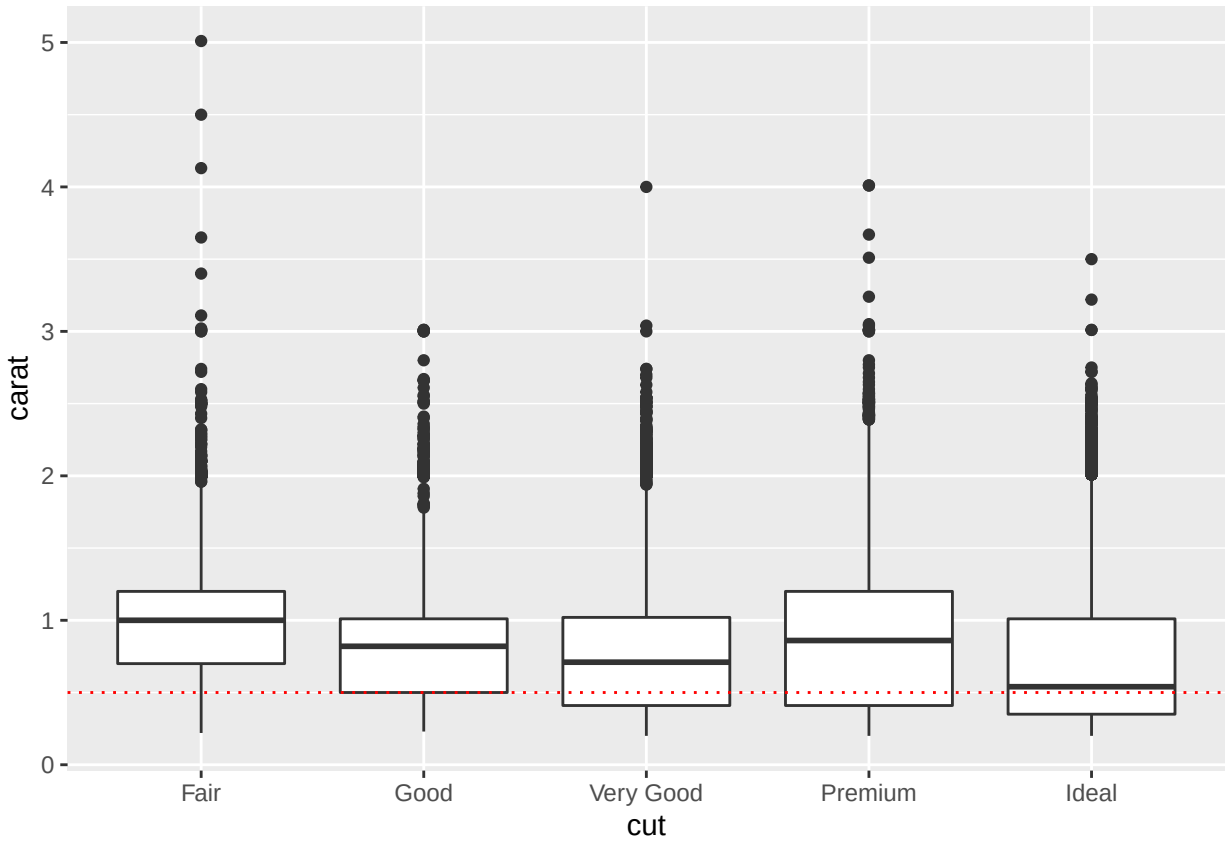

Scatterplot, with facets:

```
diamonds %>%
  ggplot(aes(x=carat, y=price)) +
  geom_point() +
  facet_wrap(~color)
```



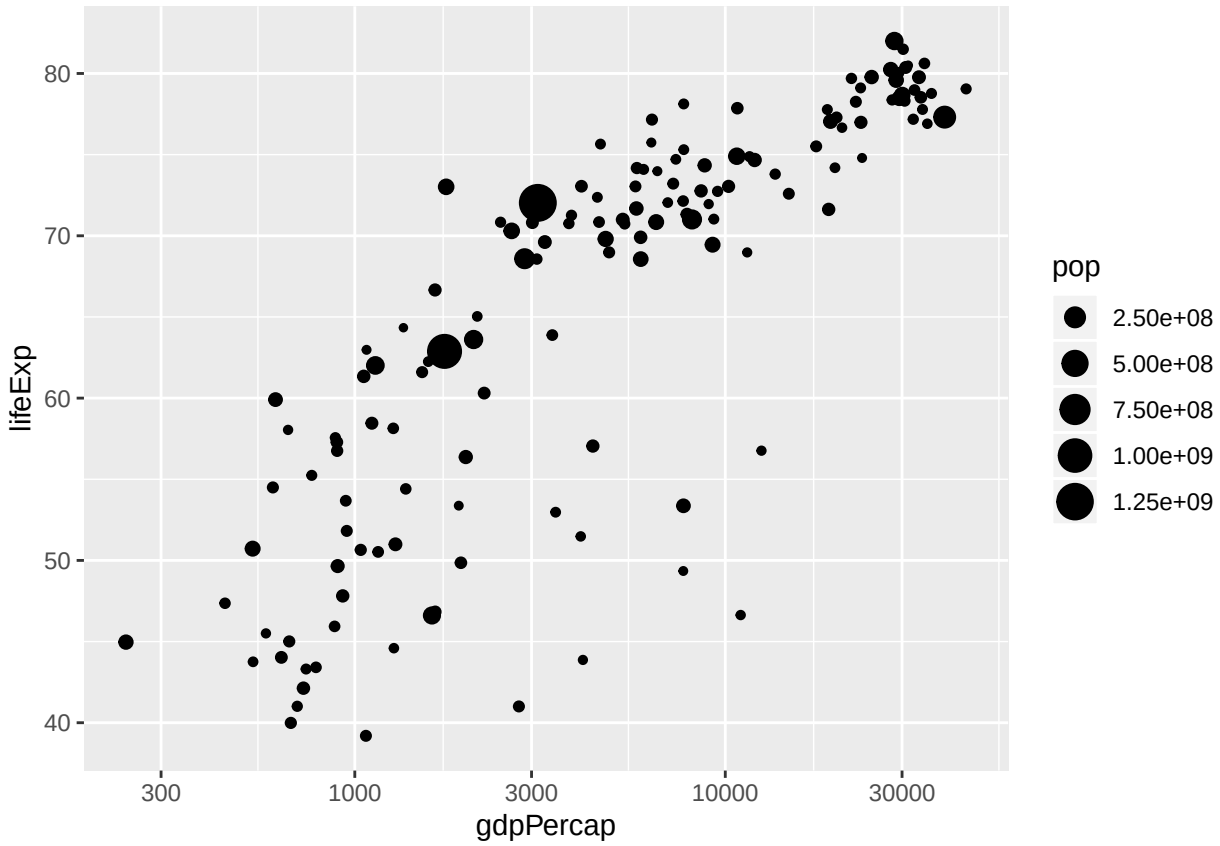

Boxplot, with added line:

```
diamonds %>%
  ggplot(aes(x=cut, y=carat)) +
  geom_boxplot() +
  geom_hline(yintercept=.5, color="red", linetype="dotted")
```




Scatterplot with logarithmic scale on x axis, with point sizes scaled (to country-size `pop`):

```
gapminder::gapminder %>%
  filter(year==2002) %>%
  ggplot(aes(gdpPercap, lifeExp, size=pop)) +
  geom_point() +
  scale_x_log10()
```




## More sampling

In the beginner session we used `sample` to shuffle a sequence of numbers, and make random samples:

```
sample(1:10, size=5)
```

```
## [1]  4 10  1  3  8
```

```
sample(1:2, size=10, replace=TRUE)
```

```
##  [1] 1 2 1 1 1 2 2 1 1 1
```

We also saw how to use `expand.grid` to create experimental designs:

```
colours=c("Red", "Green")
positions=c("Top", "Bottom")
words = c("Nobble", "Wobble", "Hobble")
expand.grid(colour=colours, position=positions, words = words)
```

```
##    colour position  words
## 1     Red      Top Nobble
## 2   Green      Top Nobble
## 3     Red   Bottom Nobble
## 4   Green   Bottom Nobble
## 5     Red      Top Wobble
## 6   Green      Top Wobble
## 7     Red   Bottom Wobble
## 8   Green   Bottom Wobble
## 9     Red      Top Hobble
```

```
## 10  Green      Top Hobble
## 11    Red   Bottom Hobble
## 12  Green   Bottom Hobble
```

---

Tidyverse contains a function which lets us sample raws from our design dataframe directly. First we make sure tidyverse is loaded:

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

And then we can use `sample_n`

```
design <- expand.grid(colour=colours, position=positions, words = words)
design %>% sample_n(100, replace=TRUE) %>% head
```

```
##   colour position  words
## 1    Red   Bottom Nobble
## 2    Red      Top Hobble
## 3  Green   Bottom Hobble
## 4    Red   Bottom Wobble
## 5  Green      Top Wobble
## 6  Green   Bottom Hobble
```

**Explanation**: `sample_n` has randomly sampled rows from our design. In this case we sampled 100, but show only the first 6 (by using `head`).

## More on Tibbles

This material was adapted from Andy Wills' RminR.


Figure 24: Kitten that might be called Tibbles. Possibly. Image: 0x010C; CC BY-SA 4.0

In the Exploring Data worksheet, you got the mean income of your sample like this:

```
cpsdata %>% summarise(mean(income))
```

```
# A tibble: 1 x 1
  `mean(income)`
           <dbl>
1         87293.
```

The answer (the mean) is the last number on the bottom right – but what about all that other stuff in the output? What the hell is a tibble? You can safely ignore the extra stuff but, if you're curious, here's what it all means…

1. The first line, `A tibble: 1 x 1`, says your output is a data frame (aka. `tibble`) with one row and one column … so, a single number. No, I don't think anyone knows why it's called a tibble.
2. The second line tells you what summary you calculated – `mean(income)`, the mean of the incomes.
3. The third line, `<dbl>` tells you that what you calculated is a number (in case you were wondering `dbl` is short for 'double precision floating point number', but that's probably more detail than you needed…).
4. The fourth line gives you the number you calculated, the mean income of `87293`. The `1` at the beginning is a row number, which you can ignore.

If you look at the number `87293.` really closely in your output, you'll notice two more things about it:

a. The first two numbers, `87` are underlined, while the others are not. This is just a way of making big numbers easier to read, much like writing 87,293 rather than 87293. This underlining shows up in your R Console window, but not on these worksheets – sorry about that!

b. There's a decimal point at the end, but no numbers following the decimal point. This lets you know that the answer has been rounded to the nearest whole number. You'll always get an answer that is correct to at least three significant figures, which is generally enough for reporting your findings. There are ways to get a more precise output if you need it, but we don't cover those in this class.

---

Here's another example. In the Group Differences worksheet, we calculate the mean income of your sample by sex like this:

```
cpsdata %>%
    group_by(sex) %>%
    summarise(median(income))
```

```
# A tibble: 2 x 2
  sex    `median(income)`
  <chr>             <dbl>
1 female           52558.
2 male             61746.
```

This can be read much the same way as the last example. These are the differences:

- Your tibble is now 2 x 2 because it has two rows (female, male) and two columns (sex, median income).
- The `<chr>` in sex column says that column contains letters (male, female) rather than numbers – chr is short for "characters" (another word for letters).

---

## Regression - extra explanations

### Description vs prediction

(This section is related to this class activity).

You ***should*** have found that the total length of the residuals for the curved lines is *smaller* than the residuals for the straight line. If this isn't the case, check your measurements.

This is because a curved line will be better **description** of the data you had when fitting it, but will be a poorer **predictor** of new data.

This is why fitting straight lines is such a common technique. We know that the line doesn't describe the data we have perfectly, but we hope it will be a better predictor of future events than anything else.

If you like, there is more detail on this here

### Worse is better

You should have found that:

- Curved lines have smaller residuals *for the original data*
- Straight lines have smaller residuals *when you swap samples*

The reason for this is that there is a ***tradeoff***:

- If we draw a curved line, to get close to the original data points, then our lines reflect peculiarities in the sample. That is, our lines are drawn to accomodate ***random variation*** in this specific sample.
- Because these random variations aren't repeated in new samples, the lines fit ***less well*** when we swap datasets.

In fact, because the straight line (mostly) ignores this sample variation it *can be* a better estimate of the real relationship in the population as a whole[3].

So worse is sometimes better: Because they were simpler, straight lines were a worse fit for our original dataset. But they were a *better* predictor in new random samples.

This is an example of ***overfitting***. By overfitting, we mean that a model (in this case the line) is too closely matched to a particular sample, and so might not be a good predictor of the population as a whole.

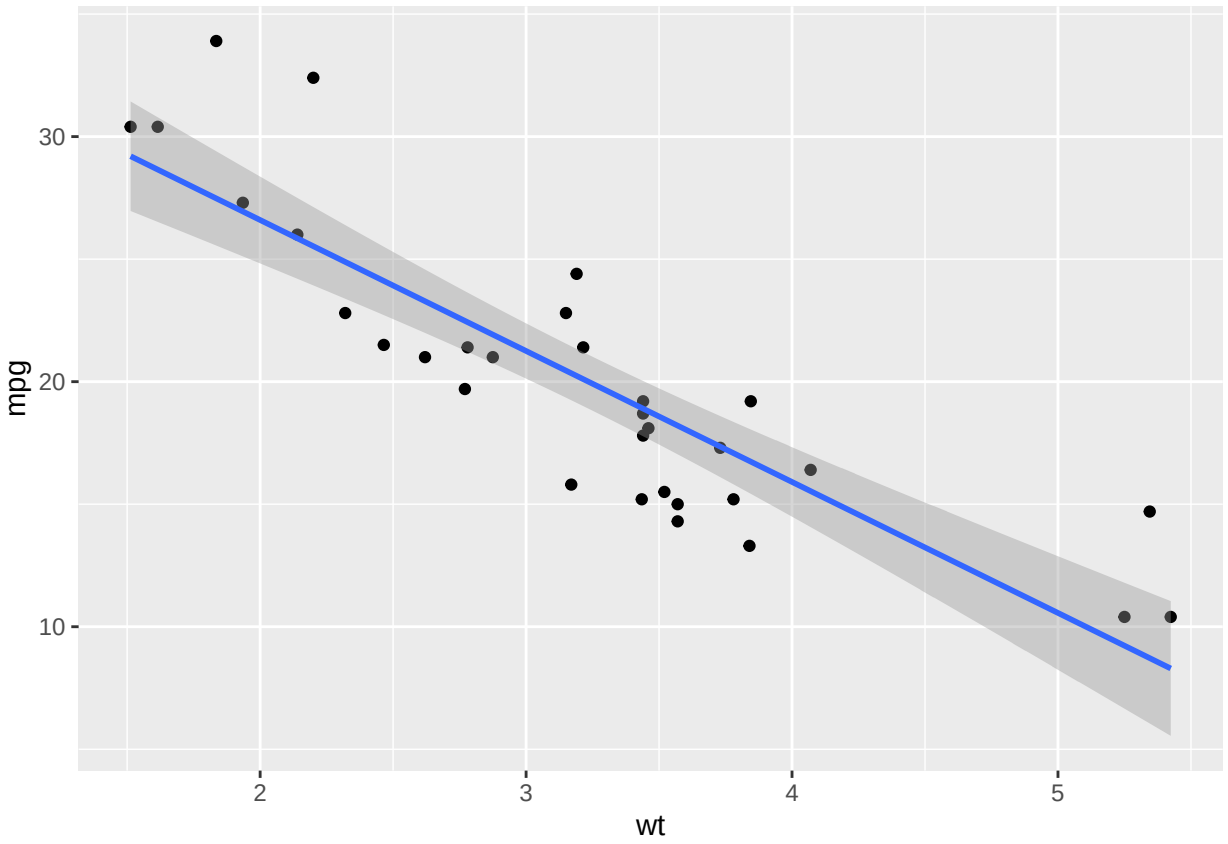
Overfitting is the reason we prefer simpler models (lines) to more complicated ones.

### The shaded area when using `geom_smooth`

If you use `geom_smoth` with `method=lm` you get a grey shaded area around the line.

```
mtcars %>%
  ggplot(aes(wt, mpg)) +
    geom_point() +
    geom_smooth(method=lm)
```

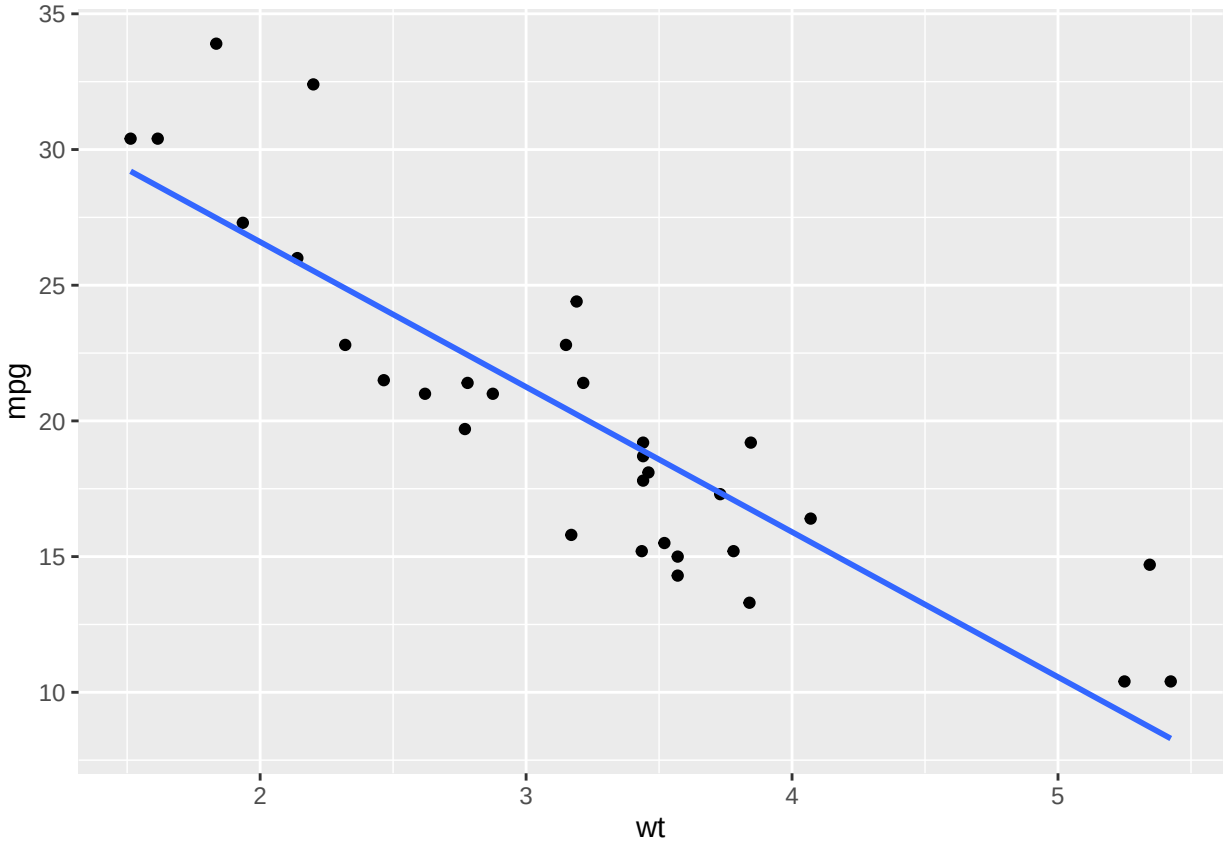
[3] It's not *always* true, but it's a good rule of thumb.




The shaded area shows the ***standard error*** of the line of best fit. This is an estimate of how confident we are about the predictions the line makes. This video explains it quite well: https://www.youtube.com/watch?v=1oHe1a3JqHw. When he uses the greek letter  he just means "slope".

If you want to hide it, you can add: `se=FALSE`:

```
mtcars %>%
  ggplot(aes(wt, mpg)) +
    geom_point() +
    geom_smooth(method=lm, se=FALSE)
```




**Checking your first predictions**

You should get something like: 42.6, 55.1 and 71.9

Don't worry about rounding errors… within 1 point is fine.

We should be most confident about the prediction for 20 hours, because we have more data in the sample which is close to that value. Our line was estimated from data which didn't have many people who worked 5 or 40 hours, so we don't really know about those extremes.

**Why don't we always use real data?**

Real data is often quite complicated, and it is sometimes easier to simulate data which illustrates a particular teaching point as clearly as possible. It also lets us create multiple examples quickly.

It *is* important to use real data though, and this course includes a mix of both simulated and real data.

**What is a formula?**

In R, **formulas** describe the relationship between variables. They are used widely, e.g. in ggplot, functions like `t.test`, and especially in model-fitting functions like `lm`.

Formulas for regression will always describe the link between one *outcome* and one or more *predictor* variables.

The outcome goes on the left, and predictors go on the right. They are separated by the tilde symbol: the `~`. When you read `~` you can say in your head *"is predicted by"*.

You can add multiple variables by separating them with a `+` symbol. So `outcome ~ age + gender` is a model where the outcome is predicted by age and gender. This doesn't add interaction terms.

If you want to include interaction terms (e.g. to let slopes vary for different groups) then you use a `*` symbol instead of a plus. So `outcome ~ age * gender` means the outcome is predicted by age, gender, and the interaction of age and gender.

There is a more technical explanation of all of the formula syntax here: https://stat.ethz.ch/R-manual/R-devel/library/stats/html/formula.html

## Bayesian estimation of the

### How does this all work?

Behind the scenes, the model is being used to *simulate* new data: `add_fitted_samples` does thousands of simulations for what the average prediction will be, and then the `mean_qi` function summarises them, giving us mean, and the 2.5 and 97.5th percentiles. That is, we say in 95% of the simulations the mean was between `conf.low` and `conf.high`.

Because of the way the model was fitted, this is the same as making real probability statements about our prediction.

It might be useful to know that these simulations are said to be *samples from the posterior distribution* — that is, the distribution of probabilities once we combined our data with our *prior probability*.

**Why doesn't this work on my laptop or at home?** The `rstanarm` package is slightly fiddly to install, and if you don't get it right then this code won't work.

We've done it for you on university computers though, and on the RStudio server that you have login details for. So you if you want to run this at home use RStudio server.

## What do confidence intervals mean?

> Imagine we replicated our study many times. We can't know what the *true mean* is. But if we calculate the confidence interval many times then we expect the *true* mean to fall within it, a certain percentage of the time.

If that seems quite a tricky thing to get your head around, you are not alone. There is a long history of misinterpreting confidence intervals in psychology.

It might help to know that in classical statistics, the probabilities that $p$ values and intervals refer to are *attributes of the test procedure itself…* and don't refer to your hypothesis (for example, that there is a difference between experimental groups, or that there is a relationship between mother and child IQ scores). That is, the $p$ value is the probability the test procedure will mislead you. It's NOT the probability your hypothesis is right or wrong.

To interpret the CI you have to assume there is a true, unknown, unchanging value for the thing you are estimating. Imagine there is a *true* slope which defines the relationship betweem mother and child IQ, and this is equal to exactly 0.52. We then need to imagine repeating a study many times. We employ thousands of monkeys to help us, and each time we run a regression and calculate the slope and the 95% confidence interval from the sample. In 95% of these replications, the confidence interval we calculate will include the true value of 0.52.

To reiterate: the 95% CI *does not* tell us the range in which the true value will be 95% of the time, which is what you probably wanted to know.

Despite frequently misinterpreting them, psychologists often use confidence intervals to report uncertainty. They are also very easy to compute, so we cover how in the workshop.

### An example with some common misinterpretations

Reality: "70% of children prefer chocolate icecream to vanilla"

Figure 25: Thousands of monkeys helping Homer run thousands of replications of his analysis.

Your hypothesis: "More children like chocolate ice cream than vanilla."

Your sample: 62 out of 100 children said they preferred ice cream.

The $p$ value says: "If children didn't have any preference in ice cream (50:50 preference) then you would only collect a sample which is this different from the expected 50:50 split about 0.02 times."

The *confidence interval* says: "Based on the sample you collected, it would be unlikely for you to collect another sample where fewer than 52 children said they preferred chocolate ice cream, or more than 71 children said they did prefer chocolate ice cream. By unlikely, we mean less than 1 time in 20."

Common misinterpretations include:

- The $p$ value is 0.0214482 so we are 98% sure children prefer chocolate ice cream.
- We are 95% sure than more than 52 in 100 children prefer chocolate ice cream.

What we'd (really) like to say is:

- It's 4 times more likely that a majority of children prefer chocolate ice cream, than that they have no preference).

But that isn't possible with classical/frequentist statistics.

## Bayesian estimation of the

### How does this all work?

Behind the scenes, the model is being used to *simulate* new data: `add_fitted_samples` does thousands of simulations for what the average prediction will be, and then the `mean_qi` function summarises them, giving us mean, and the 2.5 and 97.5th percentiles. That is, we say in 95% of the simulations the mean was between `conf.low` and `conf.high`.

Because of the way the model was fitted, this is the same as making real probability statements about our prediction.

It might be useful to know that these simulations are said to be *samples from the posterior distribution* — that is, the distribution of probabilities once we combined our data with our *prior probability*.

**Why doesn't this work on my laptop or at home?** The `rstanarm` package is slightly fiddly to install, and if you don't get it right then this code won't work.

We've done it for you on university computers though, and on the RStudio server that you have login details for. So you if you want to run this at home use RStudio server.

### Why do my results differ from yours?

Because of the way the Bayesian models are fitted, there is normally some variation in the results each time you run the program. This means different people will get (slightly) different answers.

This is usual, and nothing to worry about: the amount of variation is typically small. If you see large differences it normally means something else has gone wrong.

However, when fitting complex models we do sometimes need to minimise the amount of variation. This is quite a complex topic, not covered in this course, but if you were interested see Jon Kruschke's excellent textbook (Kruschke 2014) or Richard McElreaths' *Rethinking* book (McElreath 2018).

However, the short version is that we need to increase the number of simulations the software uses to calculate estimates and intervals. You can do this with rstanarm by setting the `iter` variable when you run the model. For example, this runs a simple model using 20,000 simulations, rather than the default of 4000:

```
stan_glm(mpg~wt, data=mtcars, iter=20000)
```

In short: the default values are often fine, but if you are worried about increasing precision there are ways to achieve it.

## Other tips

### The Environment pane

This video shows you how to look at data in the Environment pane of RStudio:

## Wider reading

If you feel anxious about returning to study after some time, or have simply forgotten all the statistics and research methods you learnt on your BSc, you might find the teaching materials from the Plymouth University BSc course useful.

Just Enough R is a companion website to these materials. It is designed to offer a more concise guide to all of the topics dealt with in PSY556, PSY558, along with other material likely to be of use to MSc and PhD students in Psychology. It's probably best used *after* you have worked through the relevant workshop sessions — although it can also be used for self study.

If you actually want to understand *how* regression works in more detail and have some maths background (e.g. A level) the Gelman and Hill book *Data Analysis Using Regression and Multilevel/Hierarchical Models* is quite advanced, but offers clear explanations of the underlying concepts.

If you'd like more details on how to actually work with 'real' data… i.e. messy, unfamiliar datasets, then Roger Peng's *Exploratory Data Analysis with R* is nice. If you'd also like to improve your understanding of R and the tidyverse, then Hadley Wickham's *R for Data Science* gets into more of the nuts and bolts of dealing with data, plotting, modelling and and programming in (modern idiomatic) R. There's also Wickham's *Advanced R* if you get really ambitious (although this isn't really a recommended text unless you have a specific problem to solve).

For a deep-dive (with lots of examples) on plotting and visualisation, see Claus O. Wilke, *Fundamentals of Data Visualization*. The *principles* outlined in part 2 are especially useful, and draw on older texts like Cleveland 1993 and Tufte 2001, bringing them up to date and providing R source code to reproduce the examples. Kieran Healy's *Data visualisation* is also good on these topics, with longer explanatory text and more examples, if you prefer.

# References


Doll, Richard, Richard Peto, Emma Hall, Keith Wheatley, and Richard Gray. 1994. "Mortality in Relation to Consumption of Alcohol: 13 Years' Observations on Male British Doctors." *BMJ* 309 (6959): 911–18. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2541157/pdf/bmj00460-0027.pdf.

Gelman, Andrew. 2008. "Scaling Regression Inputs by Dividing by Two Standard Deviations." *Statistics in Medicine* 27 (15): 2865–73. https://doi.org/10.1002/sim.3107.

Kruschke, John. 2014. *Doing Bayesian Data Analysis: A Tutorial with R, Jags, and Stan*. Academic Press.

MacPherson, Hugh, Douglas G Altman, Richard Hammerschlag, Li Youping, Wu Taixiang, Adrian White, David Moher, and STRICTA Revision Group. 2010. "Revised Standards for Reporting Interventions in Clinical Trials of Acupuncture (Stricta): Extending the Consort Statement." *Journal of Evidence-Based Medicine* 3 (3): 140–55.

McElreath, Richard. 2018. *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Chapman; Hall/CRC.

Pearl, Judea, and Dana Mackenzie. 2018. *The Book of Why: The New Science of Cause and Effect*. Basic Books.

Seidelmann, Sara B, Brian Claggett, Susan Cheng, Mir Henglin, Amil Shah, Lyn M Steffen, Aaron R Folsom, Eric B Rimm, Walter C Willett, and Scott D Solomon. 2018. "Dietary Carbohydrate Intake and Mortality: A Prospective Cohort Study and Meta-Analysis." *The Lancet Public Health* 3 (9): e419–e428. https://www.sciencedirect.com/science/article/pii/S246826671830135X.

Solbrig, Linda, Ben Whalley, David J Kavanagh, Jon May, Tracey Parkin, Ray Jones, and Jackie Andrade. 2019. "Functional Imagery Training Versus Motivational Interviewing for Weight Loss: A Randomised Controlled Trial of Brief Individual Interventions for Overweight and Obesity." *International Journal of Obesity* 43 (4): 883. https://www.ncbi.nlm.nih.gov/pubmed/30185920.

Tufte, Edward R. 2001. *The Visual Display of Quantitative Information*. Vol. 2. Graphics press Cheshire, CT.

Venables, WN. 1998. "Exegeses on Linear Models." In *S-Plus User's Conference, Washington Dc*. http://www.stats.ox.ac.uk/pub/MASS3/Exegeses.pdf.

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (1): 1–23. https://doi.org/10.18637/jss.v059.i10.