

# ***Rotman***

# INTRO TO R

R Workshop

February 3, 2021 Prepared by Jay Cao / [TDMDAL](https://tdmdal.github.io)

Website: <https://tdmdal.github.io/r-intro-rbac-2021-winter/>



Rotman School of Management  
UNIVERSITY OF TORONTO

# What's R?



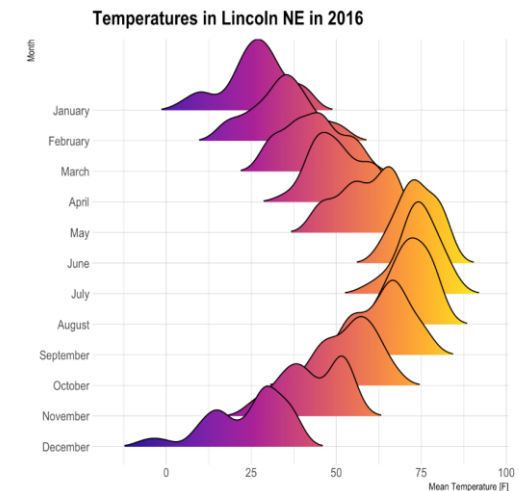
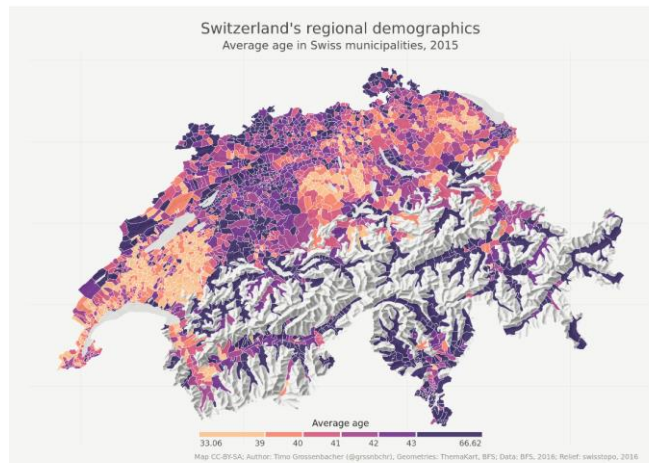
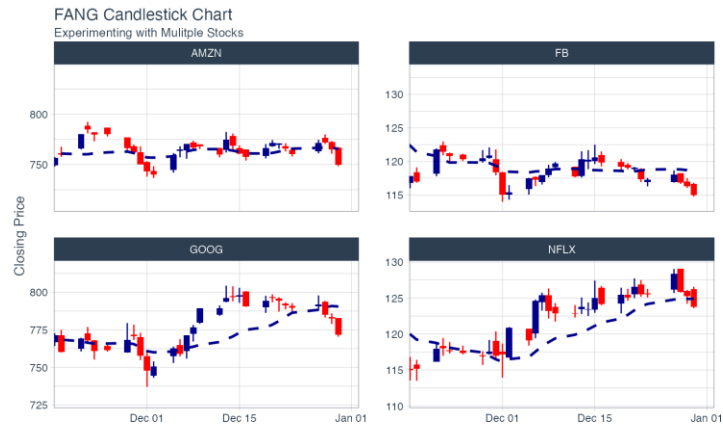
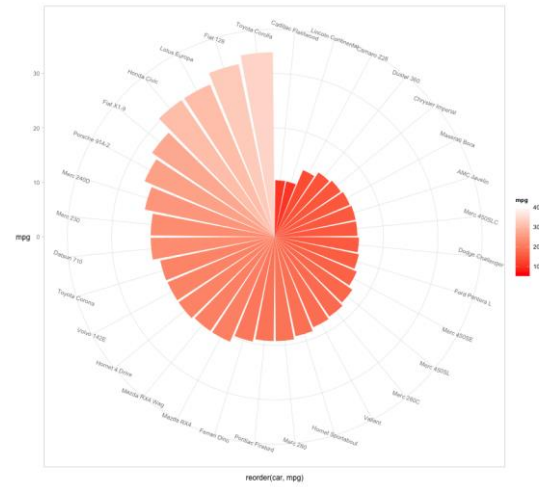
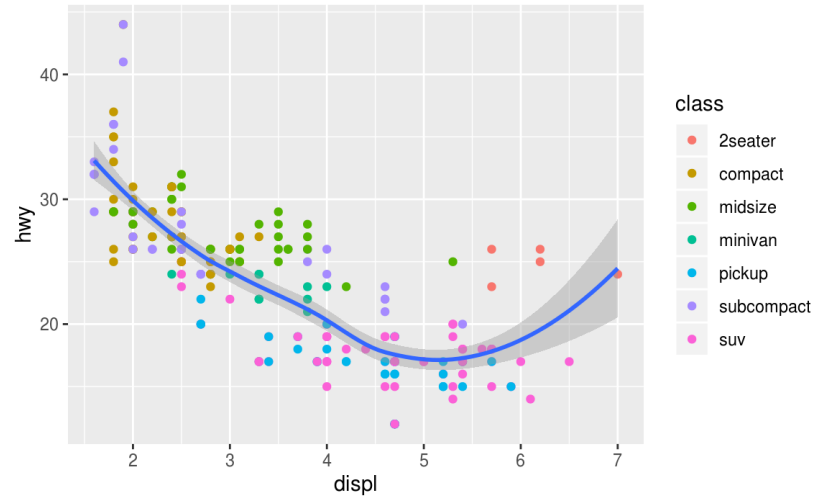
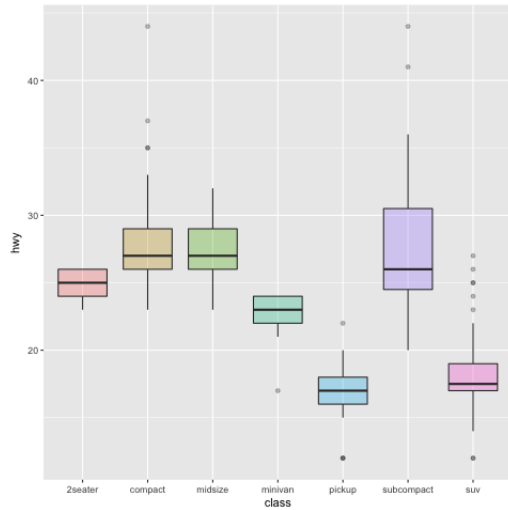
- R = a language + an eco-system
  - A free and open-source programming language
  - An eco-system of many high-quality user-contributed libraries/packages
- In the past R is mostly known for its statistical analysis toolkits
- Nowadays R is capable of (and very good at) many other tasks
  - Tools that cover the whole data analysis workflow
  - Tools for web technology...

# What can R do – Statistics & related

- Statistics & Econometrics
  - Regressions
  - Time series analysis
  - Bayesian inference
  - Survival analysis
  - ...
- Numerical Mathematics
  - Optimization
  - Solver
  - Differential equations
  - ...
- Finance
  - Portfolio management
  - Risk management
  - Option pricing
  - ...
- ...

See more R Finance and Risk Management Packages on [R Task View - Finance](#)

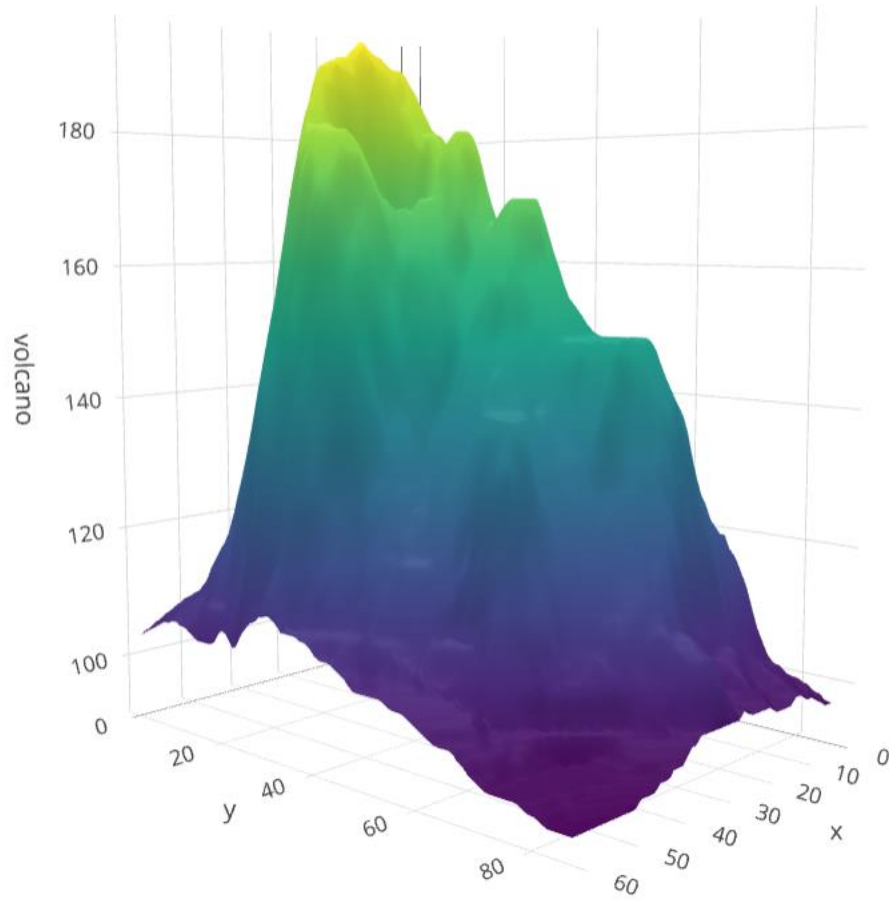
# What can R do – Graphics (static ones)



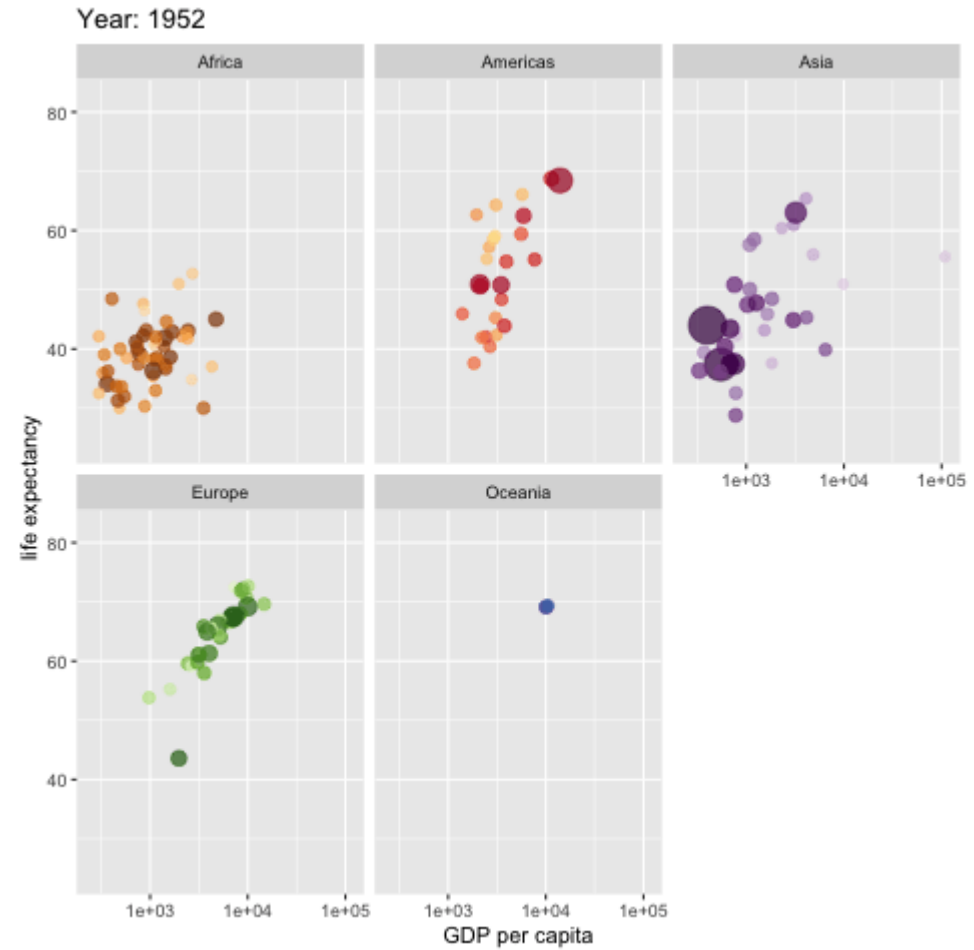
<https://www.r-graph-gallery.com/>

[https://timogrossenbacher.ch/2016/12/beautiful-thematic-maps-with-ggplot2-only/;](https://timogrossenbacher.ch/2016/12/beautiful-thematic-maps-with-ggplot2-only/)

# What can R do – Graphics (dynamic ones)



[https://plot.ly/r/3d-surface-plots/;](https://plot.ly/r/3d-surface-plots/)



<https://github.com/thomasp85/gganimate;>

# What can R do – Others

- Machine learning (e.g. interface to [Keras](#) and [Tensorflow](#))
- Natural language processing (e.g. [tidytext](#), [topicmodels](#))
- Web technology
  - Web scraping (e.g. [rvest](#))
  - API wrapper (e.g. Twitter: [rtweet](#); bigquery: [bigrquery](#); Quandl: [Quandl](#))
  - Shiny web app (<https://shiny.rstudio.com/>)
- Reporting
  - [R Markdown](#) (write reports, slides, blogs, books, etc. See a gallery [here](#).)
- ... (see [R Task View](#) for more)

# Why learn R (What can R do for You)?

- Beyond Excel Data Analysis
  - I wish Excel could...
- Automate boring repeating tasks
  - e.g., daily data collection from different sources, weekly dashboard update
- Prototype ideas
  - e.g., a novel trading strategy, a new credit risk model
- Really, find anything that interests you and use R...

# Plan for the 4 Sessions

- Overview
- Data Manipulation
- Graphs
- Time Series & Finance Applications



# Our Learning Approach – We Will Focus on

- Basics
  - e.g., R's basic data and programming structures
- Underlying principles
  - e.g., why organize data in a certain way (tidy data)
- Best practices
  - e.g., follow a consistent analysis workflow, and use R packages with consistent design, grammar and data structures

# Plan for Today

- Preparation and motivation examples
  - Setup R
  - Overview of data analysis workflow
  - Motivation examples
- Basics of R language
- Walk-through of a typical analysis workflow

# Setup R

- R on your computer
  - Install R (<https://www.r-project.org/>)
  - Install RStudio (<https://rstudio.com/products/rstudio/download/>)
- R in Cloud (run R without installation)
  - RStudio Cloud (<https://rstudio.cloud/>)
  - Google Colab (<https://colab.to/r>)

# What's RStudio?



The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for creating a graph. The code uses `tribble` to define a raw data table, processes it with `distinct` and `rename` to create `node_tb` and `edge_tb`, and then uses `create_graph` and `render_graph` to generate a graph.
- Environment Pane:** Shows the 'Global Environment' with a 'Data' section containing:

Object	Description
edge_tb	3 obs. of 2 variables
g	List of 12
node_tb	4 obs. of 1 variable
node_tb_tp	2 obs. of 1 variable
raw	4 obs. of 5 variables
- Viewer Pane:** Displays a directed graph with four nodes (1, 2, 3, 4) and three edges: 1 to 2, 2 to 3, and 2 to 4.
- Console:** Shows the execution of the code, including the `arrange` command and the `render_graph` command.

# RStudio Cloud

The screenshot displays the RStudio Cloud web interface. The browser address bar shows the URL `https://rstudio.cloud/spaces/112457/project/2046604`. The interface includes a sidebar on the left with navigation options like 'Spaces', 'Your Workspace', 'R Intro', 'New Space', 'Learn', 'Guide', 'What's New', 'Primers', 'Cheat Sheets', 'Help', 'Current System Status', 'RStudio Community', and 'Info'. The main workspace is titled 'R Intro / Workshop 1' and features a menu bar with 'File', 'Edit', 'Code', 'View', 'Plots', 'Session', 'Build', 'Debug', 'Profile', 'Tools', and 'Help'. Below the menu is a toolbar with icons for file operations and a 'Go to file/function' search bar. The code editor shows a single line of code: `1 |`. The console at the bottom displays the R startup message: 

```
/cloud/project/
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

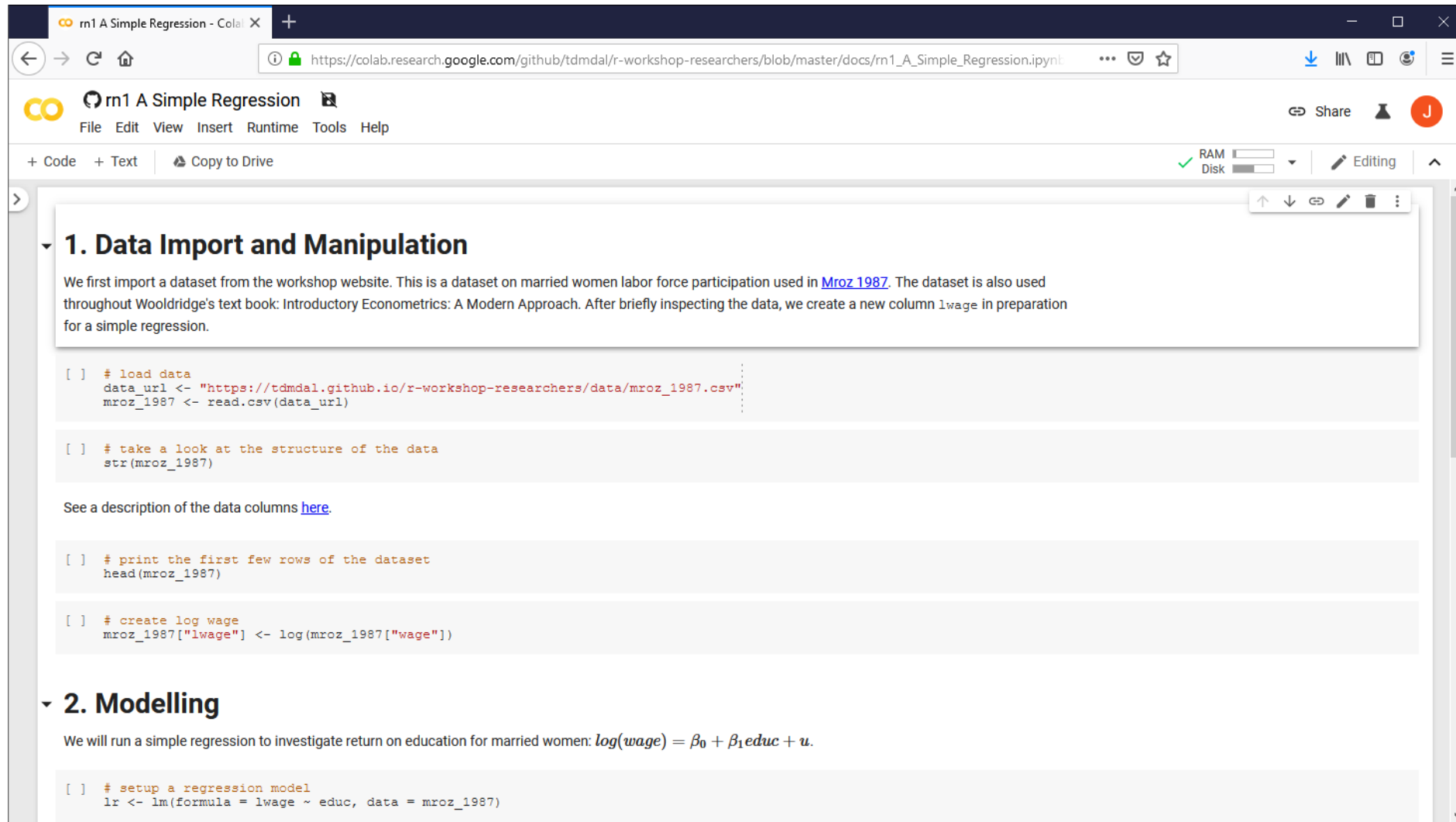
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

The right-hand side of the interface contains three panels: 'Environment' (showing 'Global Environment' and 'Environment is empty'), 'Files' (showing a file browser with a table of files), and 'Plots' (empty). The 'Files' panel shows a table with columns 'Name', 'Size', and 'Modified':

Name	Size	Modified
..		
.Rhistory	0 B	Dec 28, 2020, 4:52 PM
project.Rproj	205 B	Dec 28, 2020, 4:52 PM

# Google Colab



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `https://colab.research.google.com/github/tdmdal/r-workshop-researchers/blob/master/docs/rn1_A_Simple_Regression.ipynb`. The notebook title is "rn1 A Simple Regression". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with options for adding code and text, and a status bar showing RAM and Disk usage. The notebook content is as follows:

## 1. Data Import and Manipulation

We first import a dataset from the workshop website. This is a dataset on married women labor force participation used in [Mroz 1987](#). The dataset is also used throughout Wooldridge's text book: Introductory Econometrics: A Modern Approach. After briefly inspecting the data, we create a new column `lwage` in preparation for a simple regression.

```
[ ] # load data
data_url <- "https://tdmdal.github.io/r-workshop-researchers/data/mroz_1987.csv"
mroz_1987 <- read.csv(data_url)
```

[ ] # take a look at the structure of the data  
`str(mroz_1987)`

See a description of the data columns [here](#).

```
[ ] # print the first few rows of the dataset
head(mroz_1987)
```

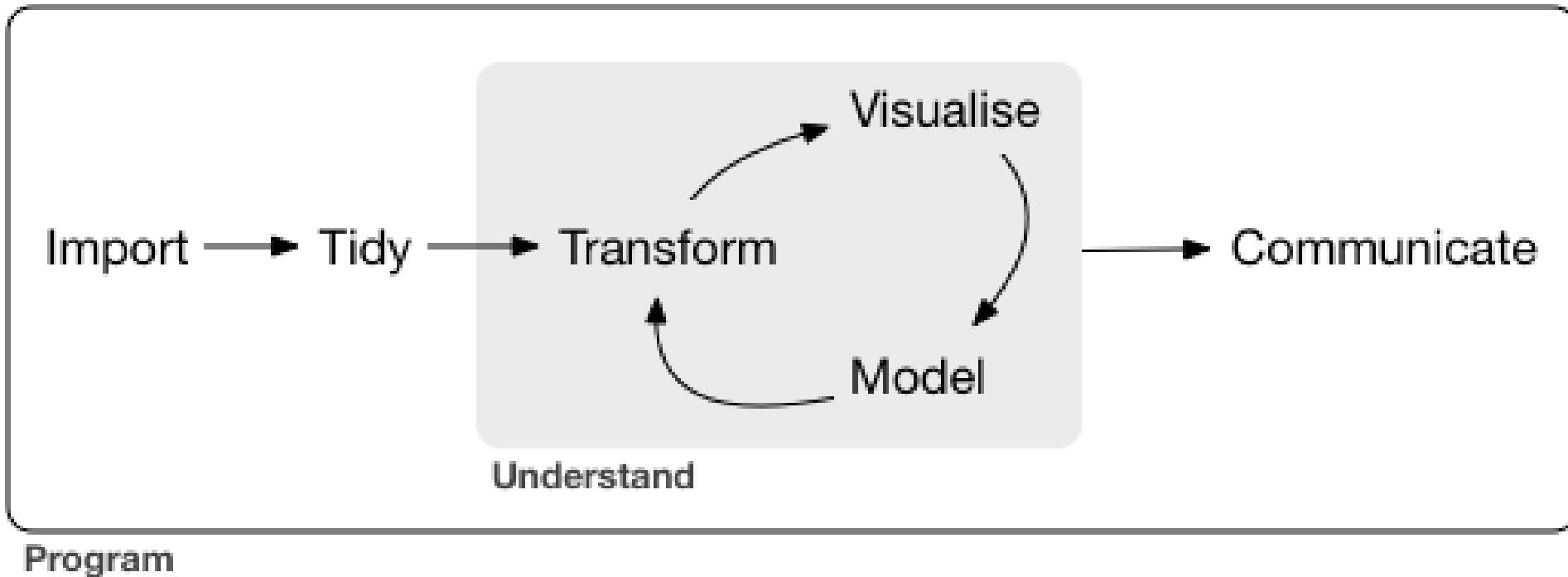
```
[ ] # create log wage
mroz_1987["lwage"] <- log(mroz_1987["wage"])
```

## 2. Modelling

We will run a simple regression to investigate return on education for married women:  $\log(wage) = \beta_0 + \beta_1 educ + u$ .

```
[ ] # setup a regression model
lr <- lm(formula = lwage ~ educ, data = mroz_1987)
```

# Data Analysis Workflow



# A Few Examples

- Housing prices and clean air - a simple regression analysis
- Analyze portfolio performance
- Look for trends in R community through Twitter
- Recognize handwritten digits - an example of deep learning



**PerformanceAnalytics  
Package**



**K Keras**

  
**TensorFlow**



# A Few Examples: What to Look For

- Focus on analysis workflow (by reading the code comments)
  - What data are imported; where are the data located
  - What data wrangling tasks are performance
  - What model is applied
  - What report and visualization are produced
- Don't focus on R syntax
- Do notice everything is done in a sequential way
  - no conditional branching or looping

# Plan for Today

- Intro to Intro
- Basics of R language
  - Expressions and assignment
  - Data structure
  - Programming structure
- Walk-through of a typical analysis workflow

# Expression and Assignment

```
# expression
```

```
2 + sqrt(4) + log(exp(2)) + 2^2
```

```
# assignment
```

```
x <- 3
```

```
y <- (pi == 3.14)
```

# R Data Structure - Overview

	Homogeneous	Heterogeneous
1-d	<b>Atomic vector</b>	<b>List</b>
2-d	Matrix	<b>Data frame</b>
n-d	Array	

# R Data Structure - Overview

	Homogeneous	Heterogeneous
1-d	<b>Atomic vector</b> →	<b>List</b>
2-d	Matrix	↓ <b>Data frame</b>
n-d	Array	

# Atomic Vectors

```
# create R vectors
```

```
vec_character <- c("Hello,", "World!")
```

<b>Hello,</b>	<b>World!</b>
---------------	---------------

```
vec_integer <- c(1L, 2L, 3L)
```

<b>1</b>	<b>2</b>	<b>3</b>
----------	----------	----------

```
vec_double <- c(1.1, 2.2, 3.3)
```

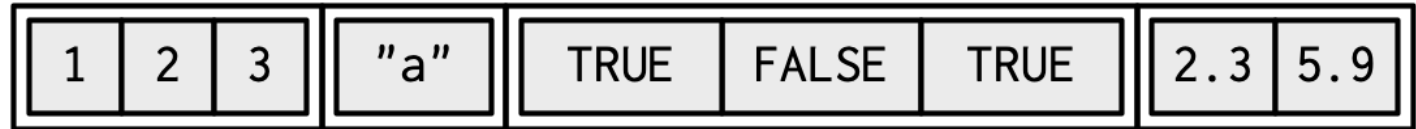
<b>1.1</b>	<b>2.2</b>	<b>3.3</b>
------------	------------	------------

```
vec_logical <- c(TRUE, TRUE, FALSE)
```

<b>TRUE</b>	<b>TRUE</b>	<b>FALSE</b>
-------------	-------------	--------------

# List

```
# create an R list
l1 <- list(
  1:3,
  "a",
  c(TRUE, FALSE, TRUE),
  c(2.3, 5.9)
)
```



# Data Frame

```
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

<b>x</b>	<b>y</b>	<b>z</b>
1	"a"	1.1
2	"b"	2.2
3	"c"	3.3



# Data Frame

```
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

x	y	z
1	"a"	1.1
2	"b"	2.2
3	"c"	3.3

# Data Frame

```
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

x	y	z
1	"a"	1.1
2	"b"	2.2
3	"c"	3.3

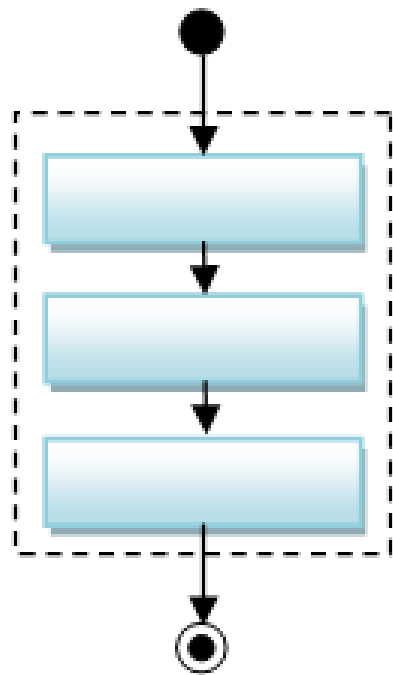
# Tibble – A Cousin to Data Frame

```
# load tibble library (part of tidyverse lib)
library(tibble)

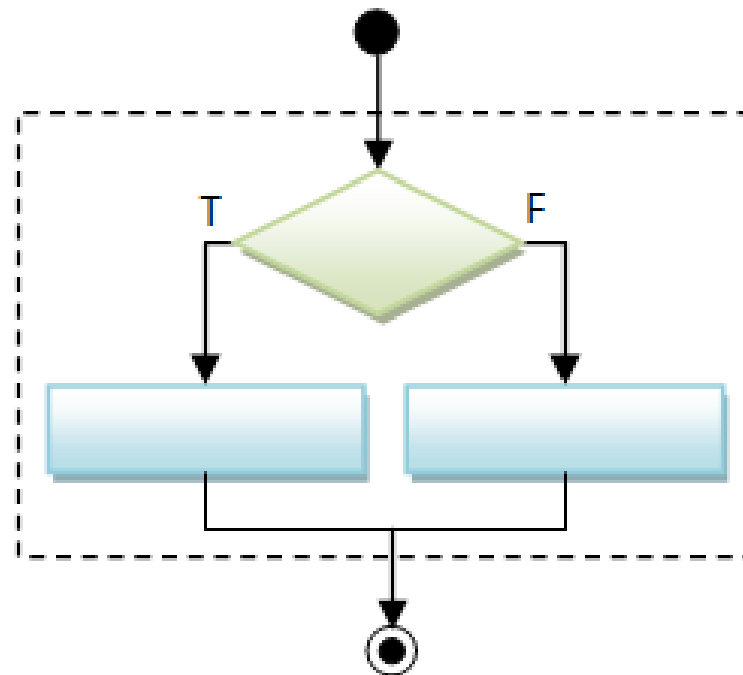
# create a tibble
tb1 <- tibble(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

x	y	z
1	"a"	1.1
2	"b"	2.2
3	"c"	3.3

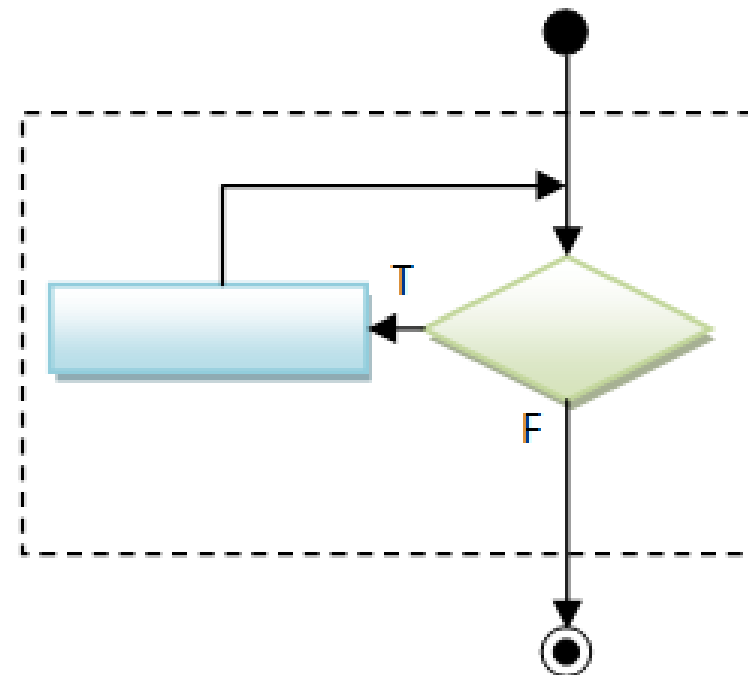
# Programming Structure: Control Flows



**Sequential**



**Conditional (Decision)**



**Loop (Iteration)**

# Sequential

- Example: Sum of Squares

$$\sum_{t=1}^3 t^2$$

```
# sum of squares  
t <- 1:3  
y <- sum(t^2)  
print(y)
```

# Sequential

- Example: Sum of Squares

$$\sum_{t=1}^3 t^2$$

```
# sum of squares  
t <- 1:3  
y <- sum(t^2)  
print(y)
```

t	1	2	3
---	---	---	---

# Sequential

- Example: Sum of Squares

$$\sum_{t=1}^3 t^2$$

```
# sum of squares  
t <- 1:3  
y <- sum(t^2)  
print(y)
```

t	1	2	3
t^2	1	4	9
sum(t^2)	14		

# Conditional (if...else...)

```
if (cond) {  
    # run here if cond is TRUE  
} else {  
    # run here if cond is FALSE  
}
```

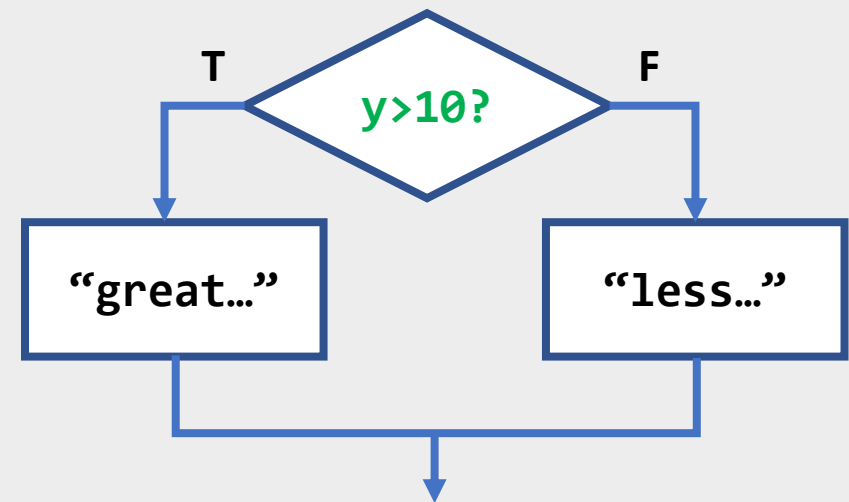
```
# y greater than 10?  
if (y > 10) {  
    print("greater than 10")  
} else {  
    print("less or equal to 10")  
}
```



# Conditional (if...else...)

```
if (cond) {  
    # run here if cond is TRUE  
} else {  
    # run here if cond is FALSE  
}
```

```
# y greater than 10?  
if (y > 10) {  
    print("greater than 10")  
} else {  
    print("less or equal to 10")  
}
```



# Conditional (if...else if...else...)

```
if (cond1) {  
    # run here if cond1 is TRUE  
} else if (cond2) {  
    # run here if cond1 is FALSE but cond2 is TRUE  
} else {  
    # run here if neither cond1 nor cond2 is TRUE  
}
```

# Iteration

```
for (var in seq) {  
  do something  
}
```

```
while (cond) {  
  do something if cond is TRUE  
}
```

```
# sum of squares  
t <- 1:3  
y <- 0  
  
for (x in t) {  
  y <- y + x^2  
}  
  
print(y)
```

# Programming Structure: Functions

- What's a function
  - a logical block of code
  - input -> output
- Why write functions
  - Reusability
  - Abstraction
  - Maintainability
- Example:  $\sum_{t=1}^n t^2$

```
# sum of squares from 1 to n
ss <- function(n) {
  t <- 1:n
  sum(t^2)
}

# calling the ss() function
print(ss(2))
print(ss(3))
```

# Programming Structure: Functions

- What's a function
  - a logical block of code
  - input -> output
- Why write functions
  - Reusability
  - Abstraction
  - Maintainability
- Example:  $\sum_{t=1}^n t^2$

```
# sum of squares from 1 to n
ss <- function(n) {
  t <- 1:n
  sum(t^2)
}

# calling the ss() function
print(ss(2))
print(ss(3))
```

# Programming Structure: Functions

- What's a function
  - a logical block of code
  - input -> output
- Why write functions
  - Reusability
  - Abstraction
  - Maintainability
- Example:  $\sum_{t=1}^n t^2$

```
# sum of squares from 1 to n
ss <- function(n) {
  t <- 1:n
  sum(t^2) # return(sum(t^2))
}

# calling the ss() function
print(ss(2))
print(ss(3))
```

# Plan for Today

- Intro to Intro
- Basics of R language
- **Walk-through of a typical analysis workflow**
  - Import and manipulate data
  - Build models
  - Report results

# Extending the regression example

- Manipulate data
  - Load data
  - Create new columns
  - Filter columns and rows
- Build models
  - Multiple linear regression
  - Regression with interactive terms
- Report and graph
  - Build a publication-ready table for regression results



# Using R libraries

- Install and load an R library

```
install.packages("Library_name")
```

```
library(Library_name)
```

- [CRAN](#) (The Comprehensive R Archive Network)
  - [CRAN Task Views](#)

# Many choices, which one to use

- Often time, many choices of functions/libraries to do one task
  - R is open and extensible!
- Example: load a csv file to a data frame
  - Use [read.csv\(\)](#) function from the `utils` library
  - Use [read\\_csv\(\)](#) function from the [readr](#) library
  - Use [fread\(\)](#) function from the [data.table](#) library
  - Use [vroom\(\)](#) from the [vroom](#) library

# Many choices, which one to use

- Start with the one most people use
- Choose one that is well maintained
  - check document, github, etc. for last update
- Choose one that suits your task

# Our Choice: extending the regression example

- Manipulate data ([tidyverse](#) eco-system)
  - Load data ([read\\_csv\(\)](#) from the [readr](#))
  - Create new columns ([mutate\(\)](#) from [dplyr](#))
  - Filter columns and rows ([select\(\)](#) and [filter\(\)](#) from [dplyr](#))
- Build models
  - Multiple regression ([lm\(\)](#) from stats library in R base)
- Report and graph
  - Build a publication-ready table ([huxreg\(\)](#) from [huxtable](#) library)

# Load a CSV file

- [read\\_csv\(\)](#) from the [readr](#)

```
read_csv(file)
```

```
e.g. hprice <- read_csv("hprice.csv")
```

- More about [read\\_csv\(\)](#)
- More about [readr](#)

# Load Data – Other file formats and sources

- [readxl](#) for Excel sheets
- [haven](#) for SPSS, Stata and SAS data
- [jsonlite](#) for JSON
- [xml2](#) for XML
- [httr](#) for web APIs
- [rvest](#) for web scraping
- [DBI](#) for connecting to DataBase engine
- ...

# Load Data – Financial Dataset

- [tq\\_get\(\)](#) from tidyquant library
  - collect financial and economic data from many online sources
    - Yahoo Finance, FRED, Quandl, Tiingo, Alpha Vantage, Bloomberg
- [simfinR](#) or [simfinapi](#) library
  - download financial statements – balance sheet, cash flow and income statement – and adjusted daily price of stocks through [the simfin project](#)
- a few others (try to look for them yourselves...)

# Data Manipulation: dplyr basics

- Filter observations: filter()
- Select variables: select()
- Reorder rows: `arrange()`
- Create new variables: mutate()
- Collapse column values to a single summary: `summarise()`
  
- Group by: `group_by()`



## Data Manipulation: filter()

```
filter(my_dataframe, condition1, ...)
```

e.g.

```
hprice_reg <- filter(hprice, price > 20000)
```

## Data Manipulation: mutate()

```
mutate(my_dataframe, new_var1 = expression1, ...)
```

e.g.

```
hprice_reg <- mutate(hprice_reg, lprice = log(price))
```

# Data Manipulation: select()

```
select(my_dataframe, var1, ...)
```

e.g.

```
hprice_reg <- select(hprice_reg, lprice, rooms)
```

## Data Manipulation: Data Pipe (%>%)

```
hprice_reg <- filter(hprice, price > 20000)
```

```
hprice_reg <- mutate(hprice_reg, lprice = log(price))
```

```
hprice_reg <- select(hprice_reg, lprice, rooms)
```

## Data Manipulation: Data Pipe (%>%)

```
hprice_reg <- filter(hprice, price > 20000)
```

```
hprice_reg <- mutate(hprice_reg, lprice = log(price))
```

```
hprice_reg <- select(hprice_reg, lprice, rooms)
```

```
hprice_reg <- hprice %>%
```

```
  filter(., price > 20000) %>%
```

```
  mutate(., lprice = log(price)) %>%
```

```
  select(., lprice, rooms)
```

## Data Manipulation: Data Pipe (%>%)

```
hprice_reg <- filter(hprice, price > 20000)
```

```
hprice_reg <- mutate(hprice_reg, lprice = log(price))
```

```
hprice_reg <- select(hprice_reg, lprice, rooms)
```

```
hprice_reg <- hprice %>%
```

```
  filter(price > 20000) %>%
```

```
  mutate(lprice = log(price)) %>%
```

```
  select(lprice, rooms)
```

# Data Manipulation: Others

- Join two data frames
  - [join\(\)](#) family in dplyr
- Reshape data frames
  - [pivot longer\(\)](#) and [pivot wider\(\)](#) in tidyr

# Regression

- Multiple regressions: [lm\(\)](#) from stats library in base R

```
my_model <- lm(y ~ x1 + x2, data)
```

- Multiple regressions with interactive terms

```
my_model <- lm(y ~ x1 + x2 + I(x1 * x2), data)
```

- Regression result summary: `summary()`



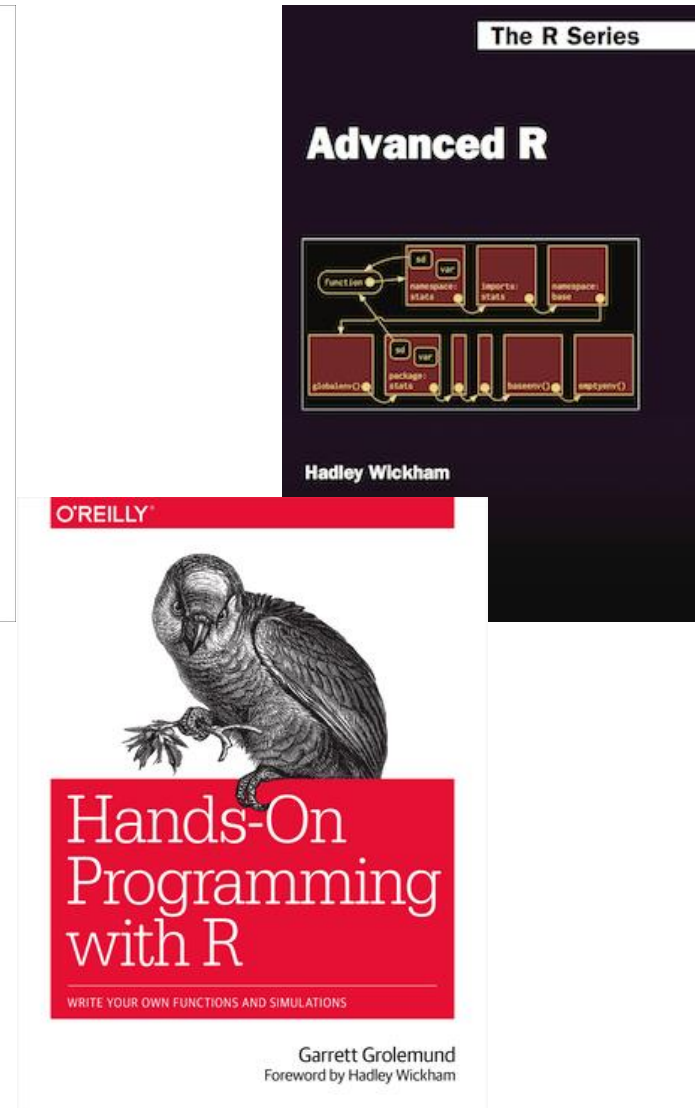
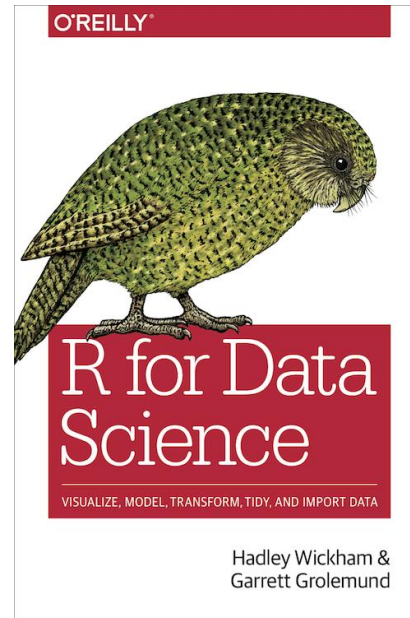
# Report

- Summary table
  - [Summary for lm\(\)](#): `summary(my_model)`
- publication-ready table: [huxreg\(\)](#) from [huxtable](#) library

```
huxtable(my_model1, my_model2, ...)
```

# Free Learning Resources - Books

- [R for Data Science](#)
- [Advanced R](#)
- [Hands-On Programming with R](#)
- Check [bookdown.org](#) often



# Free Learning Resources – Video Courses

- [RStudio Resources Site](#)
- Coursera
  - free for [UofT students](#) (mostly always free if you just audit the courses)
  - Search R and learn

# Free Learning Resources – Others

- [RStudio Education](#) ([Choose Your Learning Paths](#))
- [CRAN Task View](#)
- Twitter (a few seeds: [#rstat](#), [@hadleywickham](#), [@WeAreRLadies](#))