Rotman

INTRO TO R

R Workshop

January 27, 2020   Prepared by Jay Cao / TDMDAL

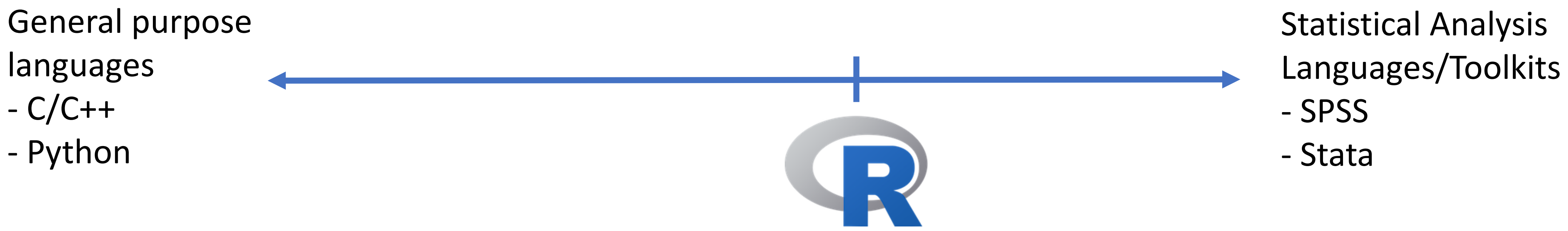
Website: https://tdmdal.github.io/r-tutorial-201920-winter/

Rotman School of Management
UNIVERSITY OF TORONTO

What’s R?

- A programming language
  - Free and open source
  - Extensible with many high-quality user-contributed libraries/packages

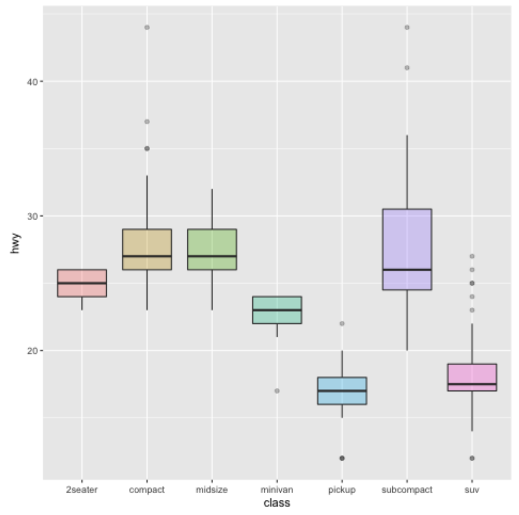
- Great for statistical analysis, graphics and many other things (ex?)

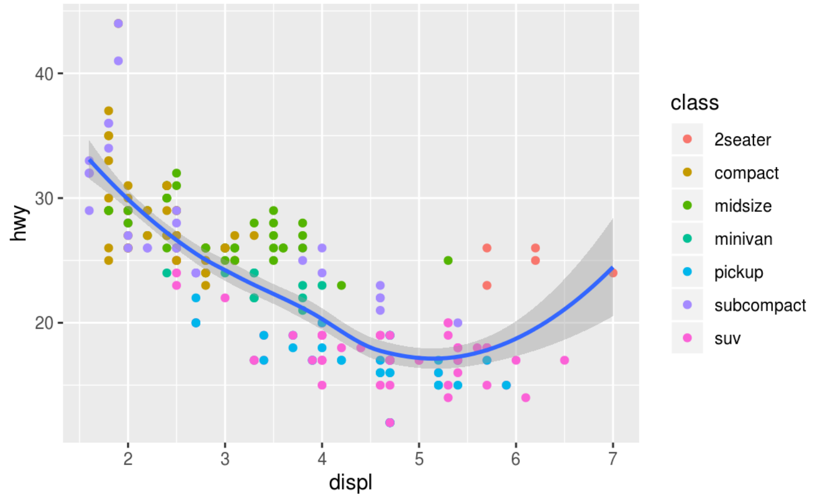

General purpose languages
- C/C++
- Python
Statistical Analysis Languages/Toolkits
- SPSS
- Stata

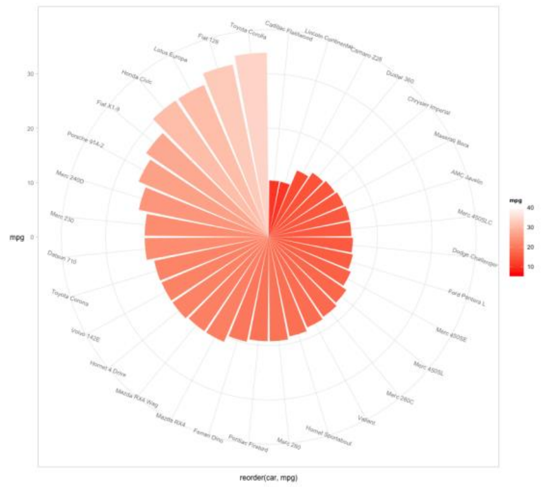
What can R do – Statistics & related

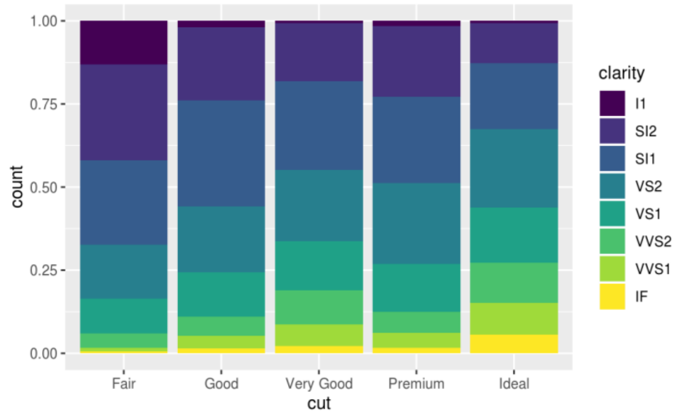
- Statistics & Econometrics
  - Regressions
  - Time series analysis
  - Bayesian inference
  - Survival analysis
  - …
- Numerical Mathematics
  - Optimization
  - Solver
  - Differential equations
  - …
- Finance
  - Portfolio management
  - Risk management
  - Option pricing
  - …
- …

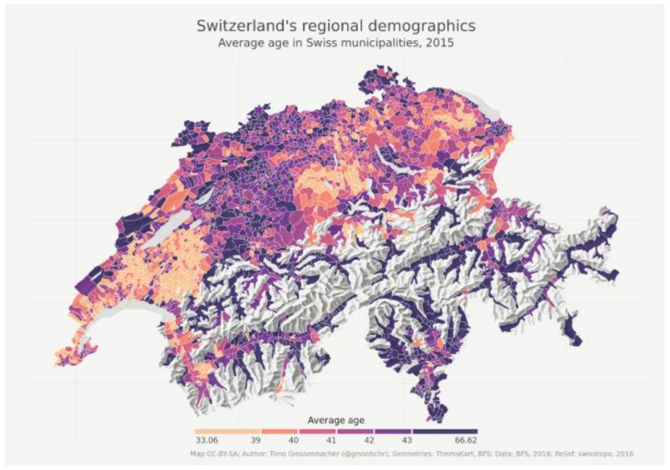
What can R do – Graphics (static ones)

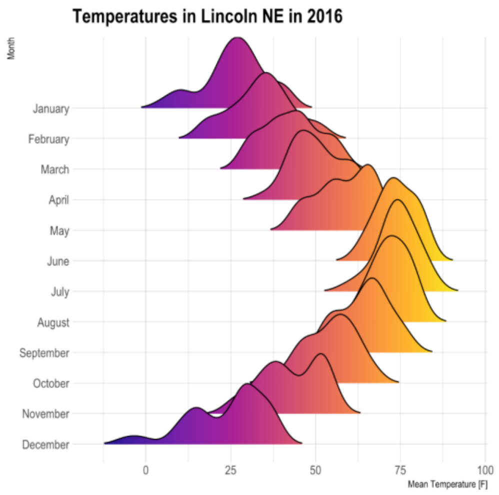








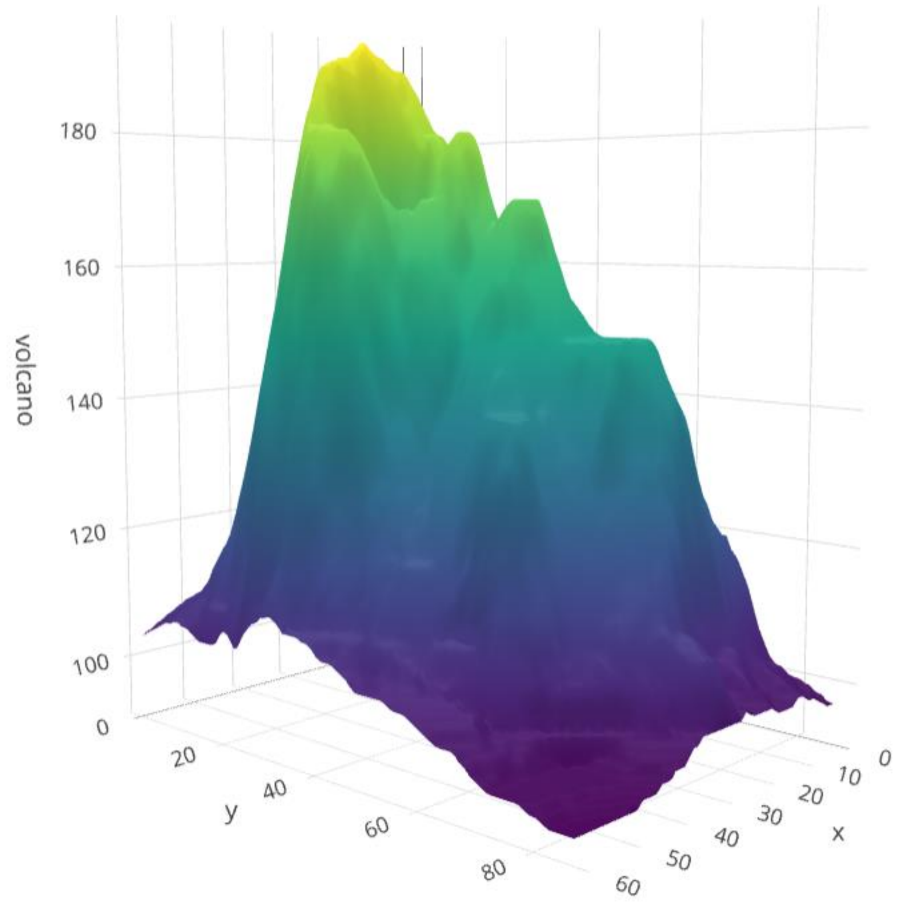

Switzerland's regional demographics
Average age in Swiss municipalities, 2015
Average age

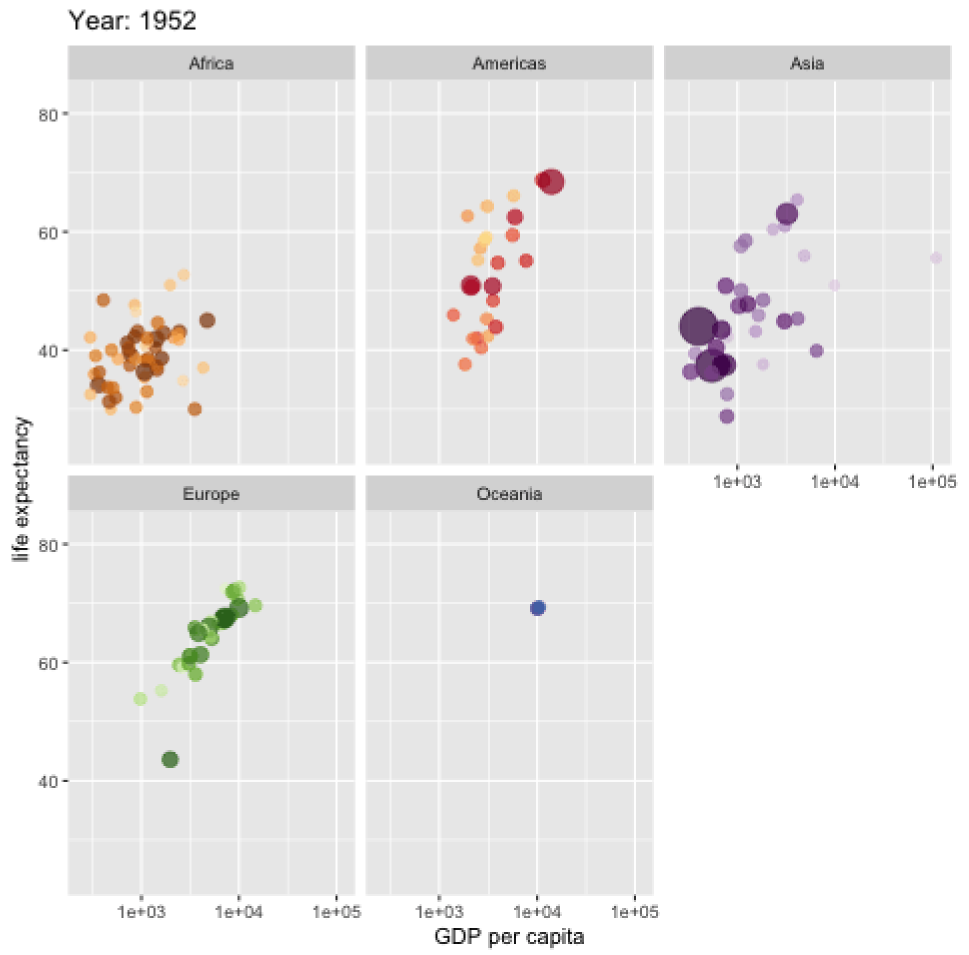


...

https://www.r-graph-gallery.com/
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://github.com/thomasp85/gganimate;

What can R do – Others

- Machine learning (ex. R interface to Keras: keras)
- Natural language processing (ex. tidytext, topicmodels)
- Web technology
  - Web scraping (ex. rvest)
  - API wrapper (ex. 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)

What can R do, **for You**?

- Beyond Excel Data Analysis
- Automate boring tasks
- Prototype ideas
- …

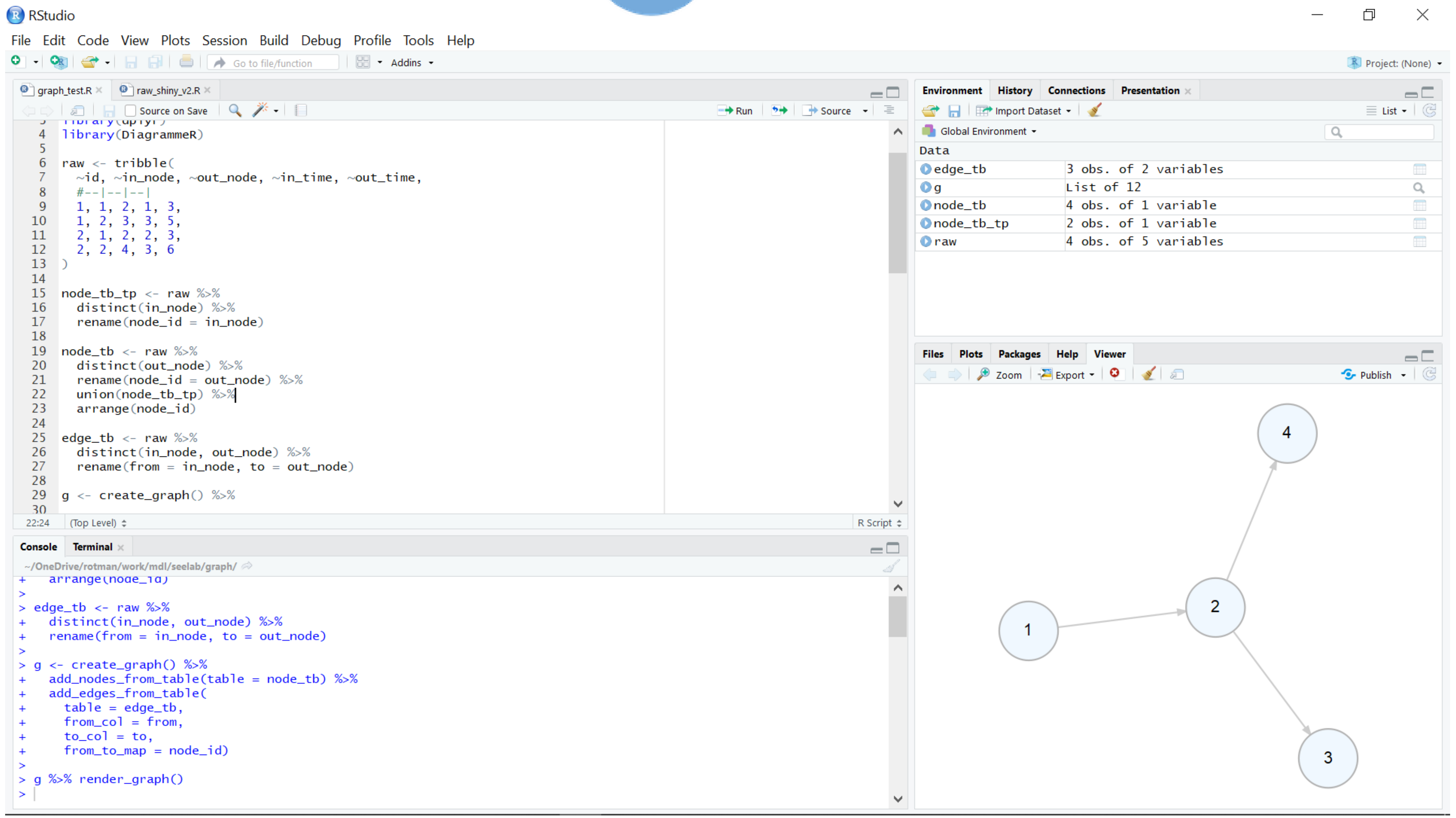
Plan for the 4 Sessions

- Overview
- Data Manipulation
- Graphs
- Time Series (Finance Applications)

Plan for Today (~2 hrs)

- Motivation: three examples
  - A simple regression (housing price and pollution)
  - Twitter API
  - Deep learning “Hello World!”
- Basics of R
  - Data structure
  - Programming structure
- A typical analysis workflow: extending the regression example
  - Import and manipulate data
  - Build models
  - Report results

What's RStudio?



```r
library(dplyr)
library(DiagrammeR)

raw <- tribble(
  ~id, ~in_node, ~out_node, ~in_time, ~out_time,
  #--|--|--|
  1, 1, 2, 1, 3,
  1, 2, 3, 3, 5,
  2, 1, 2, 2, 3,
  2, 2, 4, 3, 6
)

node_tb_tp <- raw %>%
  distinct(in_node) %>%
  rename(node_id = in_node)

node_tb <- raw %>%
  distinct(out_node) %>%
  rename(node_id = out_node) %>%
  union(node_tb_tp) %>%
  arrange(node_id)

edge_tb <- raw %>%
  distinct(in_node, out_node) %>%
  rename(from = in_node, to = out_node)

g <- create_graph() %>%
```

22:24 (Top Level) R Script

Console Terminal

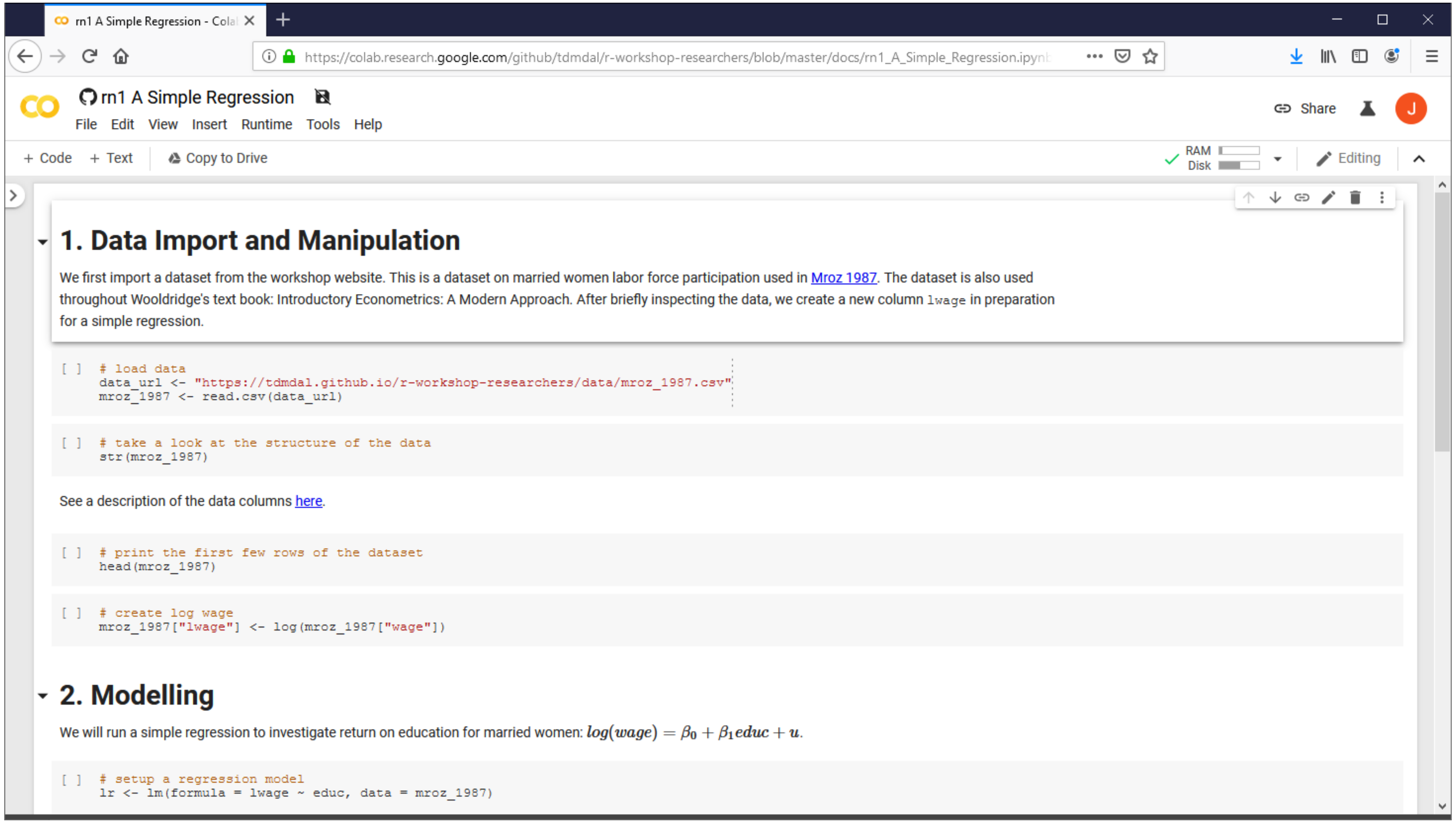
~/OneDrive/rotman/work/mdl/seelab/graph/

```
+   arrange(node_id)
>
> edge_tb <- raw %>%
+   distinct(in_node, out_node) %>%
+   rename(from = in_node, to = out_node)
>
> g <- create_graph() %>%
+   add_nodes_from_table(table = node_tb) %>%
+   add_edges_from_table(
+     table = edge_tb,
+     from_col = from,
+     to_col = to,
+     from_to_map = node_id)
>
> g %>% render_graph()
>
```

Data

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

Google Colab


rn1 A Simple Regression - Colab
https://colab.research.google.com/github/tdmdal/r-workshop-researchers/blob/master/docs/rn1_A_Simple_Regression.ipynb
rn1 A Simple Regression
File Edit View Insert Runtime Tools Help
Share
J
+ Code + Text
Copy to Drive
RAM
Disk
Editing

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"])
```

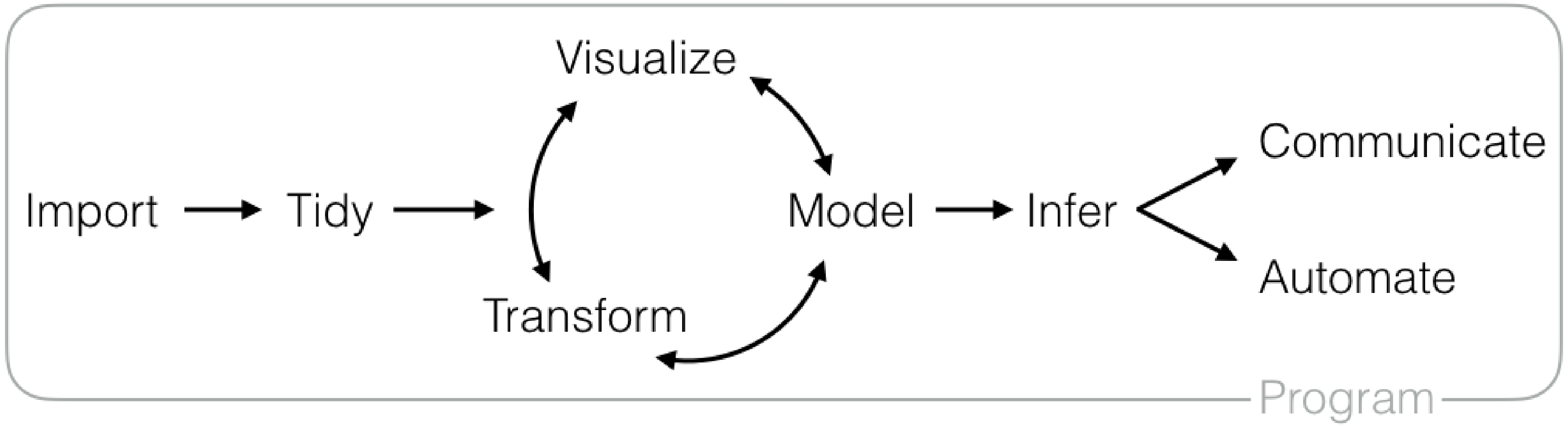
2. Modelling

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: Three Examples

- A simple regression
- Twitter API
- Deep learning “Hello World!”



https://github.com/rstudio/RStartHere

R Basics

- Data structures
- Programming structures

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	**Atomic vector**	**List**
2-d	Matrix	**Data frame**
n-d	Array	

http://adv-r.had.co.nz/Data-structures.html

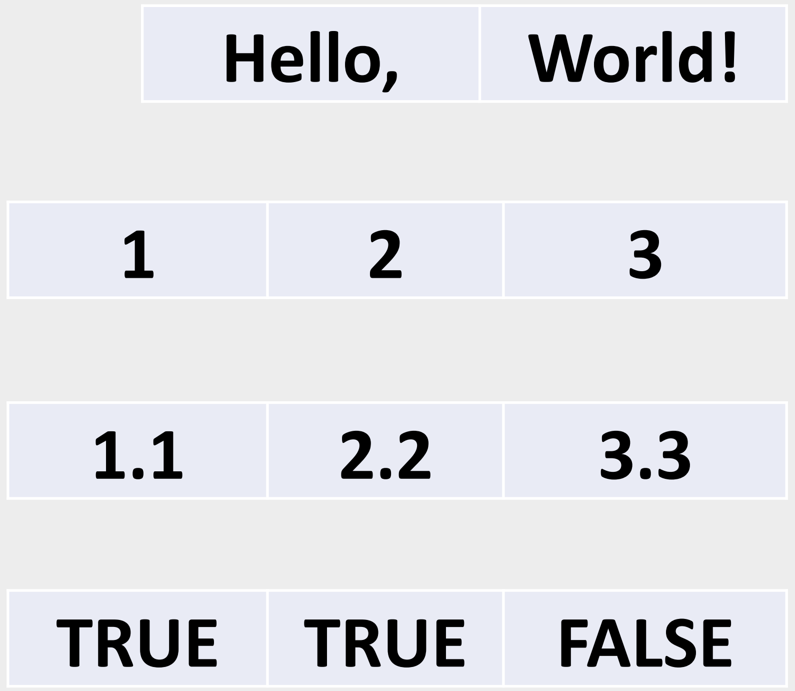
R Data Structure - Overview

	Homogeneous	Heterogeneous
1-d	**Atomic vector** →	**List** ↓
2-d	Matrix	**Data frame**
n-d	Array	

http://adv-r.had.co.nz/Data-structures.html

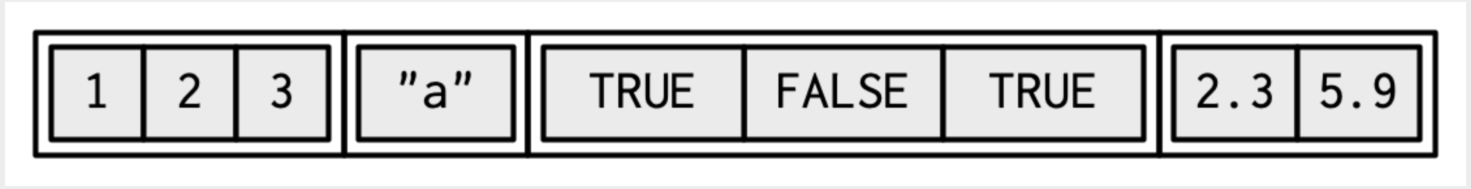
Atomic Vectors

```r
# create R vectors
vec_character <- c("Hello,", "World!")

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

vec_double <- c(1.1, 2.2, 3.3)

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



List

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



ref. https://adv-r.hadley.nz/vectors-chap.html#list-creating

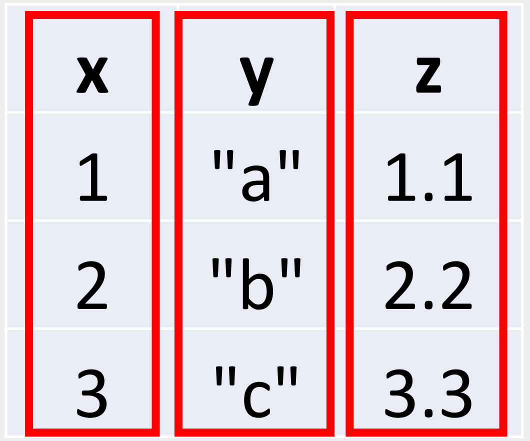
Data Frame

```r
# 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

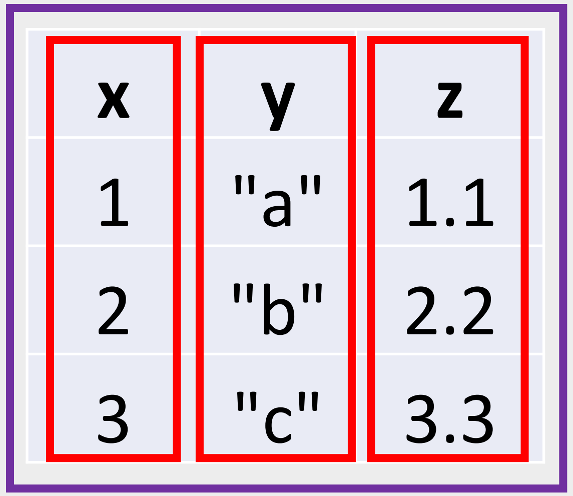
```
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```



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

Data Frame

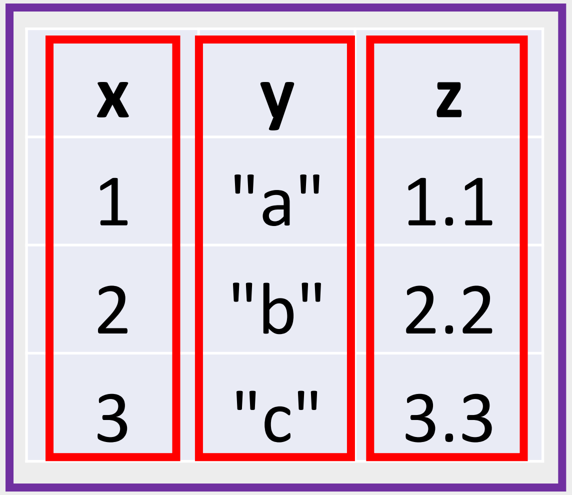
```
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```



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

Tibble – A Cousin to Data Frame

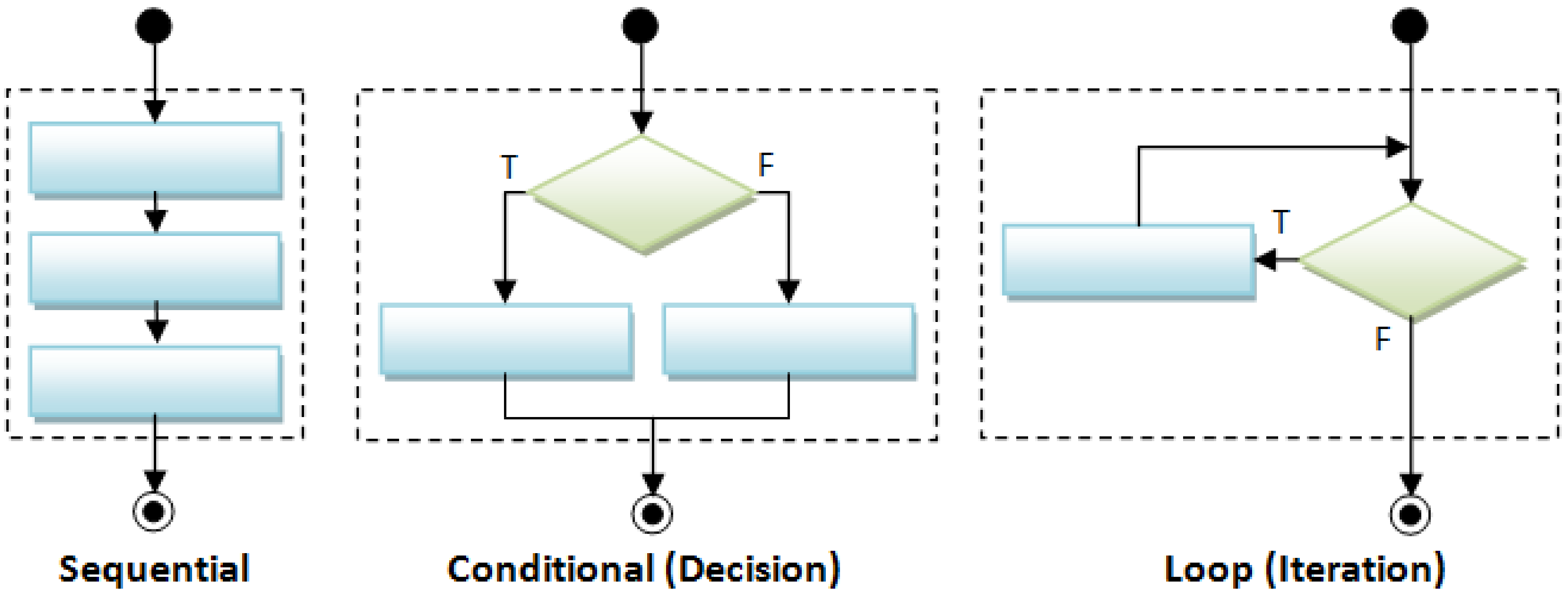
```
# load tibble library (part of tidyverse lib)
library(tibble)

# create a tibble
tb1 <- 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

https://r4ds.had.co.nz/tibbles.html#tibbles-vs.data.frame

Programming Structure: Control Flows



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…)

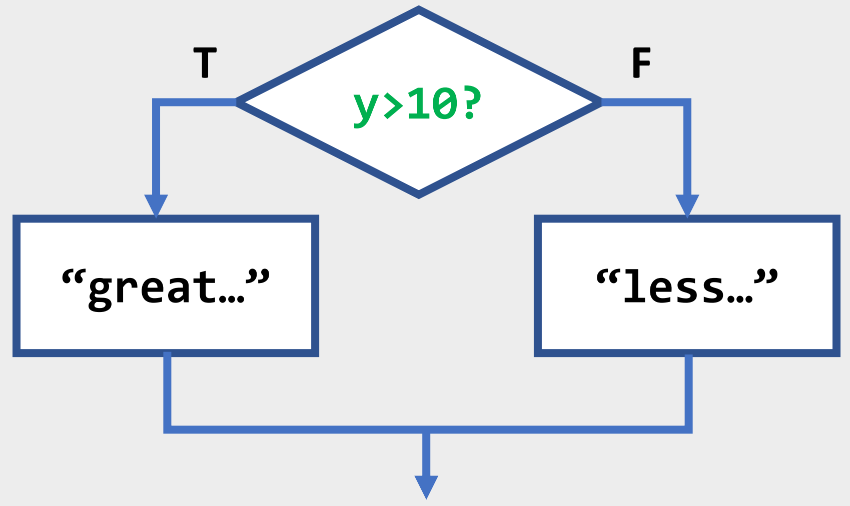
```
if (cond) {
  # run here if cond is TRUE
} else {
  # run here if cond is FALSE
}
```

```
# 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: $\sum_{t=1}^{n} t^2$

```r
# 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))
```

Extending the regression example

- Manipulate data
  - Load data
  - Create new columns
  - Filter columns and rows
- Build models
  - Multiple regression
  - IV regression
- 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 (`stargazer()` from `stargazer` library)

Load a CSV file

- `read_csv()` from the `readr`

```
read_csv(file)
```

- More about `read_csv()`

- More about `readr`

Load Data – Many other libraries

- 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
- …

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, …)
```

Data Manipulation: **mutate()**

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

Data Manipulation: **`select()`**

```
select(my_dataframe, var1, …)
```

Data Manipulation: Data Pipe (%>%)

```
iris_cleaned <- filter(iris, Species == "setosa")
iris_cleaned <- select(iris_cleaned, Sepal.Length)
```

Data Manipulation: Data Pipe (%>%)

```
iris_cleaned <- filter(iris, Species == "setosa")
iris_cleaned <- select(iris_cleaned, Sepal.Length)

iris_cleaned <- iris %>%
  filter(., Species == "setosa") %>%
  select(., Sepal.Length)
```

Data Manipulation: Data Pipe (%>%)

```
iris_cleaned <- filter(iris, Species == "setosa")
iris_cleaned <- select(iris_cleaned, Sepal.Length)

iris_cleaned <- iris %>%
  filter(Species == "setosa") %>%
  select(Sepal.Length)
```

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()`

Ref. https://faculty.chicagobooth.edu/richard.hahn/teaching/FormulaNotation.pdf

Report

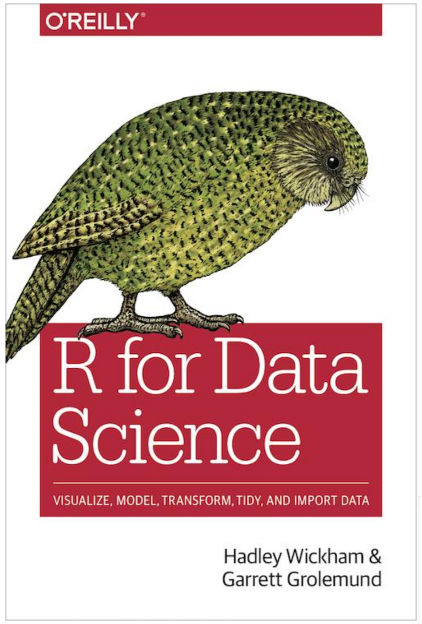
- Summary table
  - Summary for lm(): `summary(my_model)`
- publication-ready table: `stargazer()` from `stargazer` library

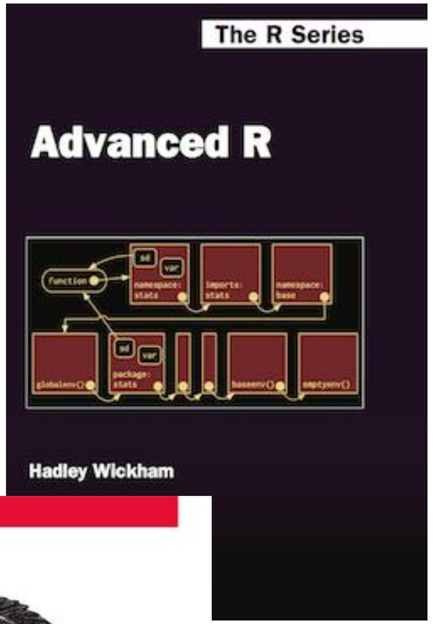
```
stargazer(my_model1, my_model2, …)
```

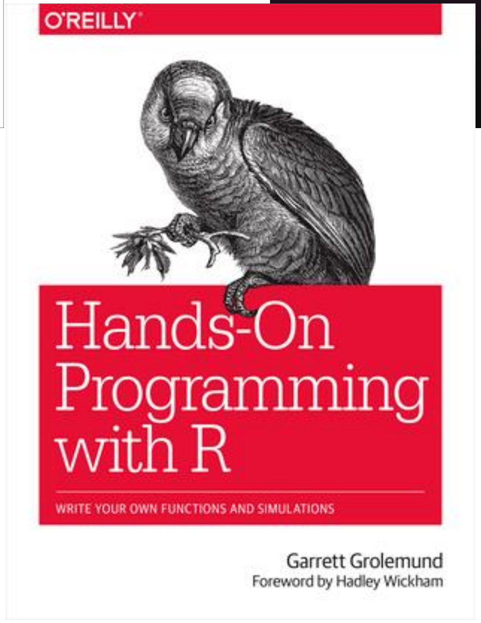
Ref. https://cran.r-project.org/web/packages/stargazer/vignettes/stargazer.pdf

Free Learning Resources - Books

- R for Data Science
- Advanced R
- Hands-On Programming with R
- Check bookdown.org often


O'REILLY
R for Data Science
VISUALIZE, MODEL, TRANSFORM, TIDY, AND IMPORT DATA
Hadley Wickham & Garrett Grolemund




O'REILLY
Hands-On Programming with R
WRITE YOUR OWN FUNCTIONS AND SIMULATIONS
Garrett Grolemund
Foreword by Hadley Wickham

Free Learning Resources – Video Courses

- RStudio Resources Site

- LinkedIn Learning (used to be lynda.com)
  - free for UofT students and Toronto Public Library users
  - Search R and learn

Free Learning Resources – Others

- CRAN Task View
- Sample notebooks / reports at http://rpubs.com/
- Twitter (a few seeds: #rstat, @hadleywickham, @WeAreRLadies)