Rotman

INTRO TO R PROGRAMMING

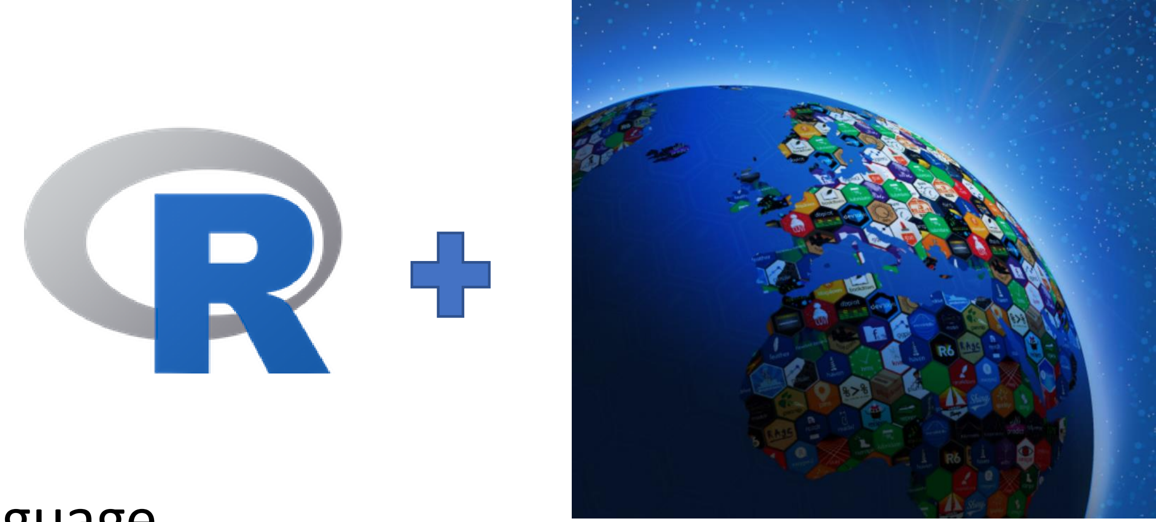
R Tutorial (RSM358) – Session 1

September 10, 2024 Prepared by Jay Cao / TDMDAL

Website: https://tdmdal.github.io/r-intro-2024-fall/

Rotman School of Management
UNIVERSITY OF TORONTO

What’s R?



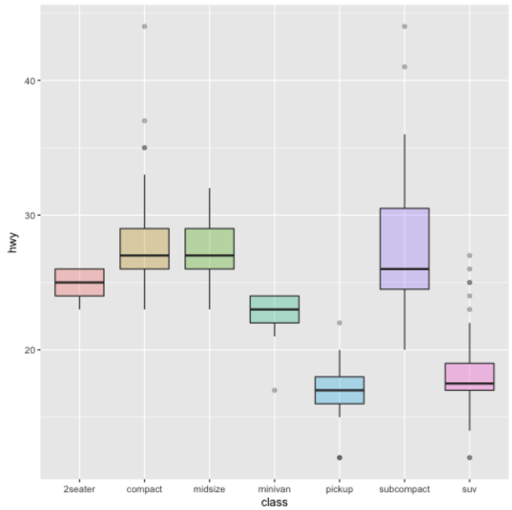
- R = a language + an eco-system
  - A free and open-source programming language
  - An eco-system of many high-quality user-contributed libraries/packages

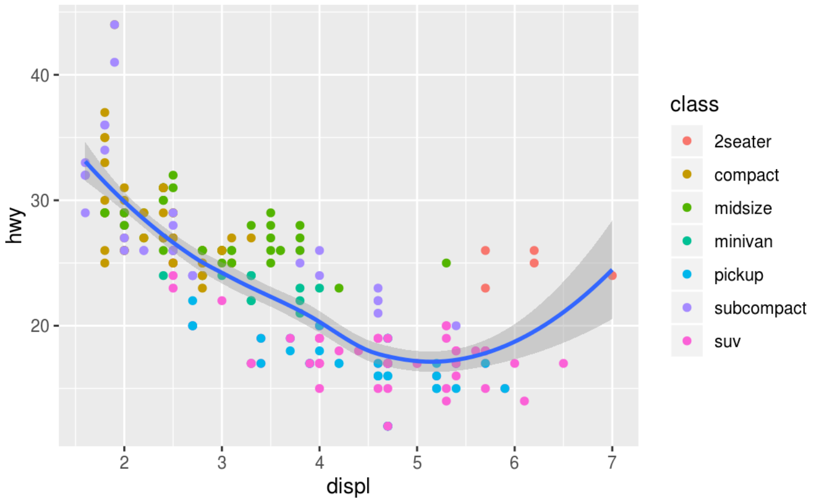
- In the past R is mostly known for its statistical analysis toolkits
- Nowadays R is capable of (and very good at) many other tasks
  - Tools that facilitates the whole data analysis workflow
  - Tools for web technology (e.g., web scraping, web app/dashboard development, etc.)
  - Many more…

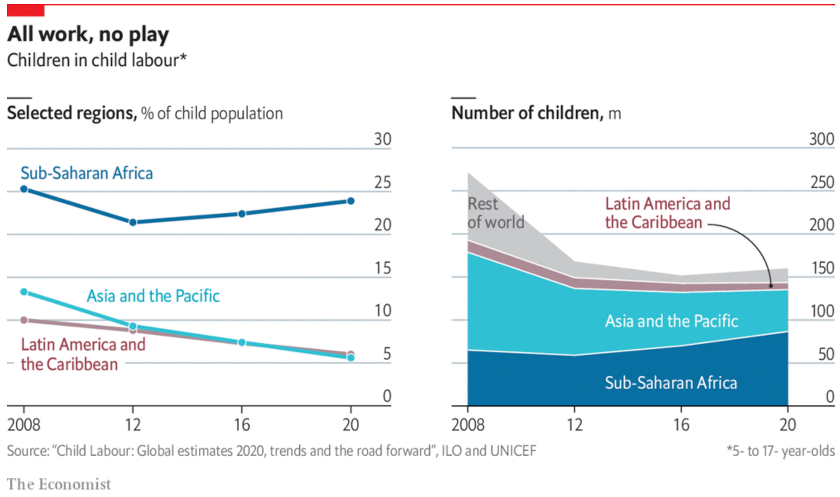
What can R do – Statistics & related

- Statistics & Econometrics
  - Regressions
  - Time series analysis
  - Bayesian inference
  - Survival analysis
  - …
- Numerical Mathematics
  - Optimization
  - Solver
  - Differential equations
  - …
- Finance
  - Portfolio management
  - Risk management
  - Option pricing
  - …
- Machine learning
  - …
- see **R Task View** for more

What can R do – Graphics





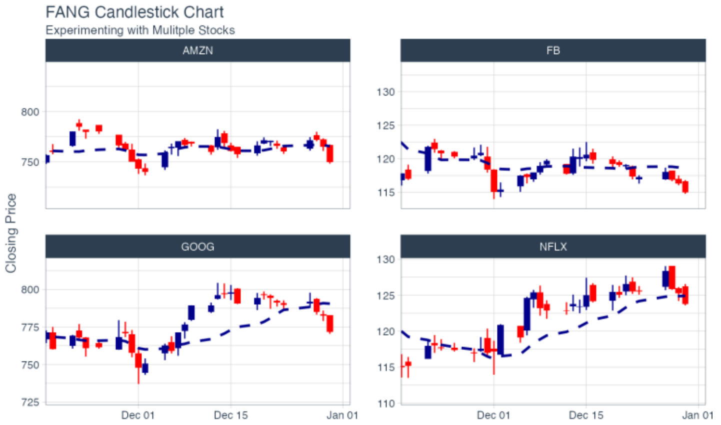

All work, no play

Children in child labour*

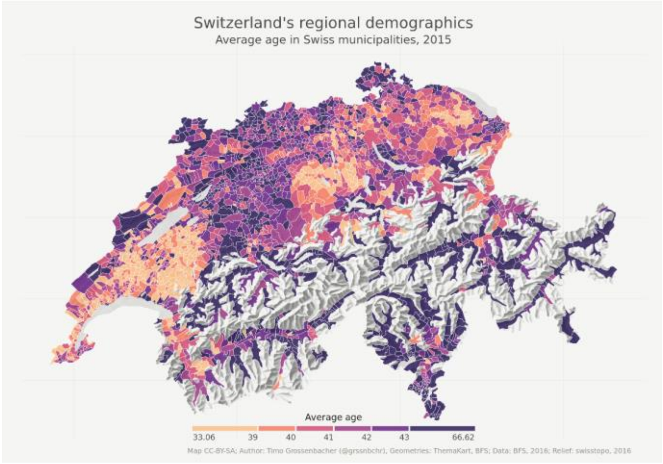
Source: "Child Labour: Global estimates 2020, trends and the road forward", ILO and UNICEF

*5- to 17- year-olds

The Economist

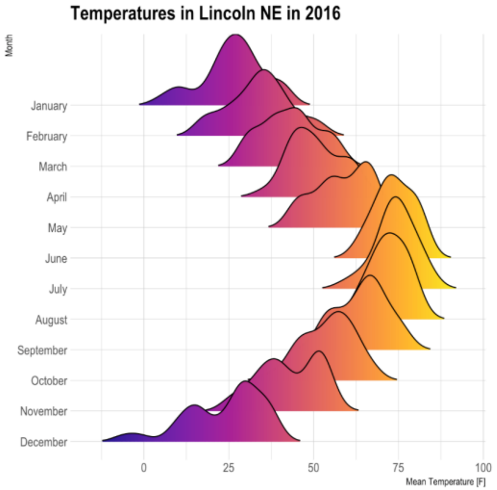

FANG Candlestick Chart

Experimenting with Mulitple Stocks


Switzerland's regional demographics

Average age in Swiss municipalities, 2015

Map CC-BY-SA; Author: Timo Grossenbacher (@grssnbchr), Geometries: ThemaKart, BFS; Data: BFS, 2016; Relief: swisstopo, 2016


Temperatures in Lincoln NE in 2016

…

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

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

Plan for Session 1

- Get started
  - Install RStudio
  - Create a project
  - Navigate RStudio
  - Install and load R packages

- Walk/work through chapter 2 lab from you textbook

- R programming basics (optional)
  - Expression and assignment
  - Basic data structures
  - Basic programming structures & functions

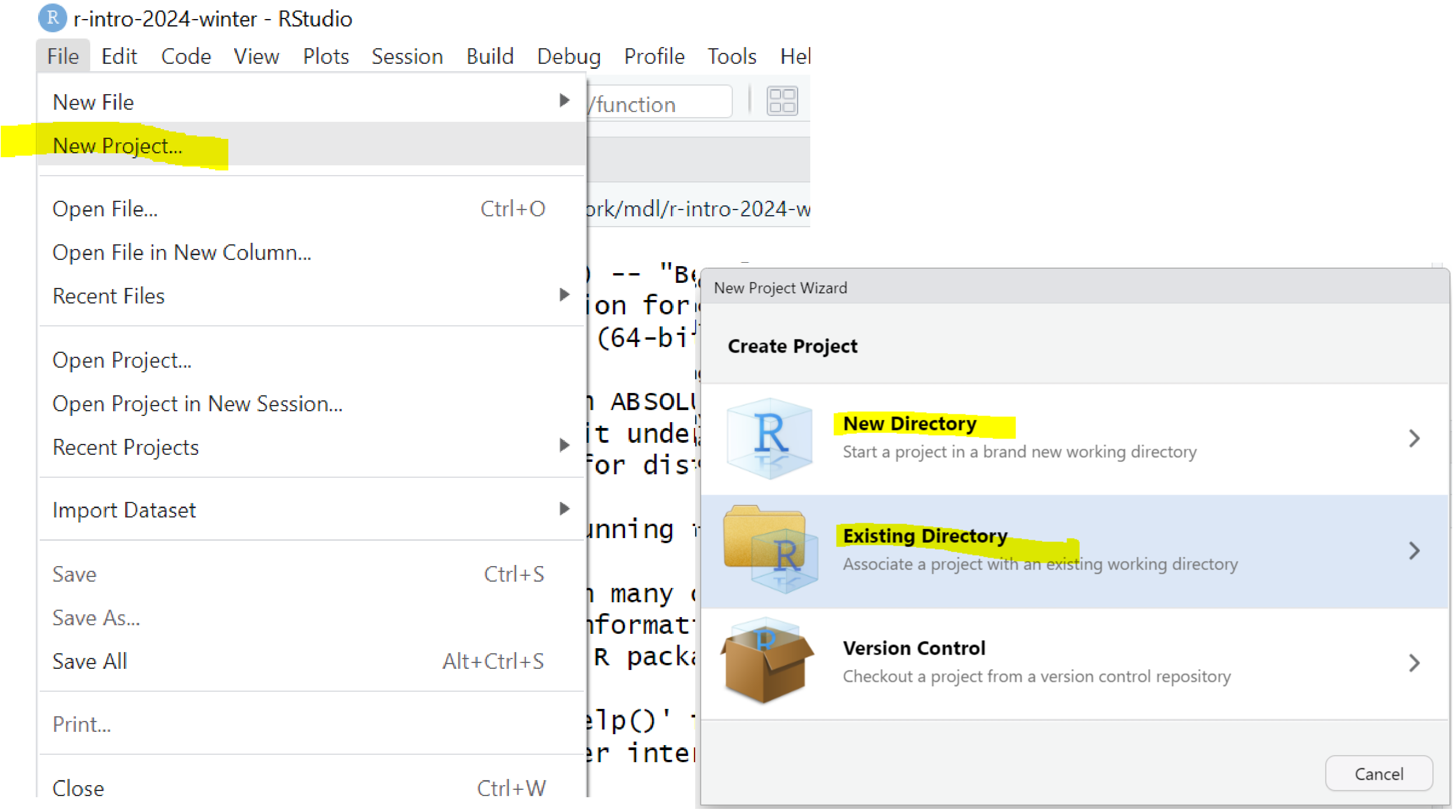
Setup R (Install R & its Coding Environment)

- **R** & **RStudio** on your **local computer** ← **Our Choice**
  - Install R (https://www.r-project.org/)
  - Install RStudio (https://rstudio.com/products/rstudio/download/)

- R & RStudio **in the Cloud** (run R without installation) ← **Backup Options**
  - Option 1: RStudio at UofT JupyterHub (https://datatools.utoronto.ca/)
  - Option 2: RStudio Cloud (https://rstudio.cloud/)

Note. In this workshop, we will may occasionally use R in Google Colab (https://colab.research.google.com/), a notebook coding environment in the cloud.

Create New Project – A Good Practice

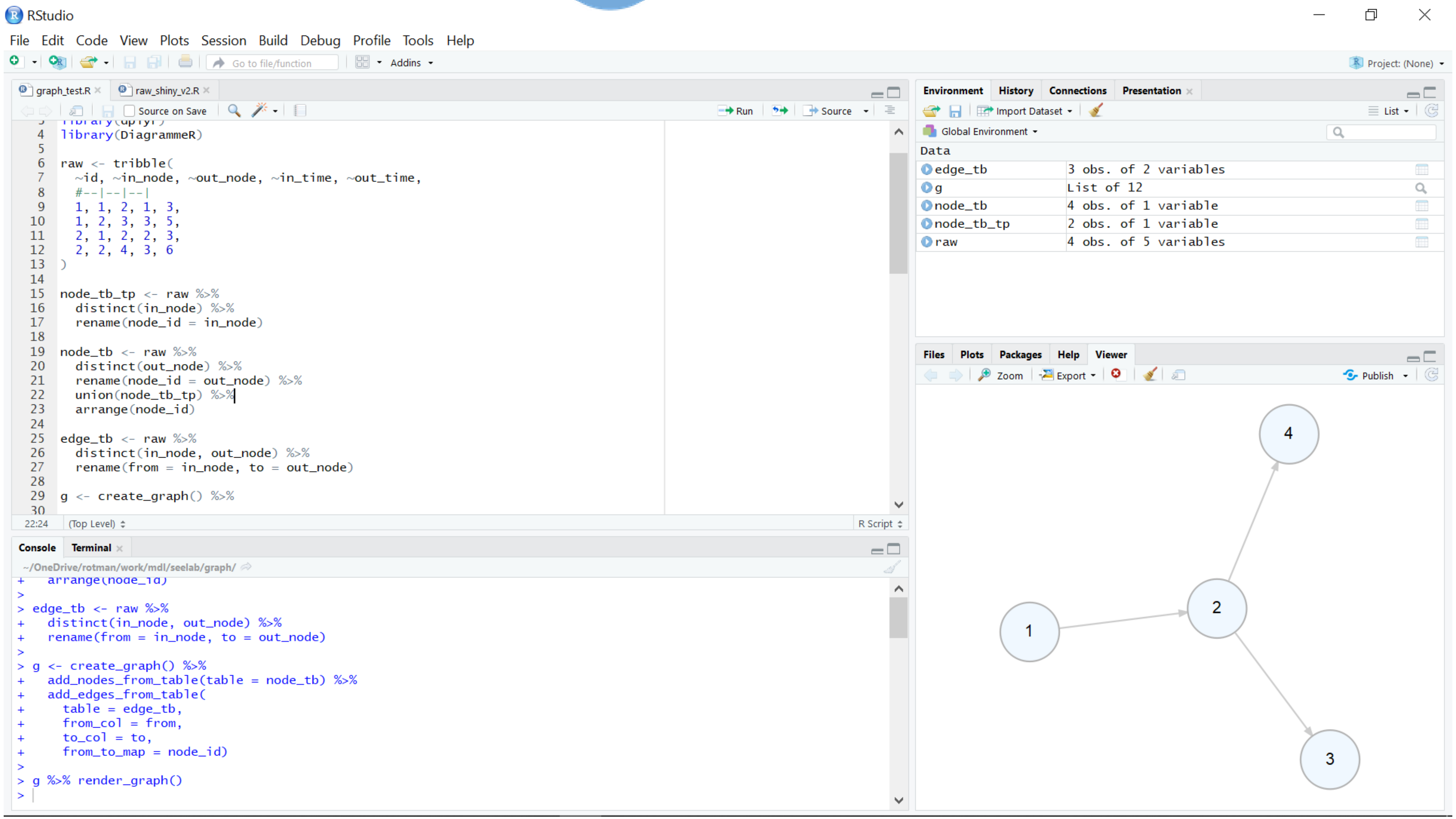


New Directory: Start a project in a brand new working directory

Existing Directory: Associate a project with an existing working directory

Version Control: Checkout a project from a version control repository

Navigate 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() %>%
```

```
+   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

Install and Load 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

Plan for Session 1

- Get started
  - Install RStudio
  - Create a project
  - Navigate RStudio
  - Install and load R packages

- Walk/work through chapter 2 lab from you textbook

- R programming basics (optional)
  - Expression and assignment
  - Basic data structures
  - Basic programming structures & functions

How to Do Well in Your Coding Assignment

- Read the relevant “theory” sections of your textbook

- Work through the relevant **lab** section of your textbook
  - Most coding questions are small variations of what’s shown in the lab section

- Your excellent textbook is free (www.statlearning.com/)
  - Many resources available on the textbook website (code, data, etc.)
  - Install the ISLR2 R package to have all the data needed for the assignments

Lab Code From Your Textbook

- Pure R code (.R files) in RStudio

- R Markdown (.Rmd files) in RStudio
  - Markdown text + code in pure text format
  - The textbook resource site also provides rendered html file

- R Jupyter Notebook (.ipynb files) in Google Colab (or Jupyter Lab, etc.)
  - Markdown text + code in special Jupyter notebook format

Textbook resource site: https://www.statlearning.com/resources-second-edition

Chapter 2 Lab Walk/Work Through Prep

- Textbook Resource Site
  - https://www.statlearning.com/resources-second-edition

- Pure R (`.R` file) in RStudio (recommended, R Markdown optional)
- Download the Auto data and/or install the ISLR2 package

- R Jupyter Notebook (`.ipynb`) in Google Colab (optional)
  - `installed.packages()`
  - `if (!require(ISLR2)) install.packages(“ISLR2")`

Plan for Session 1

- Get started
  - Install RStudio
  - Create a project
  - Navigate RStudio
  - Install and load R packages

- Walk/work through chapter 2 lab from you textbook

- R programming basics (optional)
  - Expression and assignment
  - Basic data structures
  - Basic programming structures & functions

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

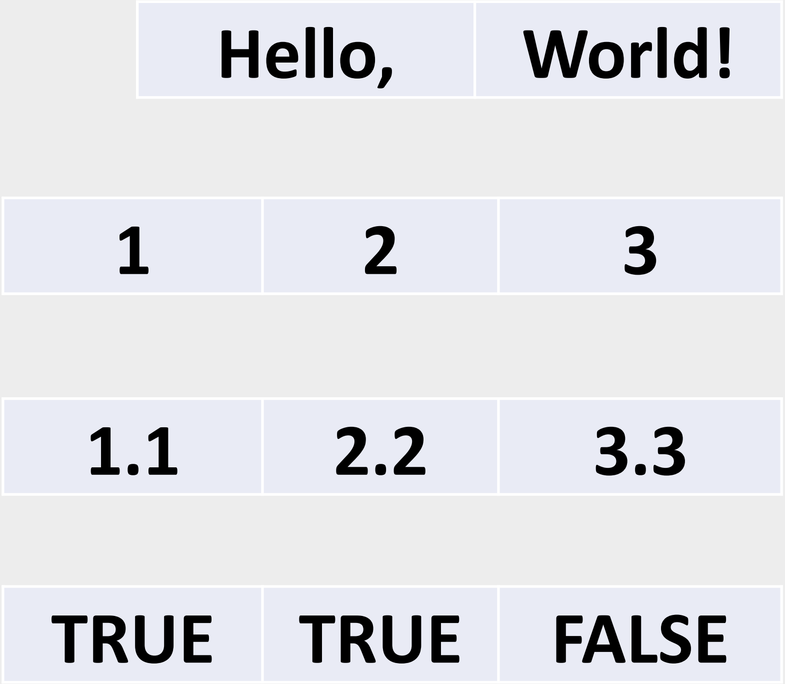
R Data Structure - Overview

	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

```
# 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)
```



Hello,	World!

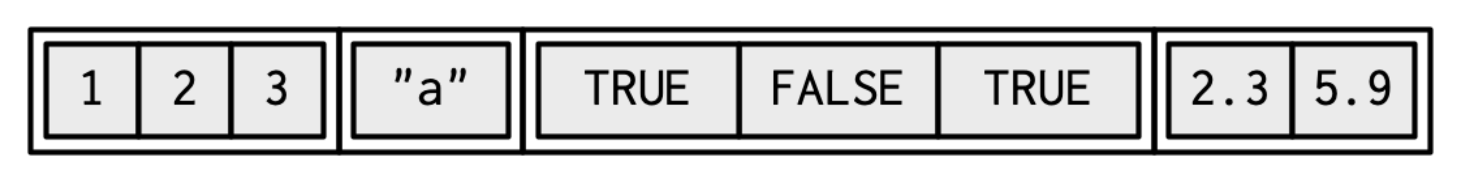
1	2	3

1.1	2.2	3.3

TRUE	TRUE	FALSE

List

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



1	2	3	"a"	TRUE	FALSE	TRUE	2.3	5.9

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

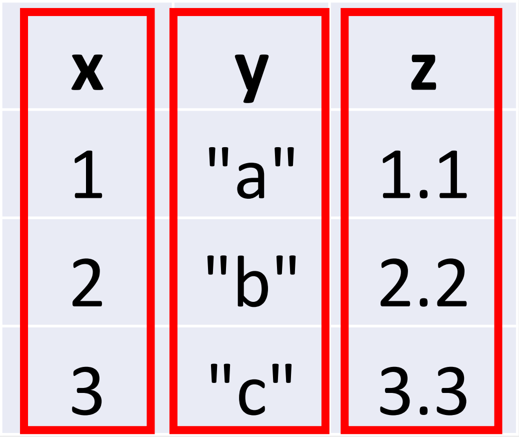
Data Frame

```
# 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

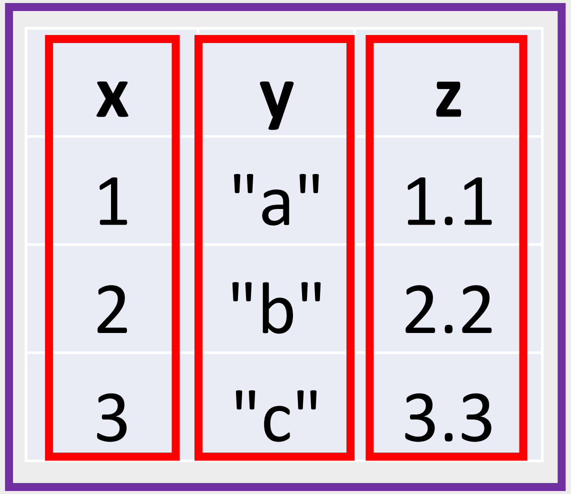
```r
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```



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

Data Frame

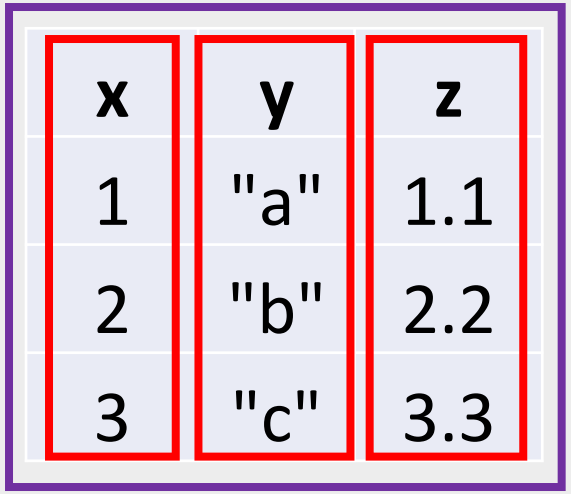
```
# create a data frame
df1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```



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

A Cousin to Data Frame - Tibble

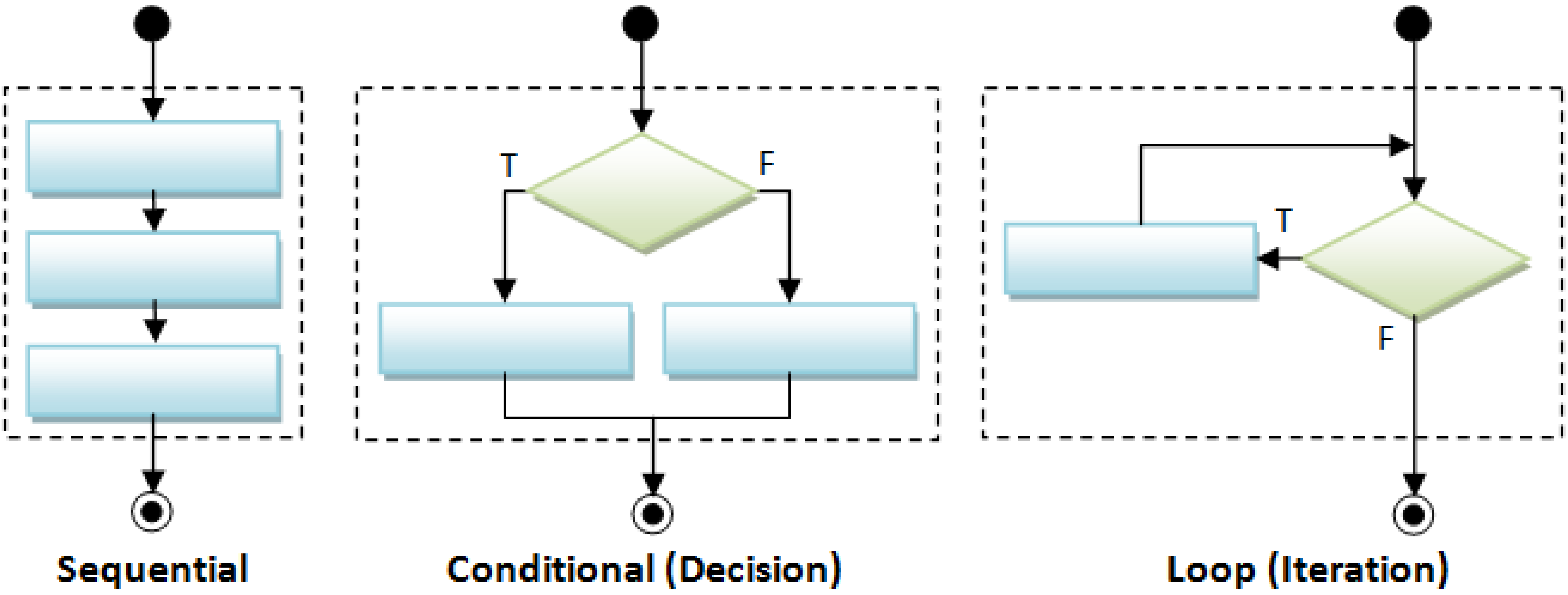
```
# load tibble library (part of tidyverse lib)
library(tibble)

# 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

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...)

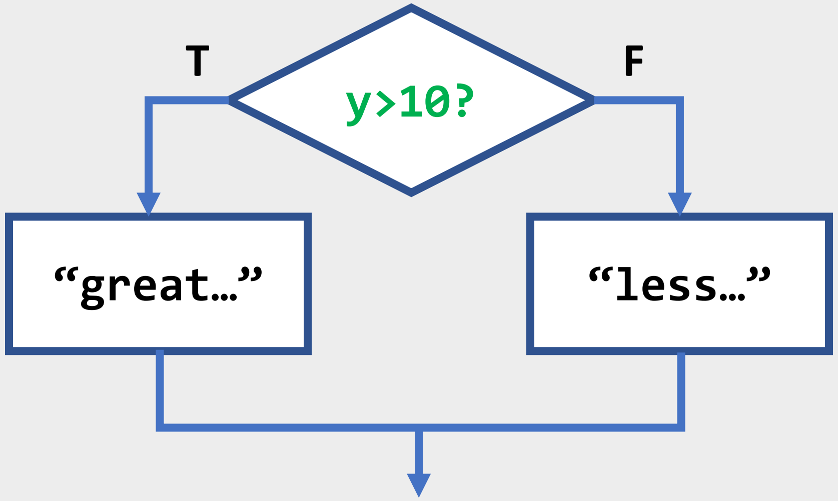
```
if (cond) {
  # run here if cond is TRUE
} else {
  # run here if cond is FALSE
}
```

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

```
# 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)  # return(sum(t^2))
}


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

Turn Ideas into Code

- Solve problems using code: three main ingredients
  - 1) Data Structure (vector, list, **data frame**, etc.)
  - 2) Programming Structure (**sequential**, conditional, iterative)
  - 3) Algorithm (sorting, searching, optimization, **modeling**, etc.)
  - Design to bind the above 3 together (functions, classes, design patterns, software architecture,…)

- Examples
  - Generate and solve Sudoku puzzles
  - Implement and backtest a trading rule/algorithm
  - **Import, manipulate, and model data**

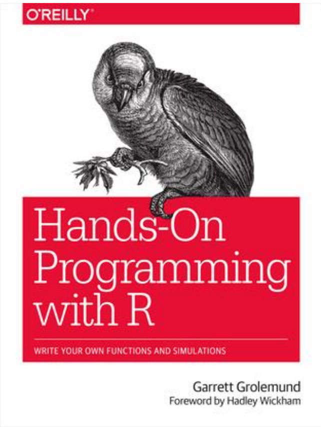
- For us (data analysis in RSM456), in most case,
  - Data frame manipulation + sequential programming flow + modeling (using algorithm already implemented by others)

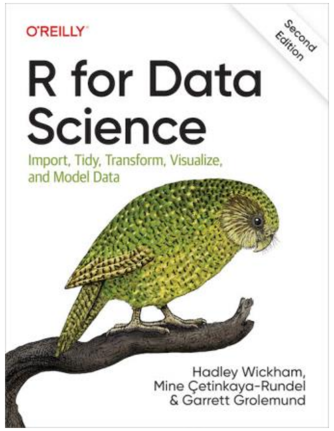
R Learning Road Map (From Zero to Hero)

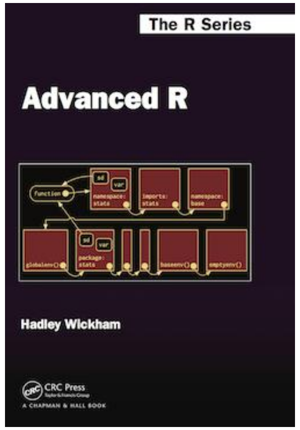
- Step 1. Basic R programming skills (Beginner)
  - Data and programming structure; how to turn an idea into code;
  - Book: Hands-On Programming with R

- 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


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


O'REILLY
Second Edition
R for Data Science
Import, Tidy, Transform, Visualize, and Model Data
Hadley Wickham, Mine Çetinkaya-Rundel & Garrett Grolemund



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