

Rotman

INTRO TO R PROGRAMMING

R Tutorial (RSM358) – Session 1

September 1, 2026 Prepared by Jay Cao / [MDAL](#)

Website: <https://rmdal.github.io/r-intro-2026-fall/>



Rotman School of Management
UNIVERSITY OF TORONTO

What's R?

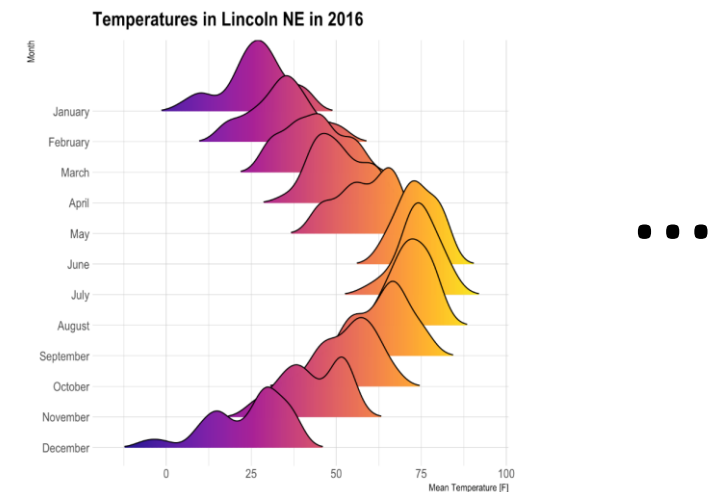
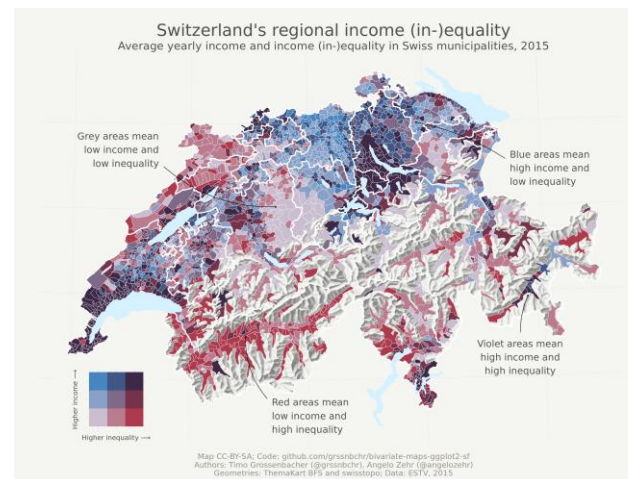
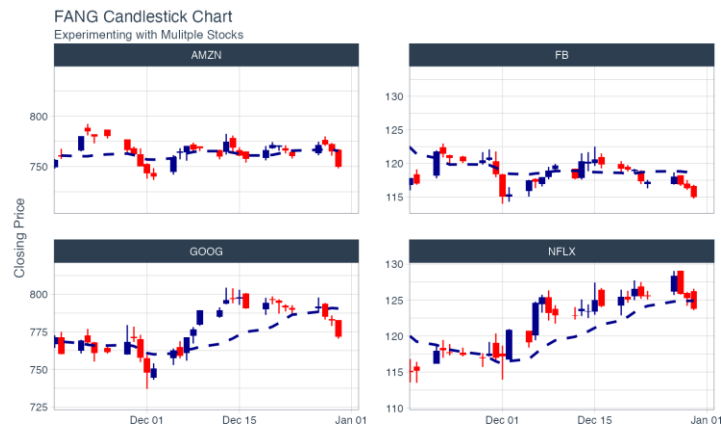
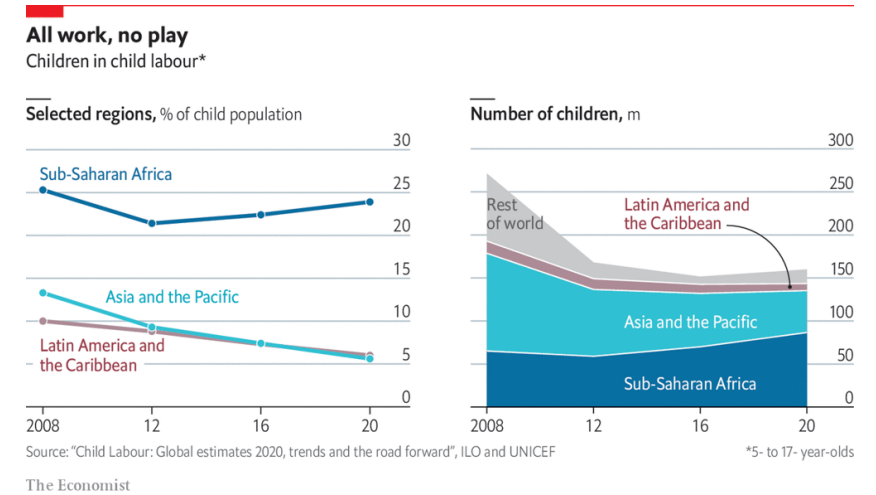
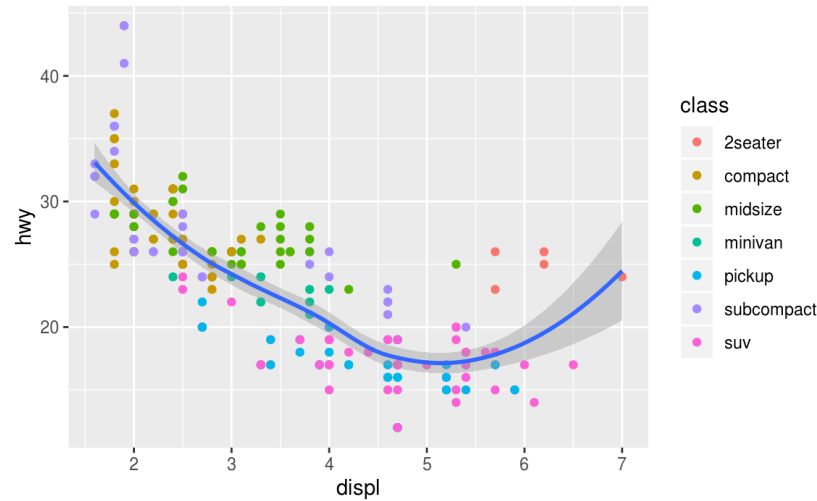
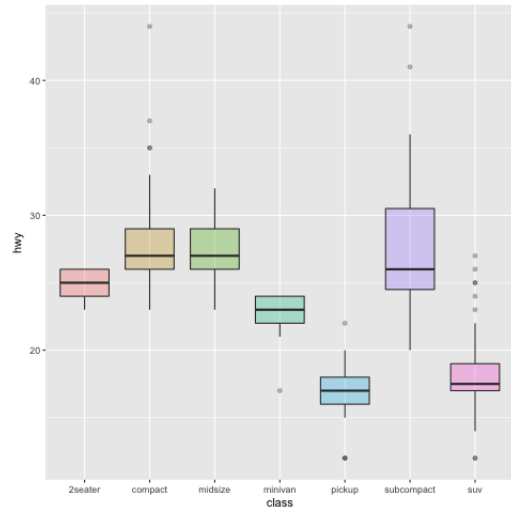


- R = a language + an eco-system
 - A free and open-source programming language
 - An eco-system of many high-quality user-contributed libraries/packages
- In the past R is mostly known for its statistical analysis toolkits
- Nowadays R is capable of (and very good at) many other tasks
 - Tools that facilitates the whole data analysis workflow
 - Tools for web technology (e.g., web scraping, web app/dashboard development, etc.)
 - Tools to facilitate use of AI/LLM, and 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



Ref: 1) <https://www.r-graph-gallery.com/>
2) <https://timogrossenbacher.ch/bivariate-maps-with-ggplot2-and-sf/>;

Plan for Session 1

- Get started
 - Install R & RStudio
 - Create a project
 - Navigate RStudio
 - Install and load R packages
- Walk through chapter 2 lab from your 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://docs.posit.co/ide/user/#rstudio-ide-oss-downloads>)
- **R & RStudio in the Cloud** (run R without installation)  **Backup Options**
 - RStudio at UofT JupyterHub (<https://datatools.utoronto.ca/>)

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

Navigate RStudio



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graph_test.R x raw_shiny_v2.R x

Source on Save Run Source

```
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
35 g %>% render_graph()
```

Source code

Environment History Connections Presentation

Global Environment

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

Environment, etc.

Files Plots Packages Help Viewer

Zoom Export Publish

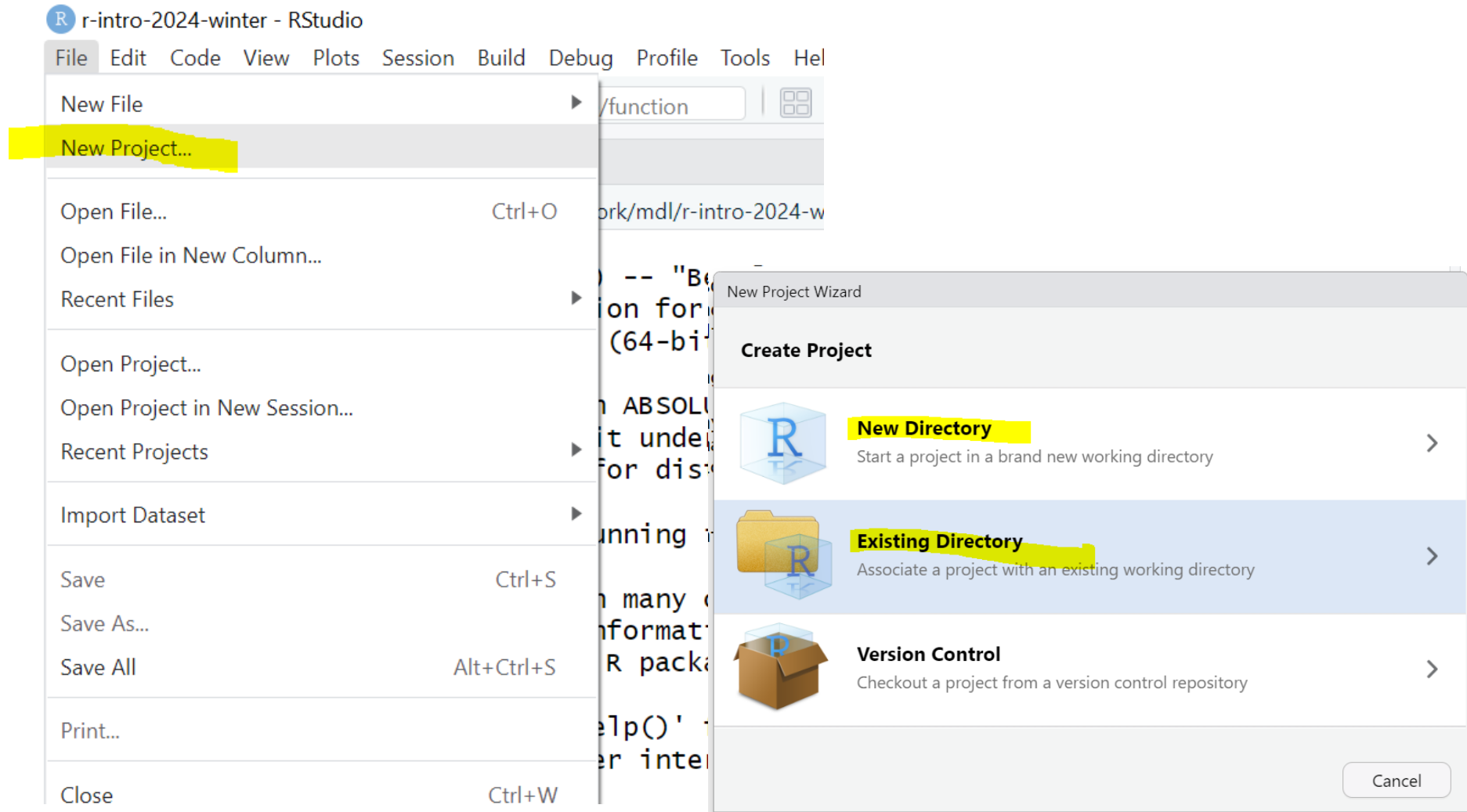
Files, Plots, etc.

```
graph LR
  1((1)) --> 2((2))
  2 --> 4((4))
  2 --> 3((3))
```

Command Console

```
+ 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()
+
```

Create New Project – A Good Practice



Install & Load R 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](#)

Install & Load R Libraries - An Example (ISLR2)

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains an R script named 'first_R.R' with the following code:

```
1 # my first R script
2
3 # load libraries
4 library(ISLR2)
5
6 # variables and expressions
7 x <- 2
8 y <- 3
9 z <- x + y
10 z
11
12 # Option 1: load the auto dataset through ISLR2 library
13 auto <- Auto
```

The line `library(ISLR2)` is highlighted in yellow.
- Console:** Shows the output of the R session:

```
R version 4.6.1 (2026-06-24 ucrt) -- "Happy Hop"
Copyright (C) 2026 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64

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.

Natural language support but running in an English locale

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.

> install.packages("ISLR2")
```

The line `> install.packages("ISLR2")` is highlighted in yellow.
- Environment:** Shows the current environment is empty.
- Files:** Displays a file explorer view of the project directory, showing various files including R scripts, data files, and documentation.

Plan for Session 1

- Get started
 - Install RStudio
 - Create a project
 - Navigate RStudio
 - Install and load R packages
- Walk through chapter 2 lab from your textbook
- R programming basics (optional)
 - Expression and assignment
 - Basic data structures
 - Basic programming structures & functions

Tackle Coding Questions in Your 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 Section From Your Textbook

- Pure R script/code (.R files) in RStudio (Good choice for beginner)
- 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 Through Prep

- Textbook Resource Site
 - <https://www.statlearning.com/resources-second-edition>
- Pure R (.R file) in RStudio (good for beginner; 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 through chapter 2 lab from your 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	

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)
```

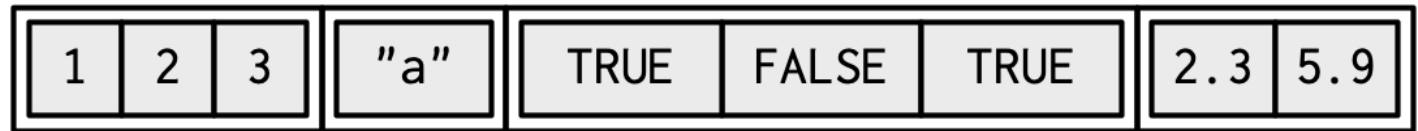
TRUE

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

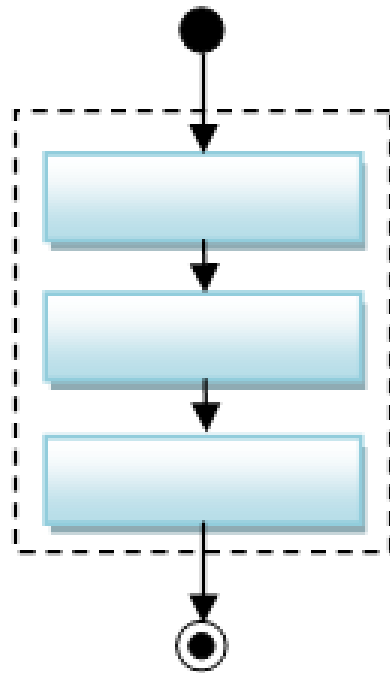
A Cousin to Data Frame - Tibble

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

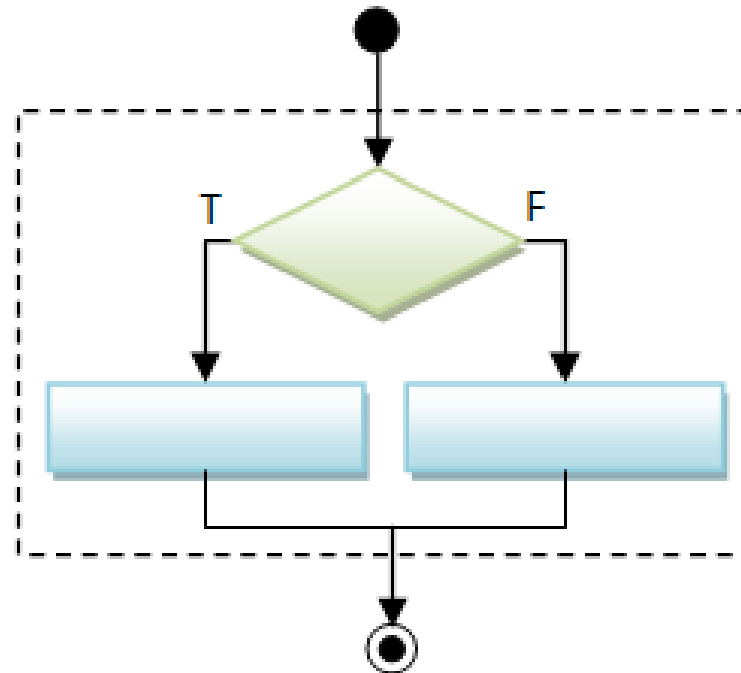
# create a tibble
tb1 <- tibble(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

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

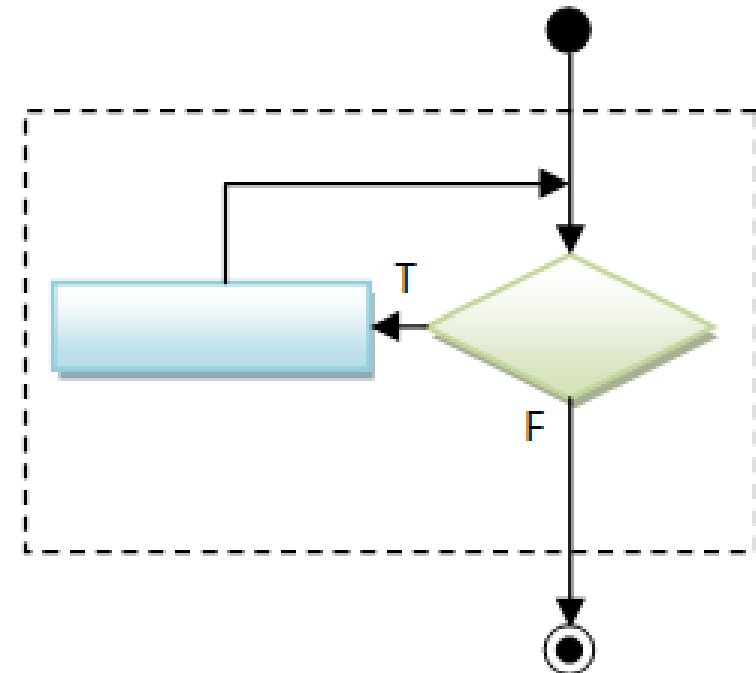
Programming Structure: Control Flows



Sequential



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...)

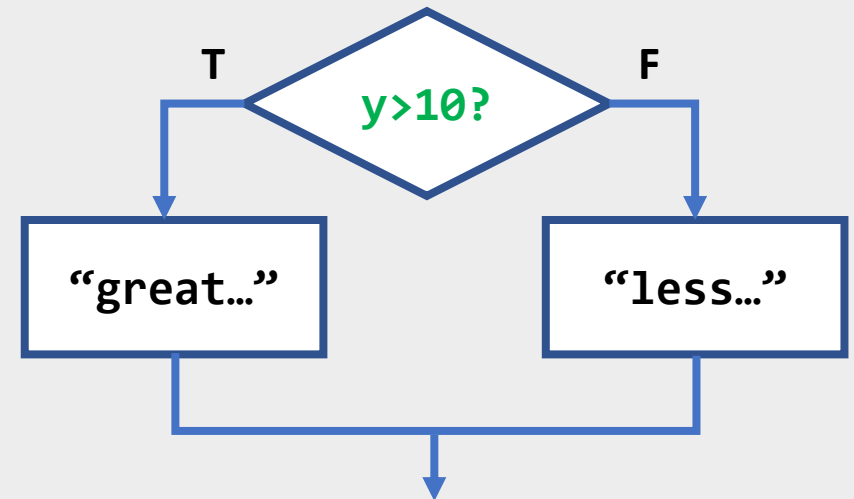
```
if (cond) {  
    # run here if cond is TRUE  
} else {  
    # run here if cond is FALSE  
}
```

```
# 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))
```

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 RSM358), in most case,
 - Data frame manipulation + sequential programming flow + modeling (using algorithms from R libraries developed by other people)

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

