R^4H_2O: R for Water Professionals: Case Study 1

Dr Peter Prevos

Online Sessions

Case Study 1: Exploring and analysing water quality data

1. Introduction to R
2. Visualising and communicating results

Case Study 2: Cleaning, exploring and analysing a customer survey

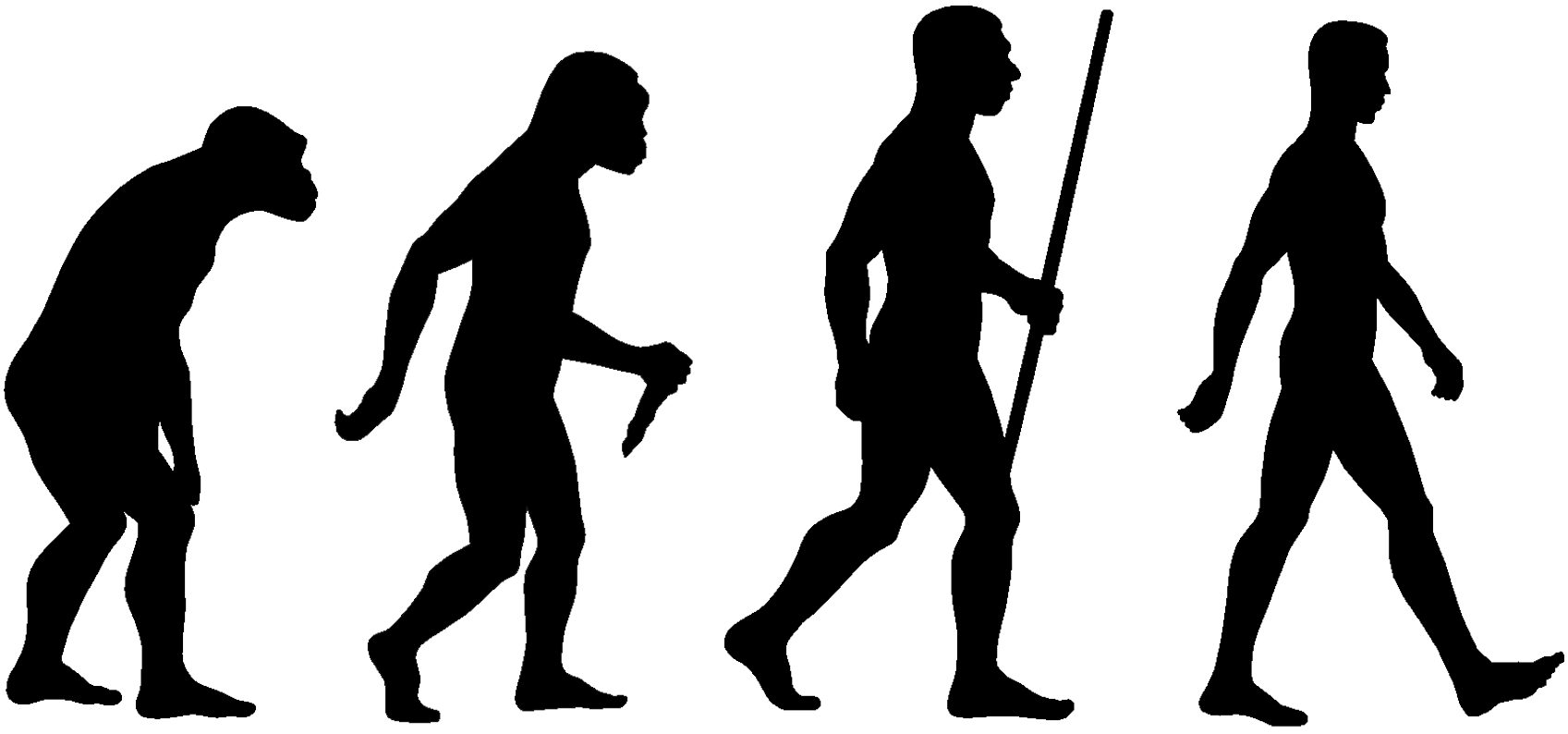
3. Cleaning and exploring data
4. Analysing and communicating results

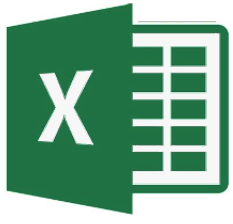
Session 1 Program

- Introduction
- Principles of Data Science
- Introduction to R
- Exploring Data
- Descriptive Statistics

Figure 1: R for Water Professionals workshop (Melbourne, 2019).

My Data Science Evolution





Resources

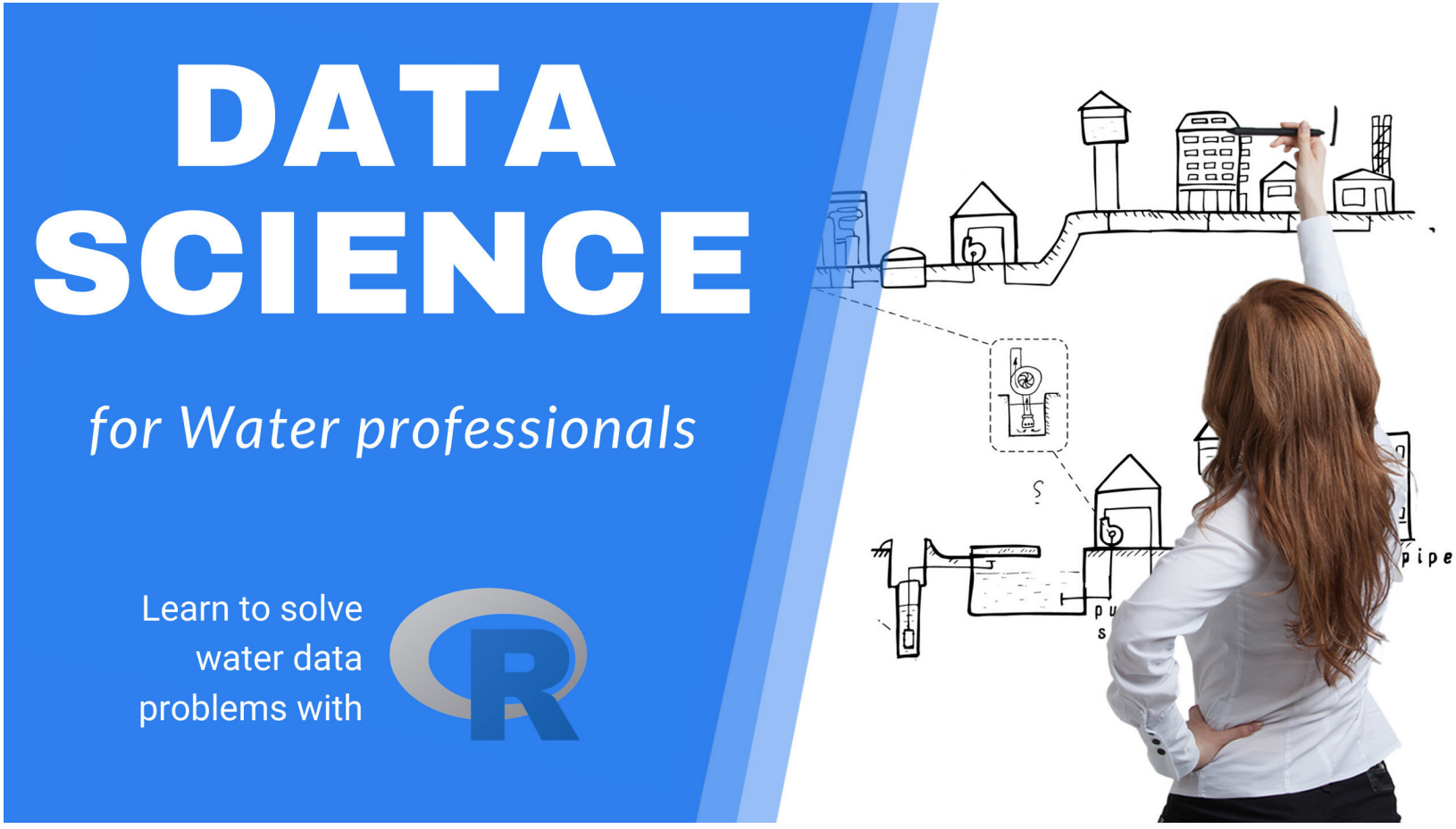
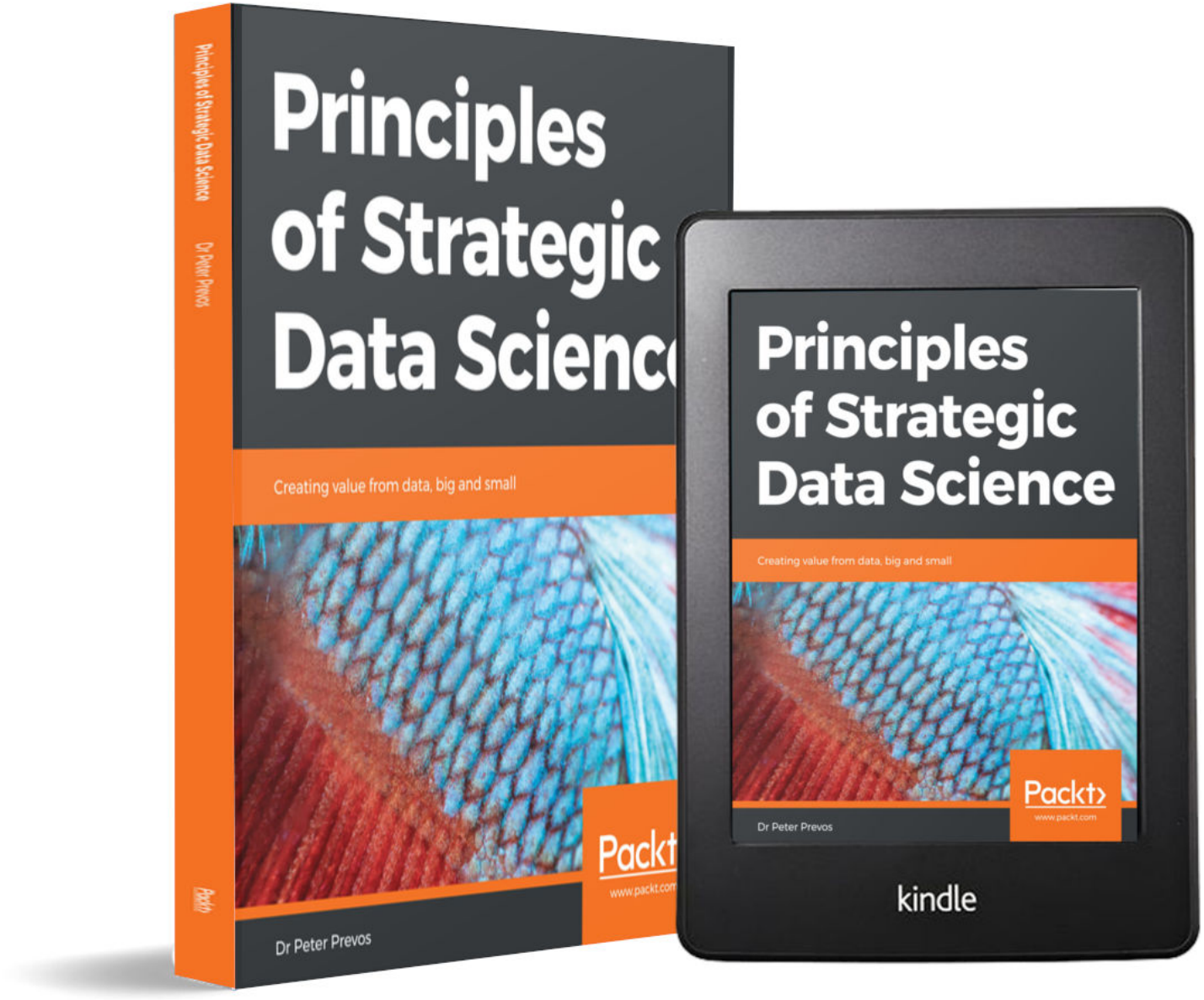


Figure 2: Register to get access to the on-line syllabus: https://leanpub.com/c/R4H2O/c/esc-vic

Principles of Data Science



What is Data Science?

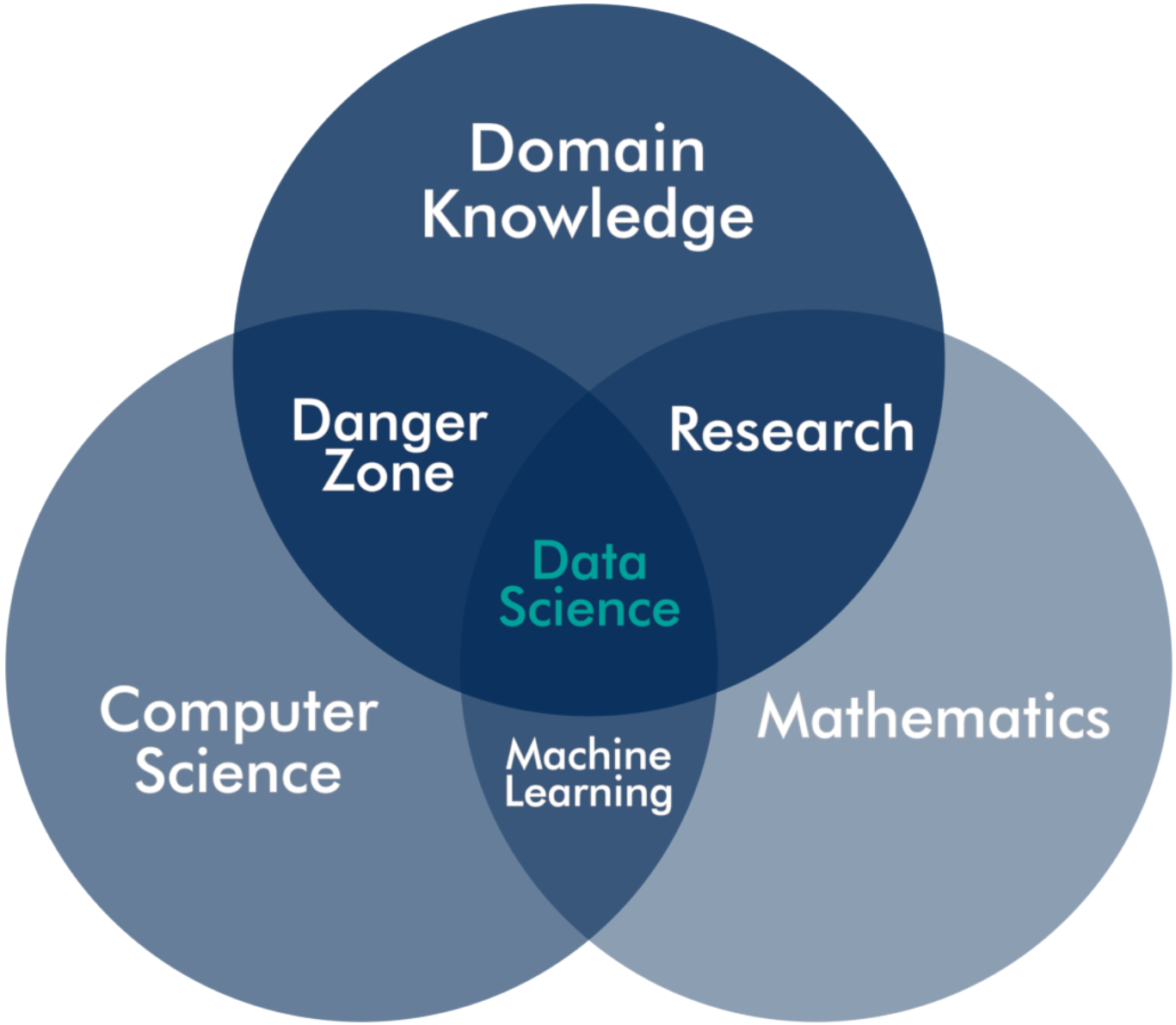


Figure 3: The Conway Venn Diagram (Drew Conway, 2013).

What is good data science?

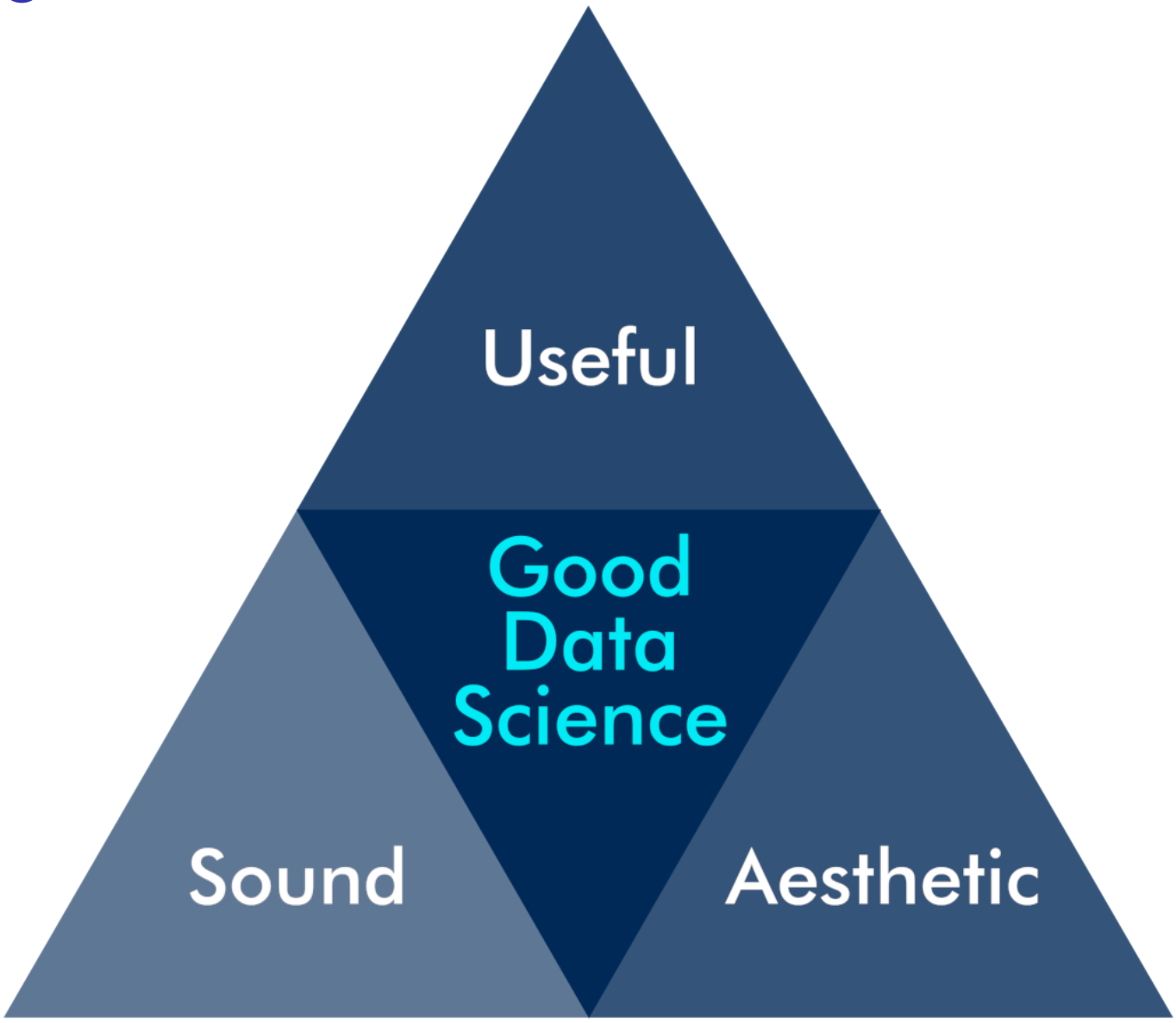


Figure 4: The Vitruvian triangle of good data science.

What is useful data science?

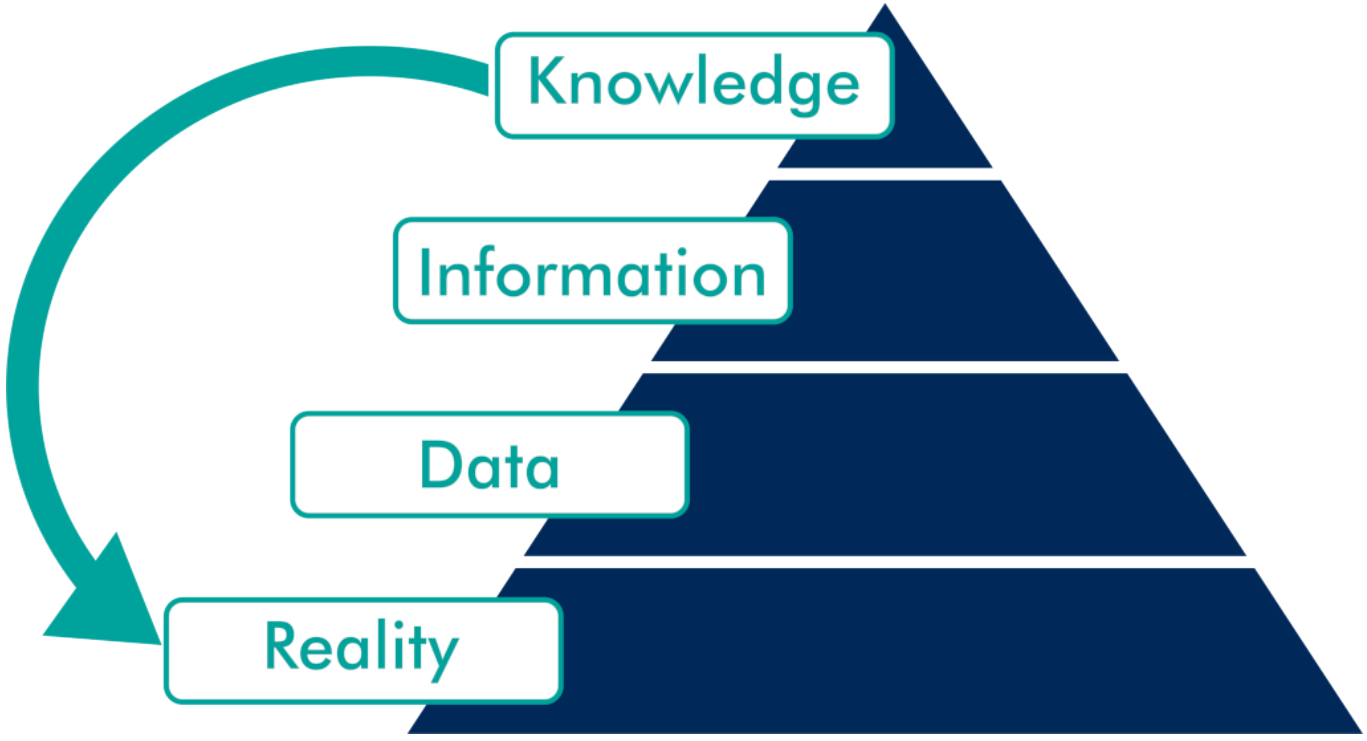


Figure 5: Modified version of the DIKW model.

What is sound data science?

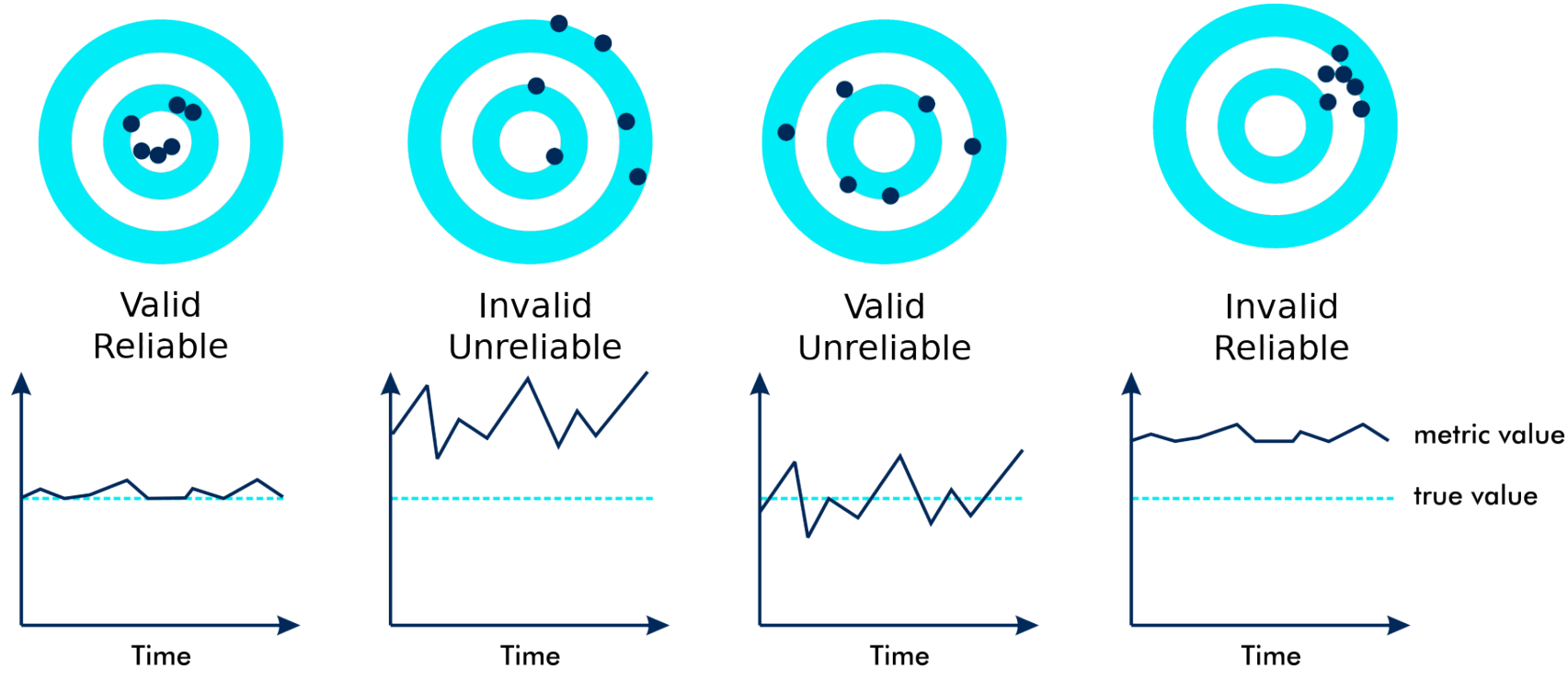


Figure 6: Validity and reliability.

What is sound data science?

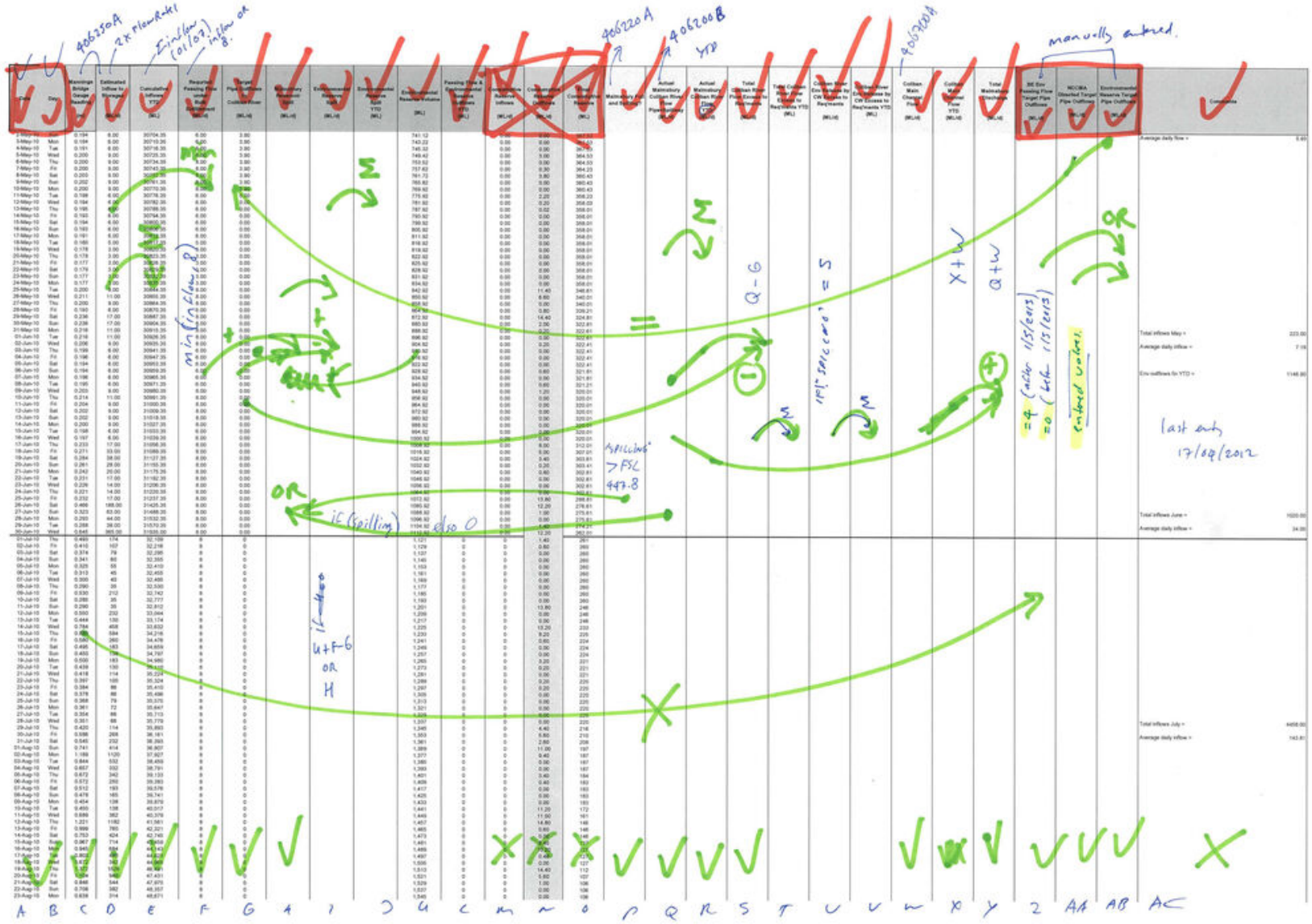


Figure 7: Reverse-engineering a spreadsheet

What is sound data science?

Reproducible code:

```
reservoirs %>%
    select(Date, River_Flow, Natural_Flow, ERV) %>%
    mutate(Date = as.Date(Date, format = "%d %m %Y")) %>%
    gather(Source, Value, -Date) %>%
    mutate(type = factor(Source == "ERV"),
           type = fct_recode(type, Flow = "FALSE",
                                   Volume = "TRUE")) %>%
    ggplot(aes(Date, Value, col = Source)) +
    geom_line() +
    facet_grid(type~., scales = "free_y")
```

What is aesthetic data science?

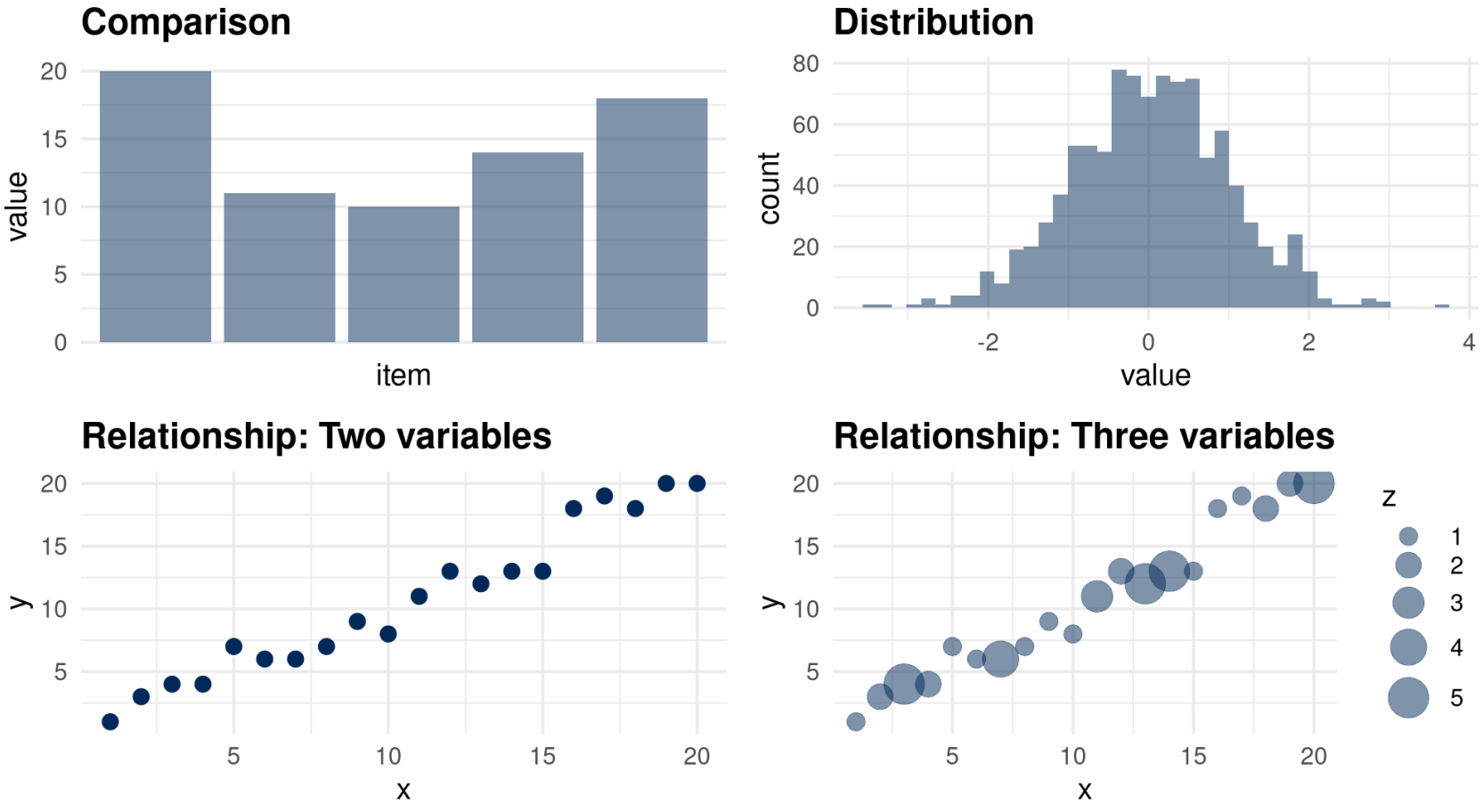


Figure 8: Data visualisation is about telling stories.

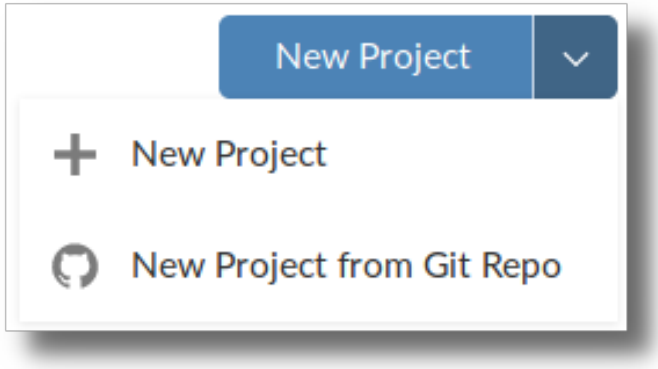
Configure R Studio

Desktop

- Install R and RStudio
- Download materials: `https://github.com/pprevos/r4h2o`
- Unzip folder
- *File* > *Open Project*
- Open the `r4h2o.Rproj` file in the downloaded folder

Cloud

- Sign-up at: `rstudio.cloud`
- *New Project* > *New Project from Git Repo*



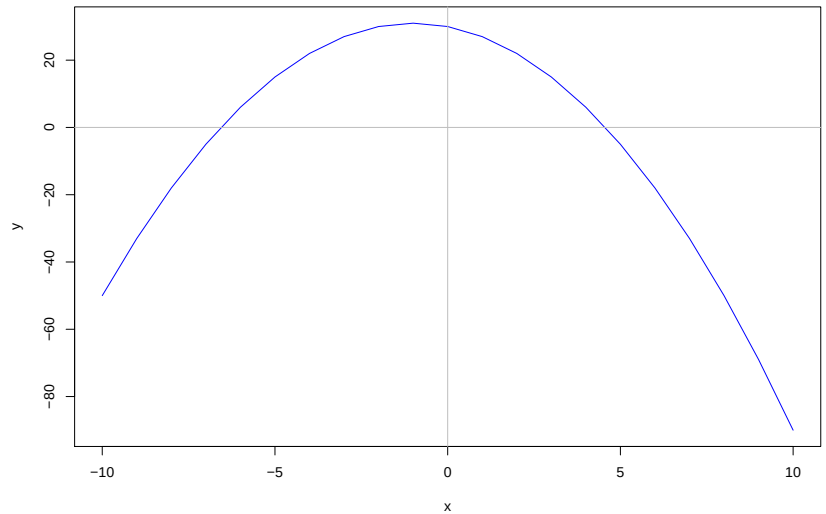
- Enter GitHub URL

Console exercise

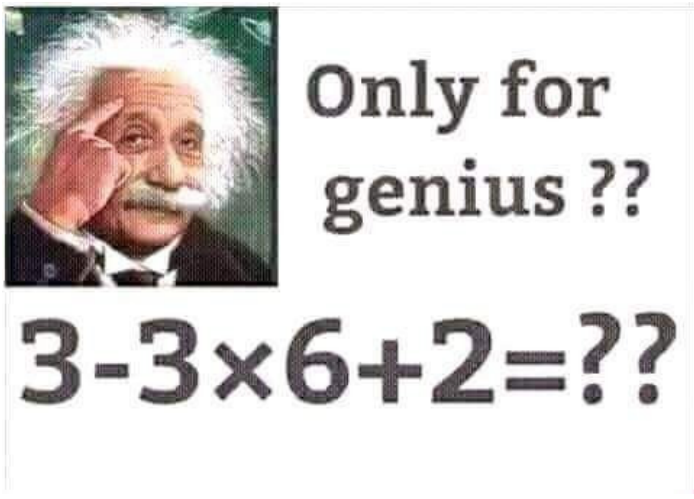
1. Enter sample code into the console (see syllabus for examples)
2. Observe the output in the console
3. Observe the environment
4. Use ↑↓ to scroll history
5. Use TAB for completion
6. Play with variations

```
x <- -10:10
y <- -x^2 -2 * x + 30

plot(x, y, type = "l",
     col = "blue")
abline(h = 0, col = "grey")
abline(v = 0, col = "grey")
```



R is Meme-Proof



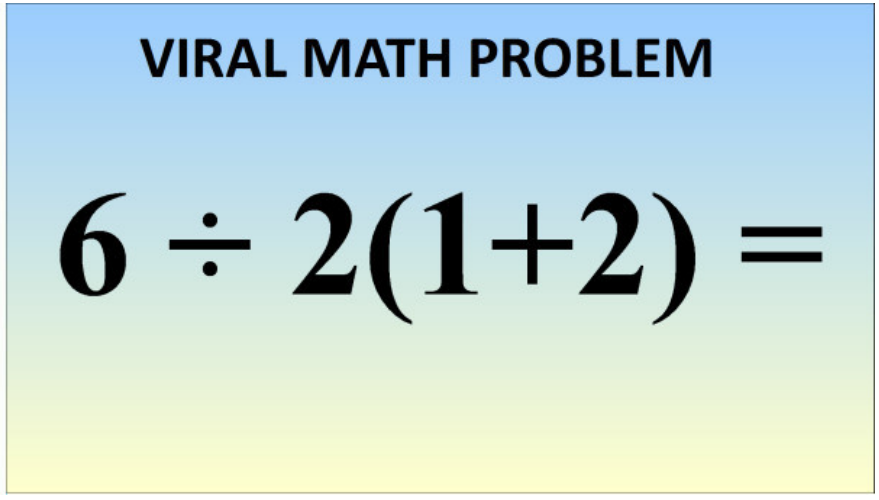


Figure 9: Aritmetic memes.

Quiz 1: Calculate Channel Flows

Determine the flow in a channel. Go to exercise 1 and answer the questions.

$$q = \frac{2}{3} C_d \sqrt{2g}\ bh^{3/2}$$

- q: Flow $[m^3/s]$.
- $C_d \approx 0.6$: Constant.
- $g = 9.81 m/s^2$
- b: Width of the weir $[m]$
- h: Water depth over weir $[m]$

Figure 10: Channel with weirplate (Photo: Coliban Water).

Scripts versus Console

- Store all code in a text file with `.R` extension
- Output in console, plots and viewer
- Use comments (start with #) to explain the code
- *File* > *New File* > *R Script*
- Open the `channel_flow.R` script in `inroduction` folder.
- Reverse-Engineer the code

```
## Question 2
h <- c(150, 136, 75) / 1000 # Create a vector
q <- (2/3) * Cd * sqrt(2 * 9.81) * b * h^(3/2)
mean(q) * 1000 # Convert to l/s
```

Reproducible Code

- Give meaningful names
- Use a consistent method, e.g.:
  - Only lower case: `channelflow`
  - Underscore for spaces: `channel_flow`
  - Camel case: `ChannelFlow`
- Use comments to explain the process
- Add links to documentation
- Automate as much as possible

The Tidyverse

An opinionated collection of R packages optimised for data science. All packages share an underlying design philosophy, grammar, and data structures.

```
install.packages("tidyverse")
library(tidyverse)
```

Load the `casestudy1.R` script in the `casestudy1` folder.

Data frames or 'tibbles'

- Rectangular data
- Variables in columns
- Observations in rows
- One variable in R environment
- Tidy data
- Read data:

```
dataframe <- read_csv(filename)
```

group	var	val
1	B	12
2	B	34
1	C	43
2	C	76
1	D	5
2	D	12

Figure 11: Data frame structure.

Filter a data frame

Town	Measure	Result
Bellmoral	THM	0.097
Bellmoral	Turbidity	0.2
Blancathey	THM	0.009
Blancathey	Turbidity	0.05
Merton	THM	0.28
Merton	Turbidity	0.1

Town	Measure	Result
Bellmoral	Turbidity	0.2
Blancathey	Turbidity	0.05
Merton	Turbidity	0.1

Figure 12: `filter(gormsey, Measure == "Turbidity")`

Quiz 2: Explore data

- Load the CSV file for the Gormsey system in the `casestudy1` folder.
- Explore the data.
- Answer the questions in Exercise 2 in your syllabus.
- You can cheat by opening the `quiz_02.R` script.

Descriptive Statistics

Safe Drinking Water Regulations 2015:

> *"the 95th percentile of results for samples in any 12 months must be less than or equal to 5.0 Nephelometric Turbidity Units."*

Guidance document:

> *"The method recommended by the department is described as the Weibull method and is the method adopted by the National Institute of Standards and Technology (NIST)."*

Percentiles

1. The data are placed in ascending order:
   $y_1, y_2, \ldots y_n$.
2. Calculate the rank of the required percentile
   - Weibull: $r = p(n+1)$
     - Excel: $r = 1 + p(n-1)$
3. Interpolate between adjacent numbers: $X_p = (1 - r_{frac})Y_{r_{int}} + r_{frac} Y_{r_{int+1}}$

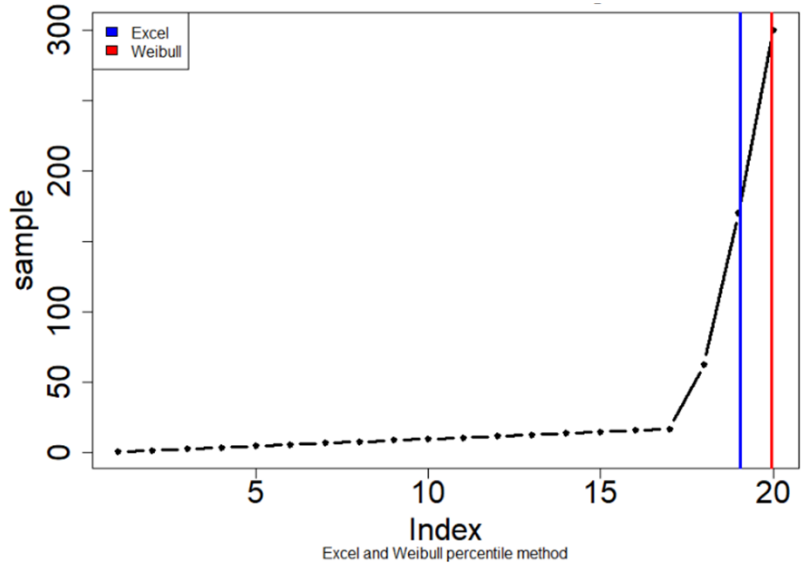


Figure 13: Explore the `percentiles.R` script in the `casestudy1` folder.

Grouping

Town	Measure	Result
Bellmoral	THM	0.097
Bellmoral	Turbidity	0.2
Blancathey	THM	0.009
Blancathey	Turbidity	0.05
Merton	THM	0.28
Merton	Turbidity	0.1

Town	Measure	Result
Bellmoral	THM	0.097
Blancathey	THM	0.009
Merton	THM	0.28

Town	Measure	Result
Bellmoral	Turbidity	0.2
Blancathey	Turbidity	0.05
Merton	Turbidity	0.1

Figure 14: `group_by(gormsey, Measure)`

Session 2 Program

- Recap
- Data Visualisation
- Creating Data Products

Figure 15: R for Water Professionals workshop (Melbourne, 2019).

Data Visualisation

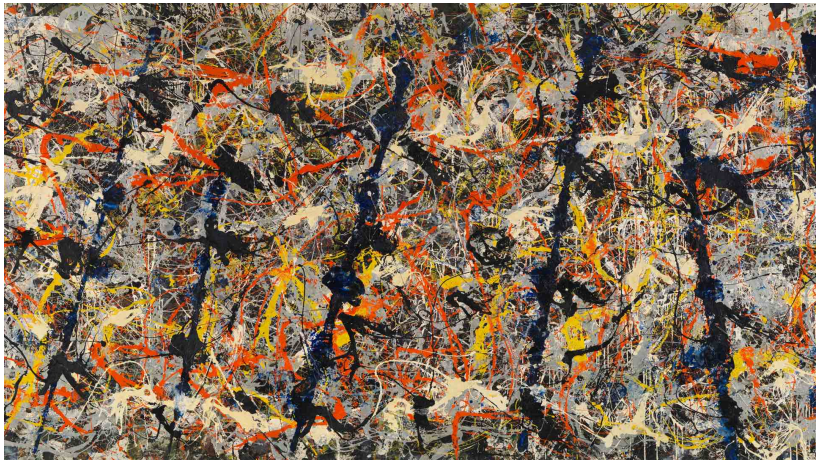


Figure 16: Jackson Pollock, *Blue Poles* (1973).

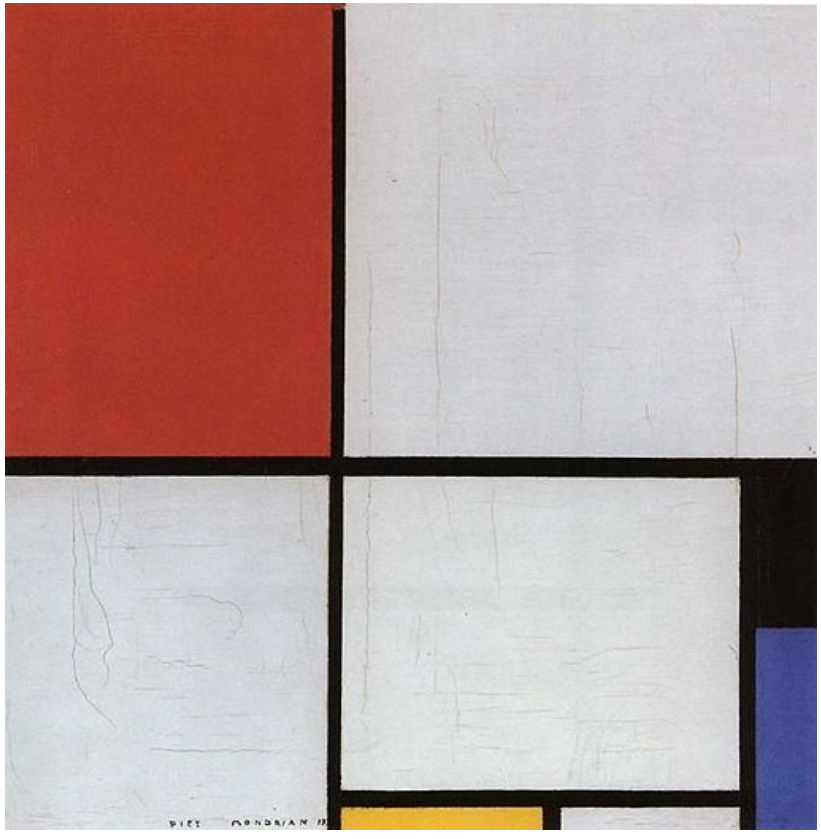


Figure 17: Piet Mondrian, *Composition in Red, Yellow and Blue* (1928)

Data-to-Pixel Ratio

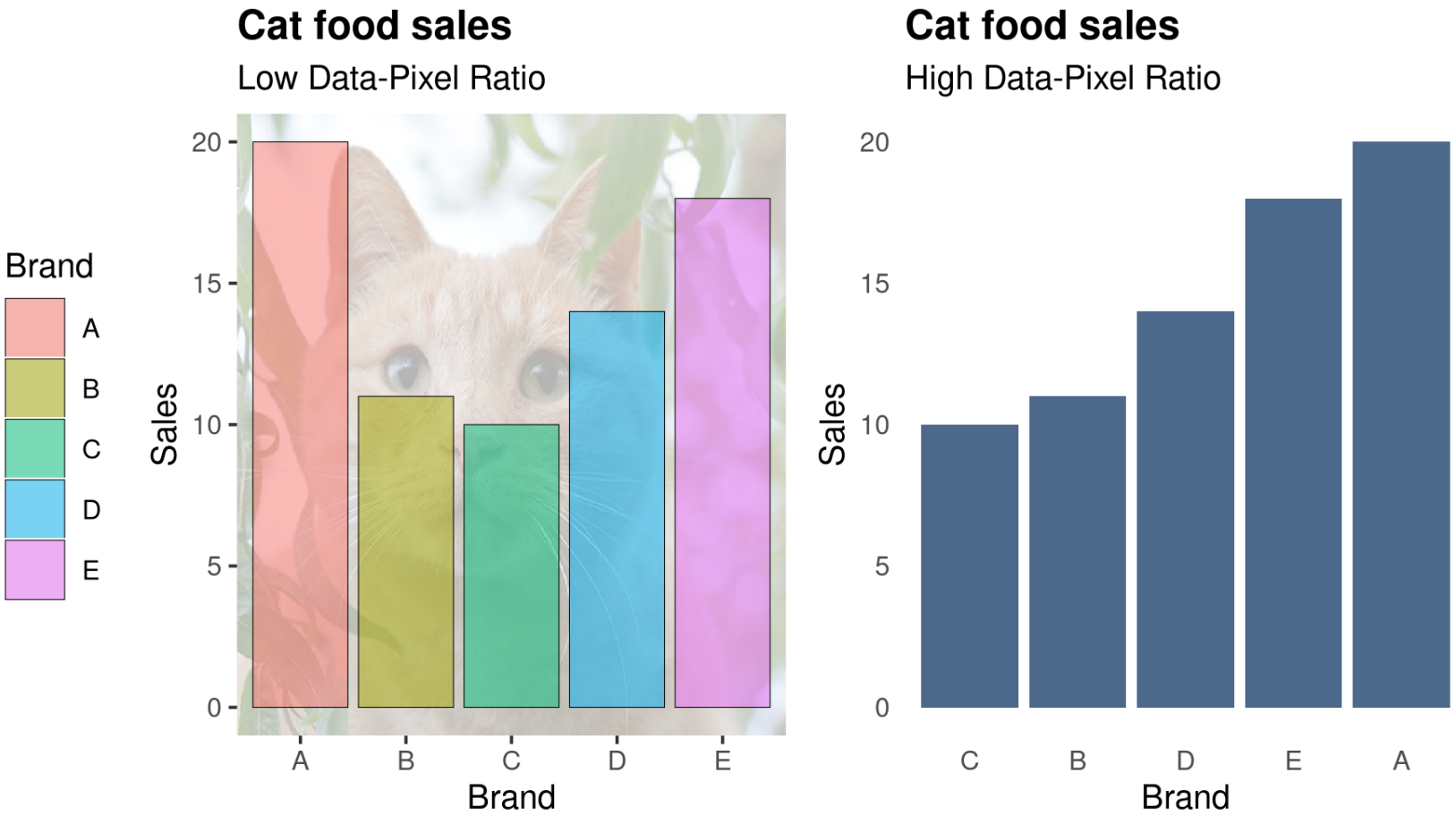


Figure 18: Maximise the data to pixel ratio for aesthetic visualisations.

Chart Chooser

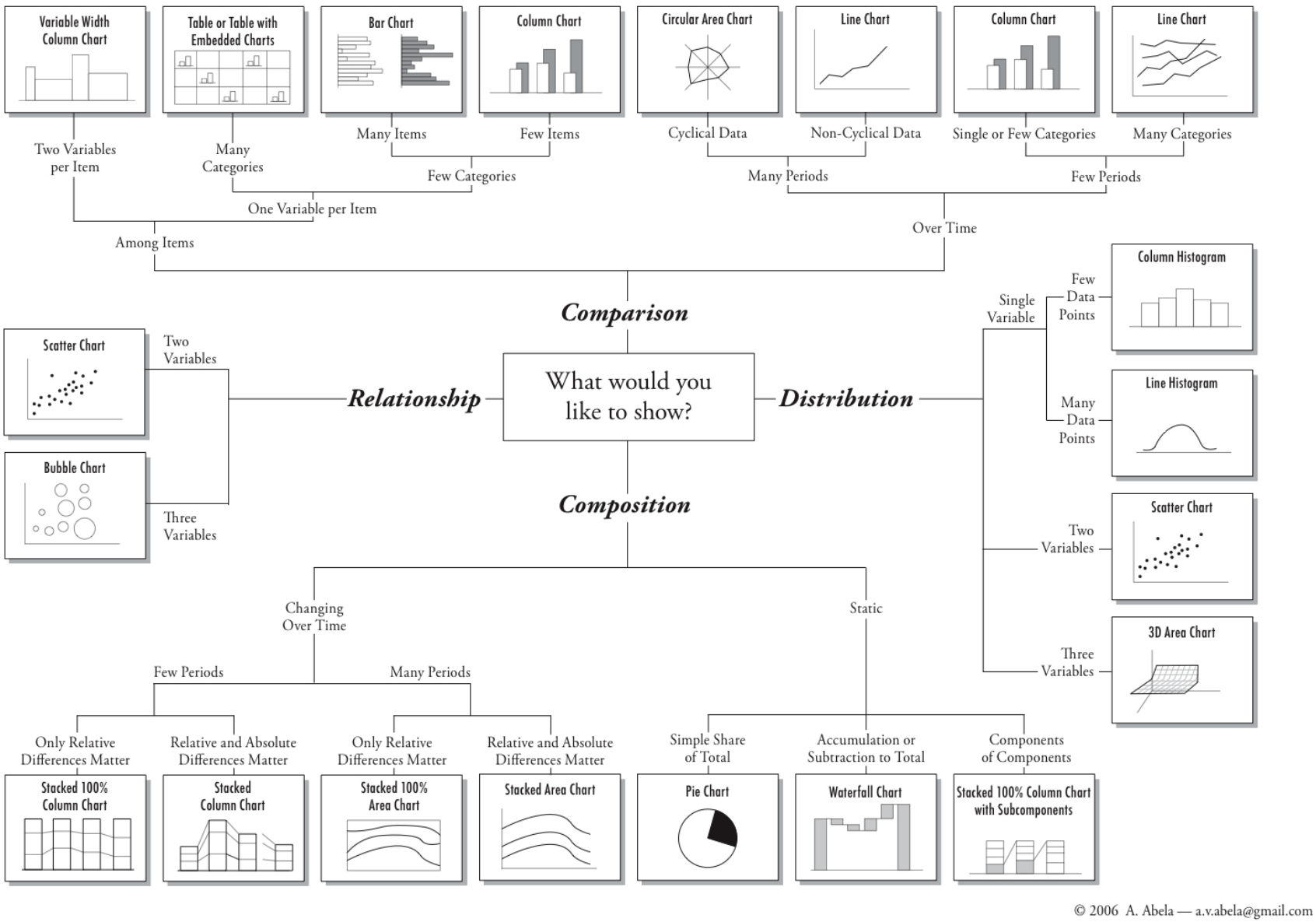
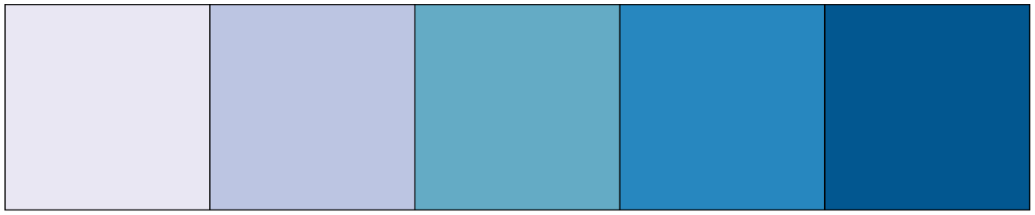


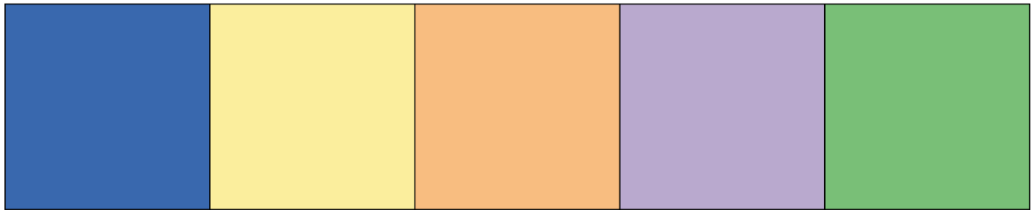
Figure 19: Chart suggestions by Andrew Abela.

Use Colours Sparingly



Sequential

Diverging



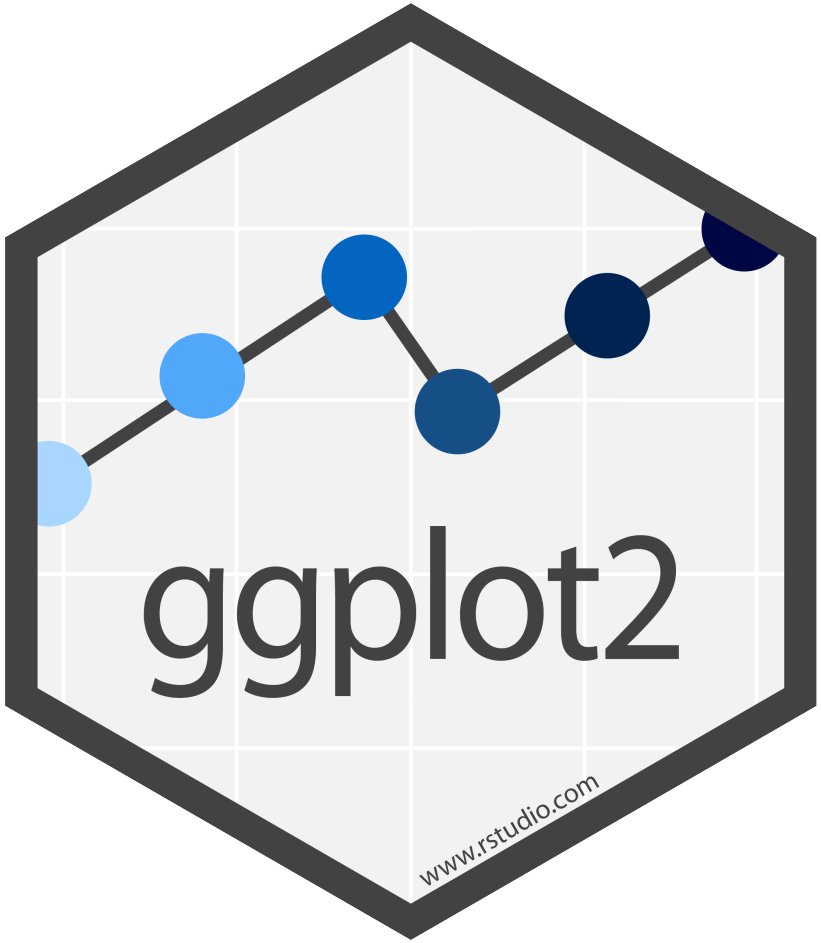
Qualitative

Figure 20: Types of colour pallets. Go to colorbrewer2.org for details.

ggplot2

- System for creating graphics, based on *The Grammar of Graphics*.
- Go to ggplot2.tidyverse.org for documentation.
- Included in the Tidyverse. You can call it separately with:

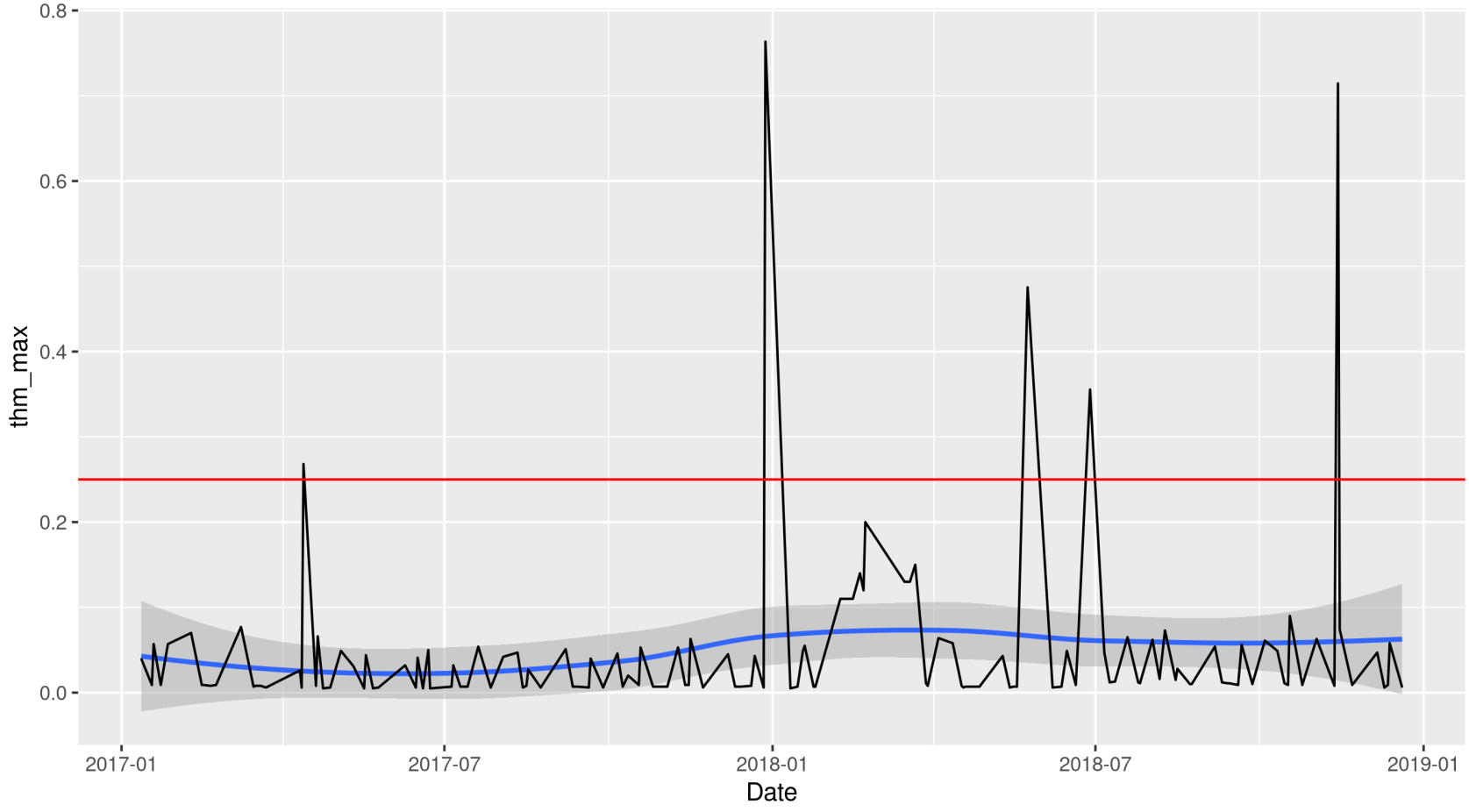
```
library(ggplot2)
```



Grammar of Graphics

Figure 21: Leland Wilkinson, *Grammar of Graphics* (2005).

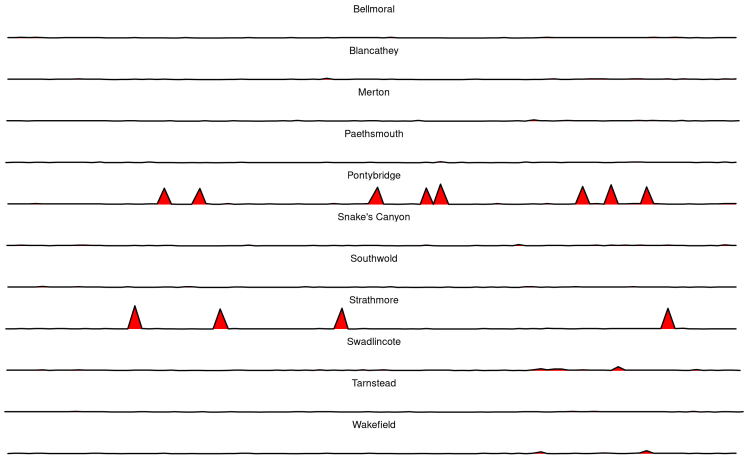
ggplot2 Example



Visualisation Exercise

Use your knowledge of the Gormsey data to create two visual data stories. Use the following four steps:

1. Explore the data and define the story you want to tell.
2. Decide on the best way to visualise the story.
3. Develop the basic visualisation.
4. Select a theme and annotate the graph.



Data Science Workflow

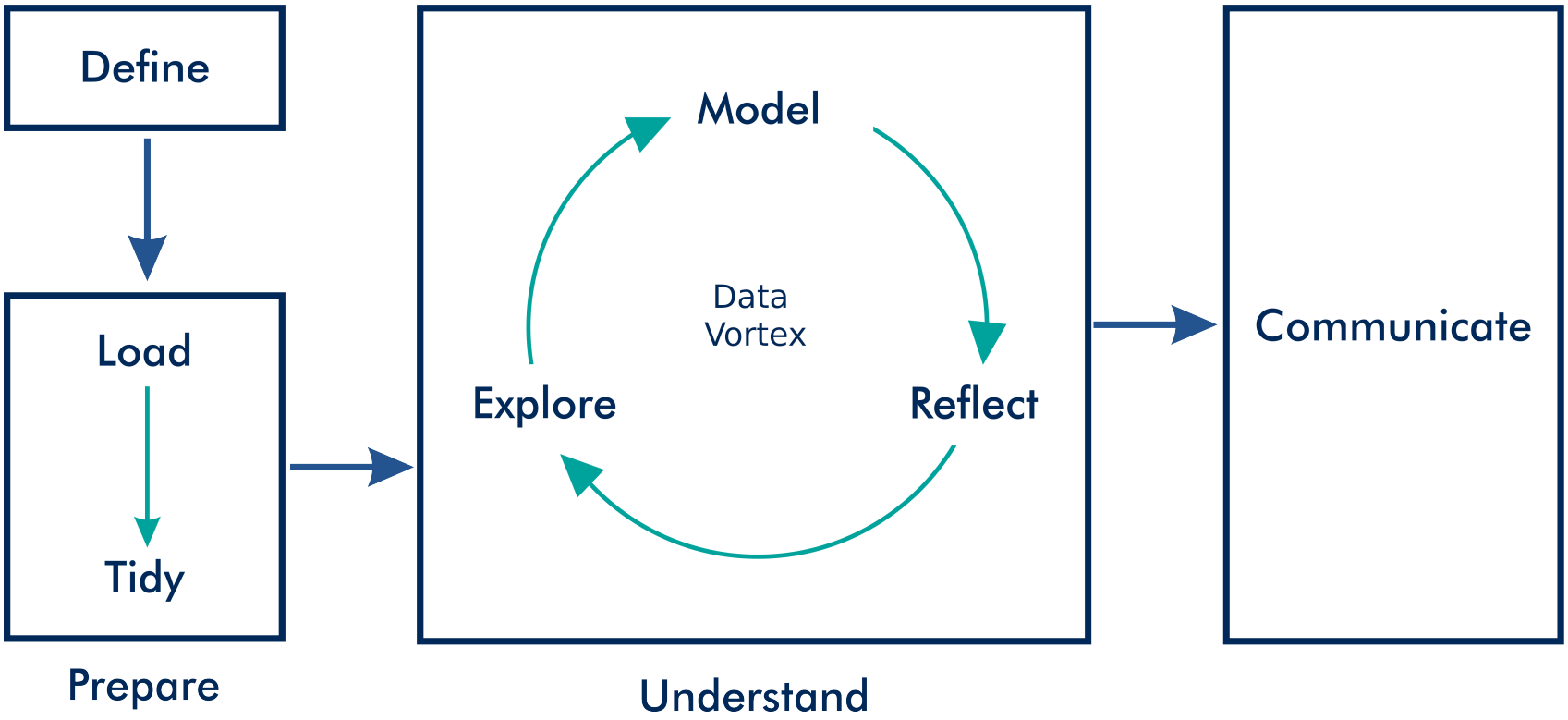


Figure 22: Data science wokflow

Data Products

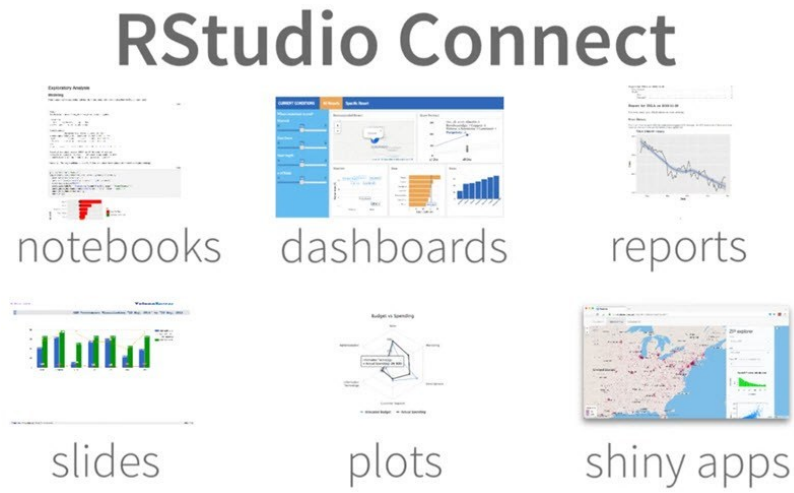
Static

- Word documents
- PowerPoint presentaions
- Web pages

Dynamic

- Shiny application
- Shiny presentation

Share - RStudio Connect - Your own server



RMarkdown

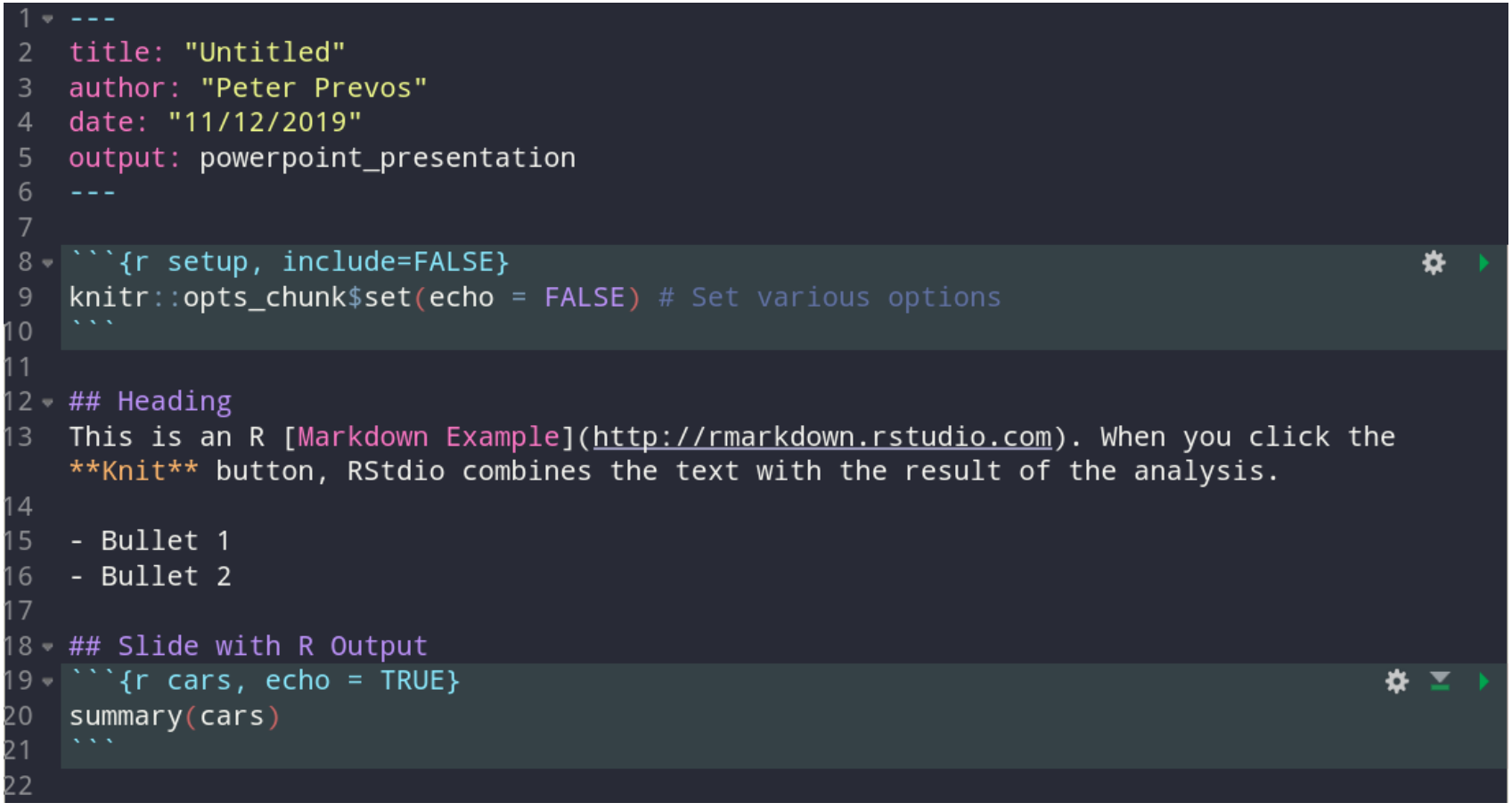
Literate programming:

- Combine prose with code
- Link the code to dynamic data
- Generate shareable output from code

Data products:

- Reports
- Web sites
- Presentations
- Applications (dashboards)

RMarkdown Syntax



```
---
title: "Untitled"
author: "Peter Prevos"
date: "11/12/2019"
output: powerpoint_presentation
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE) # Set various options
```

Heading
This is an R [Markdown Example](http://rmarkdown.rstudio.com). When you click the
Knit button, RStdio combines the text with the result of the analysis.

- Bullet 1
- Bullet 2

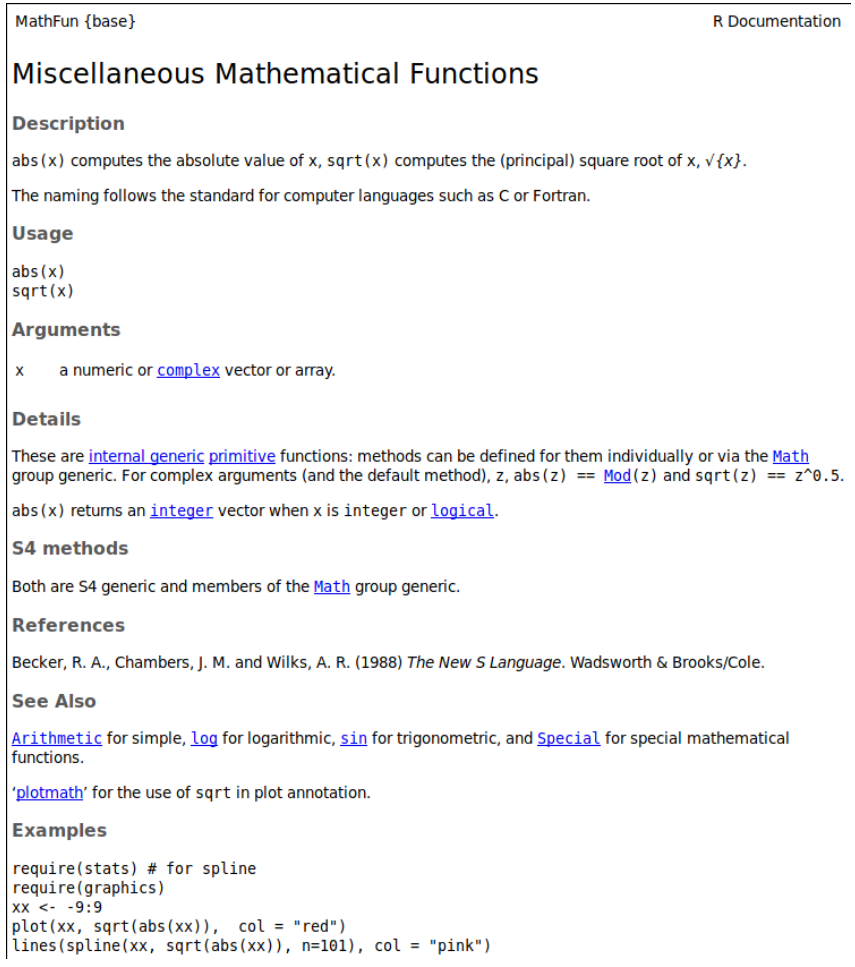
Slide with R Output
```{r cars, echo = TRUE}
summary(cars)
```

```

Figure 23: RMarkdown syntax example

# Finding Help

- Built-in `help()` function
- Cheat sheets (RStudio and Tidyverse websites)
- Use the R4H2O Slack channel
- Twitter #rstats
- Reddit rstats, rlanguage
- stackoverflow.com
- Google the problem



MathFun {base} R Documentation

## Miscellaneous Mathematical Functions

### Description

`abs(x)` computes the absolute value of x, `sqrt(x)` computes the (principal) square root of x, $\sqrt{x}$.

The naming follows the standard for computer languages such as C or Fortran.

### Usage

```
abs(x)
sqrt(x)
```

Arguments

x	a numeric or complex vector or array.

Details

These are internal generic primitive functions: methods can be defined for them individually or via the Math group generic. For complex arguments (and the default method), z, `abs(z) == Mod(z)` and `sqrt(z) == z^0.5`.

`abs(x)` returns an integer vector when x is `integer` or logical.

S4 methods

Both are S4 generic and members of the Math group generic.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

Arithmetic for simple, log for logarithmic, sin for trigonometric, and Special for special mathematical functions.

'plotmath' for the use of `sqrt` in plot annotation.

Examples

```
require(stats) # for spline
require(graphics)
xx <- -9:9
plot(xx, sqrt(abs(xx)),  col = "red")
lines(spline(xx, sqrt(abs(xx)), n=101), col = "pink")
```

Figure 24: screenshot of help window.

Mini Hackathon

To close this day, we will do a mini hackathon.

1. Create a script that results in a PowerPoint presentation about the Gormsey data.
2. Pick a story you like to tell about this data.
3. Create a RMarkdown script that results in a Powerpoint presentation.
   - Add an introduction.
   - Explore the data.
   - Share the story.