ISA 444: Business Forecasting

02: Introduction to R

Fadel M. Megahed, PhD

Endres Associate Professor
Farmer School of Business
Miami University

@FadelMegahed
fmegahed
fmegahed@miamioh.edu
Automated Scheduler for Office Hours

Spring 2023

Quick Refresher from Last Class

- ✅ Describe **course motivation** and **structure**.
- ✅ Explain the differences between **cross sectional**, **time series** and **panel** datasets.
- ✅ Describe the **components of time series** datasets.
- ✅ Explain the **forecasting steps**.

Learning Objectives for Today's Class

- Describe the syntax, data types, and data structures in R.
- Access the help for R functions (each help file has the following components: Description, usage, arguments, value, and examples).
- Utilize the project workflow in R and create R script.
- Access, subset, and create `ts()` objects in R.

Learning R (Any Programming Language)

- 🗣 **Get hands dirty**!!
- 📖 Documentation! Documentation! Documentation!
- 🔎 (Not surprisingly) Learn to Google: what that error message means (I G a lot 😂)

Source: Slide is based on Kia Ora's How I Learn a Technology.

The RStudio Interface, Setup and a Project-Oriented Workflow for your Analysis

RStudio Interface

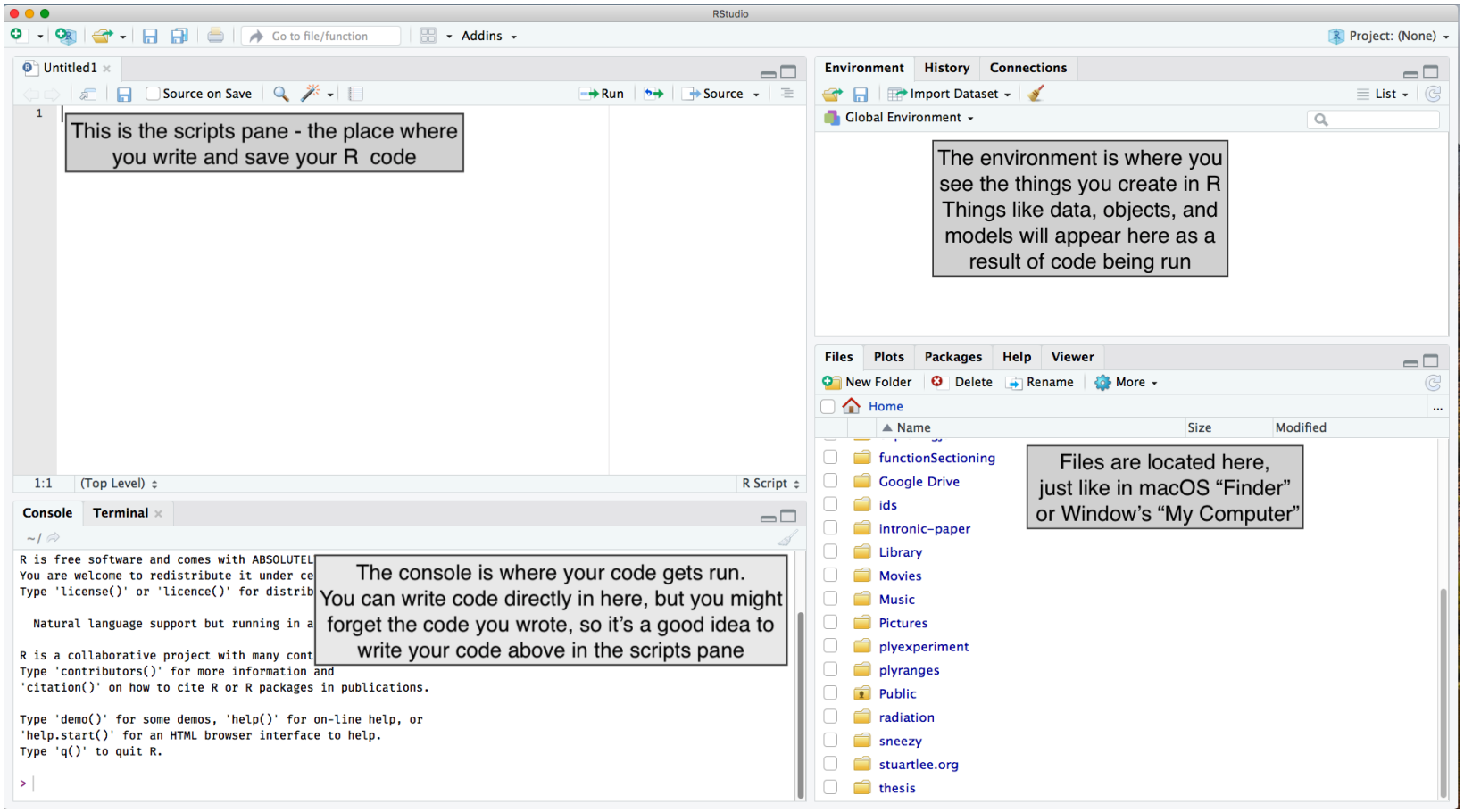
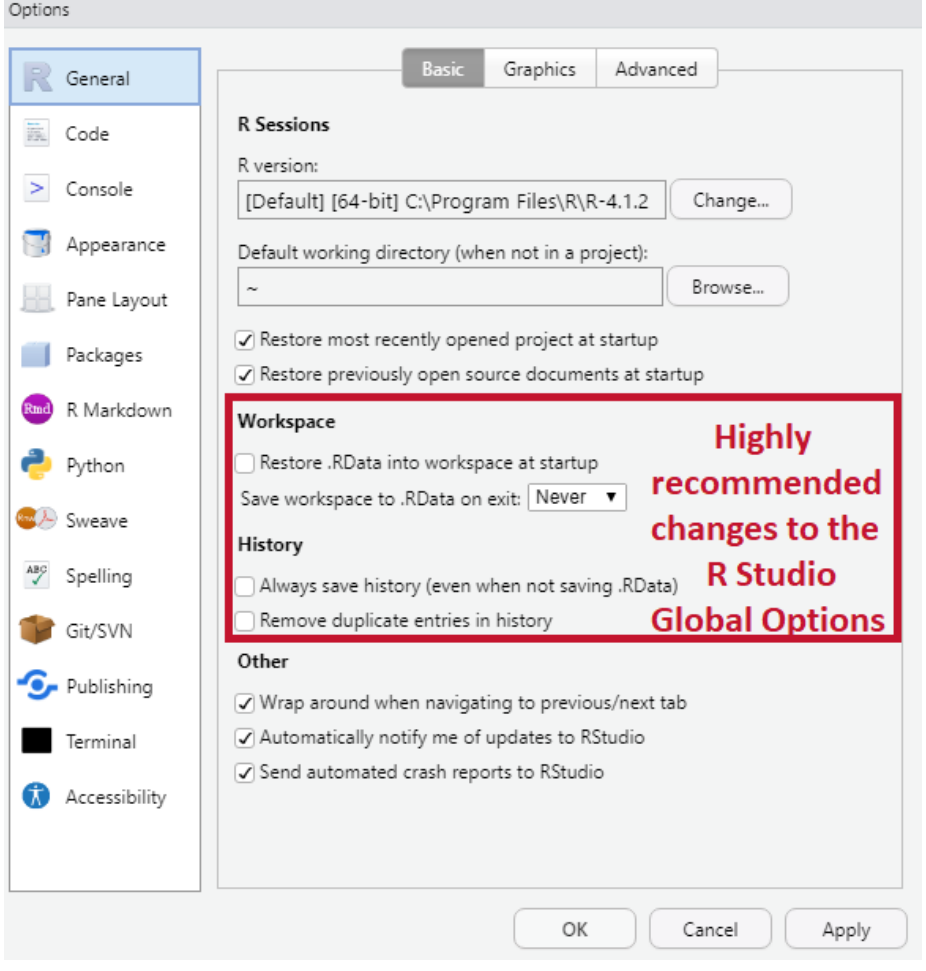


image credit: Stuart Lee

Setting up RStudio (do this once)

Go to **Tools** > **Global Options**:



Uncheck `Workspace` and `History`, which helps to keep R working environment fresh and clean every time you switch between projects.

What is a Project?

- Each university course is a project, and get your work organised.
- A self-contained project is a folder that contains all relevant files, for example my `ISA 444/` 📂 includes:
  - `isa444.Rproj`
  - `lectures/`
    - `01_introduction/`
      - `01-intro.Rmd`, etc.
    - `02_introduction_to_r/`
      - `02_intro_r.Rmd`, etc.
- All working files are **relative** to the **project root** (i.e. `isa444/`).
- The project should just work on a different computer.

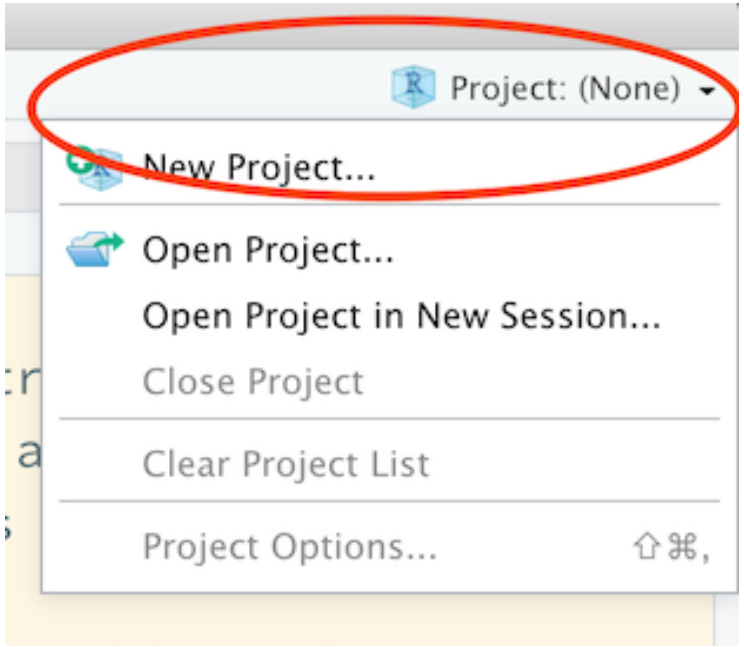
Lets Create a `.Rproj` for Our Course

02:00

- Click the **Project** icon on the top right corner

- **New Directory**/**Existing Directory** > **New Project** > **Create Project**

- Open the project



101: Syntax, Data Types, Data Structures and Functions

Coding Style

> Good coding style is like correct punctuation: you can manage without it, butitsuremakesthingseasiertoread.
> -- The tidyverse style guide

R style guide

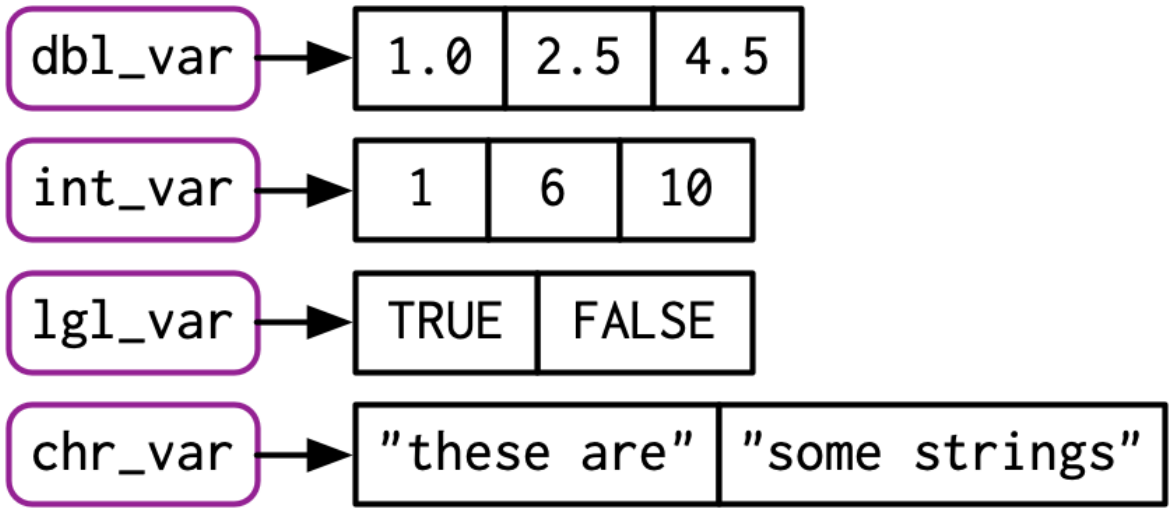
✅ `snake_case`

❌ `camelCase` (Javascript)

❌ `PascalCase` (Python)

Source: Slide is based on Earo Wang's STAT 220 Slides

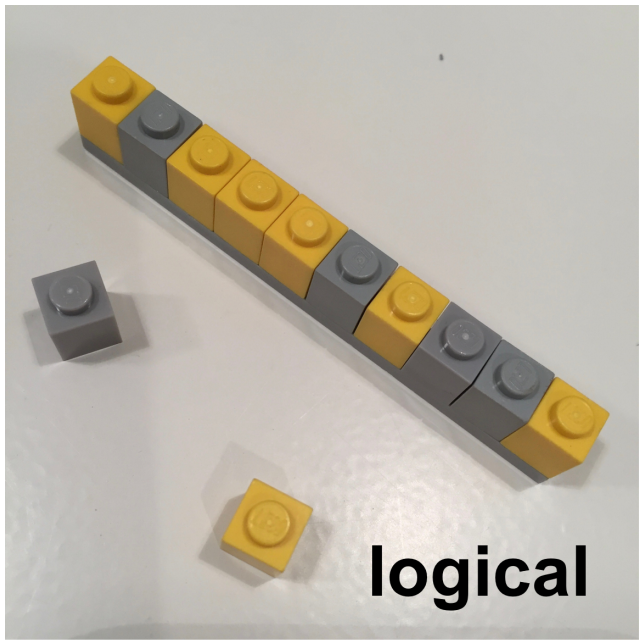
Data Types: A Visual Introduction

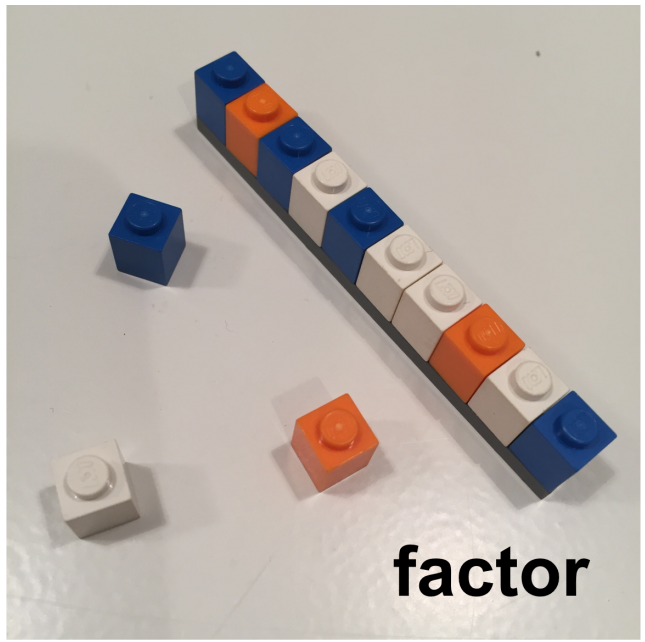


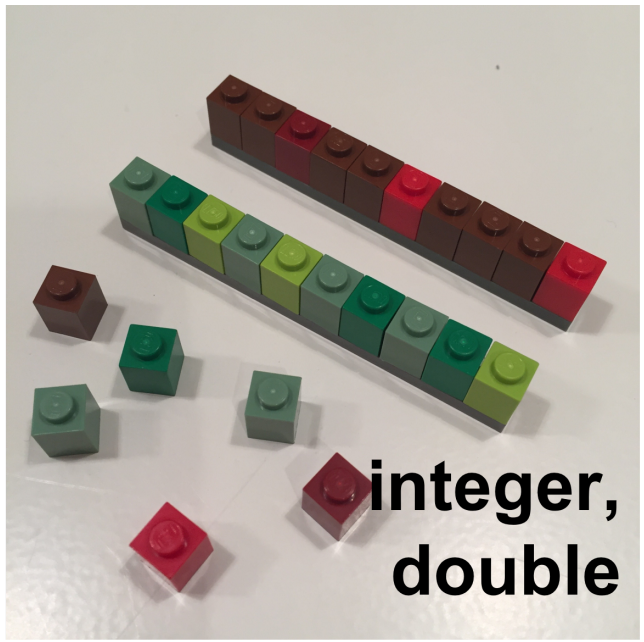
- To check the **type of** an object in R, you can use the function `typeof`.

Source: The image is from Hadley Wickham's Advanced R: Chapter 3 on Vectors

Data Types: A Visual Introduction







A visual representation of different types of atomic vectors

Source: The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

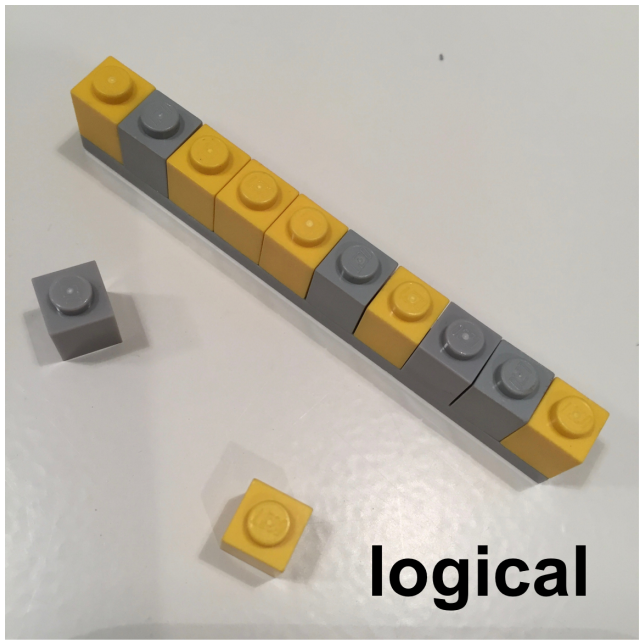
Data Types: Formal Definitions

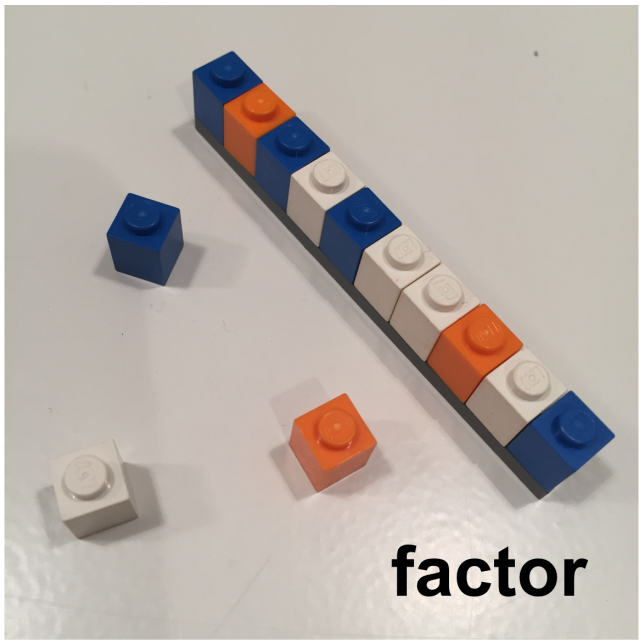
Each of the four primary types has a special syntax to create an individual value:

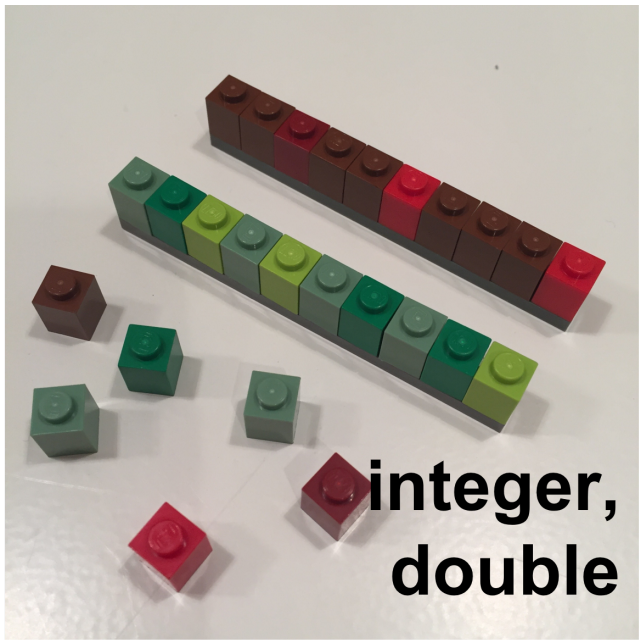
- Logicals can be written in full (`TRUE` or `FALSE`), or abbreviated (`T` or `F`).
- Doubles can be specified in decimal (`0.1234`), scientific (`1.23e4`), or hexadecimal (`0xcafe`) form.
  - There are three special values unique to doubles: `Inf`, `-Inf`, and `NaN` (not a number).
  - These are special values defined by the floating point standard.
- Integers are written similarly to doubles but must be followed by `L`(`1234L`, `1e4L`, or `0xcafeL`), and can not contain fractional values.
- Strings are surrounded by `"` (e.g., `"hi"`) or `'` (e.g., `'bye'`). Special characters are escaped with `\` see `?Quotes` for full details.

Source: The content of this slide is verbatim from Hadley Wickham's Advanced R: Chapter 3 on Vectors

Data Structures: Atomic Vector (1D)



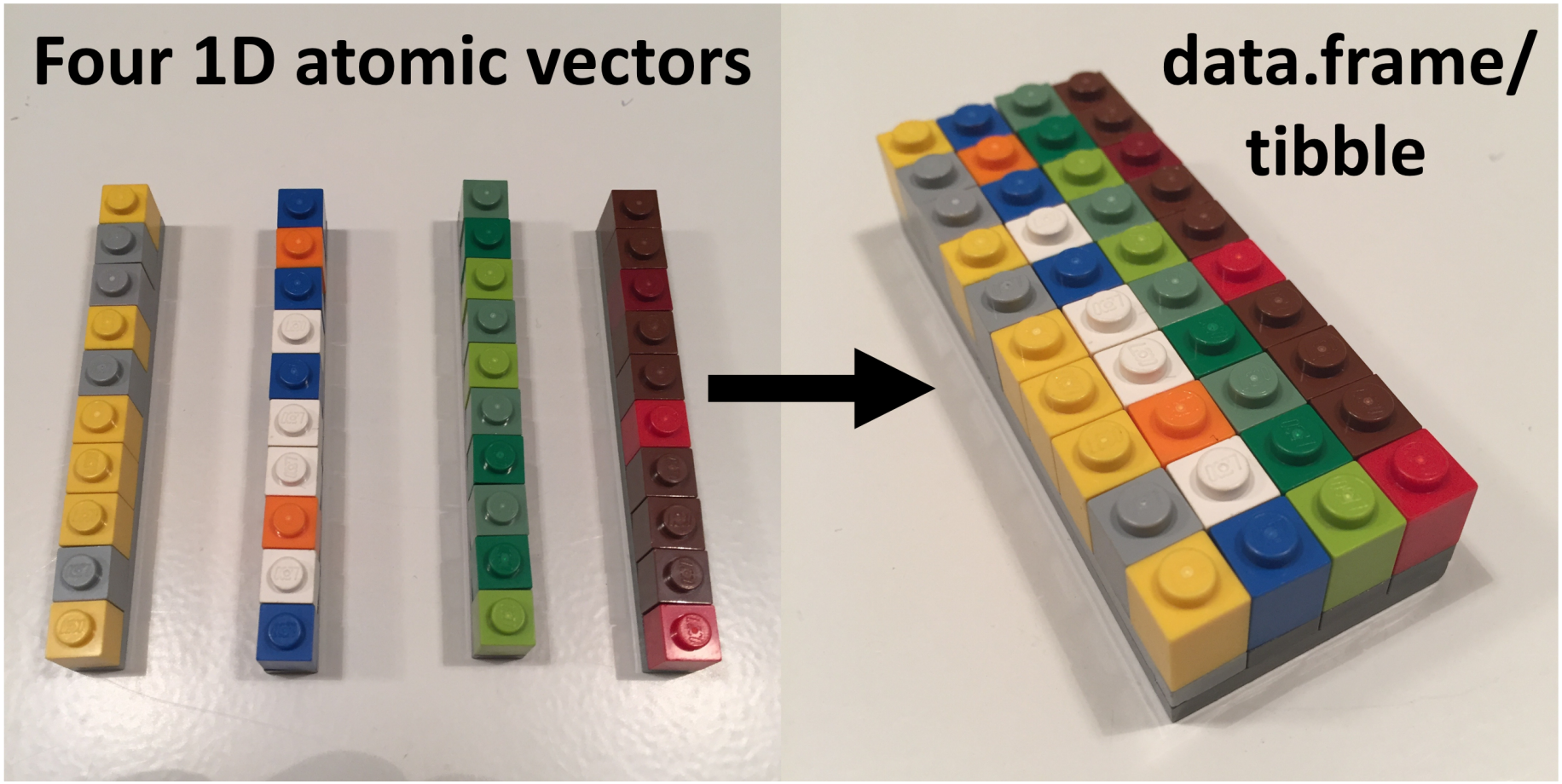




Keeping the visual representation of different types of atomic vectors in your head!!

Source: The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

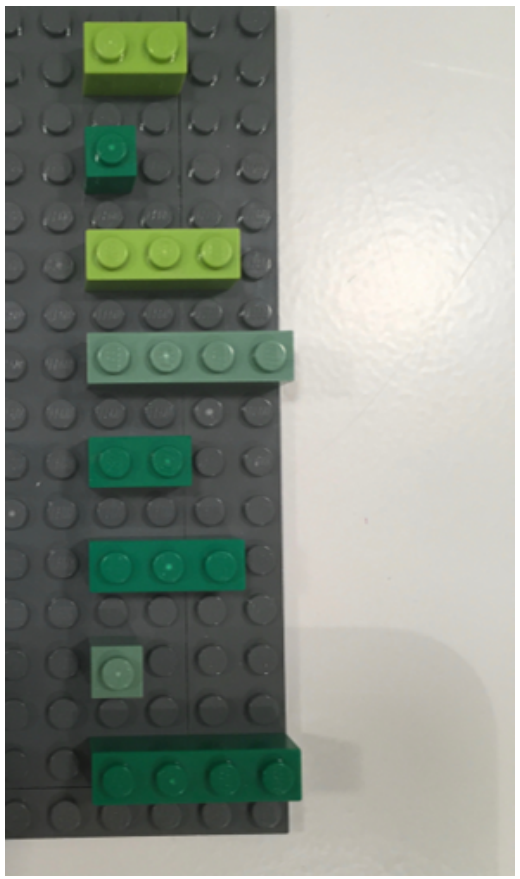
Data Structures: 1D → 2D



Source: The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

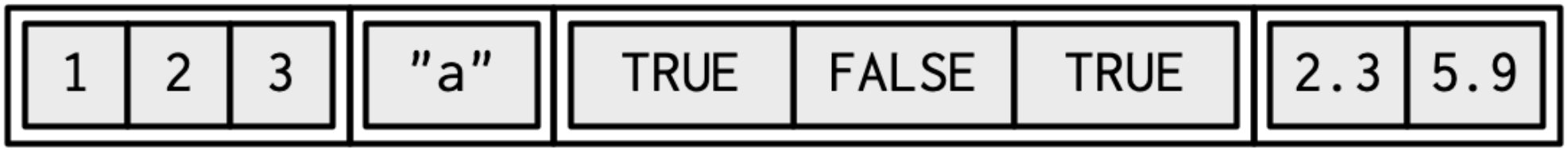
Data Structures: Lists

An object contains elements of **different data types**.



Source: The image is adapted from the excellent lego-rstats GitHub Repository by Jenny Bryan

Data Structures: Lists




```r
lst <- list( # list constructor/creator
  1:3, # atomic double/numeric vector  of length = 3
  "a", # atomic character vector of length = 1 (aka scalar)
  c(TRUE, FALSE, TRUE), # atomic logical vector of length = 3
  c(2.3, 5.9) # atomic double/numeric vector of length =3
)
lst # printing the list
```

```
## [1] "1:3"                  "a"                    "c(TRUE, FALSE, TRUE)"
## [4] "c(2.3, 5.9)"
```

Source: Image is from Hadley Wickham's Advanced R: Chapter 3 on Vectors

Data Structures: Lists

Subset by `[]`

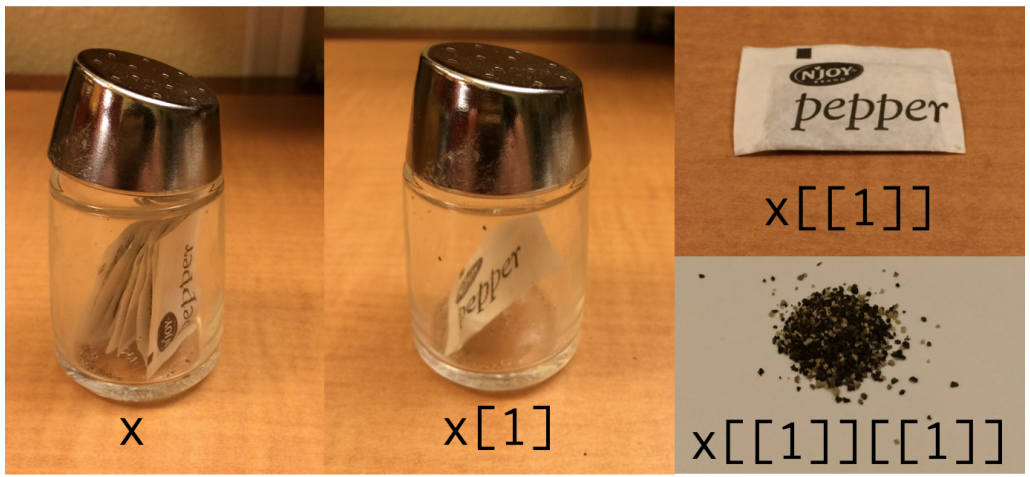
```
lst[1]
```

```
## [[1]]
## [1] 1 2 3
```

Subset by `[[]]`

```
lst[[1]]
```

```
## [1] 1 2 3
```



Sources: The slide is based on Earo Wang's STAT 220 slides and image is from Hadley Wickham's Tweet on Indexing lists in R.

Data Structures: Matrices

A matrix is a **2D data structure** made of **one/homogeneous data type.**

```
x_mat = matrix( sample(1:10, size = 4), n
str(x_mat) # its structure?
```

```
##  int [1:2, 1:2] 7 5 1 6
```

```
x_mat # printing it nicely
print('-----------------')
x_mat[1, 2] # subsetting
```

```
##      [,1] [,2]
## [1,]    7    1
## [2,]    5    6
## [1] "-----------------"
## [1] 1
```

Data Structures: Matrices

A matrix is a **2D data structure** made of **one/homogeneous data type.**

```
x_mat = matrix( sample(1:10, size = 4), n
str(x_mat) # its structure?
```

```
##  int [1:2, 1:2] 7 5 1 6
```

```
x_mat # printing it nicely
print('-----------------')
x_mat[1, 2] # subsetting
```

```
##      [,1] [,2]
## [1,]    7    1
## [2,]    5    6
## [1] "-----------------"
## [1] 1
```

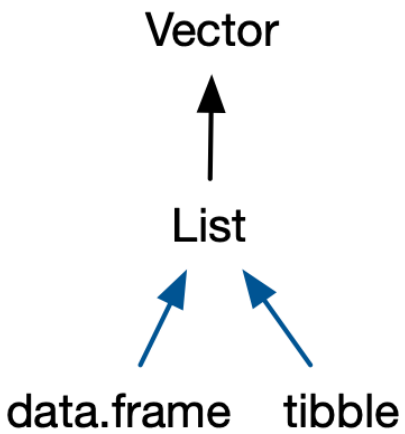
```
x_char = matrix(
  sample(letters, size = 12), nrow = 3, n
x_char
```

```
##      [,1] [,2] [,3] [,4]
## [1,] "d"  "e"  "a"  "u"
## [2,] "v"  "s"  "k"  "z"
## [3,] "b"  "n"  "l"  "t"
```

```
x_char[1:2, 2:3] # subsetting
```

```
##      [,1] [,2]
## [1,] "e"  "a"
## [2,] "s"  "k"
```

Data Structures: Data Frames



> If you do data analysis in R, you're going to be using data frames. A data frame is a named list of vectors with attributes for `(column) names`, `row.names`, and its class, "data.frame". -- Hadley Wickham

Source: Image is from Hadley Wickham's Advanced R: Chapter 3 on Vectors

Data Structures: Data Frames

```
df1 <- data.frame(x = 1:3, y = letters[1:3])
typeof(df1) # showing that its a special case of a list
```

```
## [1] "list"
```

```
attributes(df1) # but also is of class data.frame
```

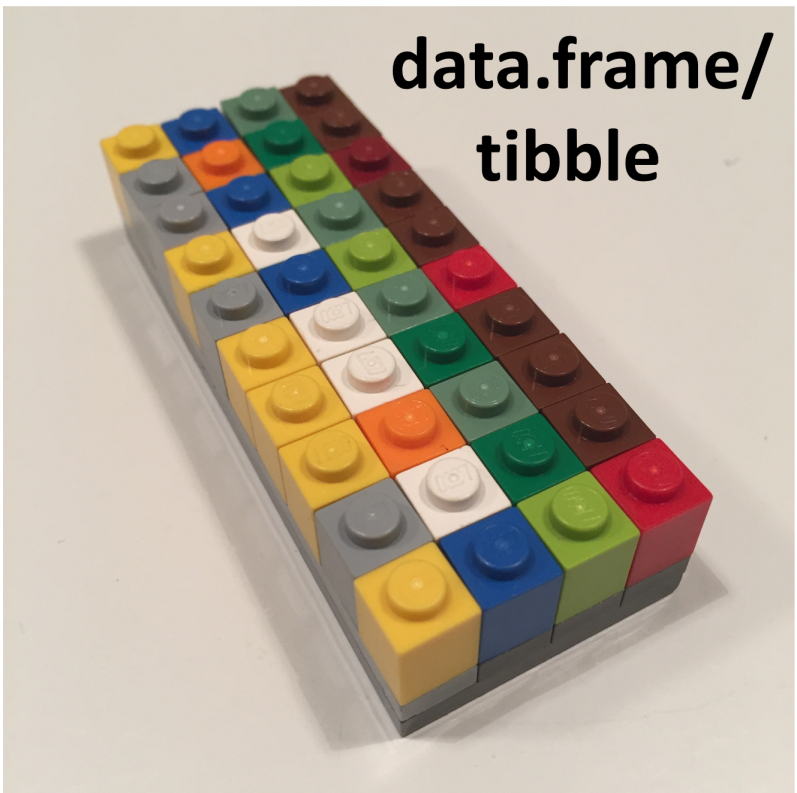
```
## $names
## [1] "x" "y"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3
```

In contrast to a regular list, a data frame has **an additional constraint: the length of each of its ~~vectors must be the same.~~** ~~This gives data frames their~~ **rectangular structure.**

Source: Content is from Hadley Wickham's Advanced R: Chapter 3 on Vectors

Data Structures: Data Frames

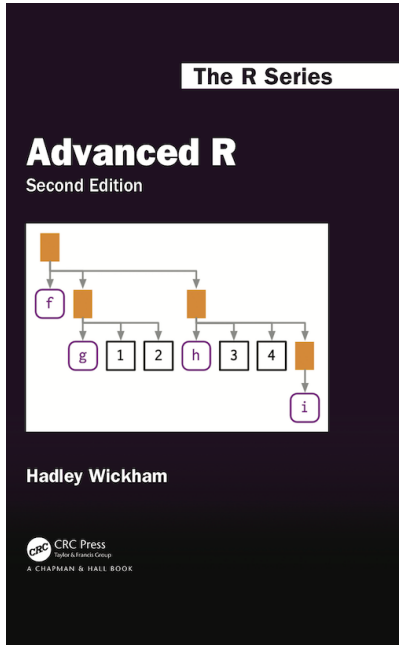
As noted in the creation of `df1`, columns in a data frame can be of different types. Hence, it is more widely used in data analysis than matrices.



Source: The image is adapted from the excellent lego-rstats GitHub Repository by Jenny Bryan

Data Structures: So What is a Tibble Anyway?

> Tibble is a **modern reimagining of the data frame**. Tibbles are designed to be (as much as possible) **drop-in replacements for data frames** that fix those frustrations. A concise, and fun, way to summarise the main differences is that tibbles are **lazy and surly: they do less and complain more**. -- Hadley Wickham



To learn more about the basics of tibble, please consult the reference below:

- Data frames and tibbles (Click and read from 3.6 up to and including 3.6.5)

Functions

A function call consists of the **function name** followed by one or more **argument** within parentheses.

```
temp_high_forecast = c(34, 36, 41, 44, 27, 32, 35)
mean(x = temp_high_forecast)
```

```
## [1] 35.57143
```

- function name: `mean()`, a built-in R function to compute mean of a vector
- argument: the first argument (LHS `x`) to specify the data (RHS `temp_high_forecast`)

Source: Slide is based on Earo Wang's STAT 220 Slides

Function Help Page

02:00

Check the function's help page with `?mean`

Class Activity

> *Please take 2 minutes to investigate the help page for `mean` in R Studio.*

```
mean(x = temp_high_forecast, trim = 0, na.rm = FALSE, ...)
```

- Read **Usage** section
  - What arguments have default values?
- Read **Arguments** section
  - What does `trim` do?
- Run **Example** code

Source: Slide is based on Earo Wang's STAT 220 Slides

Function Arguments

Match by **positions**

```
mean(temp_high_forecast, 0.1, TRUE)
```

```
## [1] 35.57143
```

Match by **names**

```
mean(x = temp_high_forecast, trim = 0.1,
```

```
## [1] 35.57143
```

Source: Slide is based on Earo Wang's STAT 220 Slides

Use Functions from Packages

```
library(dplyr)
dplyr::cummean(temp_high_forecast)
```

```
## [1] 34.00000 35.00000 37.00000 38.7500
```

```
dplyr::first(temp_high_forecast)
```

```
## [1] 34
```

```
dplyr::last(temp_high_forecast)
```

```
## [1] 35
```

Images sourced from https://www.wikihow.com/Change-a-Light-Bulb

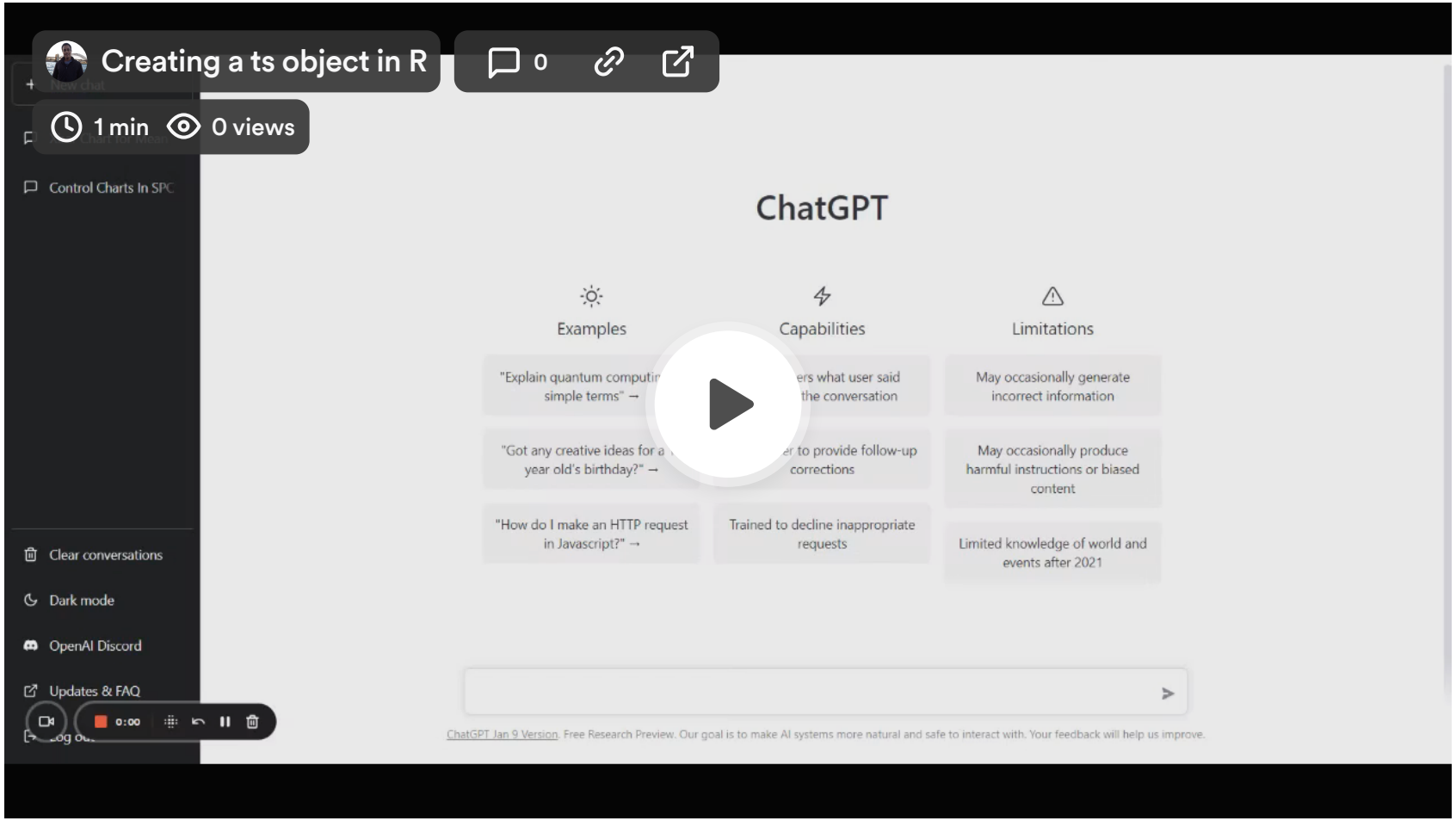
Source: Slide is based on Earo Wang's STAT 220 Slides

for time series analysis

The *ts* Object

facilitates the work with time series data by storing it in an aptly named ts object.

Create a *ts* Object by asking *ChatGPT*



Demo 1: `Nile` & `AirPassengers` Data

In class, we will create our first script where we will examine one of these two built-in datasets. In our exploration, we will:

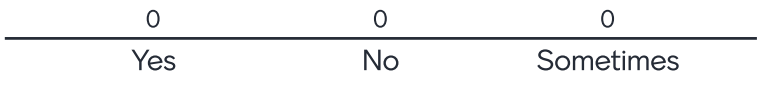
- Examine the `typeof()` the dataset.
- Examine the `class()` of the dataset.
- Examine the `length()` of the dataset.
- `print()` the data set and examine its `frequency()`.
- Subset the data using `window()` and non ts-based sub-setting techniques.
    - Useful for the concept of the entire time series vs a snippet that we discussed in Class 01.

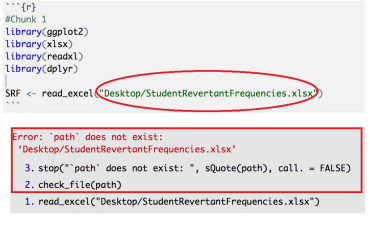
A Poll: File Paths in R

Go to **www.menti.com/alyzfp5phujn**

Do you often struggle with identifying the file path when reading data in R?

Mentimeter





```r
#Chunk 1
library(ggplot2)
library(xlsx)
library(readxl)
library(dplyr)

SRF <- read_excel("Desktop/StudentRevertantFrequencies.xlsx")
```

```
Error: `path` does not exist:
'Desktop/StudentRevertantFrequencies.xlsx'
  3. stop("`path` does not exist: ", sQuote(path), call. = FALSE)
  2. check_file(path)
  1. read_excel("Desktop/StudentRevertantFrequencies.xlsx")
```

Demo 2: Loading Time Series Data into R

- Reading CSV Files through either the `read.csv()` or `readr::read_csv()` functions.
- Reading FRED and Yahoo Finance Data using the `tidyquant` package.
- Preprocessing and converting the data into a time series object.

Bonus Tricks (if time allows)

- `file.choose()` -- not typically introduced by Miami faculty!!
- The datapasta package 🍝

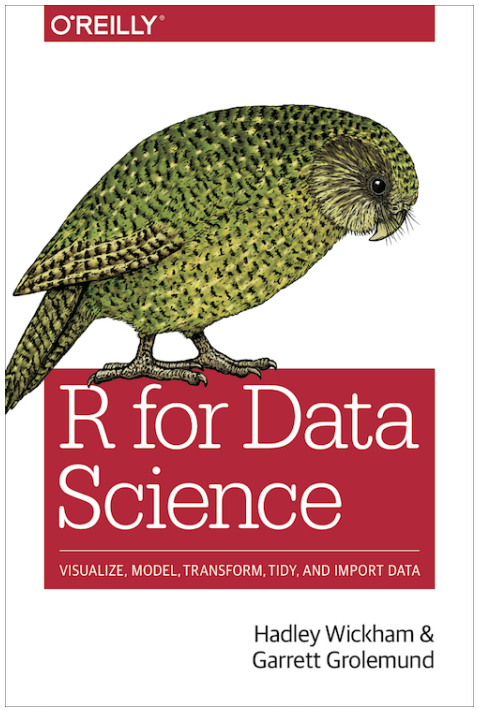
Summary of Main Points

By now, you should be able to do the following:

- Describe the syntax, data types, and data structures in R.
- Access the help for R functions (each help file has the following components: Description, usage, arguments, value, and examples).
- Utilize the project workflow in R and create R script.
- Access, subset, and create `ts()` objects in R.

Things to Do to Prepare for Our Next Class

- Go over your notes, read the **references below**, and **complete** the self-paced R tutorial.
- Complete Assignment 02 on Canvas.



- Data Visualization
- Graphics for Communication
- Dates and Times