Rotman

INTRO TO R

R Workshop – Part 1 Overview & Basics / 2

March 10, 2025 Prepared by Jay Cao / TDMDAL

Website: https://tdmdal.github.io/r-workshop-202425-winter/

Rotman School of Management
UNIVERSITY OF TORONTO

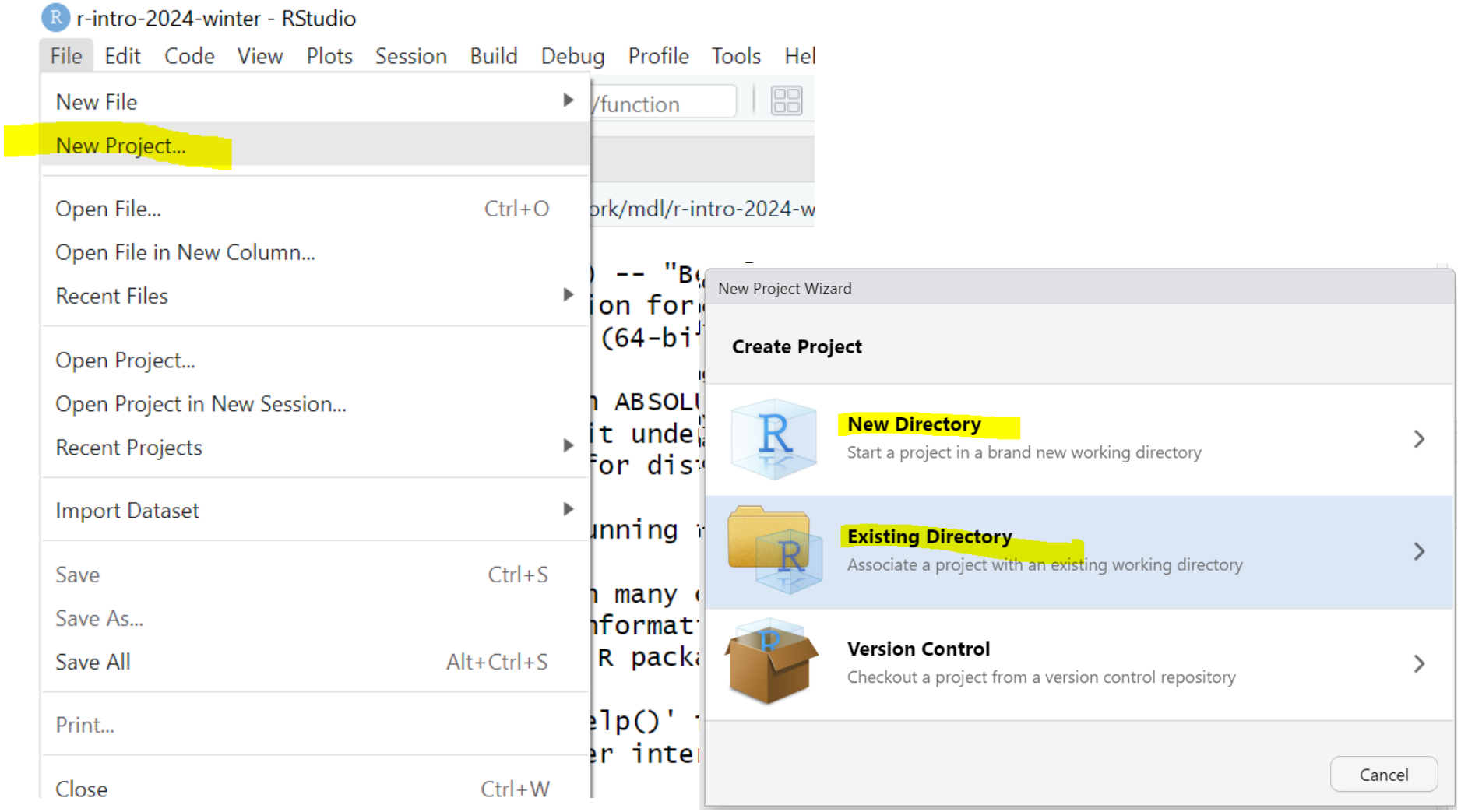
Plan for Part 1

- Intro

- Overview of R programing 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

Create New Project – A Good Practice



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!

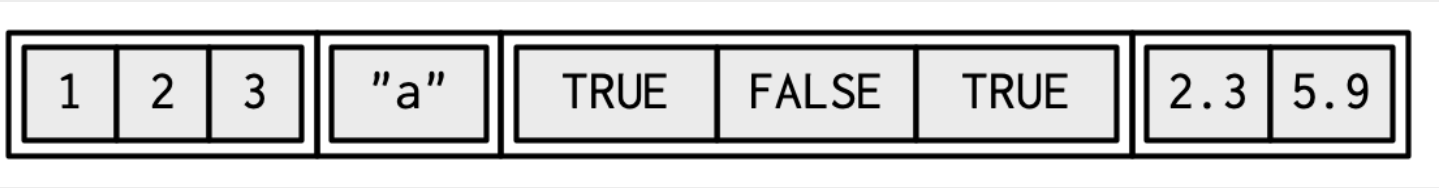
1	2	3

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)
)
```



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

```
# 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

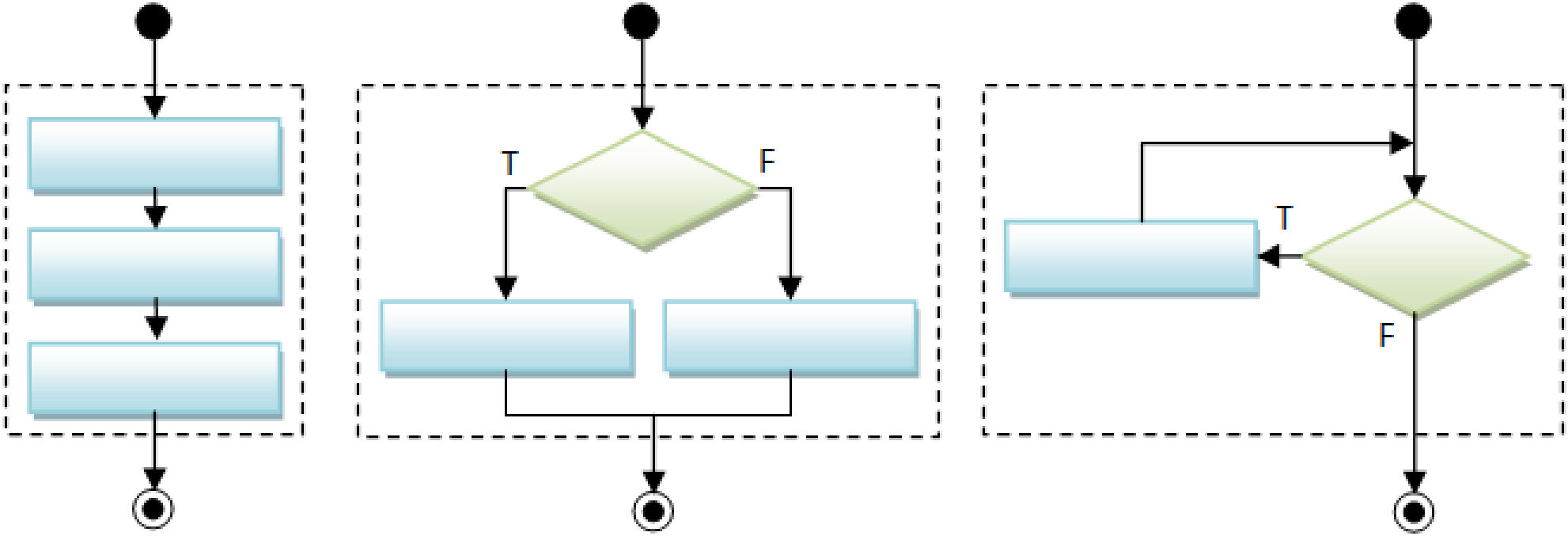
```
# load tibble library (part of tidyverse lib)
library(tibble)

# 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 | **Conditional (Decision)** | **Loop (Iteration)**

Today

Learn on your own (See Appendix)

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

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

```
# 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, in most case,
  - Data frame manipulation + sequential programming flow + modeling (using algorithm already implemented by others)

Plan for Part 1

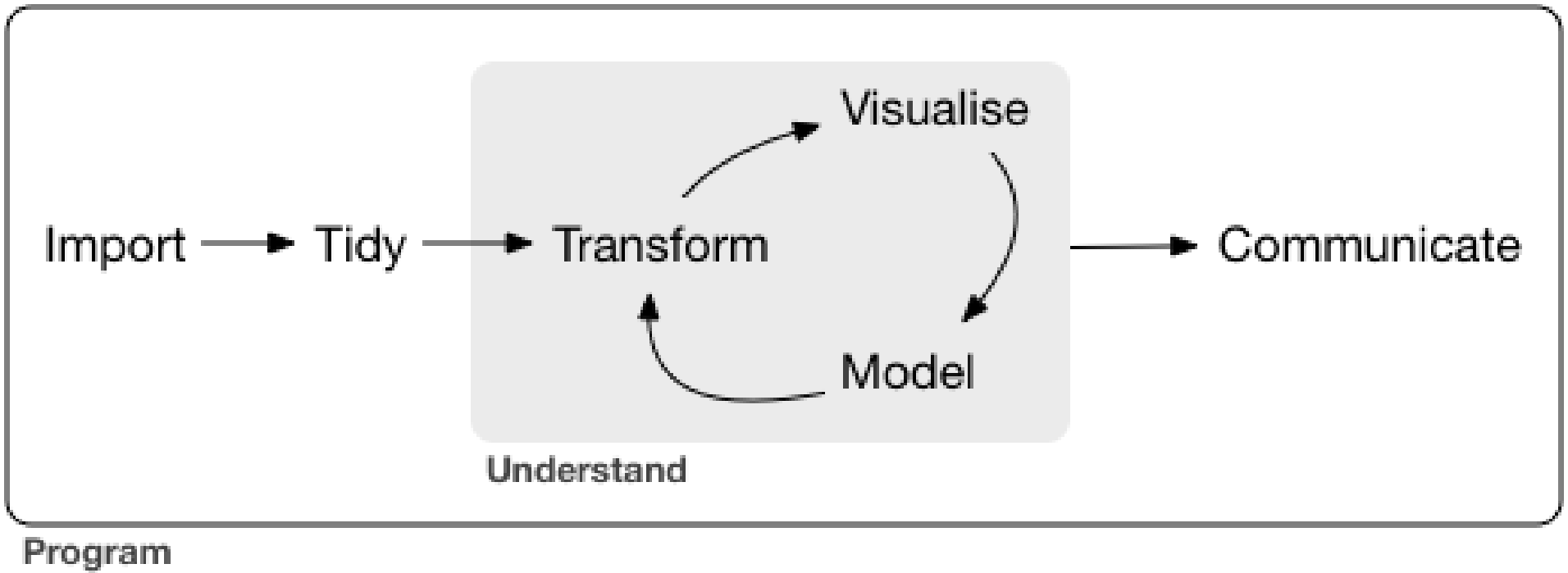
- Intro

- Overview of R programing 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



https://r4ds.had.co.nz/introduction.html

An Example: Housing Price & Clean Air

- 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

```
Obs:   506

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     accessibiliy 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 linear regression analysis

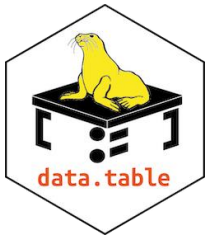
- The “modern” way
  - Use specialized packages to manipulate data and assist modeling tasks
    - Data are stored in improved data structures (mostly compatible with base R DSs)
    - Often better and consistent syntax/api across tasks than base R functions
  - This is what we will focus on

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

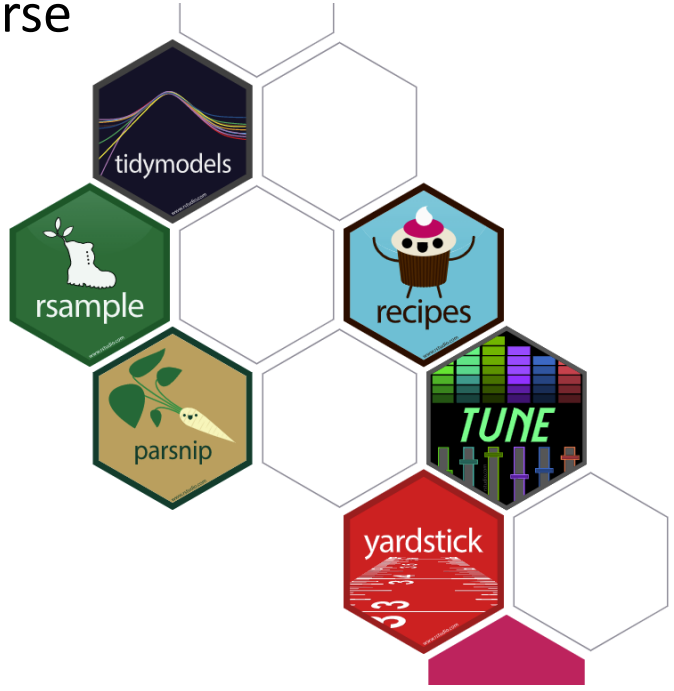


R Packages: Many choices, which one to use

- 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/Posit 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")
```

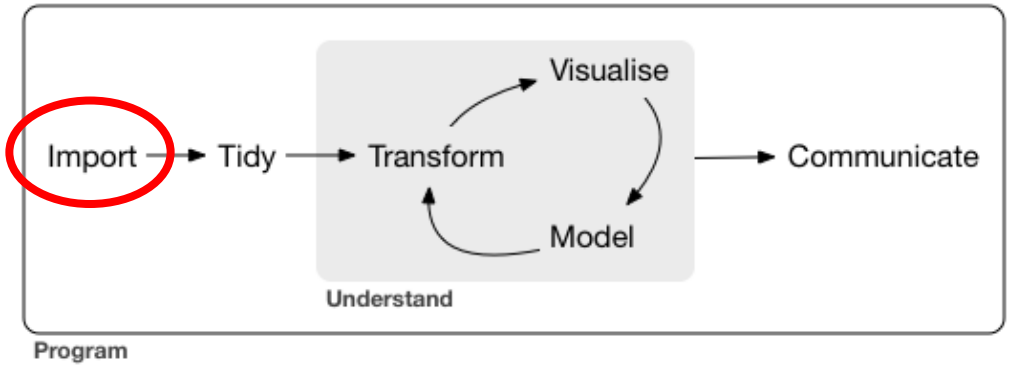
- Load an R library (before you use a library)

```
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`

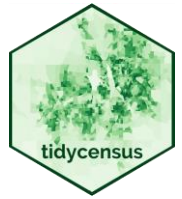
Load Data – Other file formats and sources

- readxl for Excel sheets
- haven for SPSS, Stata and SAS data
- jsonlite for JSON
- xml2 for XML
- httr2 for web APIs
- rvest for web scraping
- DBI for connecting to DataBase engine
- bigrquery for connect to Google bigquery
- …

Load Data – Financial & Census Dataset

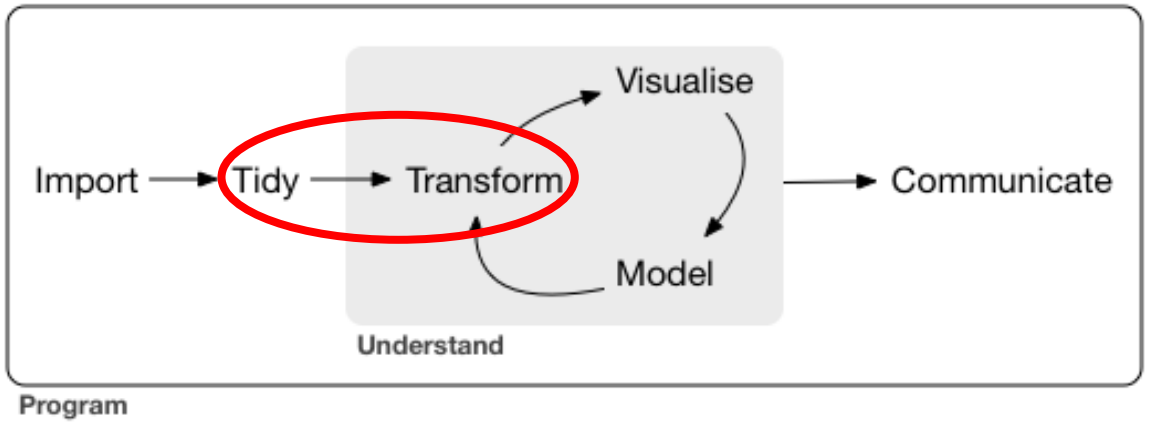
- `tq_get()` from tidyquant library 

  - 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
- tidycensus for US census data and cancensus for Canadian census data
- a few others (try to look for them yourselves…) 

Data Frame Manipulation: `dplyr` Basics

- Filter observations (rows): **`filter()`**
- Create new variables (columns): **`mutate()`**
- Select variables (columns): **`select()`**



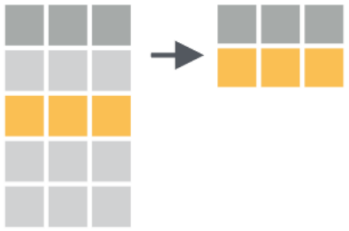
Ref. dplyr data wrangling cheat sheet

Data Frame Manipulation: dplyr Basics

- Filter observations (rows): **filter()**

filter(my_dataframe, condition1, …)

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



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

mutate(my_dataframe, new_var1 = *expression1*, …)

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

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

select(my_dataframe, var1, …)

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

Ref. dplyr data wrangling cheat sheet

Data Frame Manipulation – Base R vs dplyr

Operation	Base R	dplyr

Filter rows based on conditions	`df[which(x), , drop = FALSE], or`
	
	`subset()`
		`filter(df, x)`

Create a new column variable (from other column variables)	`df$z <- df$x + df$y, or`
	
	`df["z"] <- df["x"] + df["y"], or`
	
	`transform()`
		`mutate(df, z = x + y)`

Select column variables	`df[c("x", "y")], or`
	
	`subset()`
		`select(df, x, y)`

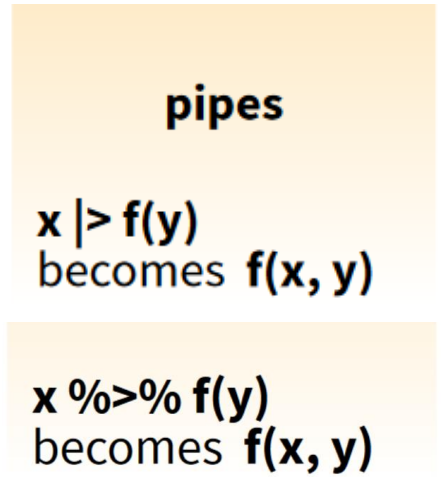
…	…	…

Source: https://dplyr.tidyverse.org/articles/base.html

Data Manipulation: Data Pipe (%>% or |>)

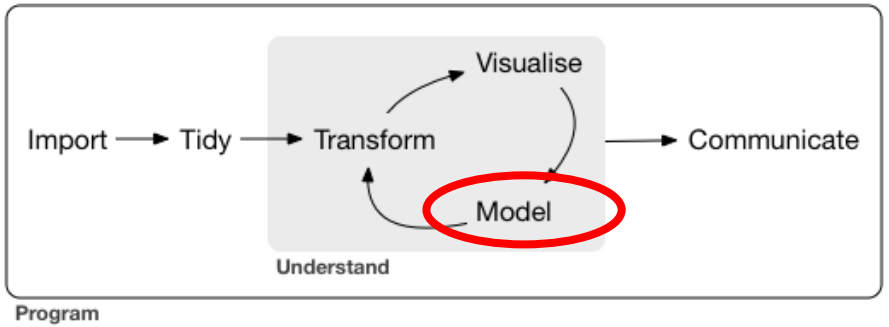
```
hprice_reg <- filter(hprice, price > 20000)
hprice_reg <- mutate(hprice_reg, lprice = log(price))
hprice_reg <- select(hprice_reg, lprice, rooms)
```

```
hprice_reg <- hprice %>%
  filter(price > 20000) %>%
  mutate(lprice = log(price)) %>%
  select(lprice, rooms)
```



Ref. dplyr data wrangling cheat sheet; pipe and |> vs %>%;

Regression



- Multiple regressions: `lm()` from `stats` library in base R

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

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

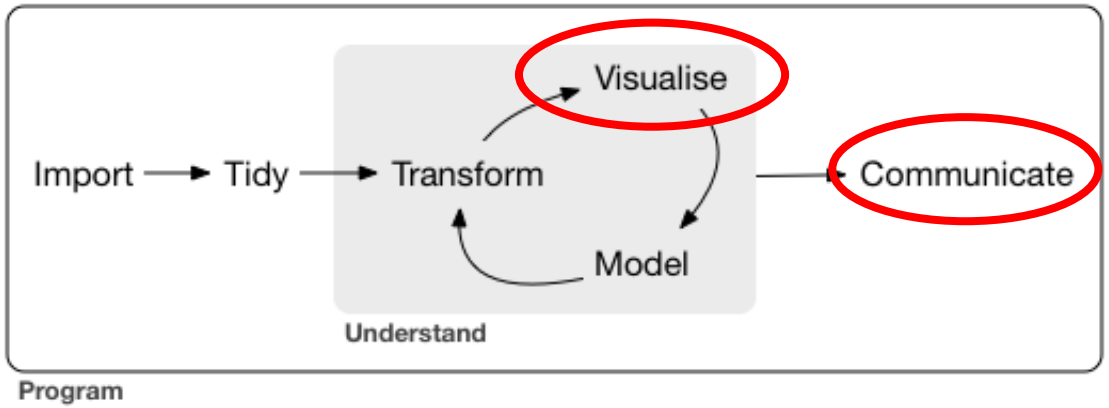
```
my_model <- lm(y ~ x1 + x2 + I(x1 * x2), data)
```

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

- Regression result summary: `summary()`

Ref 1. https://www.econometrics.blog/post/the-r-formula-cheatsheet/
Ref 2. https://stat.ethz.ch/R-manual/R-devel/library/stats/html/formula.html

Report



- Summary table
  - Summary for lm(): `summary(my_model)`

- publication-ready table: `huxreg()` from `huxtable` library

```
huxreg(my_model1, my_model2, …)
```

Ref. https://hughjonesd.github.io/huxtable/huxreg.html

Plan for Part 1

- Intro

- Overview of R programing and Data Science
  - Basics of R programming
  - Data science with R

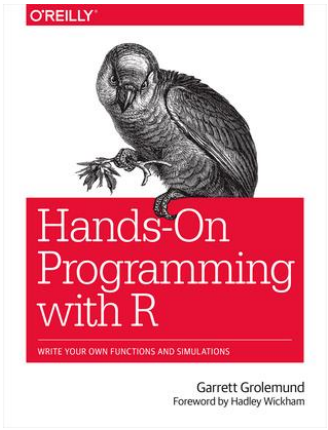
- Learning Road Map and Resources

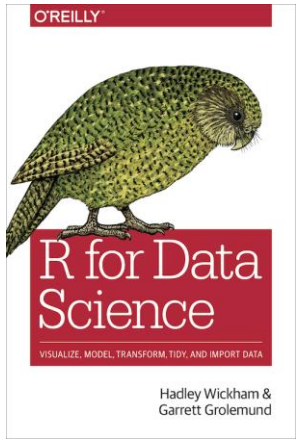
R Learning Road Map (From Zero to Hero)

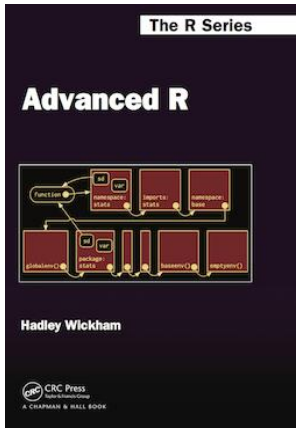
- Step 1. Basic R programming skills (Beginner)
  - Data and programming structure; how to turn an idea into code;
  - Book: Hands-On Programming with R

- Step 2. R Data Science skills (Intermediate)
  - Data wrangling, basic modeling, and visualization/reporting; Best practice;
  - Book: R for Data Science (1st ed); 2nd ed

- 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
- Talks and tutorials at posit::conf 2024
- 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)
- X/Twitter or bluesky
  - a few seeds: #rstat, @hadley.nz, @posit.co, @WeAreRLadies)

Appendix

- Programming Structure Continued
  - Conditional
  - Iteration

Conditional (if…else…)

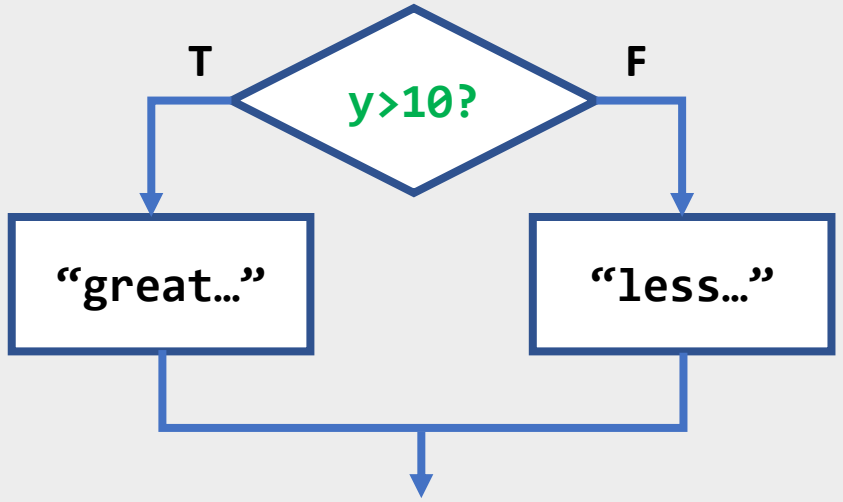
```
if (cond) {
  # run here if cond is TRUE
} else {
  # run here if cond is FALSE
}
```

```
# 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)
```