

Rotman

INTRO TO R

R Workshop

January 13, 2021 Prepared by Jay Cao / [TDMDAL](#)

Website: <https://tdmdal.github.io/r-intro-rbac-2021-winter/>



Rotman School of Management
UNIVERSITY OF TORONTO

Plan for Today (~2 hrs)

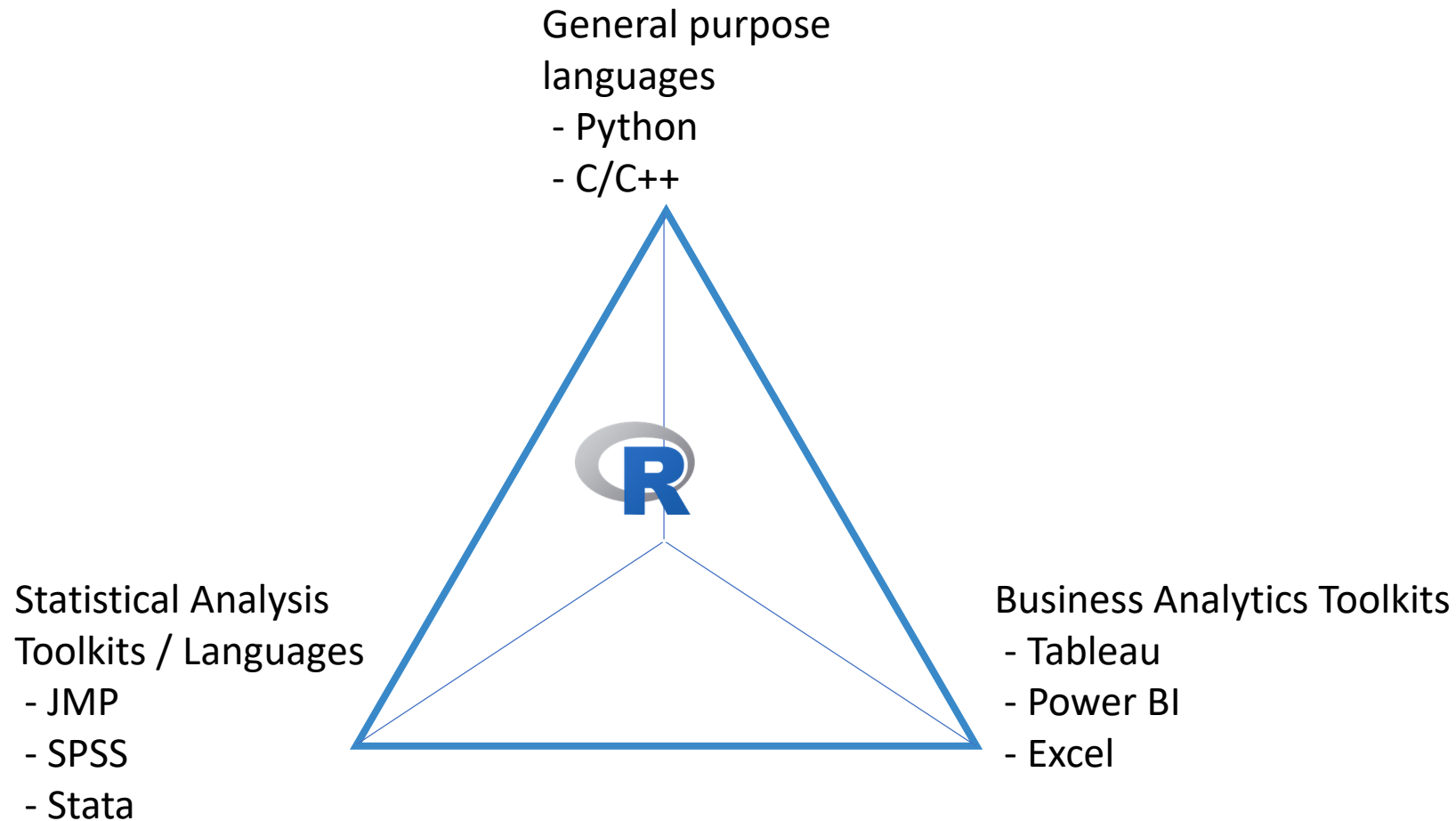
- **Intro to Intro**
 - What's R
 - Why learn R
 - Setup R
 - Three motivation examples
- Basics of R language
- Walk-through of a typical analysis workflow

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 pipeline
 - Tools for web tech...

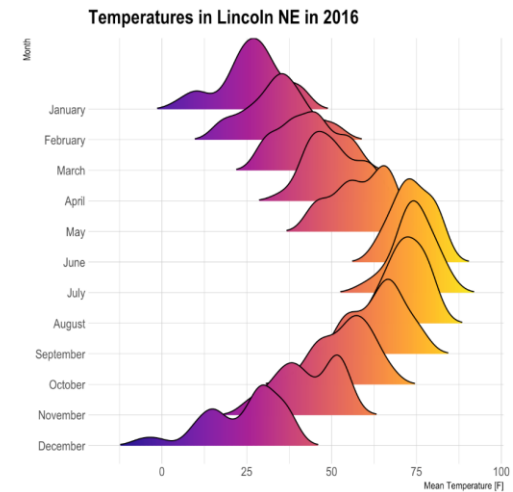
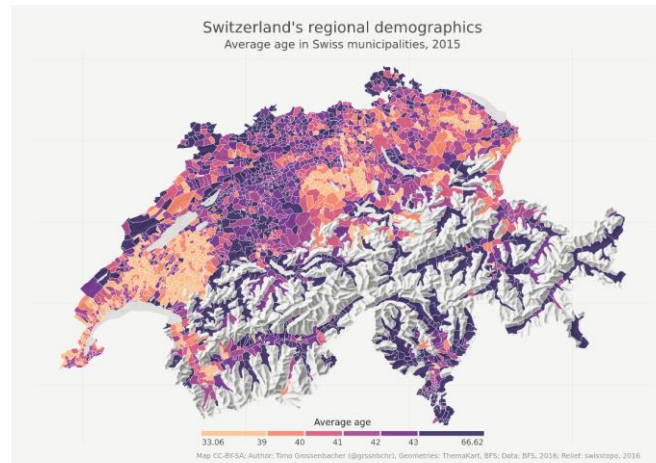
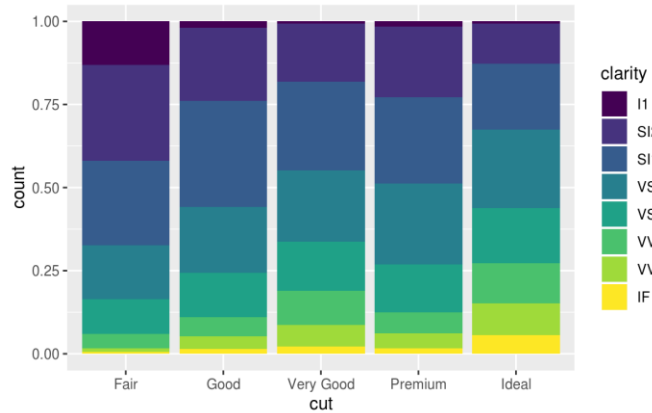
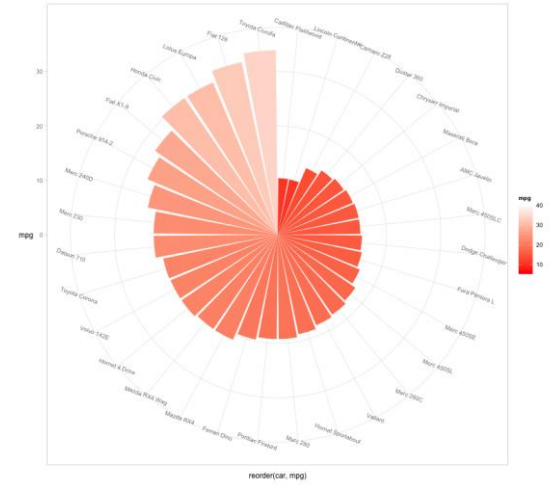
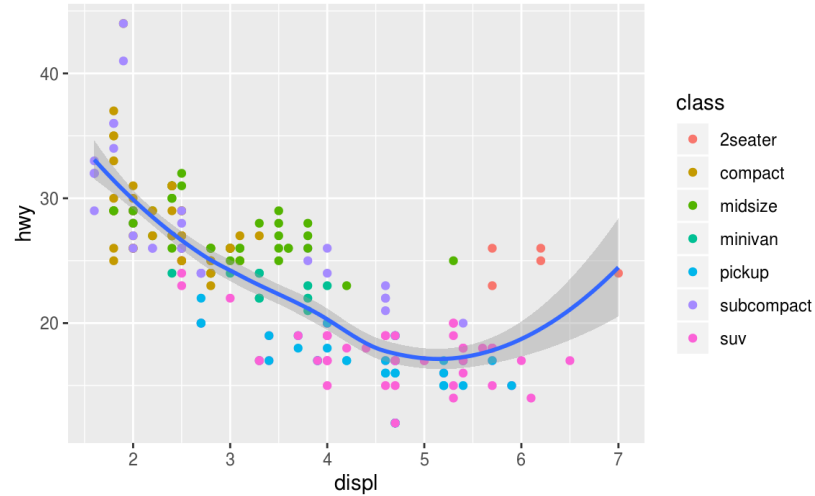
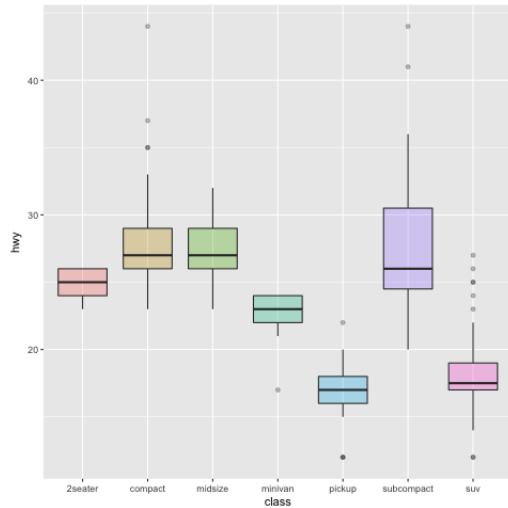
What's R?



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)

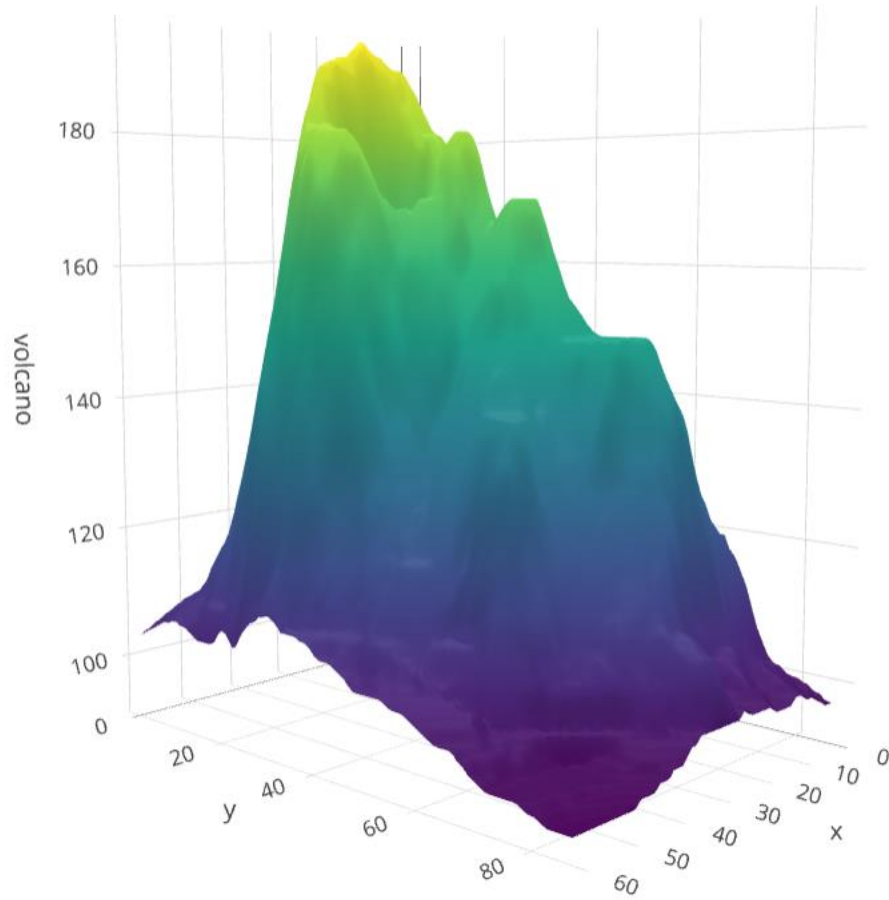


• • •

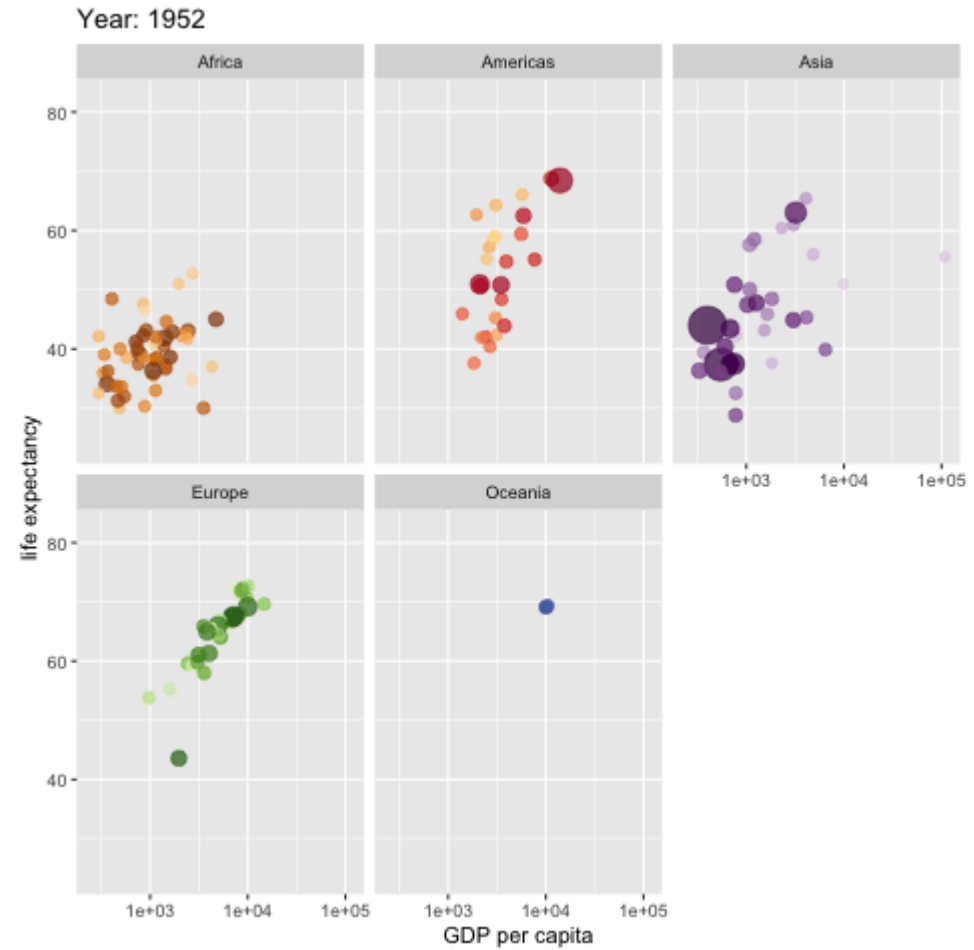
<https://www.r-graph-gallery.com/>

<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 (e.g. interface to [Keras](#) and [Tensorflow](#))
- 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)

Why learn R (What can R do for You)?

- Beyond Excel Data Analysis
 - I wish Excel could...
- Automate boring repeating tasks
 - e.g. collecting data from different sources daily
- Prototype ideas
 - e.g. trading strategy
- ...

Setup R

- R on your computer
 - Install R (<https://www.r-project.org/>)
 - Install RStudio (<https://rstudio.com/products/rstudio/download/>)
- R in Cloud (run R without installation)
 - RStudio Cloud (<https://rstudio.cloud/>)
 - Google Colab (<https://colab.to/r>)

What's RStudio?

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

graph_test.R x raw_shiny_v2.R x

```
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()
```

Environment History Connections Presentation x

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

Files Plots Packages Help Viewer

Zoom Export Publish

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

RStudio Cloud

The screenshot displays the RStudio Cloud web interface in a browser window. The address bar shows the URL `https://rstudio.cloud/spaces/112457/project/2046604`. The interface is divided into several panels:

- Left Sidebar:** Contains navigation links for 'Spaces', 'Your Workspace', 'R Intro' (selected), 'New Space', 'Learn' (with links to 'Guide', 'What's New', 'Primers', 'Cheat Sheets'), 'Help' (with links to 'Current System Status', 'RStudio Community'), and 'Info'.
- Top Panel:** Displays 'R Intro / Workshop 1' and the R version 'R 4.0.3'. It includes a menu bar (File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help) and a toolbar with icons for file operations and running code.
- Code Editor:** Shows a file named 'Untitled1' with a single line of code: `1`.
- Environment Panel:** Labeled 'Global Environment', it shows 'Environment is empty'.
- Files Panel:** Displays a file tree for the 'project' directory. It includes 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
- Console:** Displays 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.  
  
> |
```

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

Share

J

File Edit View Insert Runtime Tools Help

+ 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

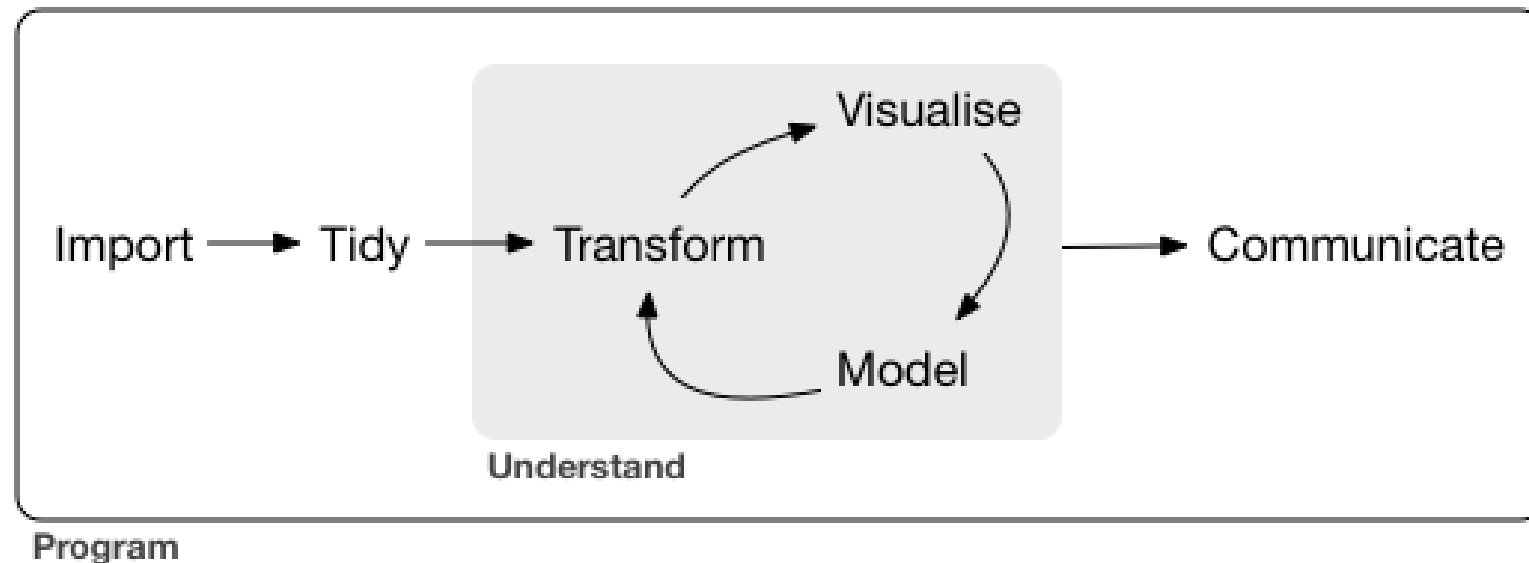
- A simple regression
- Twitter API
- Deep learning [“Hello World!”](#)



Keras



TensorFlow



Plan for Today

- Intro to Intro
- Basics of R language (minimum to get us started today)
 - Expressions and assignment
 - Data structure
 - Programming structure (see appendix for more)
- Walk-through of a typical analysis workflow

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)
```

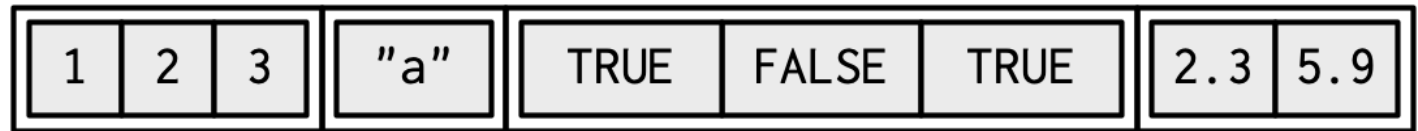
TRUE

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

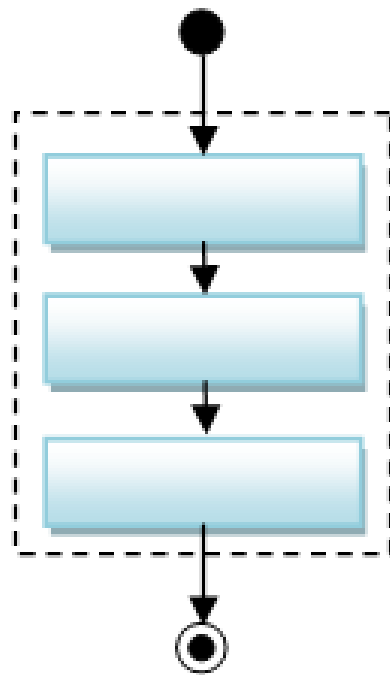
Tibble – A Cousin to Data Frame

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

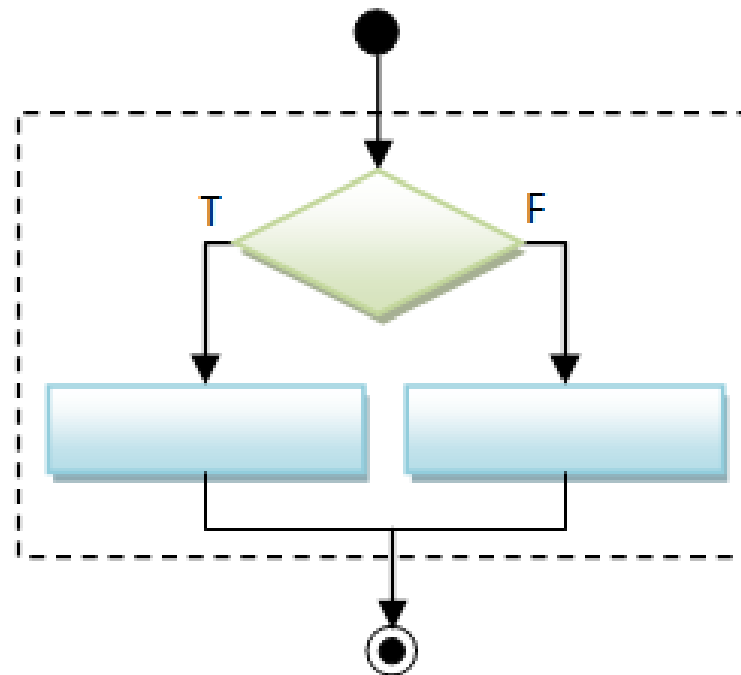
# create a tibble
tb1 <- data.frame(
  x = 1:3,
  y = letters[1:3],
  z = c(1.1, 2.2, 3.3)
)
```

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

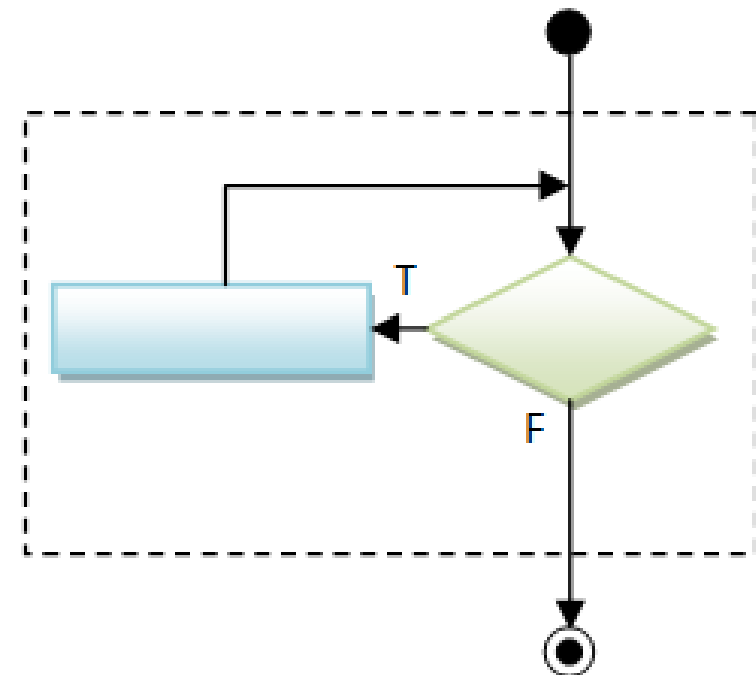
Programming Structure: Control Flows



Sequential



Conditional (Decision)



Loop (Iteration)

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))
```

Plan for Today

- Intro to Intro
- Basics of R language
- Walk-through of a typical analysis workflow
 - Import and manipulate data
 - Build models
 - Report results

Extending the regression example

- Manipulate data
 - Load data
 - Create new columns
 - Filter columns and rows
- Build models
 - Multiple linear regression
 - Regression with interactive terms
- 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 ([huxreg\(\)](#) from [huxtable](#) library)

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 – 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, ...)
```

e.g.

```
hprice_reg <- filter(hprice, price > 20000)
```

Data Manipulation: mutate()

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

e.g.

```
hprice_reg <- mutate(hprice_reg, lprice = log(price))
```

Data Manipulation: select()

```
select(my_dataframe, var1, ...)
```

e.g.

```
hprice_reg <- select(hprice_reg, lprice, rooms)
```

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)
```

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)
```


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)
```

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()`

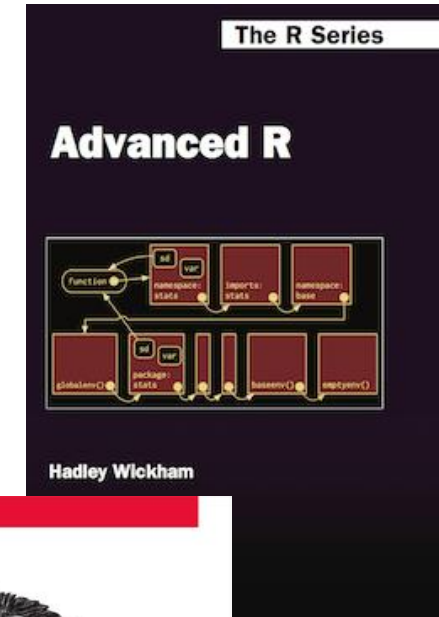
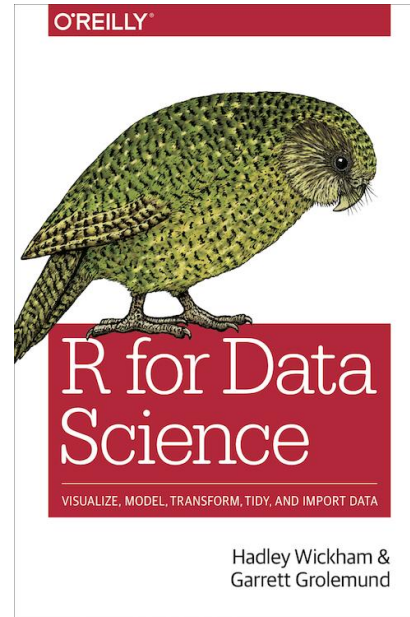
Report

- Summary table
 - [Summary for lm\(\)](#): `summary(my_model)`
- publication-ready table: [huxreg\(\)](#) from [huxtable](#) library

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

Free Learning Resources - Books

- [R for Data Science](#)
- [Advanced R](#)
- [Hands-On Programming with R](#)
- Check bookdown.org often



Free Learning Resources – Video Courses

- [RStudio Resources Site](#)
- Coursera
 - free for [UofT students](#) (also mostly free if you just audit the courses)
 - Search R and learn

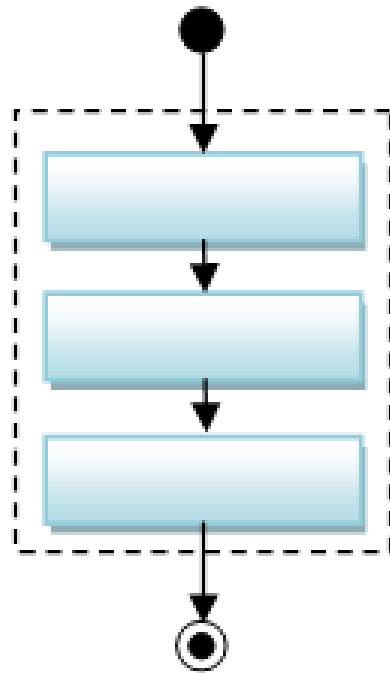
Free Learning Resources – Others

- [RStudio Education](#) ([Choose Your Learning Paths](#))
- [CRAN Task View](#)
- Twitter (a few seeds: [#rstat](#), [@hadleywickham](#), [@WeAreRLadies](#))

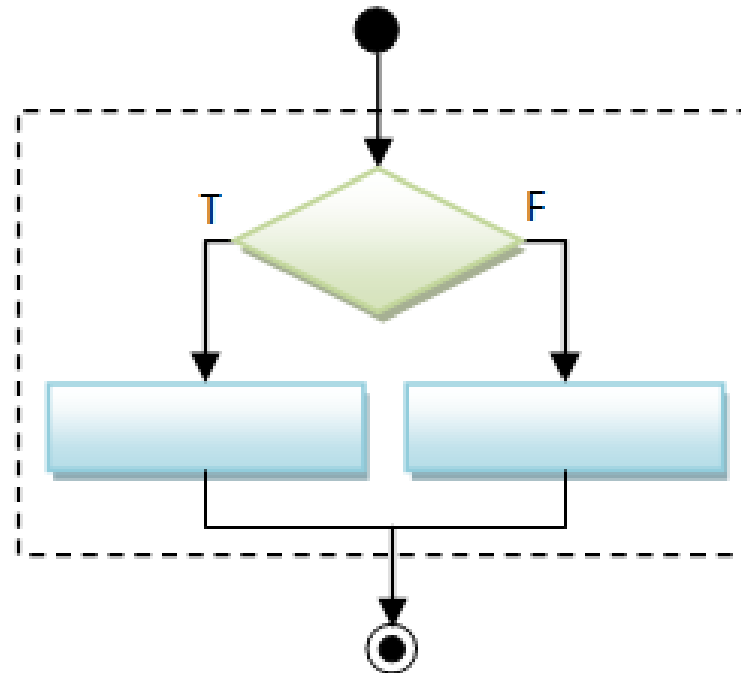
Plan for Today

- Intro to Intro
- Basics of R language
- Walk-through of a typical analysis workflow
- Appendix: more on programming structure

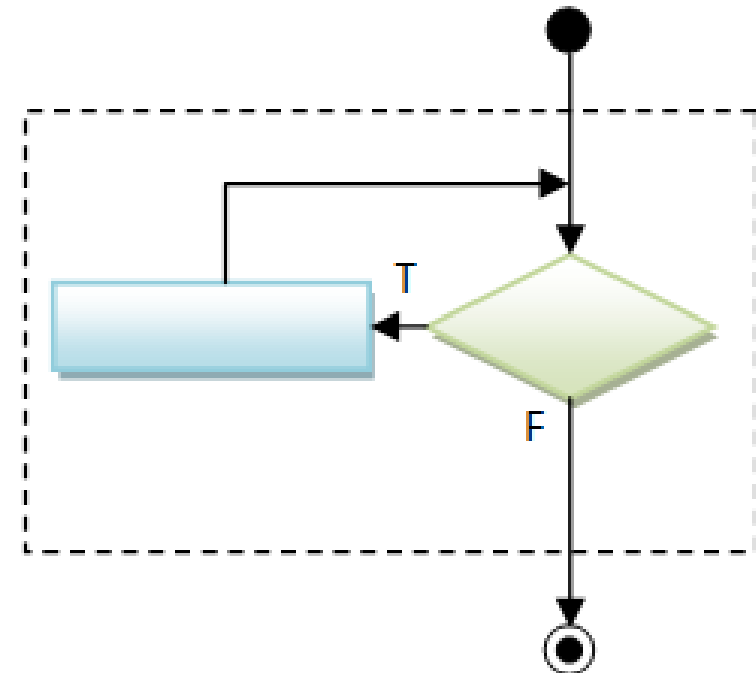
Programming Structure: Control Flows



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...)

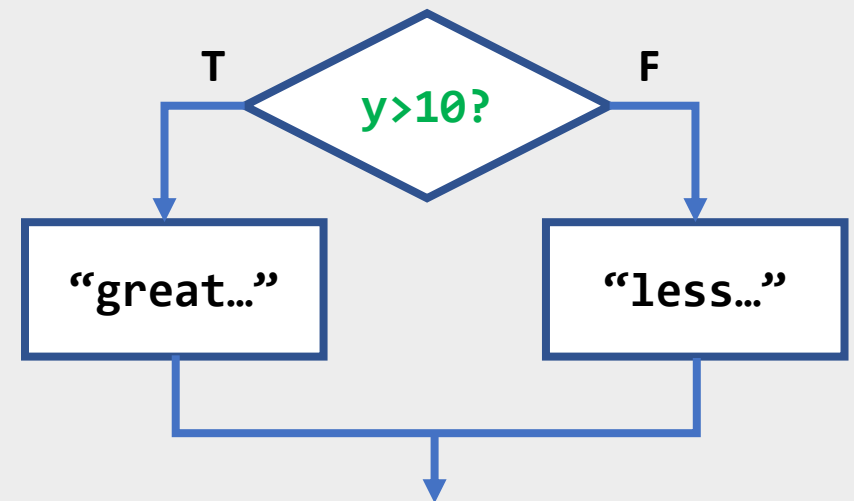
```
if (cond) {  
    # run here if cond is TRUE  
} else {  
    # run here if cond is FALSE  
}
```

```
# 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))
```