

Rotman

INTRO TO R PROGRAMMING

R Tutorial (RSM358) – Part 1 / Session 1 & 2

September 20, 2022 Prepared by Jay Cao / [TDMDAL](https://tdmdal.github.io)

Website: <https://tdmdal.github.io/r-intro-2022-fall/>



Rotman School of Management
UNIVERSITY OF TORONTO

Plan for Session 1 & 2

- Intro to Intro (today)
 - What is R and what can R do?
 - Setup R
 - R learning road map and resources
- Get started with an example: Weighted Dice (today)
 - Let's code together!
- Take Stock & more (next week)
 - Expression and assignment
 - Basic data structures
 - Basic programming structures & functions
 - Turn ideas into code

What's R?

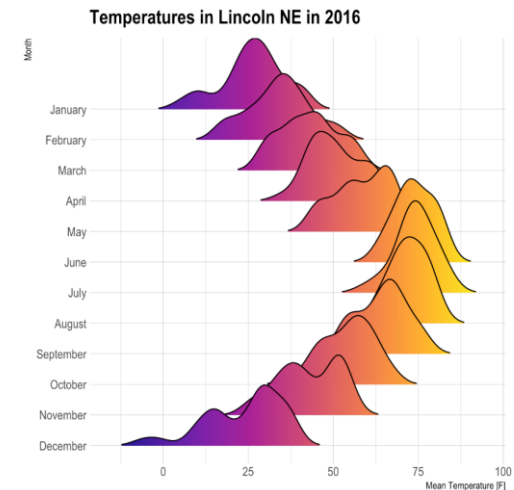
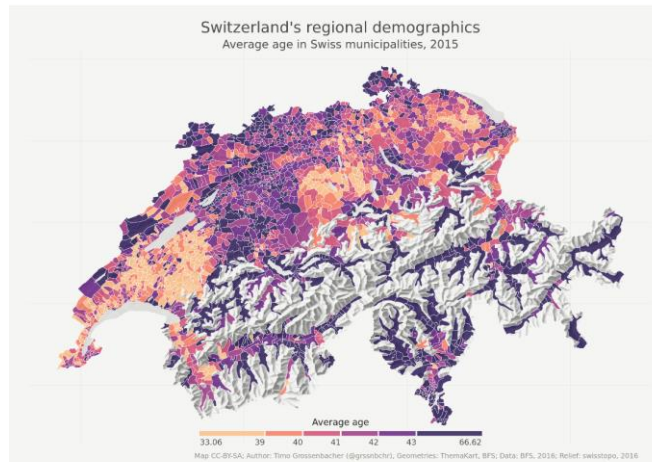
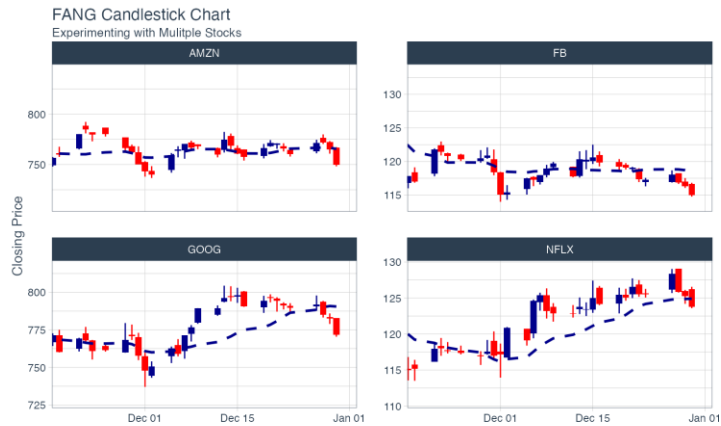
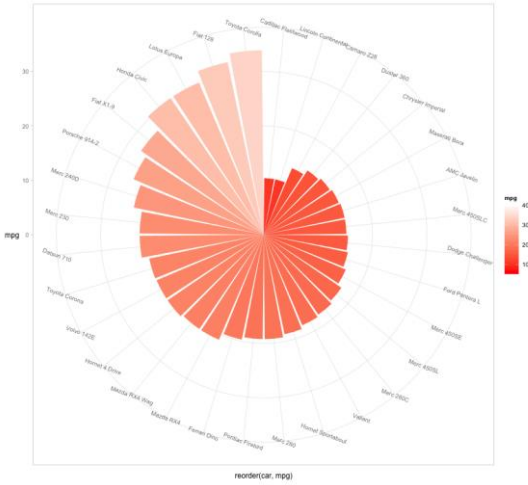
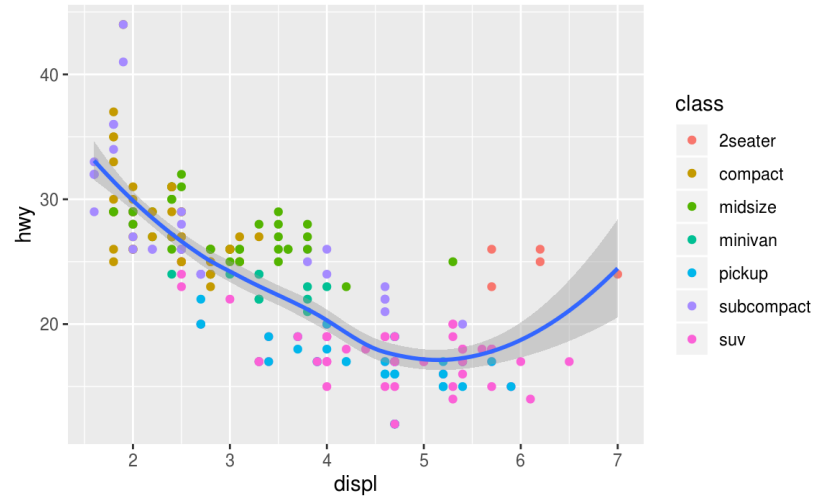
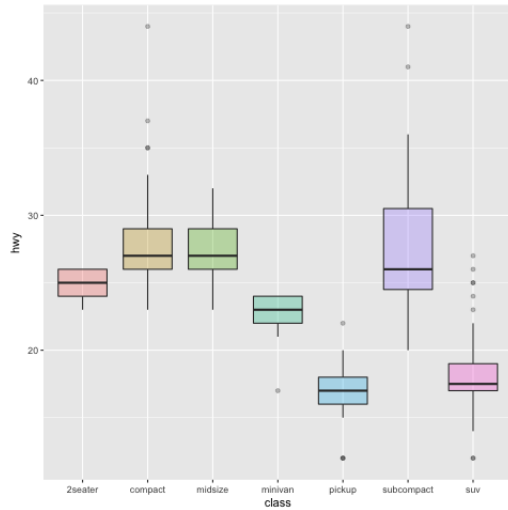


- R = a language + an eco-system
 - A free and open-source programming language
 - An eco-system of many high-quality user-contributed libraries/packages
- In the past R is mostly known for its statistical analysis toolkits
- Nowadays R is capable of (and very good at) many other tasks
 - Tools that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

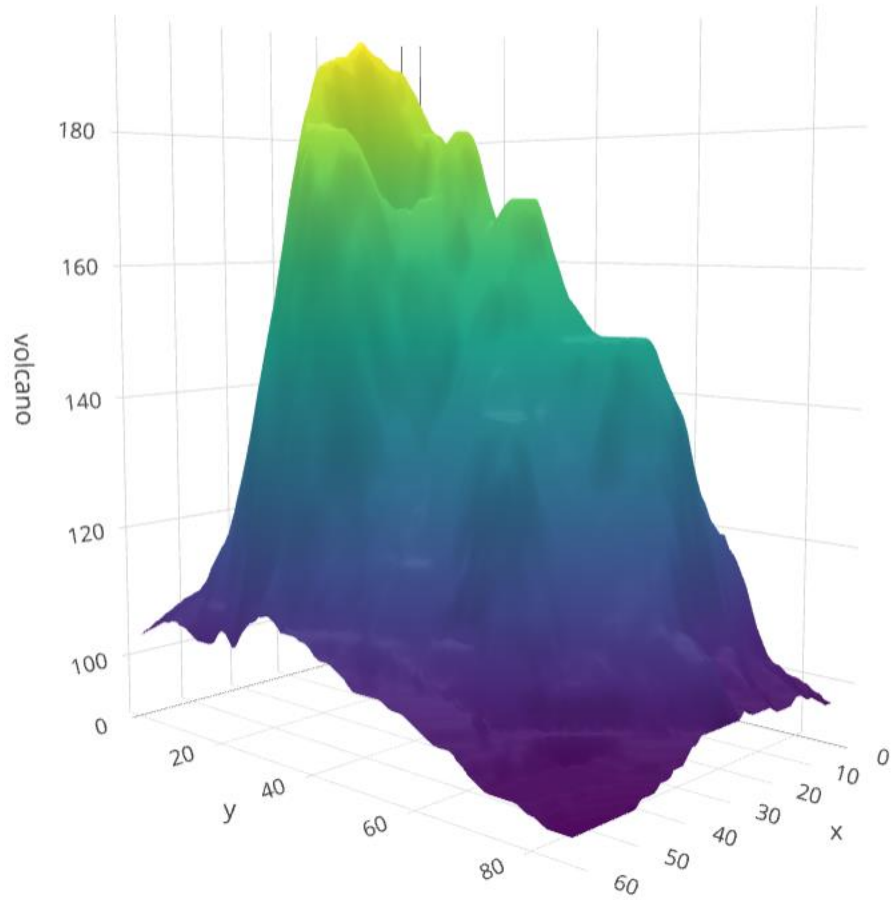
What can R do – Graphics (static ones)



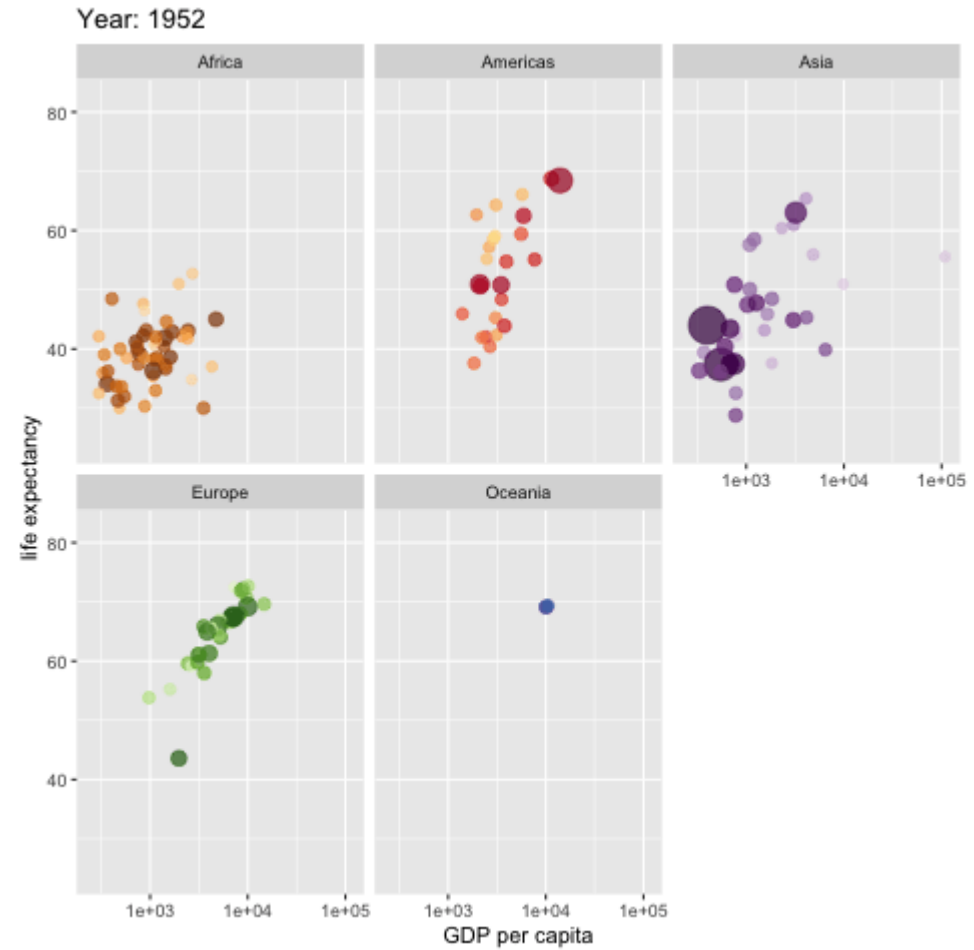
Ref: 1) <https://www.r-graph-gallery.com/>

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://gganimate.com/;](https://gganimate.com/)

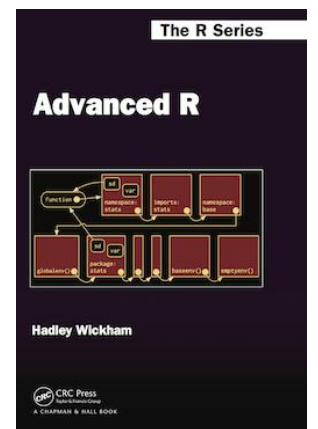
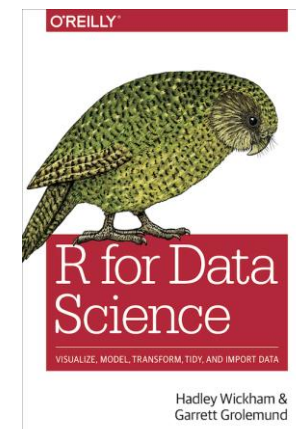
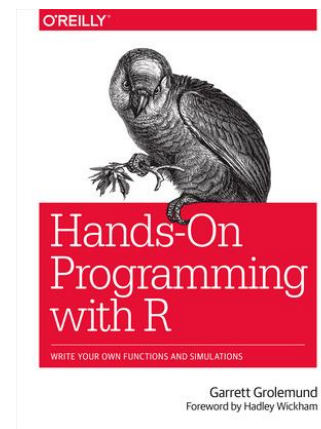
Setup R (Install R & its Coding Environment)

- R & **RStudio** on your 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>)


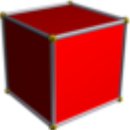

R Learning Road Map (From Zero to Hero)

- 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



Weighted Dice - Let's Get Started

- Goal: Simulate dice rolls and plot the distribution of the result
- Can handle difference kind of dice and the dice can be unfair
 - 4 faces (), 6 faces (), 8 faces (), etc.
- Can handle multiple rolls as one simulation
 - sum over the numbers rolled as 1 simulation result
- For example: a 6-face dice, 2 rolls as 1 simulation
 - 1st roll: 4; 2nd roll: 6; result for this simulation is $4 + 6 = 10$



Weighted Dice – Let's Code Together

- Learning by doing
 - Follow what I code
 - I will ask you to code variations of what I do too
- While working towards the weighted dice,
 - Get comfortable with RStudio
 - Learn some basic concepts of R

Plan for Session 1 & 2

- Intro to Intro (today)
 - What is R and what can R do?
 - Setup R
 - R learning road map and resources
- Get started with an example: Weighted Dice (today)
 - Let's code together!
- Take Stock & more (next week)
 - Expression and assignment
 - Basic data structures
 - Basic programming structures & functions
 - Turn ideas into code

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)
```

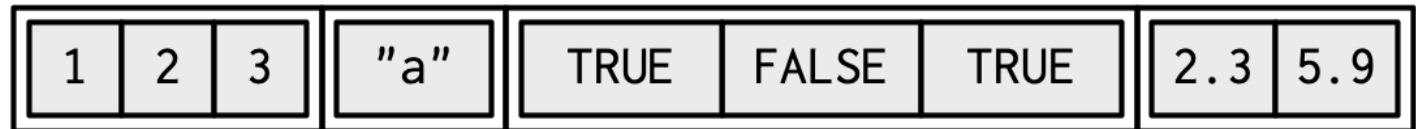
1.1	2.2	3.3
------------	------------	------------

```
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

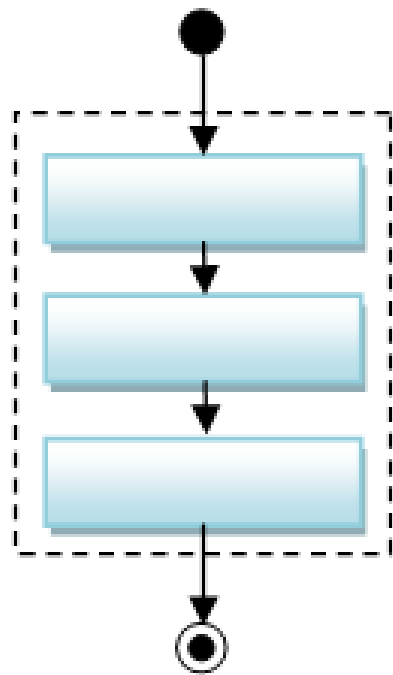
A Cousin to Data Frame - Tibble

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

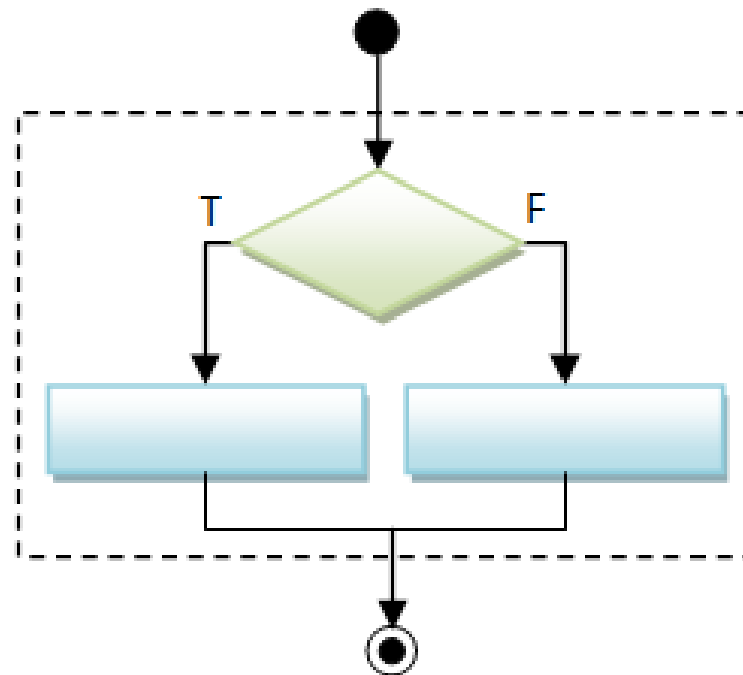
# create a tibble
tb1 <- tibble(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

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

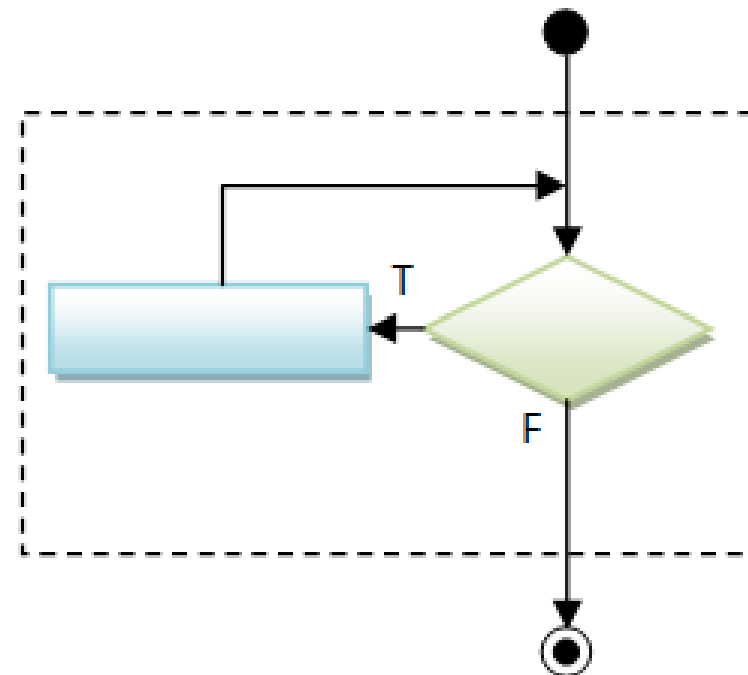
Programming Structure: Control Flows



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		

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))
```

Conditional (if...else...)

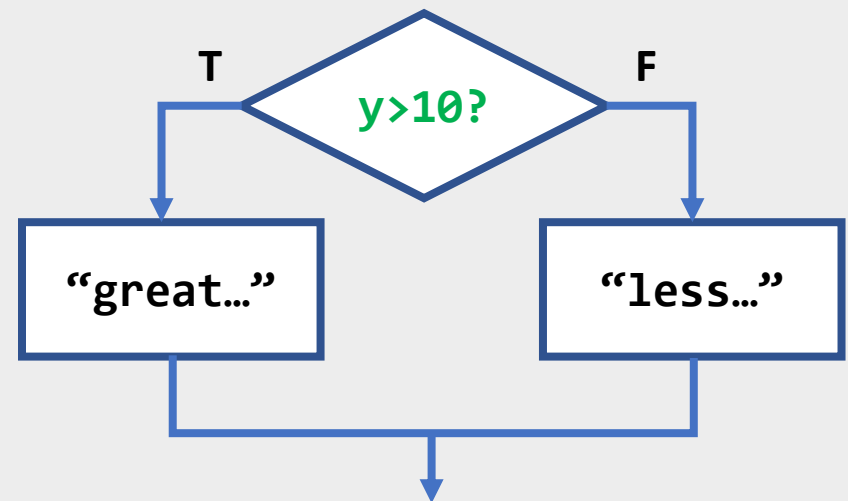
```
if (cond) {  
    # run here if cond is TRUE  
} else {  
    # run here if cond is FALSE  
}
```

```
# 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) # 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
 - Data Structure (vector, list, data frame, etc.)
 - Programming Structure (sequential, conditional, iterative)
 - 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.)