

PSY6422 Data Management and Visualisation

Tom Stafford

2023-01-16

Contents

0.1	Motivation	9
0.2	Course Aims	10
0.2.1	The Module mini-project	10
0.3	Resources for current students	11
1	Overview, assessment & module organisation	13
1.1	Class material:	13
1.2	Exercise 1 (assessed!)	13
1.2.1	Marking criteria:	14
1.2.2	Submission	14
1.3	Checklist:	14
1.4	Resources	14
1.4.1	Further reading on the motivation for a data management component of this course	14
2	R & R Studio	17
2.1	Resources	17
2.2	Exercises & Checklist	17
3	Making Graphs	19
3.1	Class material	19
3.2	Resources	19
3.2.1	Other resources	19
3.3	Exercises	20
3.4	Checklist	20

4	Project Organisation	21
4.1	Project folders	21
4.2	Naming Things	21
4.3	Research Data Management	22
4.4	Include meta-data	22
4.5	Backup and Sync	23
4.6	Resources:	23
4.7	Exercises	23
4.7.1	Warm up: Pair programming exercise	23
4.7.2	Homework: Project 1	24
4.8	Checklist	24
4.9	Project requirements	25
4.10	Choosing your data	26
4.10.1	Finding Data	26
4.10.2	Suggestions for possible project data sets	27
4.11	Choose your graph	27
4.12	Check list	27
4.13	Answers to common questions	28
4.14	Example Projects	29
4.15	Stretch goals	30
4.16	In class exercises	30
5	Data Wrangling	31
5.1	Class material	31
5.2	Exercises	31
5.3	Checklist	31
5.4	Resources	32

<i>CONTENTS</i>	5
6 Coding Principles	33
6.1 Fundamental methods	33
6.1.1 If statements	33
6.1.2 Loops	35
6.1.3 Functions	36
6.1.4 Exercises 1	38
6.1.5 More	38
6.2 Fundamental principles of good code	38
6.2.1 Readability Matters	38
6.2.2 Avoid hard coded values	41
6.2.3 Functionalise & Generalise	44
6.2.4 Ask for help	45
6.2.5 Exercises 2	46
6.2.6 Checklist	46
6.2.7 Resources	47
7 Rmarkdown	49
7.1 And this line is a heading	50
7.2 First a header	50
7.3 Second, text	51
7.4 Third, code	51
7.4.1 Options for code chunks	53
7.4.2 Include r in inline text	54
7.5 Conclusion	54
7.6 Exercises	54
7.7 Advanced	55
7.8 Checklist	55
7.9 Resources	55

8 ssh and the command line	57
8.1 Launching the Terminal	57
8.2 Changing folders ('navigating the directory tree')	57
8.3 Networking	58
8.4 Ping	58
8.5 Remote access using SSH	59
8.6 Find files, run files, copy files	59
8.6.1 switches	60
8.7 Exercises	61
8.8 Checklist	61
8.9 Resources	61
9 Git and Github	63
9.1 Motivation	63
9.2 Before the class	63
9.3 Accessing git	63
9.4 Resources for the class	64
9.5 Checklist	64
9.6 Resources	64
10 Publishing	67
10.1 Sharing your Rmd files via Github pages	67
10.2 Other sharing options	68
10.3 Checklist	68
10.4 Resources	68
10.5 2022	69
10.6 2021	69
10.7 Other topics/resources	70
10.8 Reproducibility	70
10.9 General righteousness	70
10.10 Before class	71
10.11 Exercises	71

<i>CONTENTS</i>	7
10.12Resources	71
10.13Examples of interactive graphs which use different tech (Javascript!)	72
10.14Exercises	74
10.15Context	75
10.16Key information	75
10.17Exercises	76
10.18Checklist	76
10.19Resources	76
10.20Motivation	77
10.21A worked example	77
10.22Preparation	78
10.23In class	78
10.24After class	79
Appendix	79
A Class notes	81
A.1 Ask me questions	81
A.2 Terminology: ‘Raw data’	81
B Extra Reading	83
B.1 Visualisation (theory)	83
B.2 The Reproducibility Crisis	84
B.3 Better practice	84
B.4 Project organisation	85
B.5 Coding	85
B.6 R	86
B.6.1 Hints	86
B.6.2 Courses / books	86
B.7 Making graphs (practice)	87
B.8 Presentations	87
B.9 Statistics	87
B.10 Advanced Reading, Background & Other Recommends	87
B.11 Pedagogy	88

C Notes	89
C.1 Credit	89
C.2 Colophon	89
C.3 Data Science @ Sheffield	89
C.4 Careers	90
C.5 Testimonials	90
D Class of 2020	91
D.1 Feedback	93
E Class of 2021	95
E.1 Feedback	96
F Class of 2022	99
F.1 Module project showcase	99
F.2 Advice for future students	100
F.3 “one thing you have read and enjoyed or found useful”	102
F.3.1 General Visualisation	102
F.3.2 R specific advice	103
F.3.3 Markdown	103
F.3.4 Advanced R packages	103
F.3.5 General coding / projects / reproducibility	104
F.4 Feedback on the module	104

Module Overview

PSY6422 Data Management and Visualisation is part of the MSc in Psychological Research Methods with Data Science taught at The University of Sheffield by Tom Stafford. See [here](#) for more on the different courses which offer data science, at the University of Sheffield

Spring 2022: These pages are evolving as I teach the course (in person rather than online this year!). If you find any inconsistencies please let me know. Check out the class of 2021 showcase from last academic year

0.1 Motivation

Psychological science is increasingly reliant on complex computational and statistical methods to make sense of rich behavioural data. This course aims to teach the skills which support creating robust and reproducible analyses with such methods and data.

As well as supporting sophisticated data visualisation, we aim to train you in reproducible workflows - meaning that you can reliably re-create all steps of an analysis using scripts that automate all steps between raw data and the final visualisation.

As well as being *reproducible* (by you or other researchers) your work should be *legible* (to Future You, or other researchers) and *scalable* (it should work as well on 400,000 data points as on 40).

You will need help to do this. Therefore you will use Open Source solutions - these are analysis products which have a worldwide community of people using them, and the infrastructure which supports sharing advice and solutions.

In practice, this means you are going to start by using R (you could use Python, but this module is based on R).

0.2 Course Aims

The curriculum is updated each year, but you can get the general idea of the order the topics are covered from the leftbar. By the end of this course you will have:

- Been trained in data project management – including fundamentals of data storage, synchronisation and sharing – and the importance of reproducible workflows
- Used the statistical programming language R, and RStudio, for data management, analysis and visualisation
- Been introduced to fundamental programming concepts
- Prepared data project documentation using RMarkdown
- Had an introduction to version control using git
- Published data projects to the web via github pages

There is also the opportunity to cover advanced topics, either in class or as part of your project. These could include

- Interactive visualisation with Shiny apps
- SQL
- webscraping
- animated / roll-over visualisation

You may particularly enjoy the Reading list

0.2.1 The Module mini-project

The bulk of the course assessment is to conduct and publish your own analysis project. By doing this you will have experience of combining all the skills taught on the course within a single project. This will take a data visualisation from start to finish - from raw data, through data cleaning and documentation to sharing your code and the resulting visualisation on the web.

The intention with the assessment is to ensure that every student leaves the course with something they are proud to put in their portfolio of work, something which shows what they can do and which helps with future job or course applications.

See here for more on the nature of your module project, see here for examples from previous years: class of 2020, class of 2021..

0.3 Resources for current students

Google Drive:

Includes slides and other resources, as well as these specific documents

- Timetable
- Assessment Criteria

FAQ document which I am adding to as questions come in

- FAQ for PSY6422 (including on module project)

Most information is on these pages (hosted on github, no login required)

Chapter 1

Overview, assessment & module organisation

1.1 Class material:

Materials from the class are in the PSY6422 Google Drive (UoS login required to access). Slides : slides format.

1.2 Exercise 1 (assessed!)

The exercise for this class is an initial self-assessment, which comprises part of your final grade. The deadline is in two weeks - see the timetable.

Submit using the form link provided by Tom [check your email] a document of no more than one A4 page (i.e. less than 500 words):

Aim 1: tell me what you know already

- review course webpages, write ~50 words on your experience with computers and coding

Aim 2: start thinking about data to visualise

- describe a dataset you have access to and what you'll do with it OR
- describe the kind of topic you'd like to identify relevant data to do a project on

Aim 3: set some targets for what you'd like to learn next

- describe 1 or 2 ambitions; techniques, technologies or tools you'd like to acquire

1.2.1 Marking criteria:

See the assessment document

The purpose of this assessment is to ensure you are engaging with the course and to start a conversation between us about the final assessment. I want to give you 100%, all you have to do to achieve this is follow the instructions above.

1.2.2 Submission

You submit using the form link supplied by me, as an attachment with a filename containing your surname and registration number, of the form:

surname_000000000.pdf

(file format does not need to be pdf)

1.3 Checklist:

To complete this class you should do the following :

- Review these webpages
 - identify topics covered
 - locate timetable
 - note assessment deadlines
- Complete initial self-assessment by deadline
- Check you have received email from Tom
 - Used the link to join the slack group

1.4 Resources

1.4.1 Further reading on the motivation for a data management component of this course

- Russ Poldrack's Advice for learning to code from scratch

- Neuroskeptic on a true story of data analysis error (and how you can avoid it) “if you’re doing something which involves 100 steps, it only takes 1 mistake to render the other 99 irrelevant.”
- Data Sharing and Management Snafu in 3 Short Acts

Chapter 2

R & R Studio

2.1 Resources

Statistical computing using R, the open source programming language, and the R Studio interface.

Guide: How to install R and RStudio?

<https://swirlstats.com/>

Nick KH Troubleshooting Common R Problems

2.2 Exercises & Checklist

To complete this class you should do the following :

- install R on your computer
- identify the console
 - enter a command directly (e.g. “2 + 2”)
- identify the file window
 - save a script as a .R file
 - run a single line of the script by pressing CTRL + ENTER
 - run the whole script
- identify the environment window
- identify the files/plots/etc window

- know how to install and load a library of functions (“a package”)
 - install the swirl package
- open this script
 - review line by line [as a pair exercise ideally]

Chapter 3

Making Graphs

3.1 Class material

Note: most of this class talk in workshop style, so any recordings are short and the slides are brief

The live-coding demo used these files: `class3a.R` and `class3b.R`. These are based on Healy (2018) chapters 2 and 3 respectively.

3.2 Resources

This book is recommended for the course, and for this lecture in particular:

Healy, K. (2018). Data visualization: a practical introduction. Princeton University Press.

- The whole book is available online via the link
- There are supporting materials here: <https://github.com/kjhealy/socviz>

3.2.1 Other resources

- We are going to be using the `ggplot` package for our visualisations: `ggplot2`
- Five Charts You've Never Used but Should
- guru99: Scatter Plot in R using `ggplot2` (with Example)
- `ggplot2`: Elegant Graphics for Data Analysis, the chapter on Themes
- Giulia Ruggeri, Better data communication with `{ggplot2}` Styling a chart and improving the text using `{forcats}` and `{ggtext}` and `{scales}`, Better data communication with `{ggplot2}`, part 2

- datanovia.com: GGPlot Axis Ticks: Set and Rotate Text Labels

3.3 Exercises

Review the introduction and Chapter 1 ‘Look At Data’ of Healy (2018)

Work through Chapter 2 ‘Get Started’ from 2.3 onwards (“Things to know about R”), we are ignoring the material on RMarkdown for now (we’ll get there, see here).

Work through Chapter 3 ‘Make a plot’

3.4 Checklist

Completed the exercises

- especially section 3.8 ‘Where to go next’ in Chapter 3 ‘Make a plot’

Key concepts (chapter 2):

- variables and functions
- variables are typed (e.g. can be numeric or character)
- and how you detect what type a variable is
- indexing for variables and data frames

In the console, understood

- the shortcut keys for the assignment operator `<-`
- the use of tab for autocomplete
- the use up arrow to cycle through previous commands

Key concepts (chapter 3):

- Tidydata
 - long vs wide data
 - values and keys
 - data variables can be continuous or factors
- plots have data, aesthetic mappings (‘aes’, ‘mappings’) and layers
- adding mappings, adding geoms, geom properties, other layers (e.g. labs)
- mappings are inherited by geom layers

Chapter 4

Project Organisation

4.1 Project folders

You should organise all the documentation, files, data, code and outputs for a project in a single folder. Your aim is that when you come back to this project, or want to share it, everything you need is in one place and sensibly organised.

Create subfolders for ‘data’, ‘graphs’, ‘notes’. Michael Frank’s onboarding guide explains a bit more about the logic of this.

4.2 Naming Things

Files come in different types. This is indicated by their name, not the content. Turn on the options which means you file viewers the extensions of files (so you can see the full filename is `MyProject.Doc` not just `MyProject`). How to do this: PC, Mac.

Because you will want to share this project use names that make sense to other people, as well as to you. `MyProject.Doc` is a bad name. `joeblogs_psy64222.doc` is better.

Computers sometimes care about capitalisation, which means you need to pay attention : `MyFile.doc` and `myfile.doc` may not get treated the same. Computers also sometimes care about spaces in filenames. You can include these in your filenames and wait until this messes things up later (maybe much later, maybe even never), or you can decide now to be careful about capitalisation and never use spaces in your files names. Replace spaces with underscores `My File.doc` becomes `my_file.doc`

R/RStudio may not work well if you are using Windows and have Chinese characters in your username. You can avoid this by using the latin alphabet (thanks to [this](#) for this tip)

“c:/users/chuanpeng” ✓✓✓

“c:/users/ ” ×××

Never include special characters (e.g. &).

These rules also apply to variables in computer code.

Jenny Bryan has good thoughts on Naming Things

4.3 Research Data Management

Everything recorded during an experiment, whether by you or the computer, is data. All log files. Everything.

Never delete *any* data.

Never edit the raw data files. They should exist in a folder called \raw or \data and only ever be opened, never modified in any way.

All data should be backed-up automatically, as soon as it is collected. For how to do this, see the next header

4.4 Include meta-data

When you share data you *must* include descriptions of where the data comes from and how it can be interpreted. This is sometimes referred to as *data dictionary* or *code book*. The general category is *meta-data* - data about the data.

Typically I include a file called `codebook.txt` in the same folder as the data. You can see this in the example projects shared during the module.

In your codebook you should say briefly where your data comes from (how it was collected, who by etc). You should explain what the variables are. Usually variable columns are labelled with abbreviated or compressed names. The code book is your chance to explain in full what these abbreviations mean. You should also explain what the values or levels are of the difference variables. So, for example, if you have a column called `gender` in your data. A line in your codebook might be:

```
gender : self declared gender of participant. Values are 1 =
female, 2 = male, 3 = other, 4 = prefer not to say
```

A codebook should also say how any missing values are coded and any other quirks of the data which could affect someone else's onward use.

If you are using secondary data (i.e. data from someone else) it may come with an existing codebook and you should include this in your own project, along with your copy of the data. Make notes, either in an additional, addendum, codebook or in the comments on your analysis code, about any additional considerations which should be noted about the data.

4.5 Backup and Sync

Any back up process which requires you to remember to deploy it is fragile. Use a cloud synchronisation service. Many are available, like OneDrive or Dropbox, but since you are in the University of Sheffield you should install the Google Drive Backup and Sync tool.

As well as backing up your work to the cloud, this will allow you to work on multiple machines (because changes are automatically carried across from one to the other), as well as collaborate (because multiple people can add to the same shared folder and any changes are instantly available to the whole team).

4.6 Resources:

- Jenny Bryan's presentation on Naming Things
- Michael Frank's onboarding guide
- Tidy Data organisation
- Habits and open data: Helping students develop a theory of scientific mind
- Broman & Woo (2017) Data Organization in Spreadsheets
- University of Sheffield pages on Research Data Management
- Arslan, R. C. (2019). How to automatically document data with the codebook package to facilitate data reuse. *Advances in Methods and Practices in Psychological Science*, 2(2), 169-187.
- Jenny Bryan: I love the here package. Here's why.
- Why should I use the here package when I'm already using projects?
- `rstats.wtf` : Project-oriented workflow

4.7 Exercises

4.7.1 Warm up: Pair programming exercise

: Write a script to create and save a histogram of life expectancy by continent, using the `gapminder` data (see slides for more notes)

4.7.2 Homework: Project 1

Create project containing script to generate plots from this data: `anscombe.csv` (template script here; find completed example on google drive)

Share this project with another student on the course. 2022 see this file for who

Run the project shared with you, complete this checklist 2022: deadline 0900 on 2nd of March.

Review the questions in the library Research Data Management plan for post-graduate researchers

4.8 Checklist

in general

- * Be able to organise and share project work

specifically

Understand & Use

- * file extensions, file paths
- * standard file naming conventions
- * ISO 8601 date format
- * standard folder organisation for projects

Appreciate the need for a research data management plan

Map network drives / automate backups

You should be able to

- * download a file to your Desktop (e.g. this one) and locate it on your hard drive
- * report the full path of a file
- * open a csv file as plain text
- * share a project folder with someone else

Module project

These pages are about the final project, which comprises most of the module grade. There are many useful notes below, which you will want to refer to as you complete your project

In this class we covered how to identify data for your final project for this module. We also discussed the project format and assessment criteria. Unlike other classes there is no checklist of learning outcomes.

The final project for the course contributes the bulk of your grade, as well as creating a portfolio piece for you to demonstrate what you learnt on the course. The final project is a chance to deploy all the skills taught on the course in a single complete data visualisations project, which takes us from raw data and a question through data preparation and exploration through to publishing a visualisation and the documentation which shares your findings with others. The ambition is that a project is complete and reproducible, i.e. it contains the input (the data), the output (like these max and min values, or a graph), documentation (explanation of what is what) and the whole is stitched together with an R code - the script that changes the input to the output. I should be able to run your script on my computer and produce the same output.

4.9 Project requirements

A complete project, including raw data, analysis scripts, documentation and visualisations shared in a way that allows the visualisation to be reproduced and understood

In practice, this means you should upload a project to github, containing a Rmarkdown file which explains your visualisation, as taught in the module (although alternatives are acceptable if they meet the primary criteria).

Please include these headings, or something similar, in your project.

- Data Origins

- Answering questions of where it comes from, how it was collected, what the variables mean, etc
- show the first few rows of the raw data, if possible
- Research Questions
 - in plain english, a simple statement of what question(s) your visualisation will attempt to address
- Data Preparation
 - steps taken to clean the data, exclude outliers, create summary statistics, grouped variables, etc
 - show the first few rows of the processed data, if possible
 - showing the code which does this were relevant
- Visualisation(s)
 - graph or graphs
 - documentation explaining any motivation (although good graph labelling is better than explanation in the accompanying text)
 - code for producing them
- Summary
 - Brief thoughts on what you have learnt, what you might do next if you had more time / more data

The ultimate aim is to make something that looks good and shows off your abilities. You want to produce something you are proud to show people (including future employers and supervisors)

4.10 Choosing your data

The data is an excuse for the data visualisation, so don't agonise over this, and go for simple over complex if you have a choice. I recommend you find a data set with variables which you understand.

4.10.1 Finding Data

Many papers are now published accompanied by open data. Sometimes this is indicated by an Open Data Badge. For example, take a look at the current table of contents for Psychological Science. The badges are an initiative of the Centre for Open Science

You can also search a preprint archive such as psyarxiv. Many, but not all, papers published as preprints will have open data. You can also search OSF.io directly.

Pick a topic and have a look at what is available

The other route is large publicly available resources:

- such as Our World in Data.
- Another good source of diverse, open and current data sets is the Data Is Plural newsletter
- Enjoy this crowdsourced list of Open psychological datasets
- Kaggle
- Journal of Open Psychology Data
- The General Social Survey (GSS) (US)
- Datasets APIs and open source projects related to Climate Change

4.10.2 Suggestions for possible project data sets

These are examples of large, probably messy, real-world data sets which could make good starting points for projects:

- freeCodeCamp.org survey of new programmers
- StackOverFlow survey data
- BEHACOM - a dataset modelling users' behaviour in computers
- The World Values Survey "is the largest non-commercial, cross-national, time series investigation of human beliefs and values ever executed, currently including interviews with almost 400,000 respondents."
- AnAge Database of Animal Ageing and Longevity
- The General Index - detailed linguistic data derived from 107,233,728 academic journal articles
- components.one - various tech & media datasets

4.11 Choose your graph

You only need to draw one graph, but more are permitted.

The important thing is to show - either in the graph, or in text, or both - that you have thought about the logic of what you are visualising, *how* it shows the information and *why* this important

You may find this helpful, a guide to different kinds of graphs and when they are appropriate, made by journalists at the Financial Times: A visual vocabulary

4.12 Check list

Here are some questions to ask yourself as you prepare your project

- do I care about this topic and/or data?
- do you have access to the raw data?
- can you load it?
 - if you can open it in a spreadsheet you will be definitely be able to open it in R, but you will still want to check if it is well formatted when you load it in R
 - * for example, if different cells have different amounts of information in your data won't import nicely
 - * or, if you important information is contained in cell colours then this won't be available when you import into R
- Do you understand it?
 - is there a code book?
 - do you know what the columns are?
 - do you understand the values?

4.13 Answers to common questions

1. it doesn't matter if you don't do the things you said you wanted to do in the initial self-assessment
2. you can use the data from another course, for example the course project (psy6009) 2b. if you do use the data from your research project make sure you are clear how this module project is different. I recommend not using the data from your research project.
3. the data / data questions doesn't have to be something which expresses the relationship between two variables. Any visualisation which shows something non-trivial about the data is good.
4. You don't need to do any statistical tests or model fitting. You don't need to have a set amount of data to meet statistical significance, just enough to be interesting when visualised.
5. You don't need more than a brief introduction to the data - no literature review please.
6. This is a visualisation project, not an analysis project, so you don't need to do statistical analysis. You will, however, find that a clear question or set of questions helps develop a good visualisation and/or that a good visualisation helps clarify an unclear question or set of questions.
7. There is no word limit, but remember - for a project which has maximum impact short is sweet (i.e. try and show off quality of work without quantity of text)
8. You can reuse someone else's data - this is not plagiarism. You can even redo someone else's analysis. What you **can't** do is copy and paste someone's code for your project
9. Your project doesn't have to be on a psychology/neuroscience topic, but bear in mind that domain knowledge makes a better data visualisation

project, and you - as psychology students - have domain knowledge about psychology.

10. The data does not have to be very large or very complex. Better a simple, clear, project than a complex, sophisticated project which is harder to understand. ## Getting help

Simple questions should be posted on slack, ideally in the appropriate channel. This means that any advice you get (or give) is shared with the whole class, so everyone can benefit and we avoid needlessly repeating the same mistakes as individuals. During the semester I add answers to asked questions to the Q&A document ([link here](#)). At the end of the semester I will update the webpage so all the information you need is on these pages.

For more complex problems, you will want to share your whole project - this means not just the R or Rmd file you are working on, but the data file(s) as well, as well as an auxillary notes. Please create a folder, following the project organisation scheme we covered in week 2 with subfolders for the data, figures, etc and including the .Rmd file. Share this with me and I can help.

You can either share a google drive folder or a github repo. Doing this means I have all the parts required to reproduce your issue. It also means that if you update your project in between contacting me and me looking at it then I see the very latest version, rather than working on an outdated problem.

Note that the deadline for assignments is the after the weekend, so I will not be answering questions in the days immediately before the deadline. If you want to ensure an answer to your question please ask it so here is at least one full working day before the deadline (i.e. if the deadline is after the Weekend, please ask by 5pm on the thursday).

4.14 Example Projects

Here is an example small project which gives an idea of what I mean

- SuperTues: Published, repo

Also, please look at prior project completed by students on this course:

- Class of 2020 projects.
- Class of 2021 projects.

4.15 Stretch goals

More ambitious projects, using more advanced techniques, are possible for students who are further along with coding. These include techniques like database (SQL) queries, use of APIs, web-scraping or interactive plots

An accessible way to build interactive data visualisations is using the Shiny package Here's one built for previous research project I worked on : Power analyser

4.16 In class exercises

Exercises for the class include

- In pairs, discuss what visualisation you have planned for the module project.
- In pairs, review the checklist. Ask each other, and answer as far as you can, these questions?
- In pairs, discuss the assessment criteria. Are you clear on what is needed to get marks on this project?
- Whole class, discuss suggestions for improving the assessment criteria
- Whole class, review and ask questions about the FAQ document and guidance on module projects

Chapter 5

Data Wrangling

This class is about loading, cleaning, inspecting, merging and summarising data using the Tidyverse library for data science. The majority of most data science projects is getting the data into the best form for plotting or analysis, what is called data cleaning, data munging or data wrangling.

5.1 Class material

There are no slides for this class, it is taught entirely in workshop format.

5.2 Exercises

Find the folder `/modelproject_asrs` in the google drive

- review the project organisation and files
- open the file `datawrangle_exercises.R`
- complete the tasks described

5.3 Checklist

You should leave being familiar with these concepts, and known where to look up how to implement them. Indented bullets are more advanced topics (not always covered in the exercise)

- Load data from CSV

- load data from an excel file
 - load data from a TSV file (tab separated values)
- Recognise common tidyverse functions (below)
- Use the pipe operator %>%
- Mutate: new columns by combining old ones
 - Use string functions on column values (e.g. separate on characters, take substrings)
- Select: select columns
- Filter: select rows by values
- group_by and summarise
 - Understand this is a specific example of a general method of split-apply-combine
- rename: to change column names
- convert variable types, e.g. using as.numeric
- join: to merge data frames, requires a common key between data frames
 - understand inner joins, left and right joins

Almost any data wrangling task you can imagine can be done, you just need to find the right function or functions. So the final item for this classes checklist is

- practice searching for solutions to your data wrangling problems

5.4 Resources

- Data Skills for Reproducible Science: Data Wrangling
- Claudia A Engel: Data Wrangling with R
- R for Data Science: Part II Wrangle, especially Chapter 12 Tidy Data and Chapter 13 Relational Data
- datacarpentry.org Data Wrangling with dplyr and tidyr
- RStudio: data wrangling cheatsheet (PDF)

Chapter 6

Coding Principles

This class is about two kinds of fundamental principles of coding. The first is fundamental methods of making code do what you want - if statement, loops, functions. The second is fundamental principles of good code. Although we are using R, all programming languages use similar methods (although the exact syntax differs), and the principles of good code will also apply across languages.

As well demonstrating these fundamentals, these pages also introduce the vocabulary used to discuss them. Knowing the vocabulary helps because it means you know what terms to use when searching for solutions to problems you have.

6.1 Fundamental methods

1. if statements
2. loops
3. functions

6.1.1 If statements

So far we have written simple scripts that do things in order, top to bottom

```
a <- 1 # define a variable
a <- a + 1 #add 1
print(a) # output the result
```

```
## [1] 2
```

The first block above is the code, the second block (the lines which start with `##`) is the output.

Changing which statements are run is called “flow control”. An “If statement” is a fundamental way of doing this. It allows us to specify one set statements to run if a certain conditions is met. For example

```
a <- 1 # define a variable
a <- a + 1 #add 1
if(a>4) # this is the condition which has to be met, the 'test expression'
  {print(a)} # this statement runs if the test expression is true
```

Notice there is no output. Copy the code to your own computer and run it. Now change the first line to `a <- 9` and run it again.

An If statement defines a branch in the flow of a script. The default can be nothing happening, but sometimes you want to define two alternatives. You can do this with an “If...else...statement”

```
a <- 1 # define a variable
a <- a + 1 #add 1
if(a>4){ # this is the condition which has to be met, the 'test expression'
  print(paste(a," is more than 4")) # this statement runs if the test expression is true
} else {
  {print(paste(a," is equal or less than 4"))} # this statement runs if the test expression is false
}
```

```
## [1] "2 is equal or less than 4"
```

You can actually have as many branches as you like, defining a series of test_expressions, like this

```
type_of_thing <- ''
print("Is four a lot?")
if (type_of_thing=='Murders'){
  print("yes")
} else if (type_of_thing=='Dollars'){
  print("no")
} else {
  print("Depends on the context")
}
```

```
## [1] "Is four a lot?"
## [1] "Depends on the context"
```

6.1.2 Loops

Loops repeat, either iterating over a set values, like this:

```
for (i in 1:5){  
  print(i)  
}
```

```
## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5
```

Or until some condition is met

```
i <- 1 #need to initialise a starting value  
while(i<6){  
  print(i)  
  i <- i + 1 # increment the value of the counter  
}
```

```
## [1] 1  
## [1] 2  
## [1] 3  
## [1] 4  
## [1] 5
```

Note that this second version, a “while loop” uses a test expression just like an if statement

Loops are useful wherever you might want to repeat some operation.

```
years <- 10 #how many years since you started saving  
savings <-100 #how much you start with  
interest <- 1.05 #rate of interest, ie 5% interest  
#Calculate using a loop  
for (year in 1:years){  
  savings<-savings*interest  
}  
print(paste("After", years, "years you will have £", round(savings,2))) #save more, kids  
  
## [1] "After 10 years you will have £ 162.89"
```

Lots of people advise against using loops because they can be slow and it isn't always obvious what they are doing. Alternatives often exist, like vectorisation:

```
years <- 20 #how many years since you started saving
savings <-100 #how much you start with
interest <- 1.05 #rate of interest, ie 5% interest
#Calculate using a vector
total_at_each_year=savings*interest**(1:years) #rather than a loop all the answer values
#plot(total_at_each_year,xlab="years") #bonus! We can plot, since we now have all the
```

The problem is, loops are the natural way to think about some problems. Often I first write my code with loops then, when I know what I really want to do I try and work out a way to do it with vectorisation.

6.1.3 Functions

Functions take in values (called “arguments”), do something with them, and give a value or values back in return. You have already used functions, for example the mean function

```
my_nums <- c(78,12,32,24,03,89) #just a vector of some numbers
mean(my_nums) #use the mean function to find the average
```

```
## [1] 39.66667
```

Functions always do the same thing, but give different results depending on the inputs (depending on the “arguments you pass to the function”).

You can write your own functions, and then use them (“call them”) again and again. Here is the general form of a function

```
myfunctionname <- function(input_value) {
# comment line helpfully explaining what the function does
output_value <- input_value #lines of code which do something to the input to produce
return(output_value)
}
```

Note a couple of things: when you run this code it does not produce any output, but a new object appears in the “global environment” window, top right. Like a variable, your function is now stored in the memory of the current R session.

You can call this function now. If you close R you'll need to define the function again by running the above code again (other functions are inbuilt, like `mean` and are loaded at startup, or when you use the `library` command to load a set of functions).

Now, when we call the function, we pass actual values.

```
myfunctionname(3)
```

```
## [1] 3
```

Let's make our function slightly more complicated

```
outcheck <- function(val,threshold) {
  # outlier checker
  if(val<threshold){
    output_value <- val #if value is below threshold return that value
  } else {
    output_value <- NA #otherwise, return NaN
  }
  return(output_value)
}
```

This function takes two input values, and returns a single value which depends on the relation between the two

```
outcheck(3,5)
```

```
## [1] 3
```

```
outcheck(7,5)
```

```
## [1] NA
```

6.1.3.1 A note about scope

Variables within functions are kept 'inside' the functions (within the "scope" of the function). Once you pass a value to a function it acquires the label set in the function definition. Variables defined within the function don't persist outside of it (they don't affect the "global environment")

So, for example, it doesn't matter if you have another variable called `threshold`, the threshold within the function is set by the second value passed it. Like this:

```
threshold <- 100
outcheck(7,5) #returns NA because 7 is higher than 5
```

```
## [1] NA
```

6.1.4 Exercises 1

- Write an if...else statement that prints “ODD” if the number is odd, “EVEN” if the number is even (hint: you might use the remainder function %% (try 4%%2 to see how much is left when you divide 4 by 2))
- Write a loop which goes from 10 to 20 in steps of 3
- Write a function which prints “FIZZ” if a number is divisible by 3, and “BUZZ” if it is divisible by 5 and “FIZZBUZZ” if it is divisible by 3 *and* 5
- Write a loop which counts from 1 to 100 and applies the fizzbuzz function to each number

6.1.5 More

- Lisa DeBruine, & Dale Barr. (2019, December 5). Data Skills for Reproducible Science (Version 1.0.0). Zenodo. <http://doi.org/10.5281/zenodo.3564555>: Iterations & Functions
- datamentor.io on Flow control

6.2 Fundamental principles of good code

6.2.1 Readability Matters

Your most important collaborator is you from six months ago, and they don’t answer email.

Good code doesn’t just work, it is easy to understand. This supports the code being checked for errors, modified and improved (by you as well as by other people).

To support this you should make your code readable. This means commenting your code, but also laying it out nicely, and using sensible names for variables and function. The aim is to make the code explain itself, as well as doing something. Someone who reads your code - a future you maybe, or a collaborator - needs to be able to run the code, yes, but they also need to know what you are doing and why you are doing. Look at this function, it hard to understand, right?

```
pf <- function(n){ p=1 ; if (n>1){ i = 2; while( (i<(n/2+1)) & (p==1) ) {if (n%i ==0)
```

This kind of code is very compressed. You can fit a lot in a few lines, but it is useless because nobody else will understand it, and probably the person who wrote it won’t understand it when they come back to it (and that means they will miss any bugs, or will find it hard to improve or repurpose).

Readability is improved a lot by adding some spacing and tabs. Have another go at figuring out what the code does:

```
pf <- function(n){
  p=1
  if (n>1){
    i = 2
    while( (i<(n/2+1)) & (p==1) ) {
      if (n%i ==0) {
        p=0
      }
      i=i+1
    }
  } else {
    p=0
  }
  return(p)
}
```

Now we make the variable and function names sensible:

```
primecheck <- function(num){
  isprime=TRUE
  if (num>1){
    i = 2
    while( (i<(num/2+1)) & (isprime==TRUE) ) {
      if (num%i ==0) {
        isprime=FALSE
      }
      i=i+1
    }
  } else {
    isprime=FALSE
  }
  return(isprime)
}
```

Can you tell what it does yet?

Now fully commented

```
primecheck <- function(num){
  #check if a number is prime
  # - assumes the number provided is an integer
  # - works by working through all possible divisors up to half the test number, checking if the
```

```

#
isprime=TRUE # a flag, which tracks if we think the number is prime. We start out as
# first we only need to do the complicated method for numbers great than 1
if (num>1){
  i = 2 #a counter, starting at 2 (because all numbers divide by 1)
  #use while loop to check all divisors until we've done them all or we find one (
while( (i<(num/2+1)) & (isprime==TRUE) ) {
  if (num%%i ==0) {
    #if the number divides by another number with no remainder it can't be prime, so w
    isprime=FALSE
  }
  i=i+1 # increment the counter, so we work through all possible divisors
}

} else {
  # if the number is 1 or lower it can't be prime, so we change the flag
  isprime=FALSE
}
return(isprime) #return the flag as the output of the function, 0 -> not prime, 1 -> p
}

```

It is possible to comment too much. The code above I commented so someone who wasn't an experienced programmer could read the comments and it would help them understand how the code worked (you can tell me if I succeeded). Usually a few fewer comments might make the code easier to read, with the assumption that anyone reading it has a bit of experience with the coding language. Like this

```

primecheck <- function(num){
  #check if a number is prime
  # - assumes input is integer
  isprime=TRUE # a flag, start assuming our number *is* prime
  # only check numbers > 1
  if (num>1){
    i = 2 #a counter
    #check all divisors until we've done them all or we find one
while( (i<(num/2+1)) & (isprime==TRUE) ) {
  if (num%%i ==0) {
    #no remainder -> number isn't prime
    isprime=FALSE
  }
  i=i+1 # increment the counter
}

} else {

```



```
# if the number is 1 or lower it can't be prime
isprime=FALSE
}
return(isprime)
}
```

This version is 22 lines rather than 1, but I hope you agree it is easier to work with. There's no shortage of space in R scripts, so if in doubt, put some effort in to laying things out nicely, use sensible names for variable functions and add comments. You'll thank yourself when you come back to your code (which you will always have to).

6.2.2 Avoid hard coded values

Say you were going to load some data, you could do this:

```
mydata = read.csv('/home/tom/Desktop/psy6422/mydatafile.csv')
```

Now this happens to work on my computer, but it won't on yours. The reason it won't work isn't because there is a bug in how i'm loading data, just that you don't have a file in the same place as I do. Far better, for both readability and debugging if you separate out values that might change from the commands that use them.

Like this:

```
datafile = '/home/tom/Desktop/psy6422/mydatafile.csv'
mydata = read.csv(datafile)
```

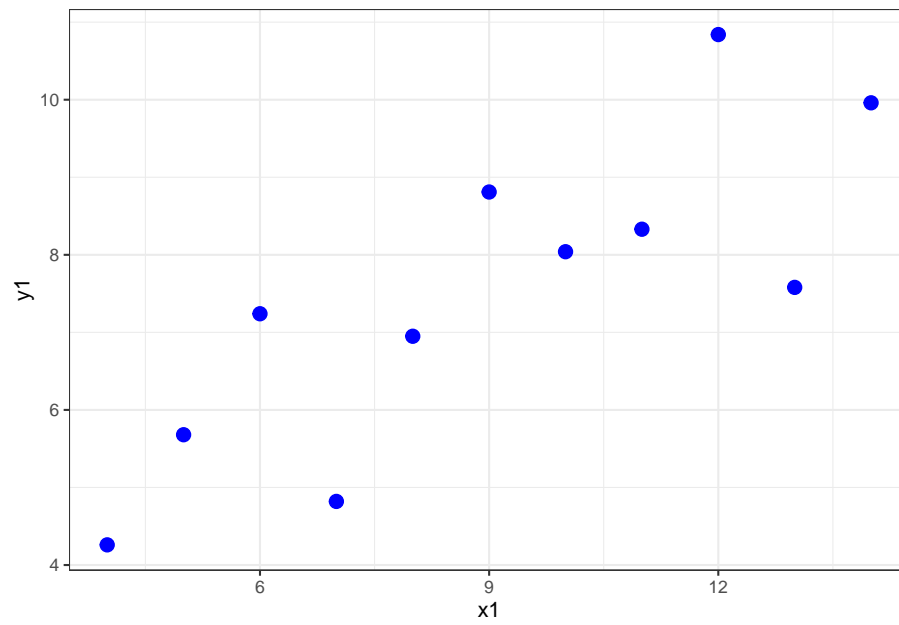
Now the second line is easier to read, and you also have a variable which you can reuse. For example maybe later in your script you want to save the name of the raw data file somewhere. You can just use:

```
label = paste('This plot generated using data from ', datafile)
```

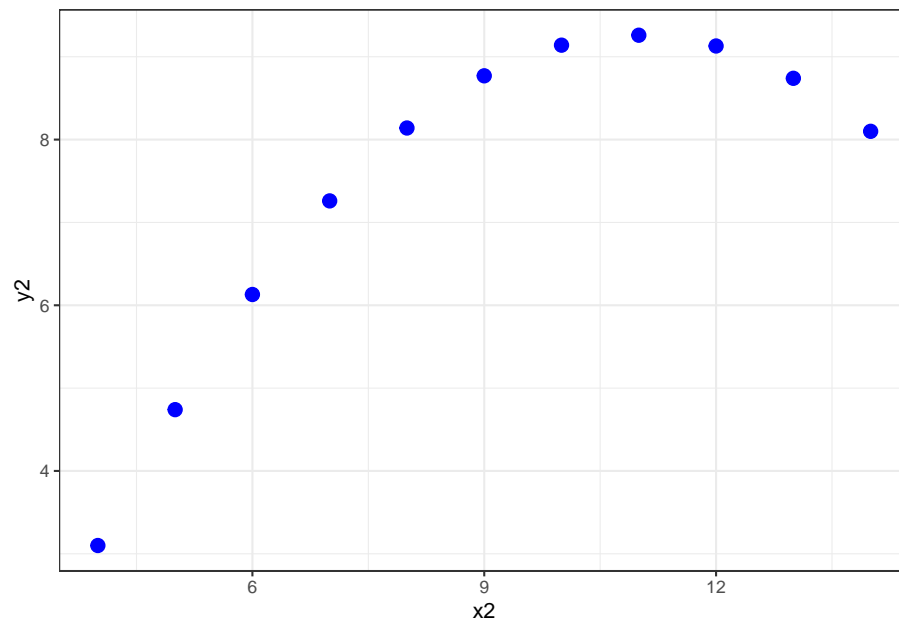
And when you use the same script for different data, both the lines loading data and recording the data file are correct.

Another example, suppose you had two plots:

```
graph1 <- ggplot(data = anscombe, mapping = aes(x = x1, y=y1))
graph1 + geom_point(color='blue',size=3) #change this line for different look
```



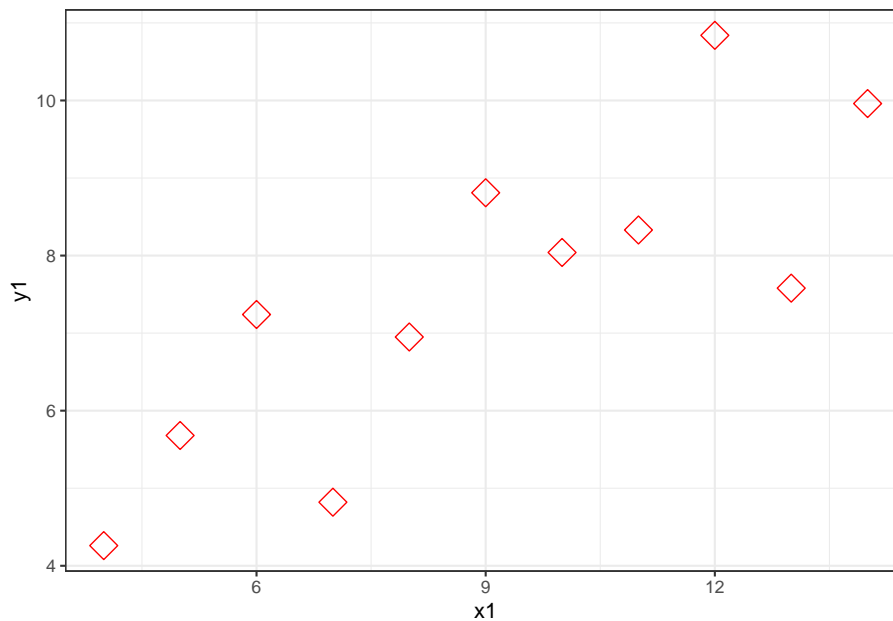
```
graph2 <- ggplot(data = anscombe, mapping = aes(x = x2, y=y2))  
graph2 + geom_point(color='blue',size=3) #change this line for different look
```



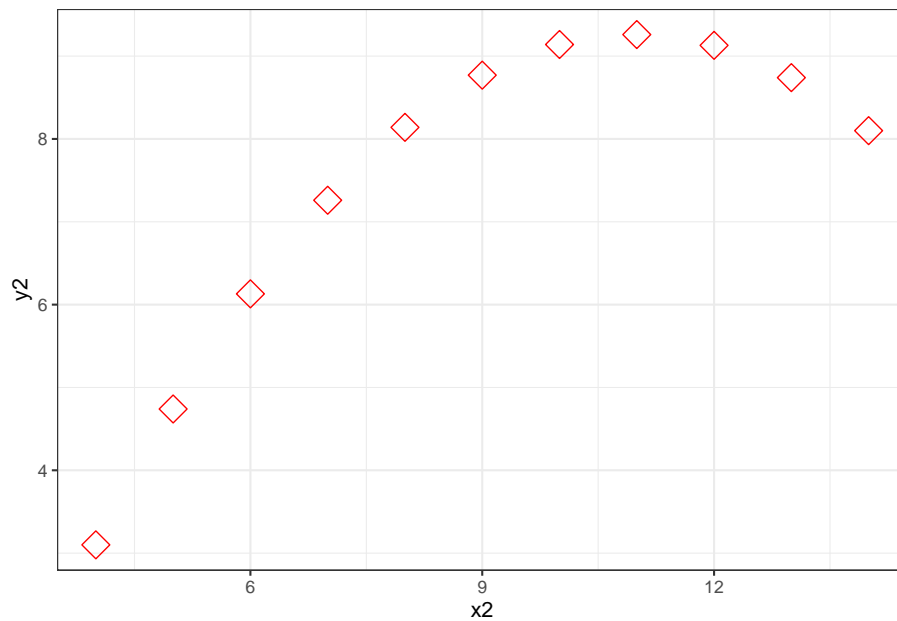
Adding variables means you only need to edit one line to change the look of both plots

```
pointcolour='red'; pointsize=5 ; pointshape = 23 #change this line for different look
```

```
graph1 <- ggplot(data = anscombe, mapping = aes(x = x1, y=y1))  
graph1 + geom_point(color=pointcolour,size=pointsize, shape = pointshape) # never change these l
```



```
graph2 <- ggplot(data = anscombe, mapping = aes(x = x2, y=y2))  
graph2 + geom_point(color=pointcolour,size=pointsize, shape = pointshape) # never change these l
```



This may seem minor, but as your code gets longer developing habits like this will save you time, and make your code easier to work with.

6.2.3 Functionalise & Generalise

If you ever find yourself using very similar lines of code, you should think about making a function. Functions make your code shorter and easier to read (and write), and they make it *way* easier to update (because when you catch a bug you can just update the code in the function, rather than every time you repeated those lines).

Functions are also an opportunity to think to yourself “what is the most general purpose way of doing what I’m doing”. Thinking like this will help you develop powerful, flexible, code which you can use to do multiple things.

Let’s look at a toy example:

```
mynumbers = c(2,3,4)

#double and add one to each number
mynumbers[1] <- mynumbers[1]*2+1 # line 1
mynumbers[2] <- mynumbers[2]*2+1 # line 2
mynumbers[3] <- mynumbers[3]*2+1 # line 3

print(mynumbers)
```

```
## [1] 5 7 9
```

This can be improved with a function

```
myfunc <- function(num){  
  #toy function, doubles and adds 1  
  return(num*2+1)  
}  
  
mynumbers = c(2,3,4)  
  
mynumbers <- myfunc(mynumbers) # all the work with 1 line!  
  
print(mynumbers)
```

```
## [1] 5 7 9
```

This code is easier to read, easier to change, and you can write new code which uses this function again.

6.2.4 Ask for help

Nobody finds this easy straight away. Learning how to find help a core programming skill (along with not giving up when it feels like you are completely stuck).

Part of this is knowing how programming people talk about stuff so you can search effectively for solutions.

If you get an error message, copy and paste it into your search.

If you are really stuck, just trying to describe your problem is a good way of identifying exactly what you want to do, and why you can't. This often means creating a 'minimal reproducible example' - the smallest script possible generates the error you're trying to solve. Often the process of working on this helps you solve the problem yourself, but if it doesn't it means you have a clear way of showing someone else what your issue is. See this [How to make a great R reproducible example](#) and this [How do I ask a good question?](#). Now you have your minimal reproducible example you can ask a friend, teacher, or post it to a forum.

If you're on this module you can do this by posting it to Slack, or if not try seeking out R groups in your city or institution. Shout out to Rladies

6.2.5 Exercises 2

- In pairs, take one person's code from a previous week's project and review for readability, using the **better code** checklist below. Write an improved version together
- In pairs, try these 'Parsons Problems' **only try the R problems** not the python (.py) ones
- Review these articles on how to make a minimal reproducible example: How to make a great R reproducible example & How do I ask a good question?.
- Make and share on slack a minimal reproducible example of your next R problem

6.2.6 Checklist

control flow

- if statements
 - if ...then statements
 - if ...then .. else statements
- for loops
- while loops
- functions
 - scope

better code

- readable code
 - indents and whitespace
 - sensible variable names
 - comments
- hard coded values aid extension, efficiency and reproducibility
- functions make code easier to read, easier to modify, reduce the likelihood of bugs

ask for help

- google the error message
- make a minimal reproducible example

6.2.7 Resources

- Patrick J Mineault & The Good Research Code Handbook Community (2021). The Good Research Code Handbook. Zenodo. doi:10.5281/zenodo.5796873
- Function tips
- Vocabulary: Names for common programming symbols
- Program better, for fun and for profit
- Prime Hints For Running A Data Project In R
- Software Carpentry: Best Practices for Writing R Code
- Nice R code: bad habits
- The Good Research Code Handbook [python orientated]
- Barnes, N. (2010). Publish your computer code: it is good enough. *Nature*, 467(7317), 753-753.
- Axelrod, V. (2014). Minimizing bugs in cognitive neuroscience programming. *Frontiers in psychology*, 5, 1435.
- Wilson, G., Aruliah, D. A., Brown, C. T., Hong, N. P. C., Davis, M., Guy, R. T., ... & Waugh, B. (2014). Best practices for scientific computing. *PLoS biology* 12(1), e1001745.
- r-bloggers.com Looping through variable names and generating plots
- r-bloggers.com Coding Principles: Updating and Maintaining Code
- Advanced R by Hadley Wickham: Style guide

Chapter 7

Rmarkdown

You need to be able to share your analysis. Comments are good for making code readable, but often you will want longer sections of text, mixed in with both the code you are running and the outputs of the code (e.g. the plots you are making with it). Do this with RMarkdown.

A RMarkdown file is a plain text file, like a R script, but with the extension `.Rmd` rather than `.R`. You can make one now in RStudio by clicking File > New File > R Markdown. Or by saving an existing `.R` script as `filename.Rmd`.

RMarkdown is the system which marks how text should look after it has been converted into a webpage, or PDF, or some other kind of document. The name RMarkdown is kind of a joke, since RMarkdown is a version of a “markup language”. Markups are the opposite of the WYSIWYG systems (like MS word or Google Docs) which you are used to.

These pages are written in Rmarkdown. You can see this individual file here in the online repository. It will really help if you read the webpage alongside the file, so you can compare the file which generates the text, using the markdown, and the output (the webpage).

Rstudio magic (called “rendering” or “knitting”) turns this file in to the webpage.

Before you can turn your `.Rmd` files into PDF you need to install TeX on your machine: use this download link

Compare the file and the webpage.

In the webpage this line is in bold. Why?

In the webpage this line is in italic. Why?

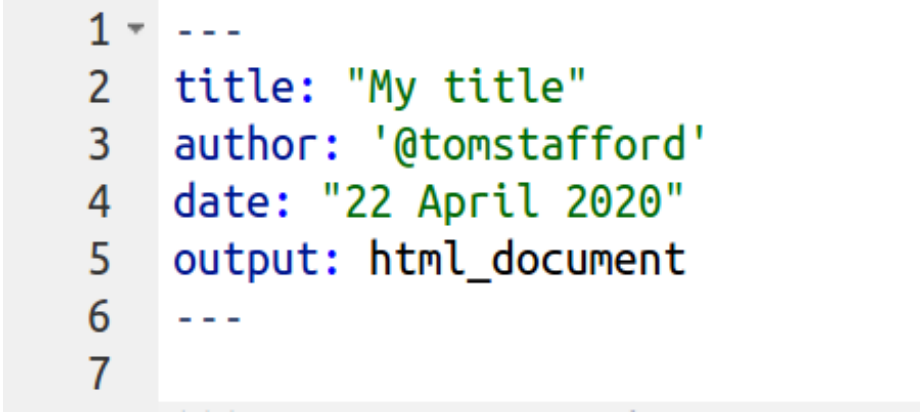
7.1 And this line is a heading

Try now creating your own Rmarkdown file by clicking File > New File > Rmarkdown in RStudio.

RMarkdown files have three components.

7.2 First a header

Aka “header material”. It looks like this, at the top of the file:



```
1 ---
2 title: "My title"
3 author: '@tomstafford'
4 date: "22 April 2020"
5 output: html_document
6 ---
7
```

Figure 7.1: Yaml matter at the top of a .Rmd file

This is called YAML and it is stuff meant to be read by the computer when the file is converted into a document to be read by humans. You can see that this is meant to be an “HTML” document (that’s the kind on a webpage), so let’s make it now. Click “knit” in RStudio (or “knit to HTML” if you are exploring the options menu).

(you’ll be prompted to give your .Rmd file a name first, if you haven’t done this already).

After a brief pause you should get a new window open, containing something that has some of the same words as your document. Notice how the YAML stuff has disappeared, and the new document now contains formatting (bold, italics, headings, etc).

Part of the benefit of markdown is that you write the document once, and can convert it to a webpage, or a PDF, or a MS Word document. Try now. Click “Knit” and select “Knit to PDF”. You get a nice PDF document, looking almost, but not entirely, like the webpage you made moments before.

7.3 Second, text

If you just write stuff in a `.Rmd` document you get text. This is the second kind of thing in a `.Rmd` document, like this.

It can contain formatting - *italics*, **bold**, etc - as well as stuff like lists and hyperlinks:

- See a long list of formatting options in this cheatsheet
- The most common options here: [Markdown Basics](#)
- This line just to demonstrate that this is a list

But the real strength of Rmarkdown is you can mix text and code

7.4 Third, code

This is the third ingredient, code and any output it produces. Like this:

```
print("Here is some R code")
a <- 6
b <- 2.3
print(a/b)
print("And the output it produces")
```

```
## [1] "Here is some R code"
## [1] 2.608696
## [1] "And the output it produces"
```

Here is another example

```
#Code to make an example graph
library(tidyverse)

#load some data

#Method 1. Load from Tom's computer
#filename <- '/home/tom/Desktop/psy6422/mydatafile.csv'
#df <- read.csv(filename)

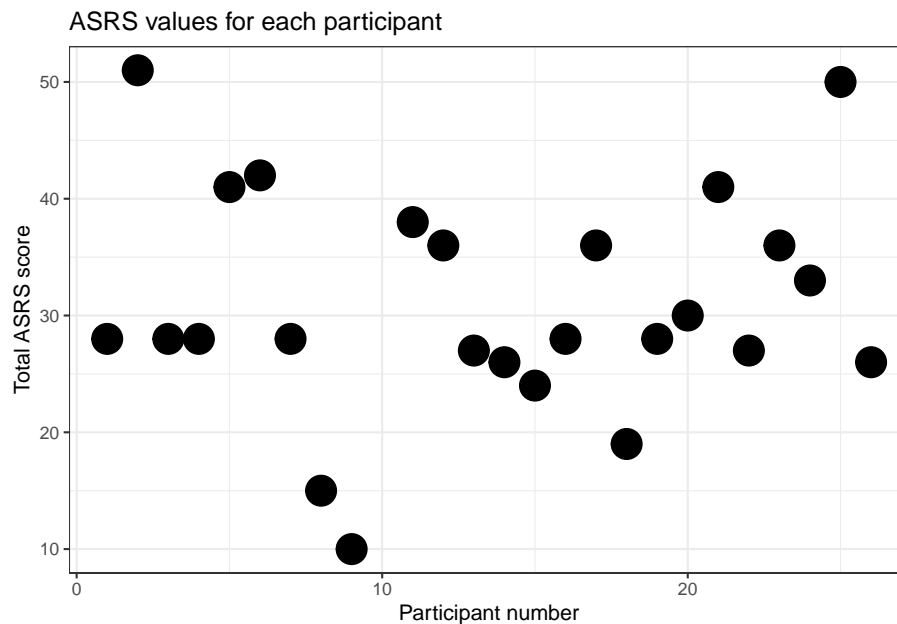
#Method 2. Load from URL https://rpubs.com/lokraj/github_csv
library(readr)
urlfile="https://raw.githubusercontent.com/tomstafford/psy6422/master/mydatafile.csv"
df <- read_csv(url(urlfile))
```

```
## Rows: 25 Columns: 4
## -- Column specification -----
## Delimiter: ","
## dbl (4): Participant Number, Total ASRS Score, Inattention subscale, Hyperac...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

#rename columns for easy labelling
df <- df %>% rename(ppt = "Participant Number", asrs = "Total ASRS Score")

#plot parameters
plottitle <- 'ASRS values for each participant'
xlab <- 'Participant number'
ylab <- 'Total ASRS score'
pointsize <- 7

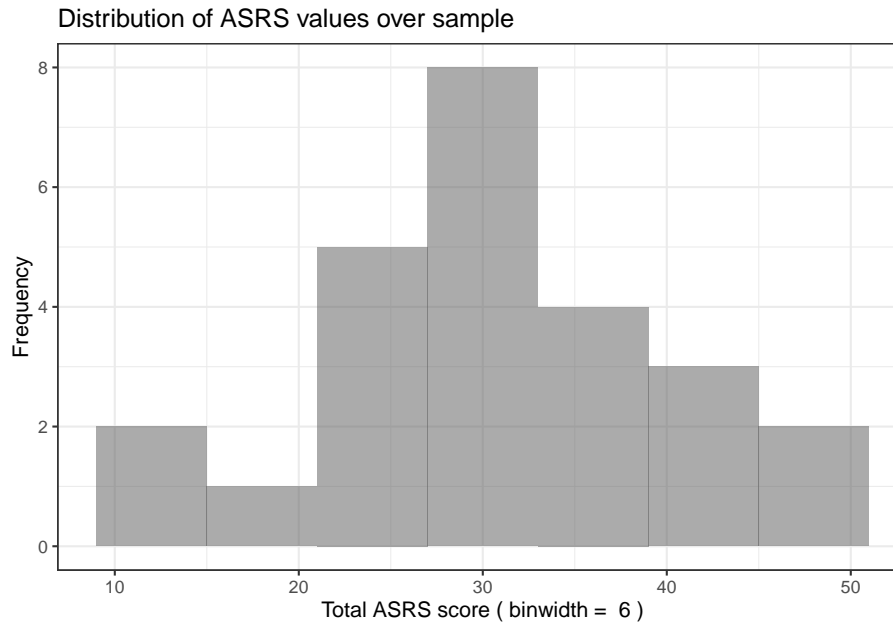
#make plot
p1 <- ggplot(data=df, aes(x=ppt, y=asrs))
p1 + geom_point(size=pointsize) +
  ggtitle(plottitle) +
  xlab(xlab) +
  ylab(ylab)
```



You don't need to show the r code, but can still include it in the document and

use it to generate figures etc.

The scatterplot above uses participant number as one of the axes, which doesn't really make any sense. A histogram is a better way of visualising the spread of scores on a single variable, so here is one:



The code to make this plot is contained in the same `.Rmd` file as this text, but I've hidden it so only the output is shown. To do this I set `echo = FALSE` for the `r` chunk in the `.Rmd` file. You'll have to look at the source file to see this, because - obviously! - in the webpage you don't see any code.

7.4.1 Options for code chunks

Other options for code chunks include

- `include = FALSE` makes the code and results invisible, but the code still runs
- `message = FALSE` hides messages generated by the code
- `warning = FALSE` hides error messages and warnings (normally you want to pay attention to these, but they can make your final document untidy)
- `fig.cap = "..."` puts a caption on any visualisation output

7.4.2 Include `r` in inline text

You can also put single values in ongoing text using Rmarkdown

Imagine you have some data

```
data = c(10,42,2,2,34,63)
```

You can then refer to it in the text. Like this: the mean of the data is 25.5

If you are viewing on the webpage you see the answer, but the file contains an instruction to calculate `mean(data)`.

Neat, eh?

7.5 Conclusion

Rmarkdown documents give you a powerful way to combine text, code and code outputs in one file. That means you don't lose track of how graphs are generated, or statistics are calculated. With RMarkdown your workflow is reproducible, so if you add new data you can update the final report with just one click. RMarkdown documents also encourage you to document fully and explain your code. You can export to different formats depending on your audience, all from the same file.

7.6 Exercises

- Create a new RMarkdown document
- Describe in text the data you are thinking of using for your assignment.
 - Use some highlights: bold, italics
 - Use headings
- Include an R chunk importing the data and showing the first few lines of the data (hint: use the `head` command)
- Include some more text describing the key variables you are interested in
- Show a graph of something (either a scatterplot, or a histogram perhaps)
- Include some inline text, reporting a mean value (as my last example above)
- Make PDF and a Word file by clicking “knit”
- Send the .Rmd file *and the PDF to me via a Slack direct message

Using RMarkdown is essential on this course. These exercises help you check you have mastered this material, as well as giving another opportunity for Tom to feed back on your project

7.7 Advanced

Set the html output options in the YAML to make your output look nicer

```
---
title: "Habits"
output:
  df_print: 'paged'
  code_folding: 'show'
---
```

7.8 Checklist

- Create RMarkdown documents
- Knit to HTML, Word and/or PDF
- Incorporate text formatting like bold, italics, headings and hyperlinks
- Incorporate inline R code (e.g. to report variable values)
- Incorporate code chunks
 - understand that these can be defined with names and/or options
 - use the `echo=FALSE` to *not* show code chunks
- Incorporate graphical output

7.9 Resources

- RStudio intro to Rmarkdown
- Xie et al (2021): R Markdown: The Definitive Guide
- RStudio RMarkdown cheatsheet
- Getting started in R markdown
- Helps with learning Markdown conventions: markdownlivepreview.com
- Making Word documents: Happy collaboration with Rmd to docx
- Install this to make PDFs work: Getting MiKTeX
- You can write beautifully APA formatted manuscripts in RMarkdown:
A minimal example of APA manuscript preparation, integrating text & analysis code, using Rmarkdown and papaja

Chapter 8

ssh and the command line

Before this class please install the University VPN and confirm it works [estimated time: 5-10 minutes]

In this class we will introduce the idea of the command line, running commands without the normal graphical user interface (GUI) and accessing remote computers using ssh (secure shell).

8.1 Launching the Terminal

On Windows or Mac, search ‘Terminal’ in the launcher. You should see a small black box (called “Command Prompt” on Windows) with a cursor. On windows you can also enter `cmd.exe` in the search box in the bottom left.

Note that the cursor appears after some information that shows a file path and/or computer identity. Just as RStudio has a working directory, a location where it looks to load or save files, so with the command line. This is very important, since many command line instructions will only work if they are run *in the right place*.

8.2 Changing folders (‘navigating the directory tree’)

Get the contents of the current directory with `dir` (on Windows) or `ls` (on Mac).

You will get a list of contents, some of these things are folders, some are files. Can you identify which is which?

You can move into a subfolder, using `cd foldername`. You can move up/back the path using `cd ..`.

You can get the current path `cd` (Windows) or `pwd` (Mac)

This way you can navigate all of the files stored on your computer. It is fiddly and annoying compared to the normal GUI method, but it is more fundamental to how computers work ‘under the hood’ and this way of interacting with computers is necessary in some situations (including, often, accessing remote computers).

Remember, like in RStudio, you can use TAB to autocomplete commands.

8.3 Networking

Computers connected to a network have an address, called an IP address, which is always four numbers separated by dots. You are used to navigating the internet using domain names such as `sheffield.ac.uk`, but these names exist for human convenience. Like the command line for individual computers, IP addresses are closer to the metal of how computer networks actually work.

Copy this IP into your browser and see which website it takes you to: `143.167.2.102`.

Your computer is also connected to the internet, so it has an IP address. You can find this out by typing `ipconfig` (Windows) or `ifconfig` (Mac) at the command line. (You can also google ‘what is my ip’ but that won’t work for what we’re about to do). This command should give you a bunch of information. Look for the IP addresses. If you are at home you may see one with the first two numbers `192.168.xxx.xxx`. If you are on campus you should see an IP of the form ‘`172.xxx.xxx.xxx`’. All computers on the University network have IPs which start with the same number.

Now launch the University VPN and look for your IP again. If you are off campus it should change, so you now have an IP which starts 172. Remember this number, we will use it later.

8.4 Ping

Computers talk to each other in lots of different ways, but you are probably mostly familiar with webpages - the “world wide web” (www) served via the http protocol.

One of the very simplest ways for two computers to talk to each other is `ping` which is just a tool to check whether two computers can pass messages back

and forth. The content of the message isn't important, it is just to check that they are both connected to the network

You can ping an IP or a website. Try it at the command line

```
ping shef.ac.uk
```

You should get back a bunch of lines will tell you how long, in milliseconds, your computer took to get a message to and from that website. Try it with somewhere further away.

```
ping wapo.com
```

 (the East coast of the USA), or

```
ping globaltimes.cn
```

 (mainland China)

But we can also ping individual machines. “sharc” is one of the high performance computing (HPC) clusters at Sheffield. You can only talk to it if you are inside the University network (i.e. you need to be on campus or turn your VPN on). Try

```
ping sharc.shef.ac.uk
```

If you've turned your VPN on correctly it will work.

8.5 Remote access using SSH

If two computers are connected, you may be able to remotely access one from the other using a command called ssh. This takes the form

```
ssh user@machine_ip
```

Tom will set up a machine for you to try this in class. If successful you should see something like this

The terminal will look similar, but commands you run will now execute on this remote machine. You may also have changed operating system, so you rather than the commands for Mac or Windows, you need to use the commands that work on the remote machine. Many HPC machines and web servers are linux machines (and so is the one we will connect to in class).

8.6 Find files, run files, copy files

This section assumes you are working on a remote linux machine

Let's use the command `find` to look on this remote computer for any R scripts. The use of `*` here is known as “a wildcard”

```
find -name '*.R'
```

We can run R on the remote computer, but you can't run RStudio (because the terminal doesn't show graphics, only text). Running software without the GUI is known as running software "headless".

R

Will launch an interactive R console. Quit with `q()`

But we can also run the script without launching R (so the will run the script and then quit, leaving us back at the terminal. You can either navigate to the containing folder and run it.

`Rscript toyscript.R`

You can also feed the script to R by specifying the full path

`Rscript home/neo/Desktop/toyscript.R`

Find out how long the script takes to run by putting `time` in front

`time Rscript toyscript.R`

Now there's no point running a simple R script remotely, but when you have very large data it can be convenient to run your analysis on a remote machine which is more powerful (or on many remote machines, which allows parallel processing).

Let's copy this script from the remote machine to our home machine. To do this you need the `scp` command and to know your username and ip address. The path to where you want the file saved comes after the colon:

`scp toyscript.R user@your_ip:/path/to/Desktop/toyscript.R`

`scp` will work on linux or mac, but not on windows.

8.6.1 switches

Often we feed additional options (sometimes called 'switches') to command line tools. So the command `espeak` speaks words aloud

`espeak "hello"`

We can read the `espeak` documentation to see what the options are. Let's adjust the pitch and speed.

`espeak -p 8 "hello but with extra gravitas"`

`espeak -s 230 "hello my name is speedy"`

8.7 Exercises

These exercises will need to be done in the week after class, since the remote machine is not turned on permanently. See slack for the IP, username and password to use for remote access

- Enter the remote IP in a browser - this shows a website served from the remote computer
- remote access it using ssh
- find the file index.html
 - hint: it is in /var/www/html
- copy to your local machine using scp
- edit using a text editor and save on your local machine
- copy back to the server
- look again at the IP in a browser to verify that you changed it (remember to press refresh/F5/clear your cache)

8.8 Checklist

- Identify and launch a terminal window
- Navigate the directory tree using command line instructions
- find your ip address
- ping an ip or website
- find files, run programmes, on remote machines
 - feed options to command line tools using switches
- copy files to/from remote machines using scp

8.9 Resources

- Five reasons why researchers should learn to love the command line
- University of Sheffield VPN
- For historical interest: In the Beginning... Was the Command Line (1999) Neal Stephenson
- If you want to experiment with linux you may start with Ubuntu Linux

Chapter 9

Git and Github

With our guest lecturer, Seb James

9.1 Motivation

Git is an incredible common tool among software developers and computer scientists. It has powerful functionality for coordinate code projects across many people. As a core data science tool I want everyone on the course to have had some exposure to it, and to recognise the key concepts (as well as the value it has). We will not be using the full functionality in the course.

9.2 Before the class

Create an account on github.com, if you don't already have one.

Install git on your machine.

- On Windows: <https://git-scm.com/download/win>
- On Mac: <https://git-scm.com/download/mac>

Until you do that you DON'T have the git features enabled in RStudio.

9.3 Accessing git

If you have RStudio you can access git using point and click (top right), or via the terminal (the tab next to Console on the bottom right).

Outside of RStudio:

- Students using windows can install: <https://gitforwindows.org/> Once installed they should be able to open a git bash window and type `git --version`
- Students on a Mac will have git if they install ‘Xcode’ and ‘Command line tools for Xcode’. They can test by opening a terminal and typing `git --version`

9.4 Resources for the class

This week’s class is taught by Seb James. Links to his materials are below

- Intro to git talk: <https://github.com/ABRG-Models/GitTutorial>
- PDF of slides: <https://github.com/ABRG-Models/GitTutorial/blob/master/talk.pdf>
- Software carpentry: <http://sebjameswml.github.io/git-novice/>

Examples of two published projects supported by Github repos

- <https://github.com/ABRG-Models/AttractorScaffolding>
- <https://github.com/ABRG-Models/linetask2014>
- https://github.com/ABRG-Models/BarrelEmerge/tree/eLife_submission1_full

9.5 Checklist

Git and github are fundamental tools which will be relevant for collaborating on reliable computational projects, but it is not essential you master the full range of capabilities they offer for this module.

By the end of this class you should

- have a github account
- installed git on your local machine
- understand why git is useful for code projects
- recognise the essential terminology: repo/repository, remotes and local, clone, branches, commit, merge, rebase, push/pull

9.6 Resources

Official documentation: [github pages](#)

Understanding Git Conceptually

`git/github` guide a minimal tutorial

Vuorre, M., & Curley, J. P. (2018). Curating research assets: A tutorial on the git version control system. *Advances in Methods and Practices in Psychological Science*, 1(2), 219-236.

Chapter 10

Publishing

Git and github allows collaboration and version control on large software projects, but it also provides a convenient way of publishing data projects to the web. This is the way we will use to produce our final projects for this module.

10.1 Sharing your Rmd files via Github pages

1. Create a new github repository (a “repo”)
 - click “initilise with README.md”
2. “clone” to your computer
 - Either use git to manage the files in the repo, (in which case you need to add, commit and push to the remote repository)
 - OR upload the files to the repo via the github.com web interface.

If you are making multiple changes to your files, using git on your computer is easier (see the class on git and github)

3. Save your analysis in a file `index.Rmd`
4. Add an .html file called `index.html`. This file can be created by clicking ‘knit’ in RStudio from your .Rmd file (see the class on Rmarkdown
 - selecting ‘knit to HTML’ if you haven’t specified this in the header

5. Upload or “Push” your files, most importantly the `index.html` file to a github repo, probably viewable at something like: <https://github.com/username/myrepo/>
 - **NOTE:** To make the repository an archive of your project you can and should add all the other files and folders, including the `.Rmd` file which you used to make the `index.html` file
6. Visit this link and, via the browser,
 - click “Settings” and scroll down to “Pages”. Under “Source” select “Branch: main”. Click Save
 - Find your pages, after a short moment, at yourusername.github.io/yourreponame
 - Example: my repo is at github.com/tomstafford/supertues, so the published pages are at tomstafford.github.io/supertues

10.2 Other sharing options

The Open Science Framework a platform which provides storage and sharing for all materials across the whole lifespan of a project. Incredibly useful for sharing, but doesn’t look as nice as github pages, so we’re not using it on psy6422.

Jupyter notebooks. Another way of mixing description, code and outputs. More common for python code.

- Example: Light reanalysis of Čavojová et al (2018)

10.3 Checklist

- Publish website using github pages
- Understand that `index.html` is the default page for a website
- Use projects in RStudio

10.4 Resources

- Cathy Gao: How to publish project online?
- Using git with RStudio
- Publishing large projects bookdown

Advanced Topics

The first ten topics of this course represent the core that I want every student to take away from the course. With these topics you should be equipped to focus on your personal data project for which makes up the majority of the grade on the course.

The remaining time in the timetable is available for supervision on this data project, and to cover more advanced topics. For these topics, rather than provide a lecture, we will work in small groups to follow exercises and tutorials which exist outside of the course. The motivation for this is two-fold. First, because all your future work will be teamwork, I want students to graduate from this course with experience working together on technical projects. Whether you are more or less confident in the technical requirements, you will learn a lot from trying to share what you know or think you know with a group. Second, a key skill for ongoing development in data science is to teach yourself. There world is rich in useful advice, tutorials and examples. Only by discovering how you can use these will you maximise your potential after you have finished this module.

10.5 2022

Topics to be determined (see below)

10.6 2021

In 2021 we voted on the advanced topics to cover in class:

- interactive plots with shiny
- animated plots with gganimate
- SQL
- python

10.7 Other topics/resources

- regexone.com Learn Regular Expressions with simple, interactive exercises.
- Performance tuning: Code performance in R: Parallelization
- Making Your First R Package

10.8 Reproducibility

Dependencies

- Package management
 - <https://groundhogr.com/>
 - renv: <https://github.com/rstudio/renv>
 - e.g. <https://luisdva.github.io/rstats/annotater/>
- Reproducible workflows
 - Targets (was Drake)
- Containers
 - e.g. docker

Code Review

- Strand, J. F. (2021, March 31). Error Tight: Exercises for Lab Groups to Prevent Research Mistakes. <https://doi.org/10.31234/osf.io/rsn5y>
- CODECHECK

10.9 General righteousness

You should use a password manager. Really. LastPass is recommended

You should know how to use a VPN

- Advice for Sheffield students here: Working remotely - information for students
- You can test by visiting this page. If you are in the University network (either on campus or via a VPN) it will offer you the chance to download the article PDF. Otherwise it will try and charge you \$35 for the privilege.

- Interactive graphs with Shiny

10.10 Before class

Please install shiny

```
install.packages("shiny")
install.packages("shinyjs")
install.packages("shinydashboard")
```

Please visit shinyapps.io and make an account (you may be able to activate an account associated with your TUoS account by clicking ‘sign in with Google’)

10.11 Exercises

We are going to follow Lisa DeBruine’s tutorial - First Shiny App.

Additional challenges / exercises

- Make scatterplot rather than histogram
- Make the app load data which you save in the project directory
- Deploy to the web with shinyapps.io
 - This page may be helpful when combining exercises 2 and 3: Why does my app work locally, but not on shinyapps.io
- Look for and share additional resources on shiny apps

10.12 Resources

- This is part of a set of Shiny Tutorials

- RStudio: Shiny User Showcase
- Example: Movie-explorer
- Example: Enhanced sensitivity to group differences with decision modelling
- The Anatomy of a Shiny Application
- Shiny widgets - useful to get an idea of what you can do

10.13 Examples of interactive graphs which use different tech (Javascript!)

- Interpreting Cohen's d Effect Size by Kristoffer Magnusson
- Example: Grid histograms of Constituencies by median age in 2017 from Oli Hawkins

- Animated plots with gganimate

This is a package developed by Thomas Lin Pedersen and colleagues. It has it's own page here: [gganimate](#)

This is the documentation. It explains how to install it, and what you can do with it, and - if you dig a bit - what the options are for the various functions.

Get started by installing:

```
install.packages("gganimate")
```

This may give you a warning about additional packages you can install to help create output files in gif or movie file format

```
install.packages("gifski")  
install.packages("av")
```

This *in turn* may ask for additional software to be installed on your machine, and the instructions will depend on what kind of operating system you are on.

Assuming you have fixed that, it's time to make some plots:

Here are three different tutorials on gganimate:

- [gganimate.com: Getting Started](#)
- [datanovia.com: How to Create Plots with Beautiful Animation in R](#)
- [rpubs.com/jedoenriquez: An Intro to Animating Charts with gganimate](#)

The first is written by the package authors. Pay attention the material on 'rendering' at the end, this is important if you want to make animations you can share.

The second gives a good exploration of the different options, nicely showing off possible effects and how you would use them with different data sets.

The third is the shortest and simplest, but note that they use a different structure for creating ggplot objects than we do (piping the data to ggplot). If you are not sure, start here.

There is some nuance in getting the animation to save. You are having trouble with this, I recommend you divide the data frame preparation, animate plot creation and rendering into separate stages. Once you have made a plot object, `p`, try this

```
animate(p,renderer=gifsqi_renderer("my_animation.gif"))
```

10.14 Exercises

- Pick some data to animate
- Control the speed of the animation
- Save the animation as a gif file and add it to a RMarkdown document (and publish to github pages)
- bonus question, why do it this way rather than use R to generate the animation on the page?
 - Use the `{frame_time}` attribute in a label
- How does `gganimate` know what values to put in `frame_time`?
 - Add small black marks to show the starting values of the data. Which option does this
- Look at the documentation and see if you can figure out what line to add to achieve this
- Hint
- Why do you need to render a gif?
 - Why can't you just right-click 'copy image' on the RStudio output? What happens if you do this?

- Database queries: SQL

10.15 Context

SQL, you can pronounce it S-Q-L or “sequel”, is a database system. It is old, in internet terms, but not going anywhere, and core to many data science jobs.

10.16 Key information

There are different flavours, such as MySQL which is used by WordPress websites, or PostgreSQL / postgres.

The common core of SQL is that it is a ‘relational’ database management systems. It is used for storing information and saying how it is connected to other bits of information. Compare this to the spreadsheet/CSV, which stores information in a list.

Note that SQL is designed for databases with potentially billions of rows of data. It demands working habits where you can’t “look” at the data all at once, as you would with a spreadsheet.

SQL has its own terminology but to start off we will think about two things

- Tables: these are rows and columns of data.
- Queries: these are commands we use to select, pull out and/or combine tables

Tables have ‘fields’ (the variable names which define what information is stored, you may be used to thinking of these as the columns of your data)

Tables also have ‘records’ (the stored information, which you be used to thinking of as the rows of your data)

10.17 Exercises

In class, I will illustrate the key concepts of SQL with an example from my own research

- Stafford, T. (2018). Female chess players outperform expectations when playing men. *Psychological Science*, 29(3), 429-436. (Preprint).

Then we will look at this interactive tutorial:

- sqlzoo.net

Specifically, we will do

- SELECT basics https://sqlzoo.net/wiki/SELECT_basics
- JOIN https://sqlzoo.net/wiki/The_JOIN_operation

10.18 Checklist

By the end of the class, you should

- Know what SQL is and why it is important for data science
- Understand key concepts: tables, queries, fields, rows
- Recognise and solve problems involving SQL commands such as SELECT, FROM, WHERE, wildcard (%), JOIN, ON,

10.19 Resources

- Select Star SQL - interactive tutorial
- w3schools.com JOIN
- Interactive tutorial sqlzoo.net/wiki/SELECT_basics
- SQLite is great for R and Shiny. The `dbmisc` package may help a bit.
- example SQL data from Kaggle: SQLite database on Mental Health in the Tech Industry

- Python

This course teaches general principles of coding and computation, and specific skills for data management and visualisation in R. Lots of people in the data science world, particularly in areas which align with computer science / machine learning, use Python.

I decided to teach this course in R because the community around a language is as important as the language. You can do anything in any language, and most things as easily in Python as in R, but Psychologists, biologists and scholars from across the social sciences tend to use R, so I'm teaching this course in R.

10.20 Motivation

In my very prejudiced opinion, there are three reasons to prefer Python to R. These are

- Conda, for package, dependency and environment management
- Pandas, the data wrangling toolkit
- Certain toolkits,
 - e.g. Machine Learning tools. Computer Scientists tend to use Python, so the latest in ML, Deep Learning etc is more accessible if you use Python (e.g. PyTorch, scikit-learn)
 - e.g. Running experiments, using psychopy

I'd probably also do anything involving talking to hardware, an API, web-scraping or bulk text processing in Python, but this may just be prejudice. Python is a good interstitial language.

10.21 A worked example

To showcase python, we'll be trying to reproduce an analysis by Oli Hawkins, showing off a support-vector machine classification of Penguin data

repo: github.com/olihawkins/penguin-models

10.22 Preparation

Before the class, please

- install Python. **note** Python underwent a major update from version 2 to 3. Make sure you install version 3+ of python.
- install miniconda
 - You MAY have installed python and conda already, when you installed R, if you did this via the Anaconda distribution.
- clone the penguin-model repo
- create a conda environment, following the instructions in the repo README.md

```
conda env create -f environment.yml
```

10.23 In class

We will use the Spyder IDE

```
conda install spyder
```

I will showcase some features of python, focusing on things which are the same or frustratingly different from R

- for loops
- zero indexing

This is useful if you can't manage to install python: Google Colaboratory

We will do some basic data wrangling with pandas and seaborn

```
conda install seaborn
```

We will run the models, and explore the data and code, focusing on hacking a complex, existing, project

10.24 After class

Oli strongly recommends Aurelien Geron's 2019 book Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow

If you want to learn Python, I recommend:

- Resource list: Python for Non-Programmers
- Pages RealPython: Learn Python Programming, By Example
- Book: Python Data Science Handbook by Jake VanderPlas
- Book: Python for Data Analysis by Wes McKinney
- Pages: chrisalbon.com
- Pages: TowardsDataScience, for example this one A Quick Introduction to the "Pandas" Python Library
- Book: Python for Experimental Psychologists by Edwin Dalmaijer
- Pages: w3Schools
- Tutorial: WTF Python ("Exploring and understanding Python through surprising snippets")
- Book: Scientific Visualization – Python & Matplotlib, Nicolas P. Rougier

Other resources:

- Nicolas P. Rougier: Matplotlib tutorial

Appendix A

Class notes

A.1 Ask me questions

Feel free to email me with questions about any aspect of the course, or problem you are having with the work, but you must include your current best answer to your own question, along with what you've tried so far (and why it didn't work).

It is easiest for me to help you with your project, either in person (see timetable) or by email if you provide a full project (e.g. the data and code as well as the output you have so far) as well as a project rationale stating what it is you are trying to do

This is helpful guidance on posing specific coding questions <https://stackoverflow.com/help/how-to-ask>

A.2 Terminology: 'Raw data'

Data before any processing has been done. Usually these means that the individual responses which create the data are recorded

The opposite is cleaned data (once it has been processed) and then after that summary data (in which responses are collapsed across individuals or into groups)

##. What should I do if my visualisation uses a new library or package?

Ideally your code should specify all the libraries it uses. These are known as "dependencies". If you use an usual library it may require specific downloading to your computer, meaning that your code won't run on computers that don't have it downloaded. There are ways of managing this, but we haven't covered

these in the course (google “docker” or “conda” or “reproducible computational environments”). As long as your code indicates what you have installed (e.g. it includes a line like

```
library(fancyplotslibrary)
```

at the beginning, don’t worry about it

Appendix B

Extra Reading

Further reading, including books, links, demos and packages. You don't need to read all of this, but you will want to dig around. If I could recommend one book to accompany the course it would be

Healy, K. (2018). Data visualization: a practical introduction. Princeton University Press.

B.1 Visualisation (theory)

Healy, K. (2018). Data visualization: a practical introduction. Princeton University Press.

Cairo, A. (2012). The Functional Art: An introduction to information graphics and visualization. New Riders.

Tufte, E. R. (2001). The visual display of quantitative information. Cheshire, CT: Graphics press.

McCandless, D. (2012). Information is beautiful. London: Collins.

Wilke, C.O. (2019). Fundamentals of Data Visualization. O'Reilly. [free online]

Rougier, N. P., Droettboom, M., & Bourne, P. E. (2014). Ten simple rules for better figures. PLoS Comput Biol, 10(9), e1003833. <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003833>

Weissgerber, T. L., Milic, N. M., Winham, S. J., & Garovic, V. D. (2015). Beyond bar and line graphs: time for a new data presentation paradigm. PLoS biology, 13(4).

- Related: Visualizing Distributions with Raincloud Plots with ggplot2

Nightingale: The Journal of The Data Visualisation Society

Podcast: Explore Explain: A Video and Podcast Series

Why you sometimes need to break the rules in data viz by Rosamund Pearce

The Encyclopedia of Human-Computer Interaction, 2nd Ed.: Data Visualization for Human Perception

The Economist newsletter: “Off the Charts” is highly recommended. Examples: better bar charts, using log scales

The Do’s and Don’ts of Chart Making

Riffe, T., Sander, N., & Kluesener, S. (2021). Editorial to the Special Issue on Demographic Data Visualization: Getting the point across—Reaching the potential of demographic data visualization. *Demographic research*. Rostock: Max Planck Institute for Demographic Research, 2021, Vol. 44.

Franconeri, S. L., Padilla, L. M., Shah, P., Zacks, J. M., & Hullman, J. (2021). The science of visual data communication: What works. *Psychological Science in the Public Interest*, 22(3), 110-161.

Lisa Charlotte Muth: What to consider when using text in data visualizations

<https://blog.datawrapper.de/text-in-data-visualizations/>

B.2 The Reproducibility Crisis

Cancer Biology Reproducibility Project <https://www.enago.com/academy/the-reproducibility-project-cancer-biology-to-replicate-only-18-studies-now/>

Economics reproducibility <https://www.wired.com/story/econ-statbias-study/>

Video: Is Most Published Research Wrong <https://www.youtube.com/watch?v=42QuXLuH3Q>

Demo: p-hacking <https://fivethirtyeight.com/features/science-isnt-broken/#part1>

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, 349(6251), aac4716.

B.3 Better practice

Munafo, M. R., et al. (2017). A manifesto for reproducible science . *Nature Human Behaviour*, 1, 0021. DOI: 10.0138/s41562-016-0021.

Markowetz, F. (2015). Five selfish reasons to work reproducibly. *Genome biology*, 16(1), 274. <https://genomebiology.biomedcentral.com/articles/10.1186/s13059-015-0850-7>

A Guide to Reproducible Code in Ecology and Evolution <https://www.britishecologicalsociety.org/wp-content/uploads/2017/12/guide-to-reproducible-code.pdf>

Gael Varoquaux: Computational practices for reproducible science <https://www.slideshare.net/GaelVaroquaux/computational-practices-for-reproducible-science>

Axelrod, V. (2014). Minimizing bugs in cognitive neuroscience programming. *Frontiers in psychology*, 5, 1435.

“our wishlist for what knowledge and skills we’d find in a well-prepared data scientist candidate coming from a masters program.” https://github.com/brohrer/academic_advisory/blob/master/curriculum_roadmap.md

Wilson, G., Bryan, J., Cranston, K., Kitzes, J., Nederbragt, L., & Teal, T. K. (2017). Good enough practices in scientific computing. *PLoS computational biology*, 13(6), e1005510.

B.4 Project organisation

Mike Frank onboarding guide <http://babieslearninglanguage.blogspot.co.uk/2017/01/onboarding.html>

Jenny Bryan’s advice on filenames: Naming Things

Emily Riederer Column naming contracts

Broman & Woo (2017) Data Organization in Spreadsheets <https://www.tandfonline.com/doi/full/10.1080/00031305.2017.1375989>

Video: Data Sharing and Management Snafu in 3 Short Acts https://www.youtube.com/watch?time_continue=2&v=N2zK3sAtr-4

Hadley Wickham: Tidy Data: <http://vita.had.co.nz/papers/tidy-data.pdf>

B.5 Coding

Readings in Applied Data Science <https://github.com/hadley/stats337#readings>

Stack overflow: asking good questions <https://stackoverflow.com/help/how-to-ask>

Stack overflow: provide a minimal, complete, verifiable example <https://stackoverflow.com/help/mcve>

Our Software Dependency Problem <https://research.swtch.com/deps>

From Psychologist to Data Scientist <https://www.neurotroph.de/2019/01/from-psychologist-to-data-scientist/>

Bret Victor: Learnable Programming: Designing a programming system for understanding programs

Top 10 Coding Mistakes Made by Data Scientists

Coding error postmortem by Russ Poldrack, McKenzie Hagen, and Patrick Bissett (August 10, 2020)

B.6 R

B.6.1 Hints

Prime Hints For Running A Data Project In R

RStudio Cheat Sheets: <https://www.rstudio.com/resources/cheatsheets/>

Here::Here https://github.com/jennybc/here_here

We are R-ladies - Twitter account with a rotating curator featuring discussions, package highlights, and tips

B.6.2 Courses / books

I recommend you start with swirl: <https://swirlstats.com/>

Lisa DeBruine, & Dale Barr. (2019). Data Skills for Reproducible Science. Zenodo. doi:10.5281/zenodo.3564348 <https://psyteachr.github.io/msc-data-skills/>

You may also enjoy:

Chester Ismay and Patrick C. Kennedy: Getting Used to R, RStudio, and R Markdown

Matt Crump: Reproducible statistics for psychologists with R

Danielle Navarro: Learning Statistics With R

* Particularly chapter 3 <https://learningstatisticswithr-bookdown.netlify.com/intro>

Data Science with R: An introductory course by Danielle Navarro

Data Science in a Box

Adler, J. (2010). R in a nutshell: A desktop quick reference. ” O’Reilly Media, Inc.”.

Intro to R (Liz Page-Gould): <http://www.page-gould.com/r/uoft/>

Grolemund, G., & Wickham, H. (2018). R for data science. * See also <https://r4ds.had.co.nz/>

B.7 Making graphs (practice)

Graphing in R (Eric-Jan Wagenmakers and Quentin F. Gronau): <http://shinyapps.org/apps/RGraphCompendium/index.php>

r-charts.com: “Over 1100 graphs with reproducible code divided in 8 big categories and over 50 chart types, in addition of tools to choose and create colors and color palettes”

Cédric Scherer: A ggplot2 Tutorial for Beautiful Plotting in R

B.8 Presentations

Kieran Healy : Making Slides

B.9 Statistics

Discovering Statistics Using R

Hox, J. (2010) Multilevel Analysis: Techniques and Applications

Statistical Rethinking: A Bayesian Course with Examples in R and Stan

model checking package: Performance

B.10 Advanced Reading, Background & Other Recommends

Data Feminism by Catherine D’Ignazio and Lauren F. Klein. The MIT Press. 2020

Rachel Thomas’s Applied Data Ethics Syllabus

Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems by Aurelien Geron (2019)

Efficient R Programming

SQL Database access dplyr

Data Visualization course by Dr. Andrew Heiss of Georgia State University

B.11 Pedagogy

Brown, N. C., & Wilson, G. (2018). Ten quick tips for teaching programming. *PLoS computational biology*, 14(4), e1006023.

Hudiburgh, L. M., & Garbinsky, D. (2020). Data Visualization: Bringing Data to Life in an Introductory Statistics Course. *Journal of Statistics Education*, 28(3), 262-279.

Appendix C

Notes

C.1 Credit

These pages based on a template published by the awesome psyTeachR team. Extra thanks to Lisa DeBruine for help.

C.2 Colophon

Find the repo here: <https://github.com/tomstafford/psy6422/>

Everything worked better when I:

- set GitHub pages to run off of Master/docs
- listened to Lisa and used bookdown at the CL, not knit in Rstudio, to render the pages: `R -e 'bookdown::render_book("index.Rmd")'`
- installed R 3.6
- installed tidyverse, bookdown, devtools and finally Webex.

C.3 Data Science @ Sheffield

The MSc in Psychological Research Methods with Data Science, of which this module is a core part, is designed to teach graduate psychologists data science skills. If you don't have a psychology background our other MSc courses in cognitive neuroscience can operate as “conversion courses”, since they are designed to teach core psychology/neuroscience skills to people from non-psychology backgrounds.

Our MSc is not the only way you can study data science at the University of Sheffield. The MSc in Data Analytics is taught in the Department of Computer Science and has a stronger focus on computational tools such as machine learning. The MSc in Data Science is in the Information School which is in the Faculty of Social Science, and has more of a focus on business decisions and using data in organisations. We also highly recommend the MSc in Social Research, run by the Sheffield Methods Institute, which teaches a full suite of quantitative and qualitative research skills. The SMI is also home to the Data Analytics and Society Centre for Doctoral Training which offers opportunities to undertake a 4-year funded integrated MSc and PhD.

C.4 Careers

Inside or outside of research, data management and visualisation skills will be good for your career. Unlike many high-technical skills, coding is highly accessible. Even in the world of software development, over half of programmers are self-taught. But coding is relevant to more than just pure software jobs, one report found that 20% of graduate-level jobs value coding as a skill, and these jobs paid on average 30% more than other graduate jobs.

More information:

- Job profile: Data Scientist
- Data Scientist: Job Description
- University of Sheffield: Career Connect
- Academic Jobs: jobs.ac.uk
- University of Sheffield Careers service: Support for international students
- Book & podcast: Build a Career in Data Science

C.5 Testimonials

Peter Carr

This is the first year the MSc in Psychological Research Methods with Data Science has run, so more soon!

Appendix D

Class of 2020

This was the first year for the new MSc in Psychological Research Methods with Data Science. PSY6422 is a key module for this course, and is designed to set students up for their independent data science project, completed over the summer.

Before they work on their summer projects, MSc students complete a mini-project on this module, and these are what is showcased below.

Most students came into this module with little or no previous experience of coding. The aim is to share powerful, flexible, computational methods which allow the production of reproducible analyses and useful data visualisations.

The module is taught via workshops, and the end point is to carry out and publish a data analysis and visualisation on a data set which interests you.

The class of 2020 was impacted by the coronavirus pandemic, which led to some classes being cancelled and the rest being moved online mid-module.

Here just a few examples of data visualisations produced by the class for their data mini-projects. Each project was different, and each student published their data, analysis code and visualisation as an online notebook (so please click the link for more details of each project).

Hala took open data from Sheffield city council and showed daily cycles in air pollution levels

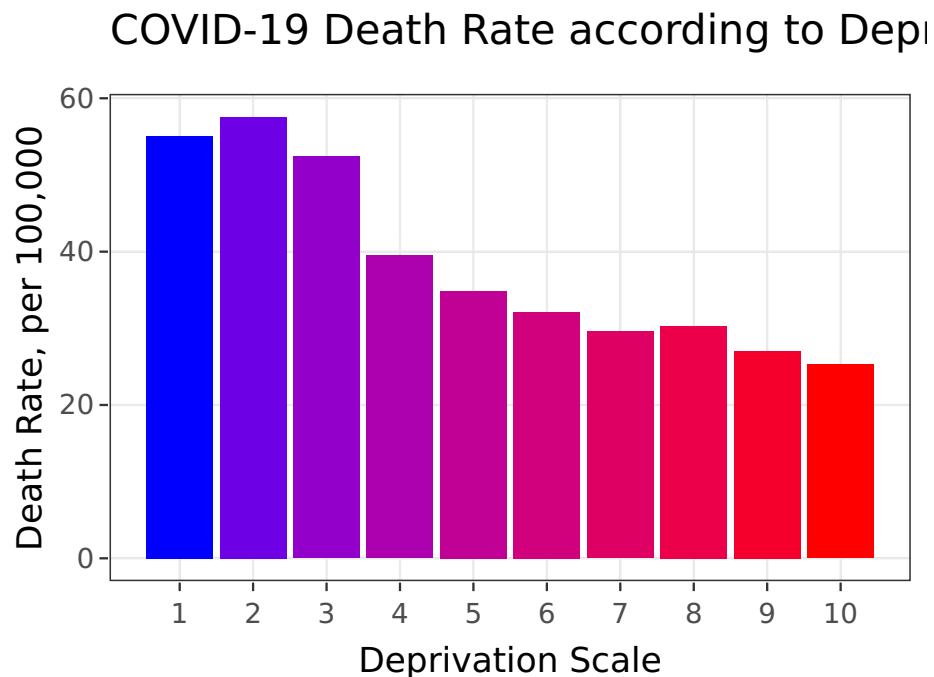
Note the spikes in N02 levels during the morning and evening rush hours.

Nabiha looked at Students' Performance in Exams

Note that these boxplots with jittered points allow you to see the group average but also the underlying data.

Rachel, looked at COVID-19 Deaths in England and produced a number of animated and interactive plots which really show off the data visualisation potential

of R/ggplot, including this shocking plot of regional rates of death against deprivation, which shows that the death rate in the most deprived areas was twice that of the rate in the least deprived areas. Hover over each bar for details.



Yidan's project mixed python and R and involved some highly technical work installing and querying an SQL database server. The topic, Skill acquisition of solo players: a case study revealed how practice mapped to skill level in players of an online strategy game:

This is the famous learning curve, showing that players improved with practice (but at a decelerating rate).

My unofficial prize for aesthetics in data visualisation went to Katie's project avocado prices, sales & distribution across the US between 2015-2018 which used a consistent, and avocado-themed, colour scheme throughout:

Adam's project is a good illustration of the power of data visualisation - it can make salient something we might know but had not fully reckoned with. In his case, his project was on Changes in average house prices and incomes in England 1997-2019:

His graph of the change in the ratio of average house price to average income shows just how unaffordable housing has become.

Several projects looked at topics which interested the students and for which public data was available. For example, Victoria looked at the World Happiness Report 2019 data, (another) Katie looked at Caffeine Intake and Mental Health, and Eleanor looked at Professional CS:GO Matches and in particular at the performance of a professional team based in Sheffield.

Others projects used open data from published research, replicating and extending the analysis the original authors reproduced (Bessie, Esra, Sam, Leah)

These are just the projects which were completed and published publicly by students on the module this year (and who gave me permission to link to their work). I am immensely proud of all the work students did on this module, which was severely disrupted by the pandemic. However far you get with data analysis and visualisation there is always more you can learn, which can be daunting, but it can also be exciting because there are new things to try, new data to look and new things to find out.

Tom Stafford, July 2020

D.1 Feedback

Selected feedback from the class of 2020

Coming from someone who has never coded before, I thoroughly enjoyed visualising data. I never thought I would be someone who would enjoy looking at data, but I found it but both useful and interesting to create graphs that visualise the data in a more digestible form.

Since I've never done coding to do anything before and I've only used point and click methods this is an obvious advantage of doing things this way so you can see where you went wrong and easily change it without having to start over again.

I have actually found myself enjoying a topic that I was convinced that I was going to hate, so for that I am thankful.

The topic I found most useful and interesting was the RMarkdown topic. I was initially extremely skeptical about this topic, my exact thoughts were; why would anyone use RMarkdown when Microsoft Word is available? I have since come to understand my naivety and learnt RMarkdown is an incredibly useful tool, which bridges the computational functionality of R with the ability to annotate code and create a running commentary of the analysis - amazing!

Before this module I had heard of R but not really understood its purpose or usefulness. I had assumed it was similar to SPSS, oh how wrong I was.

Very rewarding.

I am immensely proud of my visualisation project

I am oddly grateful for the struggles I experienced as I feel I have really got a sense of accomplishment out of completing the module. Before starting this Msc I'd have laughed at the idea of even understanding anything to do with coding or visualisation, but now I genuinely have an interest in it.

I found the topics of constructing and publishing projects in a reproducible way and via an open platform really interesting.

I also found learning how to handle data and construct graphs in R very useful as I have never manipulated data using coding, and it was interesting to learn how you can keep track of your working and avoid simple errors this way.

Finding yourself in completely disarray when trying to solve an unsolvable coding problem, but feeling like an absolute genius when said unsolvable problem is solved was very satisfying.

Thanks for the detailed feedback on our Data Visualisation projects! It was very thorough, some of the best feedback I've ever received while at university!!

Appendix E

Class of 2021

This was the second year for the MSc in Psychological Research Methods with Data Science. The course ran entirely online, but the intention was the same - to use PSY6422 to set students up for their independent data science project, carried out over the summer, but also to ensure that they left the MSc with a complete data project they could show off as part of a portfolio.

There were too many students on the module this year to show all projects, so here are a few highlights:

Ruri made this ring plot of where doctors migrate to and from:

Full project is here: Doctors' Migration in 2018. She argues convincingly that some data visualisations can be valuable when they are **not** intuitive, that ultimately demanding a reader's attention to learn to decode complexity can be rewarding.

Sebastian did a fantastic job using the Spotify API to extract data on the emotional valence of the top 200 songs for 70 countries, across the year 2020. Did the global pandemic affect the mood of the songs we listened to?

Full project is here: Visualising Valence.

Hubert's project on two meta-analytic bias detection techniques is here: Welcome to the showdown between two bias detection techniques. I'm not going to explain the visualisation here, but the project is an excellent example of data visualisation in the service of wider scientific purpose.

Alex used data from the game Overwatch to visualise how choice of game hero differed between high and low skilled players: Overwatch Hero Pick Rates:

Jarod also looked at gamers, use Steam data to show how the pandemic lockdown affected our gaming habits: How COVID-19 affected gaming habits.

A number of people made interactive visualisations, such as James, whose US State-Wide UFO Sightings from 1970 - 2020 you really need to visit to explore, but here's a static shot:

Other interactive or animated visualisations include Halleyson's Which Anime Studio Is The Best? An Examination Between 1958~2020, Ollie's visualisation of the game trajectories from One Hundred Years of Chess data, and Farzana's analysis of global data on happiness: The World Happiness Report 2021. Finally, here's an animation from Juan's Dementia Diagnosis Rates in England: A Data Story.

Again, a covid-related project and a clear demonstration of the power of visualisation to surface an important story.

Overall, I was proud of every student who struggled with R and the circumstances of the pandemic to produce a visualisation for this module. My sense from the feedback was that most people enjoyed themselves more than they thought they would, and ended up pleased with what they produced and the new skills they take forward from the module

E.1 Feedback

Here is some selected feedback from the class of 2021.

When first planning this project and working with R, I was convinced that I wouldn't be able complete the module. I pushed myself to the very best of my ability and I am proud that I was able to carry out my initial plan and complete the task. Starting this module, I had an interest in climate topics, and I am glad that I was able to use my new-found skills to make a sophisticated climate themed visualisation project at master's level. I will reflect on this module with fondness whilst simultaneously wondering how I managed to reach the end. I'm grateful to have gained my current skills - whilst they still require much improvement, I look forward to further improving as I go forward.

every topic was relevant, interesting and engaging and clearly linked to the overall aims of the course.

"This was one of my favourites modules throughout my undergraduate and postgraduate studies...I am delighted that this is a free platform and I will be able to use it even after finishing"

The magic of data analysis and visualization relies in sharing your data and findings among other scientists and for anyone to access and see. Sharing data allows other people to replicate it, refine it, learn from it or even get inspired by it and create their own. Science should be shared and using these platforms and programs is crucial to achieve so.

"Gaining a coding foundation and an understanding of how to use the statistical programming language of R to create detailed visualisations was extremely

useful, and has provided me with another skillset that will enable me to go on to create and develop further in-depth visualisations in any future research projects”

Appendix F

Class of 2022

This was the third year for the MSc in Psychological Research Methods with Data Science. The classes were back in person and we cracked on with the core motivation of PSY6422 - to set students up for their independent data science project, carried out over the summer, but also to ensure that they left the MSc with a complete data project they could show off as part of a portfolio.

F.1 Module project showcase

There were too many students on the module this year to show all projects, so here are a few highlights. Many creative, fun, interesting, challenging visualisation projects were produced but aren't shown here, but these few give a flavour of what student's got up to:

Florence's project looked at Gender bias in the cover personalities of TIME Magazine (1920-2013) [Pages](#)

Gemma's project was "Female labour force participation rates over the Last 100 Years" [Pages](#) [Repo](#)

Yibo looked at Causes of death in the world, [Pages](#)

Peter's project shows character preference for the Overwatch League 2019 [Pages](#) [Repo](#)

Luke's project is an interactive visualisation which allows you to explore "Global Levels of GDP per Capita, Population Density and Covid-19 Lockdown Severity, and their Relation to World Happiness" [Pages](#) [Repo](#)

This project looked at "Covid-19-immunised to total region population ratio in the Czech Republic" [Pages](#) [Repo](#)

Ben looked at Efficacy of Self Testing - comparing two Covid 19 prevalence metrics across stages of the pandemic in England; Pages

Natasha: A stacked bar plot showing likert scale responses to imagined traumatic scenarios, Pages

Other projects looked at the income of UK doctors compared to inflation over the last 12 years, Pages; a A History of Pitchfork Reviews, Pages; Kieran's project looked at the effect of losing crowds on football scores: Home Advantage and Covid-19: Football's Natural Experiment, Pages

F.2 Advice for future students

As part of the module assessment I asked those who took the module "what advice would you give someone starting this course?". Here's what they said:

its not as hard as you think it will be, you learn pretty quick, google helps a lot, start work on it early.

I would definitely say choose a project topic that you are interested in. I have (quite tragically) loved every minute of developing my project since I am quite interested in the topic and found myself very invested in the outcome of the project.

Try not to get frustrated! I didn't have any coding experience before starting this course, and you will be surprised at how far you can push yourself. If you told me I would be making an interactive plot at the start of this module I would've laughed.

Coding is less about technical skill, and more about mentality. It requires a great deal of patience to manage errors, and even greater patience to Google and self teach the code required to attain your goal. Mastering coding is less about knowing syntax off by heart, but having an effective procedure for accessing resources and managing problems when they arise.

The one advice I would give to someone starting this course is to not be afraid to ask questions. I initially felt out of place and that I should not ask questions as it would expose the fact that I am a beginner at coding. However, there are many people that are beginners in this course, and it is better to ask questions than not understand, especially when doing the final project.

Google will solve many of your problems (such as when trying to write code, or fix technical issues and errors). Keep everything organised (including code, and files. This will make it far more manageable). Don't get frustrated if something doesn't work at first (either look for help online, or ask someone).

I would say to attend every class and make the most of them by asking for help if needed. I have a habit of giving up straight away if I do not understand anything so struggled a lot in class, which made it harder when it came to

the final project. However, after a lot of hard work I have managed to create something I am proud of and never thought I'd be able to accomplish, so if I can do it anyone can!

The course stimulates independent learning, so I would advise everyone to be curious and go after things they find interesting. I would advise future students to ask about everything they are interested in during the workshops - looking things up and troubleshooting takes much longer when one needs to do it alone. Finally, I would encourage everyone to meet up with their peers to discuss their work - they can get a lot of good ideas from the others, and their perspective, too.

Working with a partner from the beginning is also really nice because it makes it so much more enjoyable, and you don't feel like you're being left behind if there's two of you at the same level.

It is helpful to find a friend to code with, especially one who might have more experience than you who can guide you when you are stuck.

First and foremost, I would say not to panic and get discouraged, especially when had some coding in the past and did not necessarily like it. The learning process takes time, just like with any language, you need to understand its rules. In the end, you will code and visualise what you want, even exceeding your initial expectations.

Firstly, don't be afraid to write code and think positively about the meaning of each statement. Secondly, talk to Tom and Luke a lot as that is a very helpful thing to do. Finally, mastering the content of this course will help a lot in whatever industry you enter in the future.

Be patient and don't panic. When I first started coding, I would get hung up on every small error and get really frustrated. As I started to code for my final project, I realised that errors aren't always big problems, it could be something as small as a comma in the wrong place. In my opinion, coding is as much about patience and detail detection as it is about skill. I would also suggest to someone starting the course to not expect the skills to develop immediately. It was only once I began to put together the skills that I learned throughout the course that everything began to piece together and slowly make sense.

It will be very challenging and sometimes quite stressful, but the outcome is something you'll really be proud of if you stick with and push through. Overall, I really enjoyed this module and I now understand the thrill coders get when they've been stuck on a problem for a while and finally get their code to work.

Get involved. Coding can be really scary at first, but you will get as much out of it that you put into it. Plus, there is a genuine feeling of accomplishment once you get your head around some of the concepts involved in coding!

Google is your friend! There are endless help forums online offering advice and snippets of code that can help fix/create code to produce amazing plots. A lot

of forums often give alternative work rounds for code that may not be parsing or rendering correctly, with explanations, offering an alternate route if you are stuck. If you can imagine it, there's generally a code for that somewhere online!

I would recommend, especially if this is the student's first experience with the data visualisation process, to study regularly instead of just a few days before the submission of the final project. It can be helpful to break up extensive thematic sections into smaller chapters. For example, instead of learning all the functions of `ggplot2`, it would be better to learn how to use a specific layer, such as `geom_bar()`, at a time.

Yes it's daunting, and yes it's challenging especially if you have no experience, but don't let it stop you. Once you start engaging properly in this module you'll learn it quickly and start to love it

F.3 “one thing you have read and enjoyed or found useful”

I also asked the students for recommended reading:

F.3.1 General Visualisation

Video: How to avoid death By PowerPoint | David JP Phillips | TEDxStockholmSalon

Schwabish, J. (2021, Feb 8). Five Charts You've Never Used but Should. PolicyViz.

“Data to Viz leads you to the most appropriate graph for your data. It links to the code to build it and lists common caveats you should avoid” data-to-viz.com/

Visual Vocabulary: Designing with data ft-interactive.github.io/visual-vocabulary/

Economist.com Mistakes, we've drawn a few: Learning from our errors in data visualisation

Cairo, A (2012). The functional art: an introduction to information graphics and visualisation (1 st ed.). New Riders.

IBM.com: Learn how data visualization can improve understanding and analyses, enabling better and faster decision making.

F.3.2 R specific advice

Ruggeri, G. (2021, January 29). Better data communication with {GGPLOT2}](<https://giulia-ruggeri.medium.com/better-data-communication-with-ggplot2-92fbcfea2c6e>). Medium.

Datanovia: “TOP R COLOR PALETTES TO KNOW FOR GREAT DATA VISUALIZATION” extremely useful for the choice of the most appropriate colour palette for my final project ().

DeBruine, L. & Barr, D. (2019). Data Skills for Reproducible Science (1.0.0). Zenodo. <https://doi.org/10.5281/zenodo.3564555>

PDF: RStudio: Data Wrangling Cheatsheet

r-statistics.co by Selva Prabhakaran Top 50 ggplot2 Visualizations

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. (2 nd ed.). Springer.

Holtz, Y. (2022). The R Graph Gallery – Help and inspiration for R charts

Chinese R language forum

discover: a package of interactive tutorials <https://www.discover.rocks/discover/>

F.3.3 Markdown

dataquest.io: R Markdown cheatsheet

bookdown.org/yihui/rmarkdown/

F.3.4 Advanced R packages

Paldhous, (2016). ‘From R to interactive charts and maps’

Creating a SQLite database for use with R

Hailperin, K. (2019). Animate ggplots with gganimate:: CHEAT SHEET

Shiny Tutorials. Psyteachr.github.io.

R Studio: Learn Shiny

Mastering Shiny: mastering-shiny.org/index.html * Particularly mastering-shiny.org/reactivity-objects.html

F.3.5 General coding / projects / reproducibility

<https://www.geeksforgeeks.org>

PDF: Jenny Bryan’s “Naming Things” produced for a ‘Reproducible Science Workshop’.

stackoverflow.com (“any errors that I had somebody else most likely had the same and there was always a few different solutions provided.”)

- e.g Krabel, T. (2018). Make plotly annotation font bold
- e.g Formatting mouse over labels in plotly when using ggplotly

Broman, K. W., & Woo, K. H. (2018). Data organization in spreadsheets. *The American Statistician*, 72(1), 2-10.

F.4 Feedback on the module

Here is some selected feedback from the class of 2022.

I went from not knowing how to code to creating my own data-wrangling script in the span of a few months, and I owe it to the teaching style employed during our sessions

I really enjoyed the freedom to self direct on this course. Spending hours trying to learn how to properly implement a function correctly became a lot more satisfying when I was working on a project of my own design.

For the first time in my life, I really liked the ‘homework’ exercises. The mini projects such as the anscombe one really helped me work with R because there was a goal and something tangible to work with. I find it quite challenging to ‘think’ about how to code something, but having hands on experience with data really helps contextualize it for me so I was able to translate those skills to my own project!

I have enjoyed this course and feel it will be beneficial for my career in the future. The lab classes helped my learning but it was also crucial that I had access to video run-throughs of some sections like publishing as these aided me when I was struggling. I was very pleased with how clear the expectations were throughout the course; the distinct parts of the coursework were broken down clearly and explained giving me a good understanding of how I needed to approach the task.

This course gave me a great understanding of the impact visualizations can have within your research.

Tom and Luke's help was invaluable. Having two people within the class that could guide us made all the difference when facing certain problems.

I really enjoyed the final project. During classes I struggled a lot with the content but the final project I was able to go over everything we had learnt in my own time and understand it fully. This gave me a massive sense of accomplishment

I strongly believe that showing live demonstrations of the coding was a very beneficial way of learning. This is because it allowed us to go back and watch the recording if we were stuck; listening to the explanations of the code, and why some things are done in certain ways, allowed me to further my understanding in this field a lot faster as I did not find myself stuck on the basic things. Furthermore, I also found it beneficial that there was some room for self-teaching, in terms of the more advanced topics. This is because it allowed us to set a threshold as to how much coding we could take on and understand, without being overwhelming. It allowed me to push myself only if I thought I could handle it, and then produce a visualisation that I was proud to show friends and family. This is extremely advantageous as it also demonstrates my ability to take on new challenges.

Specifically, I appreciated the opportunity to ask Tom and Luke questions in-person when I was confused, as they helped me understand the philosophy behind how one should approach coding issues; they didn't just tell me the answer – which would have crippled me in the long term – they gave me hints on how I could solve it myself.