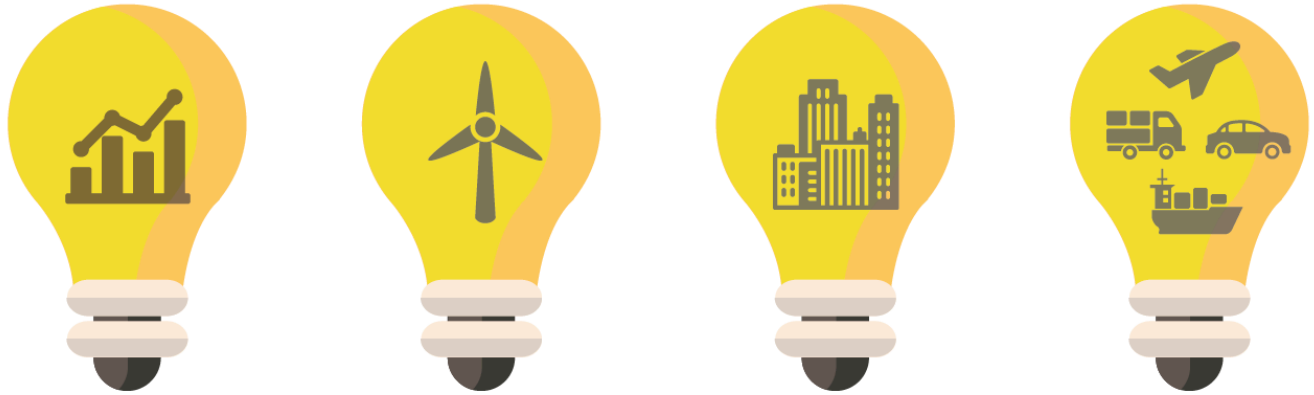
BENV0091 Lecture 1: Introduction

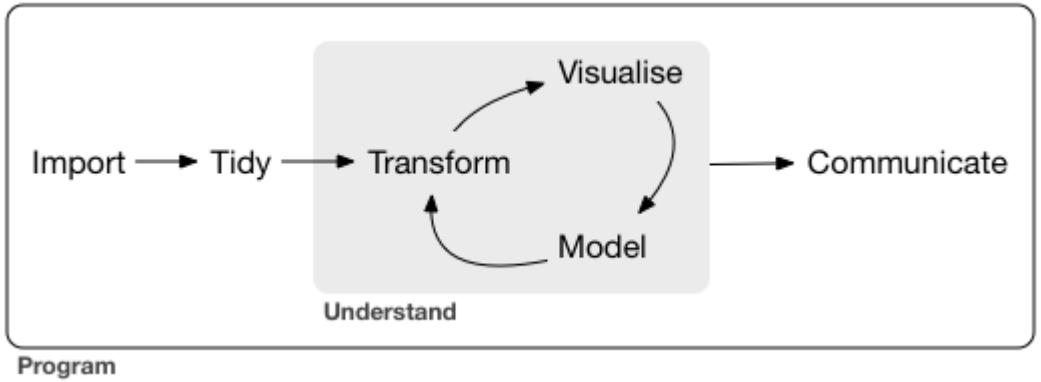
Patrick de Mars



Lecture Part 1: Course Overview

Course Aims

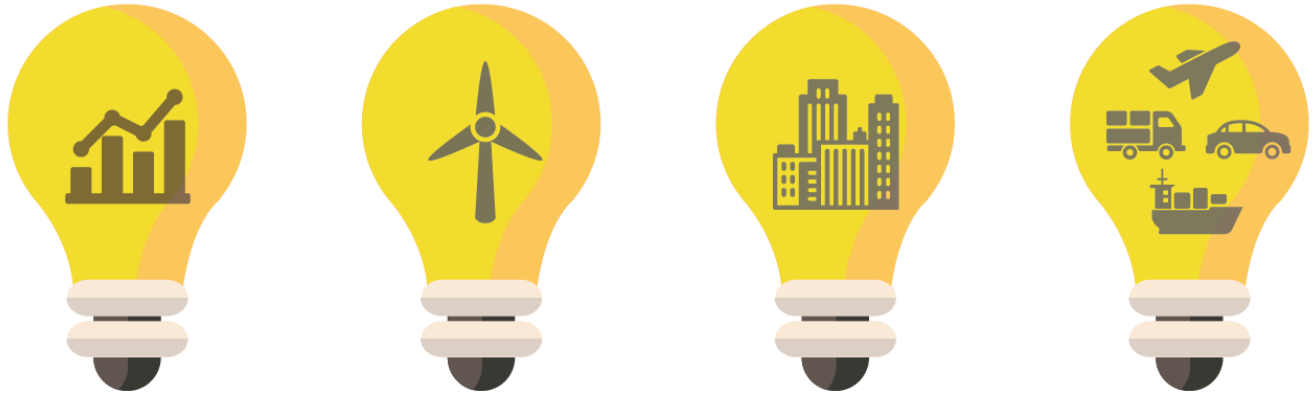
- Build a toolbox for energy data science!
- Learn to use data effectively:
  - Handle data **confidently** and **carefully**
  - **Draw insights** from data
  - **Solve problems** with data



R for Data Science by Hadley Wickham and Garrett Grolemund

Course Overview

- Programming in R
- Data wrangling
- Data visualisation
- Tools and best practices for data science
- Time series
- Data retrieval (APIs and web scraping)
- Supervised learning
- Unsupervised learning
- R Shiny
- Guest lecturer!

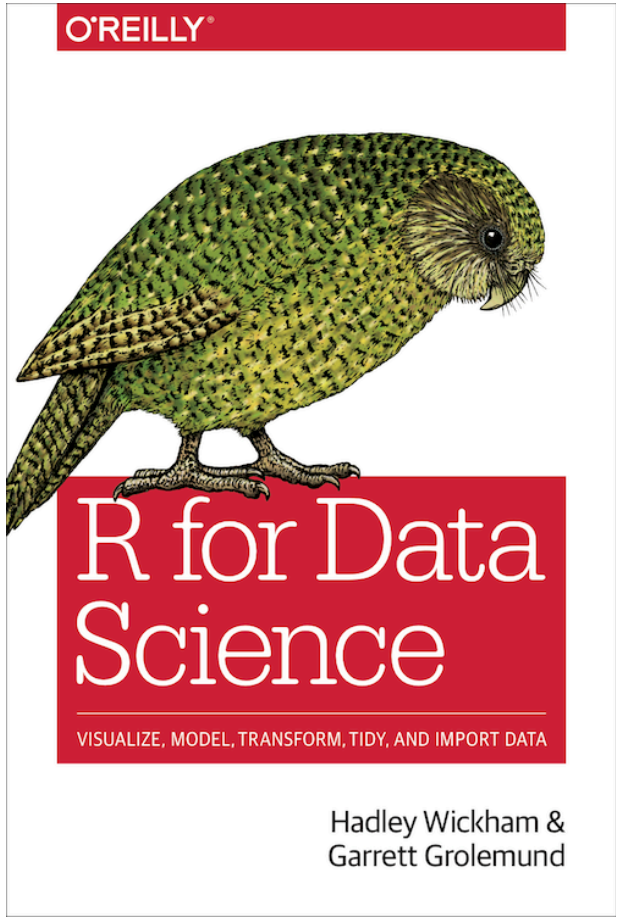


Assessment

- Group project 😬
- One report of 5000 words
- Collaborative effort: one report, one grade
- Groups to be assigned before reading week!

Textbook

- R for Data Science by Hadley Wickham and Garrett Grolemund
- It is free! https://r4ds.had.co.nz/
- But it won't teach you everything… get familiar with **Stack Overflow**



Lecture Part 2: Introduction to R

Task: Install R and RStudio

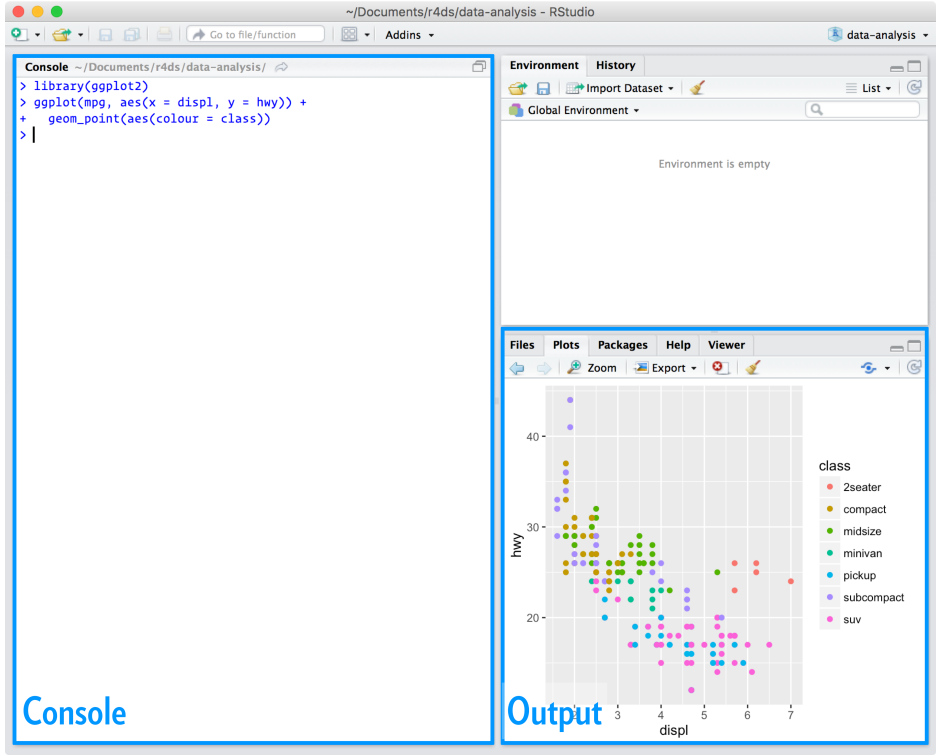
- Go to: http://www.r-project.org
- Go to CRAN
- Choose a (local) mirror
- Download R for your operating system
- Open RStudio

Why R?

- Designed for data science
- Quick to pick up
- Powerful when you need it to be
- Encourages statistical rigour
- Widely used by academics and professionals
- Enables you to work efficiently
- Recommended reading: *How R Helps Airbnb Make the Most of Its Data*

RStudio: Integrated Development Environment (IDE)

- Lets you view/write code, files, plots, notebooks (more on this later) all in one place
- Task: do some calculations in the console
- `print(x)` is a **function** that displays `x` on your screen
- Inputs to a function are called **arguments**
- Task: print hello world!



Function: a reusable piece of code to perform a particular task

Type `?print` to read the documentation for print()

Variables

- A name we give to some value
- R uses <- to assign variables
- We can manipulate and update variables
- Variables have **types** – these determine how we can operate on them
- We can convert between types using `as.character()`, `as.numeric()` etc.
- Logical variables can take two values: TRUE or FALSE which are **not** characters!
- Task: get familiar with creating variables; changing type; using mode(x)

Type	Examples
numeric	3, 12.9, 5e-7
character	“hello world”, “3.3”
logical	TRUE, FALSE

x <- 2 assigns variable x to value 2

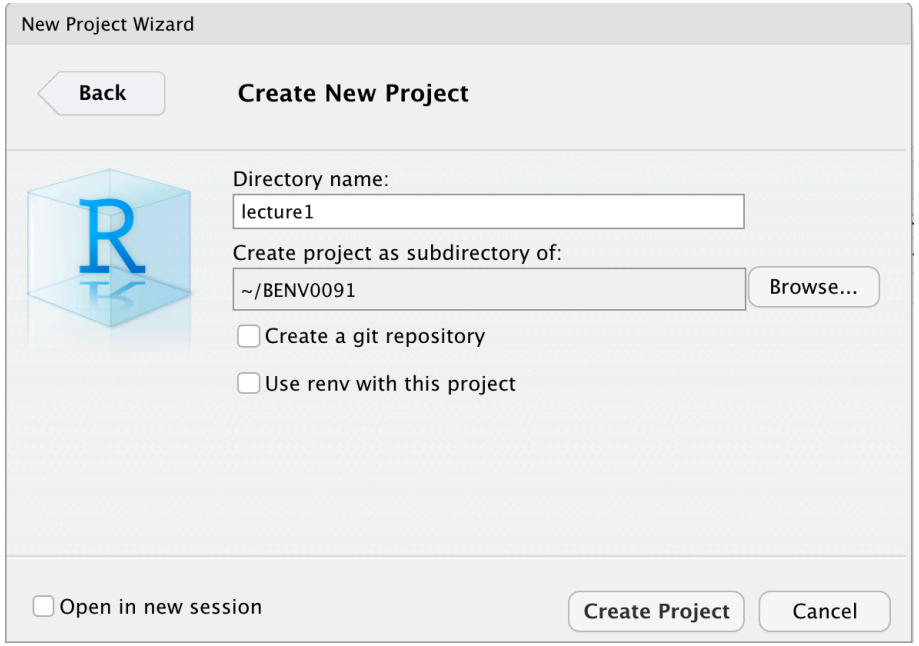
Use mode(x) to get type of x

Convert between types using as.character(), as.numeric()…

== think “is exactly equal to"

Create an R Project

- Create a new directory called BENV0091 somewhere sensible on your machine
- Go to File > New Project > New Directory and call it lecture1, and make it a subdirectory of BENV0091
- Projects load your settings including **working directory** when you open it
- Working directory determines where R will look for files: it is often the cause of bugs!
- Task: make sure your working directory is set to '…/BENV0091/lecture1'



Use getwd() to print your working directory

Use setwd(x) to manually change your working directory to x

Scripts

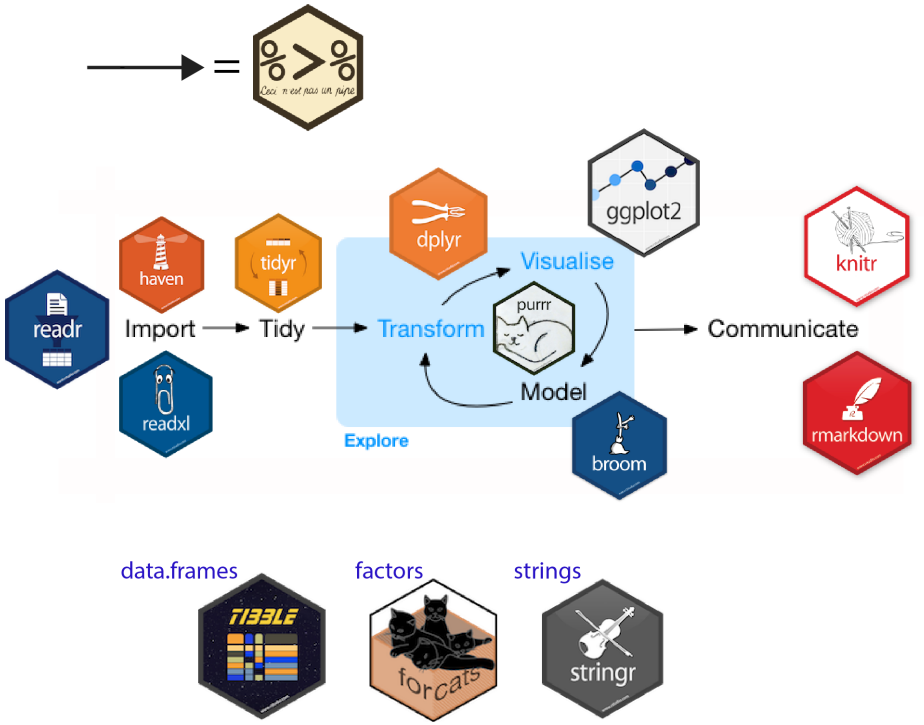
- So far we have only worked in the **console**
- Once we have run those commands, they are lost forever :(
- Scripts are simply text (.R) files for storing code
- A script should be **standalone** – performing a task from start to finish with no errors on a fresh startup!
- Open a new script and save it as variables.R in your lecture1 directory
- Add **comments** by beginning a line with #
- Task: write some variable assignments in variables.R and click **Source.** Check there are no errors!
- To be 100% sure that a script runs standalone, restart R (clears all saved variables) and then Source

Cases

- I recommend using **snake_case** for naming files and variables.
- You **may** want to use a different case like **PascalCase** for directory names
- **Whatever you do: AVOID SPACES!**

The Tidyverse

- Package: a collection of functions
- Tidyverse: a collection of packages
- Most (but not all) of what you need for data wrangling and visualisation is provided by tidyverse
- Install tidyverse
- Load up tidyverse
- Cheat sheets: https://www.rstudio.com/resources/cheatsheets/



Use install.packages(x) to install package x (remember x is a string!)

Use library(x) to load package x (this time x is NOT a string!)

Part 3: A Simple Data Science Project

Our World In Data, Data

- We will now begin a simple data science project that will teach you the basics of reading and wrangling data
- Create new folder ‘data’ inside BENV0091/lecture1
- Download owid_co2.csv from Moodle and put it in BENV0091/lecture1/data
- Create a new script analysis.R and save it in BENV0091/lecture1
- Load the tidyverse at the beginning of your script

Use library(tidyverse) to load tidyverse

Reading CSV Data

- We can import data into a **data frame** (aka **tibble**), a special type of **object** used in R to store data
- Read the OWID CO2 data CSV file and store it in an object called `co2` using <- **(write in your script!)**
- Show the first few rows of the data frame
- Each column is a **variable**
- Each row is an **observation**

Use read_csv(x) to load CSV with name x into a data frame

Type head(df) or glimpse(df) to get a concise look at the data

skim(df) is also a very useful function from the skimr package

Reading Files: Common Fixes!

- Make sure your working directory is set correctly
- Make sure the filename is a string (has quotes)

Indexing a Data Frame

- We can retrieve a value at row i and column j and even change it
- Try changing the value of position [3,2] to 0. Try changing it to “hello”. *Now change it back!*
- Can also index by column name: e.g. co2[10, ‘country’]
- **However avoid numeric indexing wherever possible!**
- You can also retrieve an entire column by name with the $ sign (returns a 1D **vector**)

Return value at row i, column j with df[i,j]

Use df[i,] to return all of row i; use df[,j] to return all of column j

Use df$var to get the entire column with name var

Manipulating data: some dplyr functions

- Too many to mention! But here are a few:
  - rename(df, new_name = old_name)
  - select(df, column1, column2,…)
  - mutate(df, new_variable = …)
  - transmute(df, new_variable = …)
  - filter(df, condition)
  - arrange(df, var)
  - count(df, var)
  - sample_n(df, n)
- All these functions **return** a dataframe - you must assign it to something with <- if you want to keep it
- Try the exercises on the right
- Reminder: https://www.rstudio.com/resources/cheatsheets/

EXERCISES 01

1. Rename co2 to Mtco2 (megatonnes of CO2)
2. Create a new column: Gtco2 (gigatonnes of CO2)
3. Arrange the df in descending order by Gtco2
4. Remove the “World” data
5. Filter to 2019 only and only countries with > 1 GtCO2
6. How many observations are left?

Joining Data

- We often need to combine multiple datasets
- To do this, we need at least one variable to match across the two datasets
- In general you should specify what variable(s) you want to join on with `by = `

left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matched = "na"**)** Join matching values from y to x.

right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na"**)** Join matching values from x to y.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na"**)** Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), …, keep = FALSE, na_matches = "na"**)** Join data. Retain all values, all rows.

Joining CO2 and Energy data

- Open a fresh script and save it as join.R
- Download the owid_energy.csv file and put it in your data directory
- **Writing in your script:** load the energy data to a new data frame called `energy`
- **Writing in your script:** load the CO2 data as before, in a data frame called `co2`
- Try combining energy and co2 with:
  - left_join()
  - right_join()
  - inner_join()
  - full_join()
- Try varying the order of co2 and energy
- Notice how many rows your joined dataframe has in each case
- **Writing in your script:** combine co2 and energy with left_join(co2, energy) and assign it to `df`

We can specify a list by using:
c(1, TRUE, 2, 'hello)

left_join(df1, df2, by = c(var1, var2))
joins together df1 and df2 by matching columns var1 and var2

NAs

- NAs represent missing data points
- How you deal with NAs is often a very important decision (more later in the course)
- For now, it's good to know how to identify where NAs are and how to remove them
- Count how many NAs there are in each column of your combined data frame
- Now remove all rows with NAs

is.na(x) checks whether x (or the values in x) are NA

is.na(df) can be combined with colSums(df) to count NAs

Use drop_na(df) to remove all rows with NA

Use drop_na(df, var) to remove all rows with NA in column var

The skimr package has skim() for counting NAs (and some other useful functions)

Pipes

- You can string together multiple functions with the pipe operator
- The output of the first function is passed as the **first argument** to the second function (and so on)
- Can make code much more concise and readable
- Try piping df into a sequence of any two functions, e.g.:
  - left_join() and count()
  - filter() and sample_n()
  - drop_na() and mutate()

f(x) %>% g() passes the output of f(x) as the first argument to g()

x <- f(x) %>% g(y)
is the same as:
x <- g(f(x), y)

Exercises 02

Use tolower(x) to change x to lower case

Use write_csv(df, f) to save df to location f

1. Drop World from the data and remove NAs
2. Create new variables for CO2/E, E/GDP, CO2/capita and GDP/capita
3. Set all country names to lower case
4. Save your combined data frame to a new file in the data directory
5. Which country had the highest GDP/capita in 2000?
6. Which country had the largest per capita CO2 emissions in 1965?
7. What proportion of countries had a GDP per capita of under $1000 in 1990?
8. What was the percentage change in global CO2 emissions between 1965 and 2016?

R Markdown

- R Markdown files combine text and code
- You can **knit** (render) them in a number of neat formats including pdf and html
- Task: try opening a new RMarkdown file and make some changes
- A worksheet will be posted on Moodle as a .Rmd file for you to work through

Tips for Improving your coding

- Read R for Data Science!
- Ask your pals and read their code
- R4DS also posts data every Tuesday as part of the Tidy Tuesday project
- Practise!