

Rotman

INTRO TO R

R Workshop – Part 1 Overview & Basics

March 2, 2023 Prepared by Jay Cao / [TDMDAL](#)

Website: <https://tdmdal.github.io/r-workshop-202223-winter/>



Rotman School of Management
UNIVERSITY OF TORONTO

Plan for the 4-Session Workshops

- Part 1: Overview & Basics (Session 1, 2)
- Part 2: Data Manipulation (Session 2, 3)
- Part 3: Data Visualization (Session 3)
- Part 4 - 1: Modeling Workflow (Session 4)
- Part 4 - 2: Time Series & Some Finance Applications (Session 4)

Plan for Part 1

- Intro
 - What is R and what can R do?
 - Setup R
 - Motivation examples
- R programming and Data Science
 - Basics of R programming
 - Data science with R
- Learning Resources and Road Map

What's R?



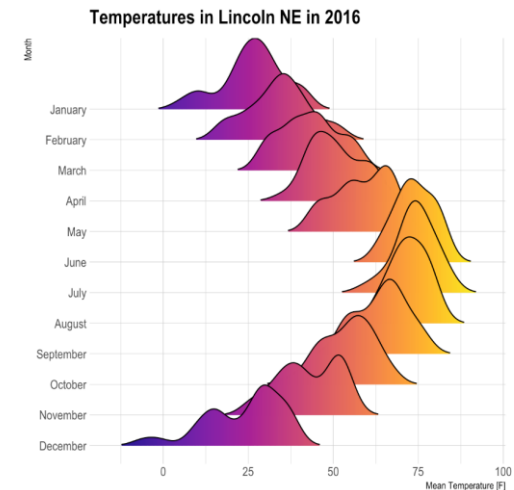
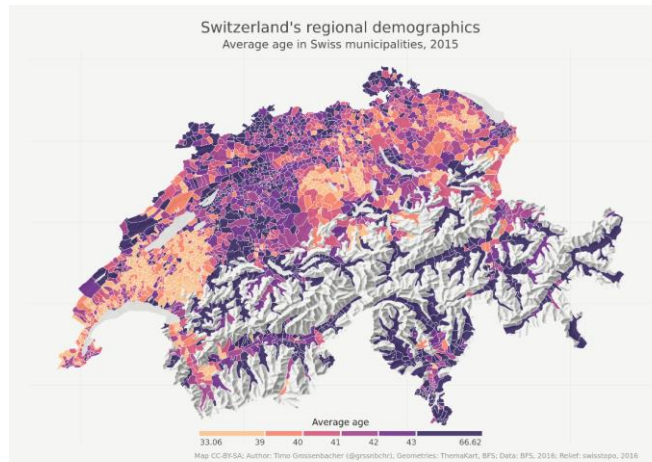
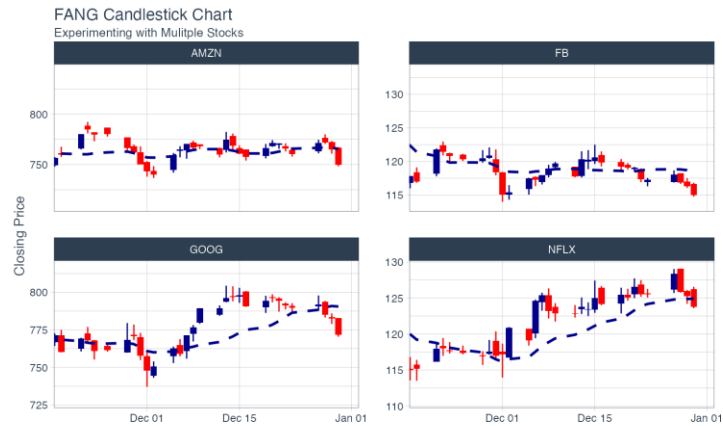
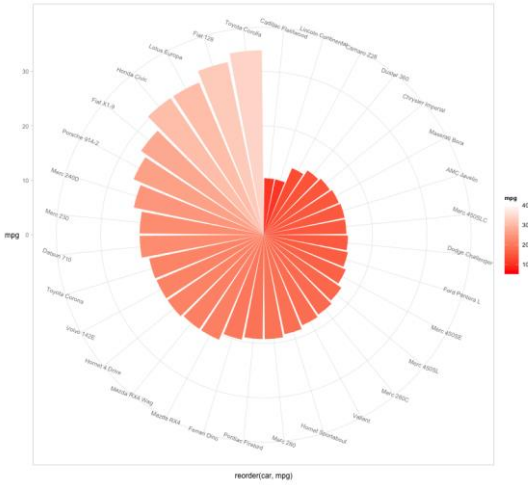
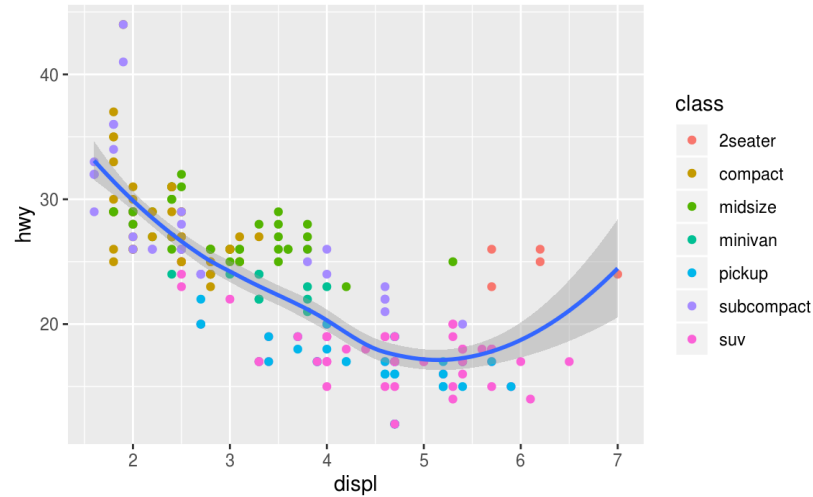
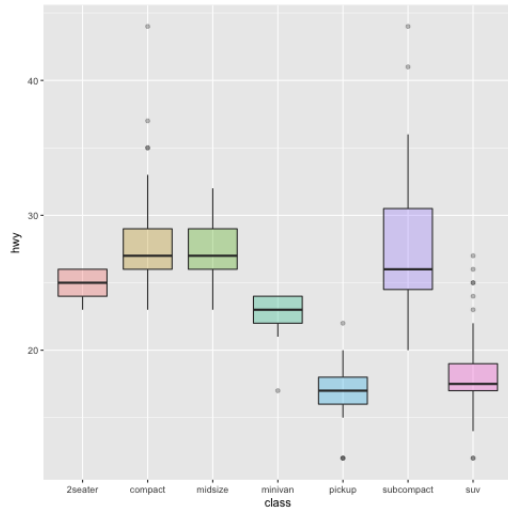
- R = a language + an eco-system
 - A free and open-source programming language
 - An eco-system of many high-quality user-contributed libraries/packages
- In the past R is mostly known for its statistical analysis toolkits
- Nowadays R is capable of (and very good at) many other tasks
 - Tools that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

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 Empirical Finance Packages on [R Task View - Finance](#)

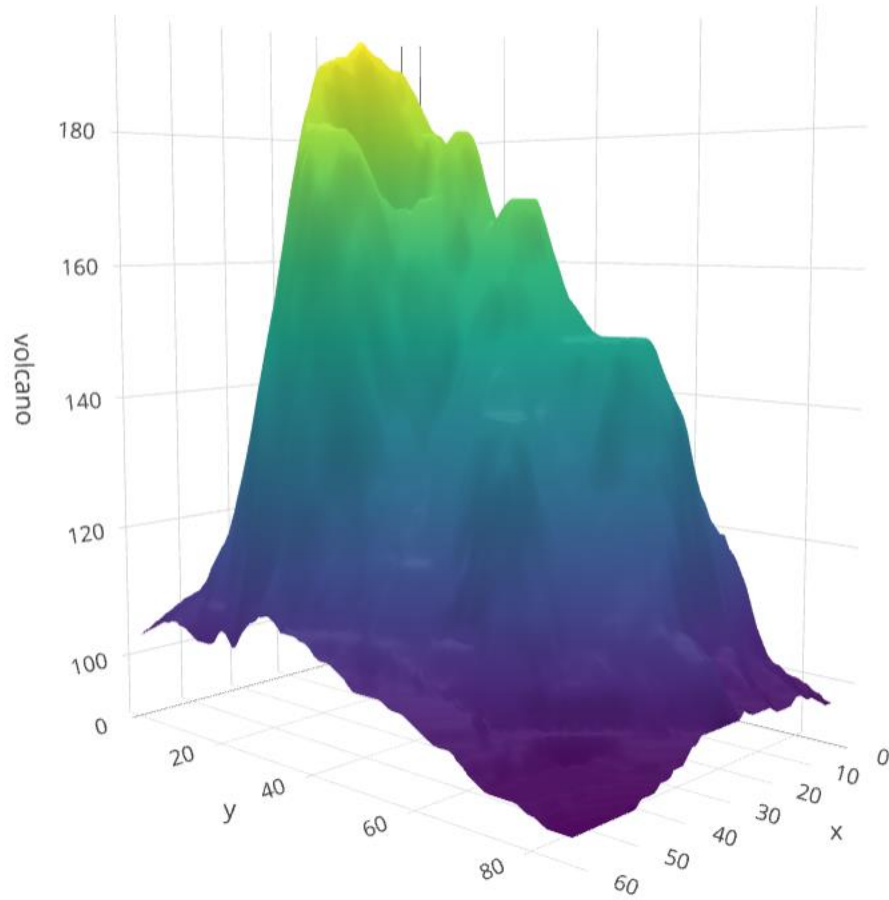
What can R do – Graphics (static ones)



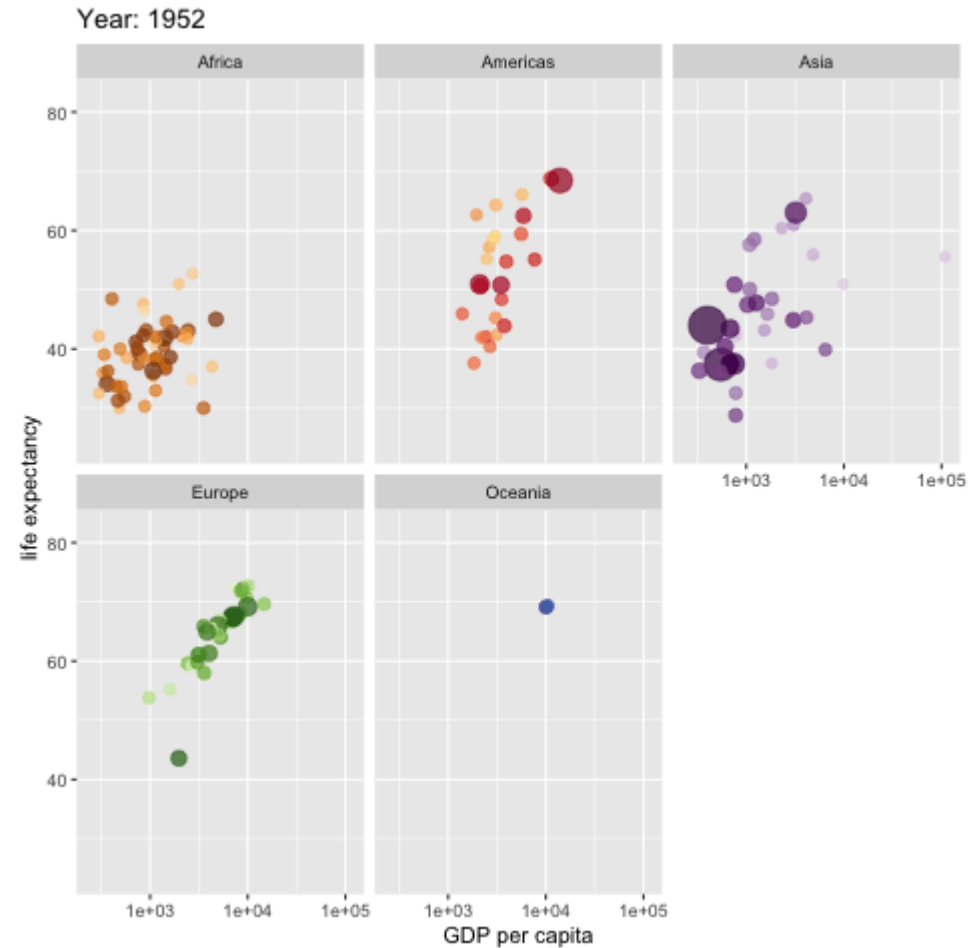
Ref. 1) <https://www.r-graph-gallery.com/>

2) <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/)

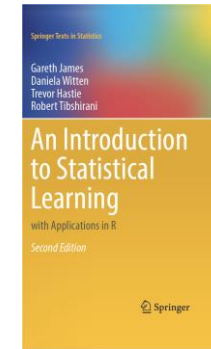


<https://github.com/thomasp85/gganimate;>

What can R do – ML & NLP

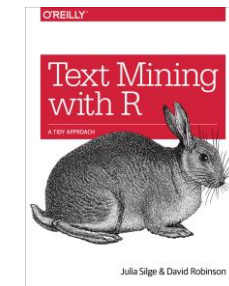
- Machine learning

- Statistical learning (clustering, decision tree, etc.)
 - [An Introduction to Statistical Learning \(with Applications in R\)](#)
- Deep learning (neural networks)
 - [Tensorflow for R](#) (via [reticulate](#), an R to Python interface)
 - [Torch for R](#) (natively from R; similar as PyTorch in Python)



- Natural language processing

- Packages (e.g., [tidytext](#), [topicmodels](#))
- Books (e.g., [Text Mining with R](#), [Supervised ML for Text Analysis in R](#))



1. See more R Machine Learning Packages on [R Task View - ML & Statistical Learning](#)
2. See more R Natural Language Processing Packages on [R Task View - NLP](#)

What can R do – Web & Reporting

- 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](#).)
 - [Quarto](#) (new authoring tool; multi-language and multi-engine;)
- ... (see [R Task View](#) for more)

R vs Excel and BI Tools vs Python



- Excel & Business Intelligence (BI) Tools (e.g., Tableau, Power BI, etc.)
 - 2-D tables as basic data structure
 - Good UI (User Interface) and minimum programming
 - Limited modeling tools
 - Not easy to reproduce an analysis (because it's hard to store UI clicks)
 - Not flexible enough for complicated analytics problems, i.e., problems with
 - Many data cleaning steps/pipelines
 - Many different models to try



Power BI Desktop

- Python

- Python is more general purpose; R is more specialized in statistical analysis
- R is much easier to learn (in my opinion)





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 Part 1

- **Intro**
 - What is R and what can R do?
 - **Setup R**
 - Motivation examples
- Overview of R programming and Data Science
 - Basics of R programming
 - Data science with R
- Learning Resources and Road Map

Setup R (Install R & its Coding Environment)

- R & **RStudio** *on your local computer*  **Our Choice**
 - Install R (<https://www.r-project.org/>)
 - Install RStudio (<https://rstudio.com/products/rstudio/download/>)
- R & **Notebook** *in the Cloud* (run R without installation)
 - Option 1: Google Colab (<https://colab.to/r>)  **Our Choice**
 - Option 2: UofT JupyterHub Notebook (<https://jupyter.utoronto.ca/hub/login>)
- R & RStudio *in the Cloud* (run R without installation)
 - Option 1: RStudio Cloud (<https://rstudio.cloud/>)
 - Option 2: UofT JupyterHub RStudio (<https://jupyter.utoronto.ca/hub/login>)

Setup R

- R & RStudio *on your local computer* (most of you should choose this one)
 - Install R (<https://www.r-project.org/>)
 - Install RStudio (<https://rstudio.com/products/rstudio/download/>)
- R & RStudio *in the Cloud* (run R without installation)
 - RStudio Cloud (<https://rstudio.cloud/>)
 - UofT JupyterHub RStudio (<https://jupyter.utoronto.ca/hub/login>)
- R & **Notebook** *in the Cloud* (run R without installation)
 - UofT JupyterHub Notebook (<https://jupyter.utoronto.ca/hub/login>)
 - Google Colab (<https://colab.to/r>)

What's RStudio?

The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for creating a graph from a tribble table.
- Environment:** Lists objects in the Global Environment: `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), and `raw` (4 obs. of 5 variables).
- Viewer:** Shows a network graph with four nodes (1, 2, 3, 4) and directed edges: 1 → 2, 2 → 3, and 2 → 4.
- Console:** Shows the execution of the R code, including the `render_graph()` command.

```
1 library(Diagrammer)
2
3 raw <- tribble(
4   ~id, ~in_node, ~out_node, ~in_time, ~out_time,
5   #--|--|--|
6   1, 1, 2, 1, 3,
7   1, 2, 3, 3, 5,
8   2, 1, 2, 2, 3,
9   2, 2, 4, 3, 6
10 )
11
12 node_tb_tp <- raw %>%
13   distinct(in_node) %>%
14   rename(node_id = in_node)
15
16 node_tb <- raw %>%
17   distinct(out_node) %>%
18   rename(node_id = out_node) %>%
19   union(node_tb_tp) %>%
20   arrange(node_id)
21
22 edge_tb <- raw %>%
23   distinct(in_node, out_node) %>%
24   rename(from = in_node, to = out_node)
25
26 g <- create_graph() %>%
27   add_nodes_from_table(table = node_tb) %>%
28   add_edges_from_table(
29     table = edge_tb,
30     from_col = from,
31     to_col = to,
32     from_to_map = node_id)
33
34 g %>% render_graph()
```

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 area is titled 'R Intro / Workshop 1' and features a menu bar with options: File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help. Below the menu bar is a toolbar with icons for file operations and a 'Go to file/function' search box. The R version is indicated as 'R 4.0.3'. The central pane shows a code editor with a single line of text: '1 |'. The right-hand side 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 |

The bottom panel is the 'Console', showing 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.
> |


RStudio at UofT Jupyterhub

 Jupyter Notebook RStudio JupyterLab'. There is an orange 'Log in to start' button and a link to 'JupyterHelp' for support. At the bottom, there is a welcome message and logos for Jupyter and RStudio."/>

JupyterHub

https://jupyter.utoronto.ca/hub/login

Not syncing



UNIVERSITY OF
TORONTO
JUPYTERHUB

Operated by 2i2c.org



After logging in, open: Jupyter Notebook RStudio JupyterLab

Log in to start

Use [JupyterHelp](#) to open tickets for support questions.

Welcome to the new University of Toronto **JupyterHub for Teaching** site.

A proof of concept service, developed as a partnership between the [Office of the CIO](#) (Information Technology Services), the Faculty of Arts & Science's new Computational and Data Science



R Notebook in 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 like "+ Code", "+ Text", and "Copy to Drive", 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)
```

R Notebook at UofT Jupyterhub

JupyterHub

https://jupyter.utoronto.ca/hub/login

Not syncing



UNIVERSITY OF
TORONTO
JUPYTERHUB

Operated by zi2c.org



After logging in, open: Jupyter Notebook RStudio JupyterLab

Log in to start

Use [JupyterHelp](#) to open tickets for support questions.

Welcome to the new University of Toronto **JupyterHub for Teaching** site.

A proof of concept service, developed as a partnership between the [Office of the CIO](#) (Information Technology Services), the Faculty of Arts & Science's new Computational and Data Science



Plan for Part 1

- **Intro**
 - What is R and what can R do?
 - Setup R
 - **Motivation examples**
- Overview of R programming and Data Science
 - Basics of R programming
 - Data science with R
- Learning Resources and Road Map

A Few Examples

- Analyze portfolio performance
- Perform simple sentiment analysis on earning call transcripts
- Recognize handwritten digits - an example of deep learning



**PerformanceAnalytics
Package**



A Few Examples: What to Look For

- Focus on analysis workflow (by reading the code comments)
 - Import and manipulate data
 - Model data
 - Report and visualize results
- Don't focus on R syntax
- Do notice everything is done in a sequential way
 - no conditional branching or looping

Plan for Part 1

- Intro
- Overview of R programming and Data Science
 - Basics of R programming
 - Expression & Assignment
 - Data Structure
 - Programming Structure (control flow & function)
 - Turn ideas into code
 - Data science with R
- Learning Road Map and Resources

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 | |

R Data Structure - Overview

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

Atomic Vectors

```
# create R vectors
```

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

| | |
|---------------|---------------|
| Hello, | World! |
|---------------|---------------|

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

| | | |
|----------|----------|----------|
| 1 | 2 | 3 |
|----------|----------|----------|

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

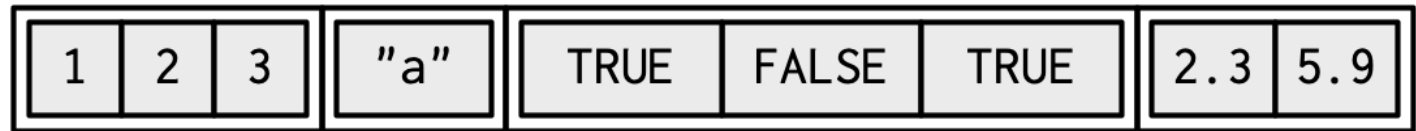
| | | |
|------------|------------|------------|
| 1.1 | 2.2 | 3.3 |
|------------|------------|------------|

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

| | | |
|-------------|-------------|--------------|
| TRUE | TRUE | FALSE |
|-------------|-------------|--------------|

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)
)
```

| 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 |

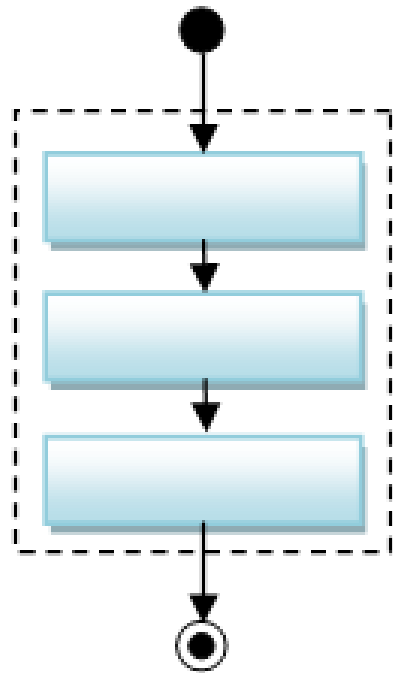
A Cousin to Data Frame - Tibble

```
# 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)
)
```

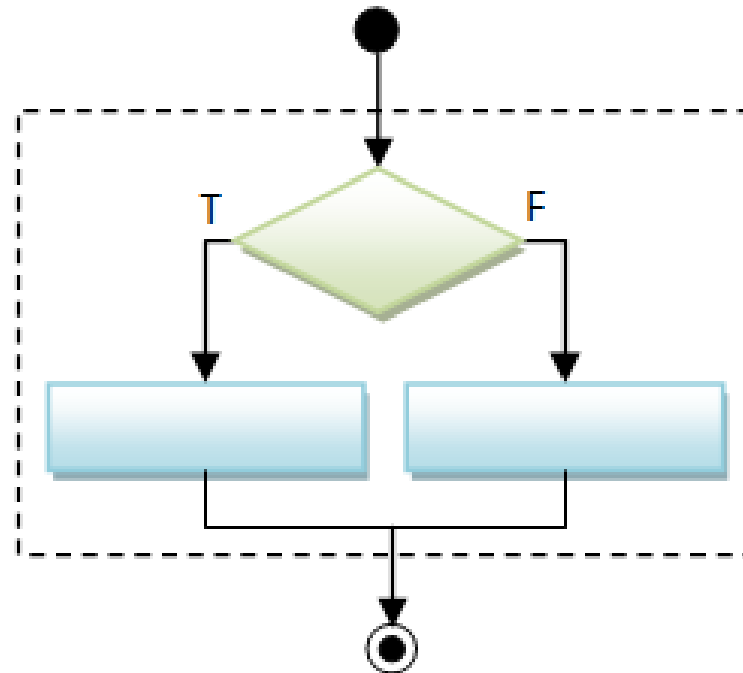
| x | y | z |
|---|-----|-----|
| 1 | "a" | 1.1 |
| 2 | "b" | 2.2 |
| 3 | "c" | 3.3 |

Programming Structure: Control Flows



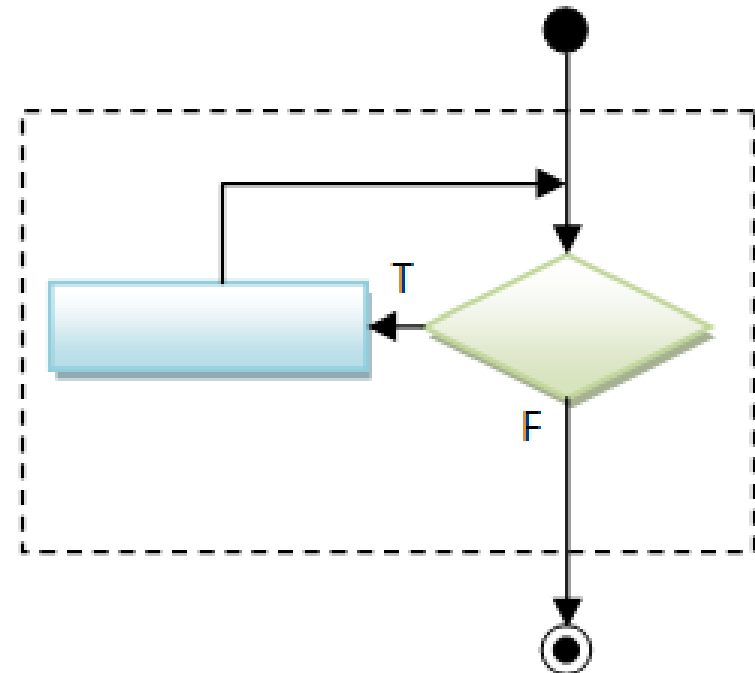
Sequential

Today



Conditional (Decision)

Learn on your own (See Appendix)



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 | | |

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))
```

Turn Ideas into Code

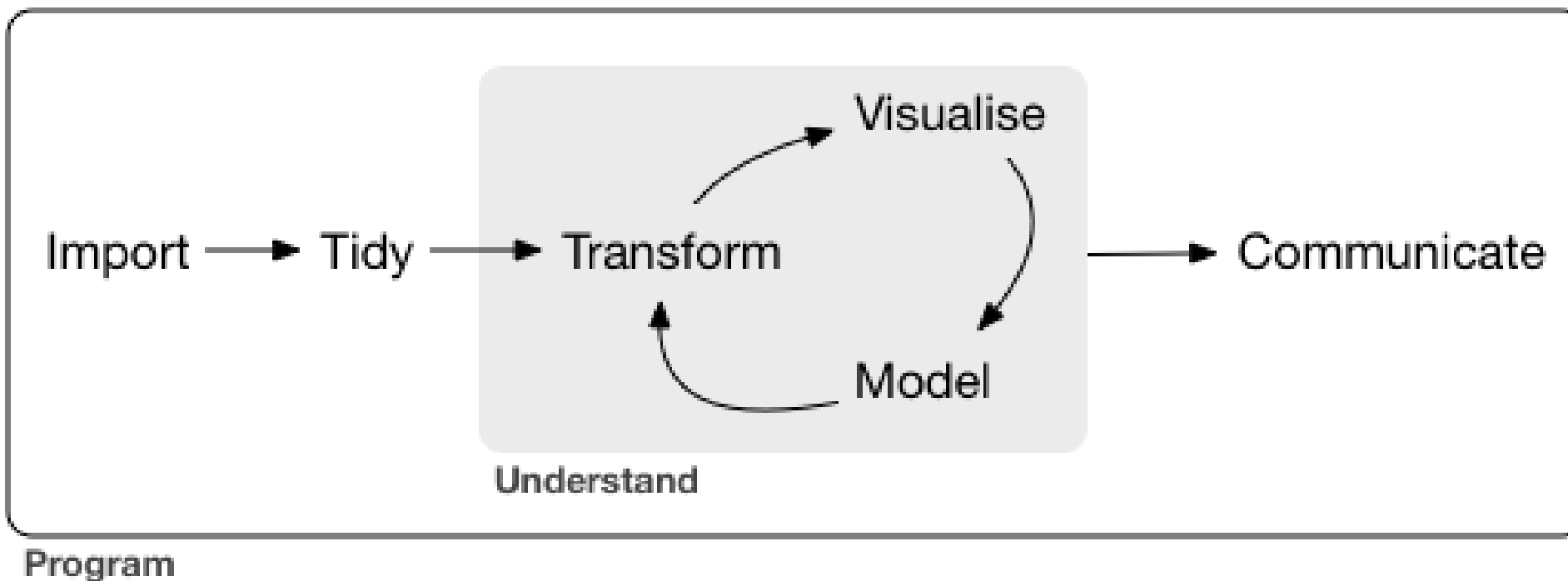
- Solve problems using code: combine three main ingredients
 - Data Structure (vector, list, data frame, etc.)
 - Programming Structure (sequential, conditional, iterative, functions)
 - Algorithm (sorting, searching, optimization, etc.)
 - Design to bind the above 3 together (functions, classes, design patterns...)
- Examples
 - Generate and solve Sudoku puzzles
 - Implement and backtest a trading rule/algorithm
 - Import, manipulate, and model data
- For us, in most cases, we solve problems by
 - Using other people's algorithm implementations (i.e., functions from R packages)
 - Simple design to combine algorithms, data & programming structures to model data (slightly easier, but still need practices to write good code.)

Plan for Part 1

- Intro
- Overview of R programming and Data Science
 - Basics of R programming
 - Data science with R
 - A Typical data analysis workflow
 - Choice of R packages
 - An example: regression analysis
- Learning Road Map and Resources

Data Science/Analysis Workflow

- Use this workflow to organize your thoughts and code



An Example: Housing Price & Clean Air

Obs: 506

- Manipulate data
 - Load data
 - Create new columns
 - Filter columns and rows
 - Build models
 - Multiple linear regressions
 - Report and graph
 - Build a publication-ready table for regression results
- | | |
|-------------------|--------------------------------------|
| 1. price | median housing price, \$ |
| 2. crime | crimes committed per capita |
| 3. nox | nitrous oxide, parts per 100 mill. |
| 4. rooms | avg number of rooms per house |
| 5. dist | weighted dist. to 5 employ centers |
| 6. radial | accessibility index to radial hghwys |
| 7. proptax | property tax per \$1000 |
| 8. stratio | average student-teacher ratio |
| 9. lowstat | % of people 'lower status' |

Many Ways to Achieve the Same Goal

- The “pure” R way
 - Mostly use functions/packages in [R standard library](#) (those shipped with R)
 - for structuring and manipulating data
 - for modeling if possible (e.g. regressions)
 - An example of a simple linear regression
- The “modern” way
 - Use specialized packages to manipulate data and assist modeling tasks
 - Data are stored in improved data structures (in most cases still compatible with base R data structure)
 - What we will focus on

R Packages: Many choices, which one to use

- Often, a task can be achieved using functions in different libraries
 - R is open and extensible!

- Example: load a csv file to a data frame/tibble/data table

- Use [read.csv\(\)](#) function from the `utils` library in Base R

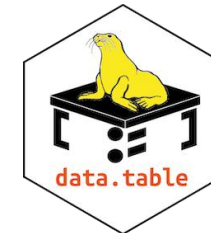


- Use [read_csv\(\)](#) function from the [readr](#) library



- Use [vroom\(\)](#) from the [vroom](#) library

- Use [fread\(\)](#) function from the [data.table](#) library



R Packages: 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 date
 - packages maintained by companies (e.g., RStudio Co.) or academic teams
- Choose one that suits your task

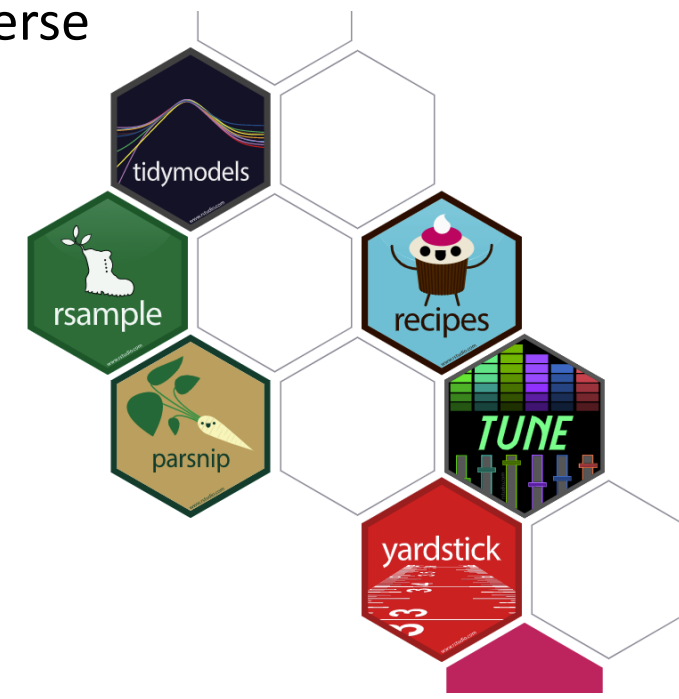
Great Choice for Data Science Work

- Tidyverse

- “an opinionated collection of R packages designed for data science”
- “All packages share an underlying design philosophy, grammar, and data structures.”
- Handle data manipulation, visualization, and much more
- an eco-system: many package developers started to follow tidyverse principles too

- Tidymodels

- “a collection of packages for modeling and machine learning using tidyverse principles”
- Manage modeling process but does not do modeling itself



Our Choice: 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)

Using R packages/libraries

- Install an R library (only need to install a library once)

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

- Load an R library (before you use a library)

```
library(Library_name)
```

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

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
- ...

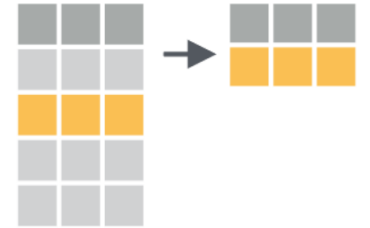
Load Data – Financial Dataset

- [tq_get\(\)](#) from tidyquant library
 - collect financial and economic data from many online sources
 - Yahoo Finance, FRED, Quandl, Tiingo, Alpha Vantage, Bloomberg
- [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 (rows): **filter()**

```
filter(my_dataframe, condition1, ...)  
e.g., hprice_reg <- filter(hprice, price > 20000)
```



- Select variables (columns): **select()**

```
select(my_dataframe, var1, ...)  
e.g., hprice_reg <- select(hprice_reg, lprice, rooms)
```



- Create new variables: **mutate()**

```
mutate(my_dataframe, new_var1 = expression1, ...)  
e.g., hprice_reg <- mutate(hprice_reg, lprice = log(price))
```

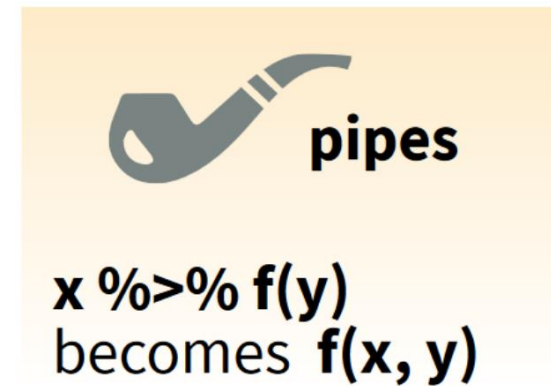


Data Manipulation: Data Pipe (%>%)

```
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)
```



Regression

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

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

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon_i$$

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

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon_i$$

- 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: [huxreg\(\)](#) from [huxtable](#) library

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

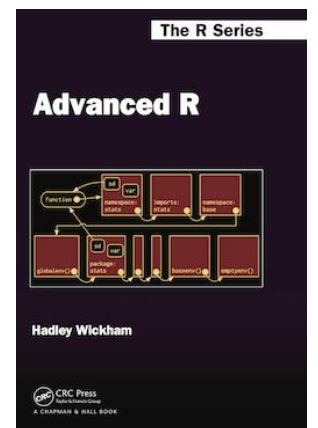
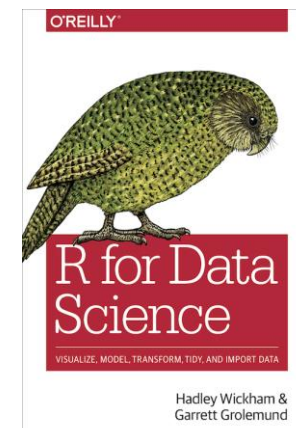
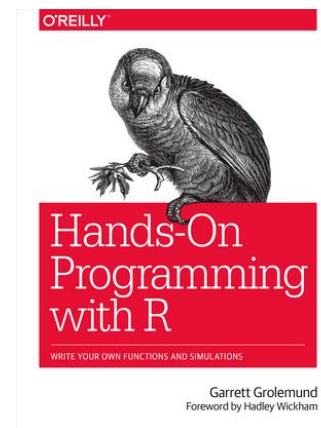

Plan for Today

- Intro to Intro
- Overview of R programming and Data Science
 - Basics of R programming
 - Data science with R
- Learning Resources and Road Map

R Learning Road Map (From Zero to Hero)

- Step 1. Basic R programming skills (Beginner)
 - Data and programming structure; how to turn an idea into code;
 - Book: [Hands-On Programming with R](#)
- Step 2. R Data Science skills (Intermediate)
 - Data wrangling, basic modeling, and visualization/reporting; Best practice;
 - Book: [R for Data Science](#)
- Step 3. Take your R Skill to the next level
 - Book: [Advanced R](#)

Ref. For other free R books, check bookdown.org often



Learning Approach

- Learn the underlying principles
 - e.g., why organize data in a certain way
- Learn best practices
 - follow a consistent analysis workflow

Free Learning Resource

- [RStudio Education](#)
 - [Choose Your Learning Paths](#)
- [RStudio Video Resources Site](#)
- More free R books? Check bookdown.org often
- Coursera: Search R and learn
 - free for [UofT students](#) (mostly always free if you just audit the courses)
- Twitter (a few seeds: [#rstat](#), [@hadleywickham](#), [@WeAreRLadies](#))

Appendix

- Programming Structure Continued
 - Conditional
 - Iteration

Conditional (if...else...)

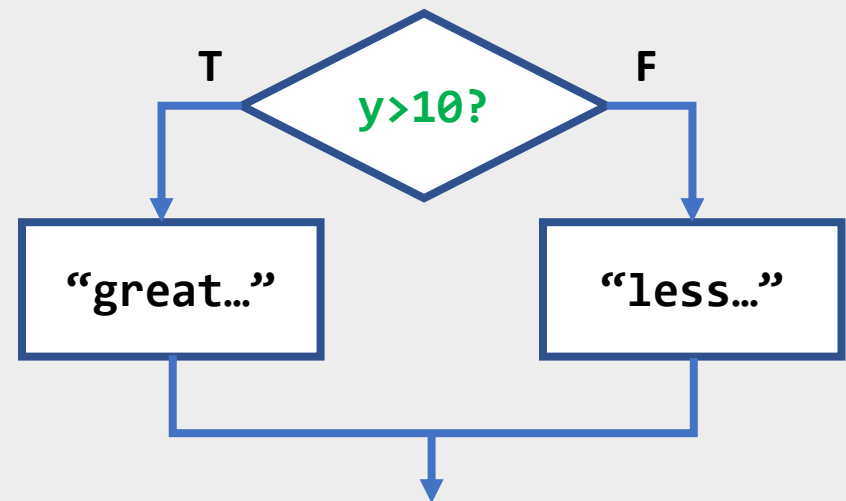
```
if (cond) {  
    # run here if cond is TRUE  
} else {  
    # run here if cond is FALSE  
}
```

```
# 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)
```