Data Science Practice

STATS 369 Coursebook: Week 1

Plan for this week

- **[L01] Introduction**
  - Course logistics
  - Course overview
  - (Quick) introduction to R
- **[L02] Reproducibility**
  - Reproducible data analysis
  - (r)markdown
  - Examples
- **[L03] Package `{tidyverse}`**
  - Tidy data (or not)
  - `{tidyverse}` operations

Course logistics

Lectures

Instructors

Weeks 1 to 6

Liza Bolton `liza.bolton@auckland.ac.nz`

Time & Location

📍 BIO100

Wed, 4 to 5 PM
Thurs, 1 to 2 PM
Fri, 10 to 11 AM

Weeks 7 to 12

Lisa Chen `lisa.chen@auckland.ac.nz`

📚 Textbooks

We use two textbooks in this course. Both are available as free e-book version.

- *R for Data Science* by Grolemund and Wickham (note: there is now a second edition, but the first edition is what our references are to, currently)
- *An Introduction to Statistical Learning* (2nd Edition), by James, Witten, Hstie, and Tibshirani

Labs

Time & Location 📍 302.190

Fri, 12 to 1 PM
Fri, 1 to 2 PM
Fri, 2 to 3 PM

Choose one time to attend. Liza will not be available at the 1 to 2 PM lab, but there will be a tutor. Attendance is not required but is highly recommended. Labs will not be recorded.

There will be **10** labs, starting from **week 1** (this Friday).

Structure

Each lab has hand-on exercise; and it is to be **handed in electronically** a week after it is given out. Labs are mainly based on the previous week's lecture.

Assessments

- 10 labs, worth a total of 6% (graded on 'reasonable attempt' basis) — **6%**
- Assignments — **24%** (4 * 6% each)
- Midterm test (25 August, in-person) — **20%**
- Final exam (Set by Exams Office, in-person) — **50%**

Restricted book

Both your test and exam will be 'restricted book'. This means you can bring a single A4 sheet (you can write on both sides) with any notes you would like to bring in with you.

Overview of the course

Course overview

- Introduction
- Data manipulation and visualisation (in R)
  - tidy data concept
  - `{tidyverse}`
  - `{ggplot2}`
- Modelling (in R)
  - regression
  - trees
  - neural networks
- Ethics
- Review

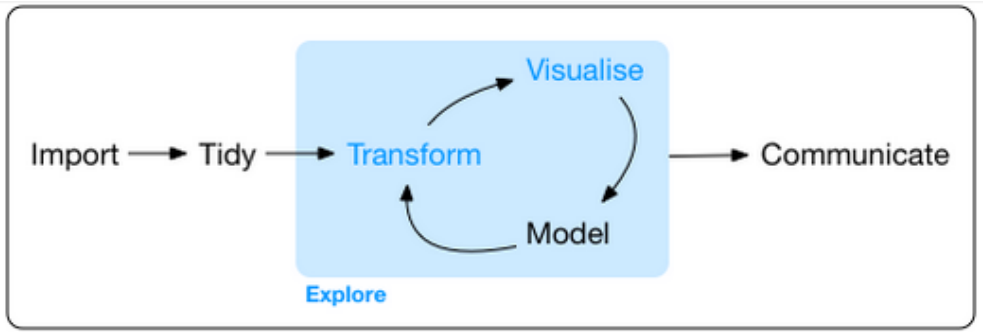


Image Sourced from the 'R for Data Science' book

Introduction to R

Why R?

Currently, there are two major languages (commonly) used in data science: R and Python

- We used R in this course, but you really should learn both (use the right tool for the right job).
- R is easier, but might be slower.
- Python is a more general programming language.
- R has better packages with new/common statistical toolkits.

Why people ❤️ R (optional)

Here are some videos from University of Toronto faculty & students about why they love using R. (I used to teach there, and you'll notice a lot of Southern Hemisphere accents for a Canadian Uni! 😉)

Why people ❤️ R (continued)

Why I love R - Monica Alexander

Why I love R - Liza Bolton

RStudio

Data analysis IDE built on top of (mainly) R. FYI -- the current version also supports SQL, Python etc.

Supports `rmarkdown` for reproducible data analysis.

Uses a feature called *Projects* (`.RProj`) which makes directory management easier.

Resource

Introduction to R

Undergraduate student-developed modules from the University of Toronto

Lecture 2

Plan for this week

- **[L01] Introduction**
  - Course logistics
  - Course overview
  - (Quick) introduction to R
- **[L02] Reproducibility**
  - Reproducible data analysis
  - (r)markdown
  - Examples
- **[L03] Package `{tidyverse}`**
  - Tidy data (or not)
  - `{tidyverse}` operations

Reproducible Data Analysis

Why do we need reproducibility?

Any data analysis work or report is likely to be rerun -- especially out in the real world!

- your boss has new problem
- your data set gets updated (possibly by your colleagues)
- review or QA may want a (slightly) different analysis
- revisit of the past work (e.g. re-do data cleaning)

We need tools and means for re-doing and updating data analyses *easily* and *reliably*.

Version control

It is important! Especially when you are collaborating with others in a (large) project.

- you want to keep track of the work you (and/or your colleagues) have done
- you want to be able to 'revert' back to older version when you can/need to
- you want an easy way to explore ideas without affecting the current (stable) work progress.

Git

All data scientists should learn and use Git in their day-to-day work.

Here is a useful page to get you started.

(r)markdown

Markdown

A simple text markup language that allows the conversion to many output formats (the original output format is HTML).

- `*italics*` *italics*
- `**bold**` **bold**
- `~~strikeout~~` ~~strikeout~~
- mathematical formulas e.g. `E = mc^2`,

$$E = mc^2$$

rmarkdown

`rmarkdown` is an extension that supports R code chunks to be embedded and run, and output to be rendered as part of the report.

A quick tour

rmarkdown quick tour by RStudio

Lecture 3

Plan for this week

- **[L01] Introduction** 👍
  - Course logistics
  - Course overview
  - (Quick) introduction to R
- **[L02] Reproducibility** 👍
  - Reproducible data analysis
  - (r)markdown
  - Examples
- **[L03] Package `{tidyverse}`**
  - Tidy data (or not)
  - Common data operations using `{tidyverse}`

Tidy Data (or not)

Tidy data

A dataset is a collection of **values**, which is recorded by:

- a *variable* (or *attribute*, or *feature*); and
- an *observation* (or *record*)

Variables contains all values that measure the same attribute (e.g., height, weight) across units (e.g., a group of people).

An **observation** contains all values measured on a same unit (e.g., a person).

Tidy data has one row per observation and one column per variable, and one table per type of observational unit.

In the real world ...

Data is often **NOT** tidy.

Tidy data is natural for **computing,** but it may or may not be *human friendly*. It is often useful to split a variable across columns to simplify comparisons.

If your data source is a human-readable tables, it may need tidying.

Here is an example of untidy data (named `alarms_count.df`).

alarm_id	d.2023.05.17	d.2023.05.18	d.2023.05.19	d.2023.05.20	d.2023.05.21	d.2023.05.22
alarms_N	322	548	199	40	513	664
alarms_O	253	342	577	668	309	575
alarms_Y	53	138	525	132	703	281
alarms_D	185	671	114	259	32	55
alarms_X	555	155	741	694	358	272
alarms_H	1	40	337	390	302	479
alarms_G	473	75	698	614	418	278
alarms_V	456	458	738	52	675	608
alarms_J	650	64	240	302	518	305
alarms_S	563	398	562	154	95	451

Here is an example of tidy data (named `alarms_info.df`).

alarm_ID	alarms_region
alarms_N	AKL_North
alarms_O	AKL_Central
alarms_Y	Waiheke
alarms_D	AKL_South
alarms_X	AKL_Others
alarms_H	AKL_North
alarms_G	AKL_Central
alarms_V	Waiheke
alarms_J	AKL_South
alarms_S	AKL_Others

(Potential) questions of interest

- What is the total number of alarms per region each day?
- Assuming that an alarm that occurs over 400 times/day is highly frequent, what is the proportion of highly frequent alarms in each region per day?
- What is the average number of alarms per day in each region?
- Which days of the week has the highest and the lowest average number of alarms across Auckland?

Common data operations using `{tidyverse}`

{tidyverse} package

{tidyverse} is *An opinionated collection of R packages designed for data science*

Pro:

- Good abstractions for data management and summarising
- Database interfaces for larger datasets
- Some operations are optimised (i.e., faster)
- Consistent syntax and design
- Trendy

Con:

- Not so good for modelling (yet)
- Might be a pain to use in programming

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

Common operations

- reshaping: `pivot_wider`, `pivot_longer`...
- subsetting and addition: `select`, `filter`, `mutate`...
- joining: `left_join`, `right_join` ...
- grouping: `group_by`, `summarise`...

Reshaping

`pivot_longer` function can *gather* columns into rows; `pivot_wider` function does the opposite -- it *spread*s rows into columns. (Note, both `gather` are `spread` functions were retired and replaced by `pivot_longer` and `pivot_wider` respectively)

```
knitr::kable(head(alarms_count.df[,1:7]))
```

alarm_id	d.2023.05.17	d.2023.05.18	d.2023.05.19	d.2023.05.20	d.2023.05.21	d.2023.05.22
alarms_N	322	548	199	40	513	664
alarms_O	253	342	577	668	309	575
alarms_Y	53	138	525	132	703	281
alarms_D	185	671	114	259	32	55
alarms_X	555	155	741	694	358	272
alarms_H	1	40	337	390	302	479

```
alarms_count_long.df <- alarms_count.df %>%
  gather('date', 'frequency', -alarm_id)
knitr::kable(head(alarms_count_long.df))
```

alarm_id	date	frequency
alarms_N	d.2023.05.17	322
alarms_O	d.2023.05.17	253
alarms_Y	d.2023.05.17	53
alarms_D	d.2023.05.17	185
alarms_X	d.2023.05.17	555
alarms_H	d.2023.05.17	1

Piping operator

`%>%` is called a piping operator. It takes the left-side object as the *first* argument (or . argument) for the right-hand side function. E.g.

`x %>% myFun(y)` is the same as `myFun(x, y)` `x %>% myFun(y, ., z)` is the same as `myFun(y, x, z)`

Piping makes the code more readable, especially for data wrangling tasks (examples to come).

Reshaping (Cont.)

`separate` separates one column into multiple columns; `unite` does the opposite -- it units multiple columns into one.

```
knitr::kable(head(alarms_count_long.df))
```

alarm_id	date	frequency
alarms_N	d.2023.05.17	322
alarms_O	d.2023.05.17	253
alarms_Y	d.2023.05.17	53
alarms_D	d.2023.05.17	185
alarms_X	d.2023.05.17	555
alarms_H	d.2023.05.17	1

```
alarms_count_long.df <- alarms_count_long.df %>%
  separate(date, c('prefix', 'year', 'month', 'day'))
knitr::kable(head(alarms_count_long.df))
```

alarm_id	prefix	year	month	day	frequency
alarms_N	d	2023	05	17	322
alarms_O	d	2023	05	17	253
alarms_Y	d	2023	05	17	53
alarms_D	d	2023	05	17	185
alarms_X	d	2023	05	17	555
alarms_H	d	2023	05	17	1

Subsetting and adding columns

`select` subsets columns from a dataset; `filter` subsets rows from a dataset. `mutate` generates new columns (often based on existing columns).

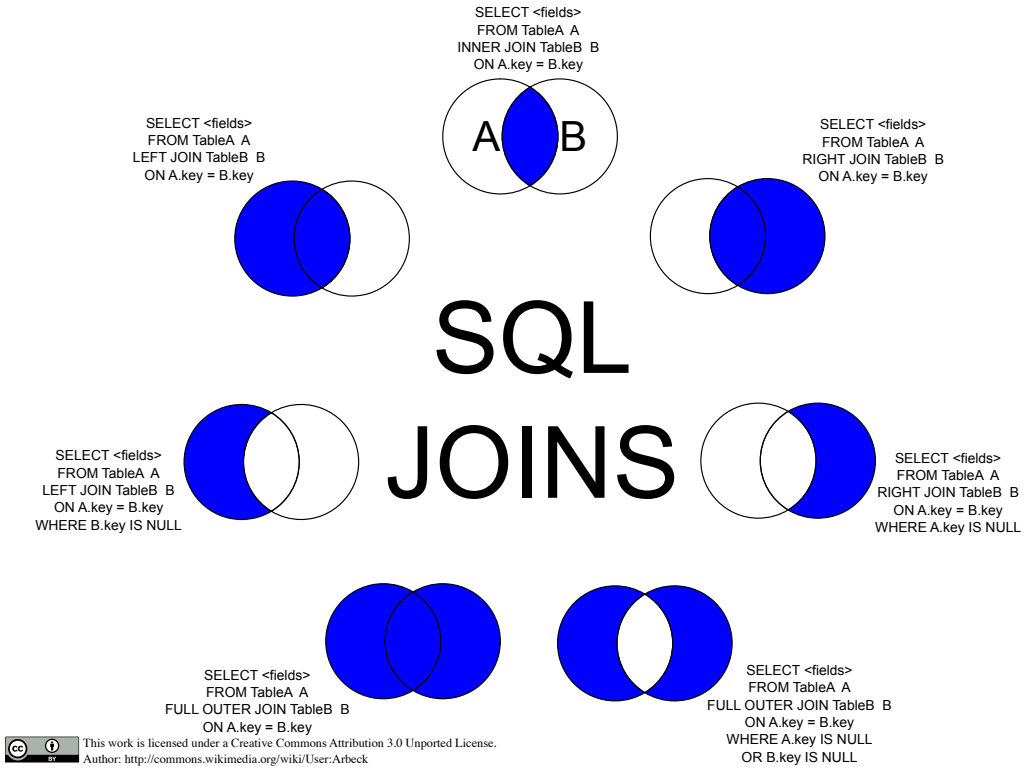
```
alarms_count_sa.df <- alarms_count_long.df %>%
  select(-prefix) %>%  # remove 'prefix'
  mutate(category = if_else(frequency >= 400, 'High', 'Low')) # add 'category'

knitr::kable(head(alarms_count_sa.df))
```

alarm_id	year	month	day	frequency	category
alarms_N	2023	05	17	322	Low
alarms_O	2023	05	17	253	Low
alarms_Y	2023	05	17	53	Low
alarms_D	2023	05	17	185	Low
alarms_X	2023	05	17	555	High
alarms_H	2023	05	17	1	Low

Joining

The joining in `tidyverse` is very similar to joins in SQL. i.e.



```
names(alarms_count_sa.df)
```

```
## [1] "alarm_id"  "year"      "month"     "day"       "frequency" "category"
```

```
names(alarms_info.df)
```

```
## [1] "alarm_ID"      "alarms_region"
```

```
alarms_merged.df <- alarms_count_sa.df %>%
  left_join(alarms_info.df, by = c('alarm_id' = 'alarm_ID'))
```

```
knitr::kable(head(alarms_merged.df))
```

alarm_id	year	month	day	frequency	category	alarms_region
alarms_N	2023	05	17	322	Low	AKL_North
alarms_O	2023	05	17	253	Low	AKL_Central
alarms_Y	2023	05	17	53	Low	Waiheke
alarms_D	2023	05	17	185	Low	AKL_South
alarms_X	2023	05	17	555	High	AKL_Others
alarms_H	2023	05	17	1	Low	AKL_North

Summarising

`group_by` groups the dataset into chunks, `summarise` performs a summary that we specify (e.g., mean, median, max, min) within the chunk.

```
alarms_summarised.df <- alarms_merged.df %>%
  group_by(year, month, day, alarms_region) %>%
  summarise(total = sum(frequency),
            percent_high = 100*sum(category == 'High') / n())
```

```
knitr::kable(head(alarms_summarised.df))
```

year	month	day	alarms_region	total	percent_high
2023	05	17	AKL_Central	726	50
2023	05	17	AKL_North	323	0
2023	05	17	AKL_Others	1118	100
2023	05	17	AKL_South	835	50
2023	05	17	Waiheke	509	50
2023	05	18	AKL_Central	417	0

Questions of interest -- revisit

We have answered the following two questions.

- What is the total number of alarms per region each day?
- Assuming that an alarm that occurs over 400 times/day is highly frequent, what is the proportion of highly frequent alarms in each region per day?
- What is the average number of alarms per day in each region?
- Which days of the week has the highest and the lowest average number of alarms across Auckland?

Raw data -- revisit

```
knitr::kable(alarms_count.df[,1:7])
```

alarm_id	d.2023.05.17	d.2023.05.18	d.2023.05.19	d.2023.05.20	d.2023.05.21	d.2023.05.22
alarms_N	322	548	199	40	513	664
alarms_O	253	342	577	668	309	575
alarms_Y	53	138	525	132	703	281
alarms_D	185	671	114	259	32	55
alarms_X	555	155	741	694	358	272
alarms_H	1	40	337	390	302	479
alarms_G	473	75	698	614	418	278
alarms_V	456	458	738	52	675	608
alarms_J	650	64	240	302	518	305
alarms_S	563	398	562	154	95	451

```
knitr::kable(head(alarms_info.df))
```

alarm_ID	alarms_region
alarms_N	AKL_North
alarms_O	AKL_Central
alarms_Y	Waiheke
alarms_D	AKL_South
alarms_X	AKL_Others
alarms_H	AKL_North

Code -- putting everything together

```
alarms_insight.df <- alarms_count.df %>%
  gather('date', 'frequency', -alarm_id) %>%
  separate(date, c('prefix', 'year', 'month', 'day')) %>%
  select(-prefix) %>%
  mutate(category = if_else(frequency >= 400, 'High', 'Low')) %>%
  left_join(alarms_info.df, by = c('alarm_id' = 'alarm_ID')) %>%
  group_by(year, month, day, alarms_region) %>%
  summarise(total = sum(frequency),
            percent_high = 100*sum(category == 'High') / n())
```

```
knitr::kable(head(alarms_insight.df))
```

year	month	day	alarms_region	total	percent_high
2023	05	17	AKL_Central	726	50
2023	05	17	AKL_North	323	0
2023	05	17	AKL_Others	1118	100
2023	05	17	AKL_South	835	50
2023	05	17	Waiheke	509	50
2023	05	18	AKL_Central	417	0

Resources

- `rmarkdown` cheatsheet (or this more screen reader friendly version)
- `dpylr` cheatsheet (or this more screen reader friendly version)
    - `tidyr` cheatsheet (or the more screen reader friendly version)

  Note: Translated versions of these cheatsheets are available in Chinese for `dplyr` and `tidyr`. Several of the cheatsheets are available in additional languages, as well.