Getting Started with Data
Week 2

PH 700A, Spring 2025

Rick Calvo

Table of contents

0.1 Overview

- Syntax Files
- Working with Data
- Loops
- Tidyverse
- Explore

0.2 Syntax Files

- All syntax files are just text documents with a different extension for language
- Can be opened in Notepad
- Contains code and your comments
- Typically does not contain results – ONLY CODE

0.3 Markdown for Data Analysis

- A markup language (text encoding) used to format documents
- **Markdown** formatting occurs on a *plaintext* file
- The markdown file is rendered to generate a formatted document
- Markdown has rules in that spacing/indention and characters have specific usage
- *Allows you to embed code and results together for presentation*

0.4 Quarto

- A language-agnostic markdown scripting tool by **Posit**
- Allows you to run code and provide commentary
- Documents can be easily disseminated
- Helps keep track of changes over time
- Formats include:
  - PDFs
  - Web presentation

- Dynamic result documents
- Static websites

0.5 Text Formatting

Headers start with hashtags #

- `*italics*`
- `**bold**`
- `***bold italics***`
- `superscript^2^`
- `subscript~2~`

For more, view: https://quarto.org/docs/authoring/markdown-basics.html

0.6 Code Chunks

0.7 Working With Tabular Data

- Data organized as a two-dimensional table of rows and columns
- Columns can hold data of different types
- Variables are the columns, observations are the rows

- two-dimensional 'cells' can can be addressed using a coordinates-style system

`dataframe[row,col]`

0.8 Accessing Columns

```
# random data generation
df <- data.frame (
  workout = c("Curls", "Pushups", "JumpingJacks", "Curls"),
  reps = c(12, 20, 40, 13),
  sets = c(3, 5, 10, 3),
  date = c("Monday", "Monday", "Tuesday", "Tuesday")
)
```

Column data can be viewed in multiple ways

- Single brackets by column number `df[1]`
- Double bracket using column text name `df[["workout"]]`
- Dollar sign using column name `df$workout`

0.9 Accessing Specific Columns

0.9.1 Addressing

```
df[1]
```

```
       workout
1        Curls
2      Pushups
3 JumpingJacks
4        Curls
```

0.9.2 Column Name

```
df[["workout"]]
```

```
[1] "Curls"        "Pushups"      "JumpingJacks" "Curls"
```

0.9.3 Internal Object

```
df$workout
```

```
[1] "Curls"        "Pushups"      "JumpingJacks" "Curls"
```

0.10 Loops

For performing repetitive tasks using a set of operations.

0.10.1 "For" Loops

for loops are to iterate a task over a defined set of numeric values or items in a list

```
for (i in 1:5) {
    print(i)
}
```

0.11 Loops, Continued

0.11.1 "While" Loops

while loops establish a *condition* that must be met for the loop to work

```
i <- 1
while (i <= 5) {
  print(i)
  i <- i + 1
}
```

0.12 Looping Over Variables

Previously, we learned that items in an object can be "addressed" by order number or name.

We can create a list of names in an object and iterate over each.

```r
columnNames <- names(df)

for (x in 1:10) {
  str(columnNames[[x]])
}
```

```r
varlist <- c("var1", "var2", "var3", ...)

for (variable in varlist) {
  print(variable)
}
```

0.13 Nesting Loops

Advanced topic, but you can embed loops within loops that have a variety of conditions

```r
for (yr in 2016:2025) {

  for (file in filenames) {

    tempitem <- paste0("file", yr)

    ... do something with tempitem...

  }
}
```

0.14 Tidyverse Package

A huge compilation of useful packages

package	description
dplyr	data manipulation
tibble	tabling system
tidyr	data cleaning
stringr	string manipulation
readr	data import
forcats	factor manipulation
purrr	functions and vectors
ggplot2	graphics and visuals

package	description

0.15 Loading Packages

`library(packageName)`

```
library(tidyverse)
library(explore)
```

```
dataframe %>% explore()
```

0.16 Pipes

What are these?

%>%

|>

Allows for the nesting of functions

Essentially an R way of shorthand to minimize keystrokes

0.17 Pipe Usage

Sends whatever is on the left side to be incorporated into a function on the right side

Left side could either be a **function** or an **object**

Right side must be a **function**

leftSideItem %>% rightSideFunction

But since you usually want to store your results, it's likely an object

0.18 Targetting Variables for Description w/ Tidyverse

Use the `select()` subfunction in `dplyr`

```
dataframe %>% select(var1, var2, ...) %>% function()
```

0.19 Explore Package

Allows interactive visualization of basic descriptive data.

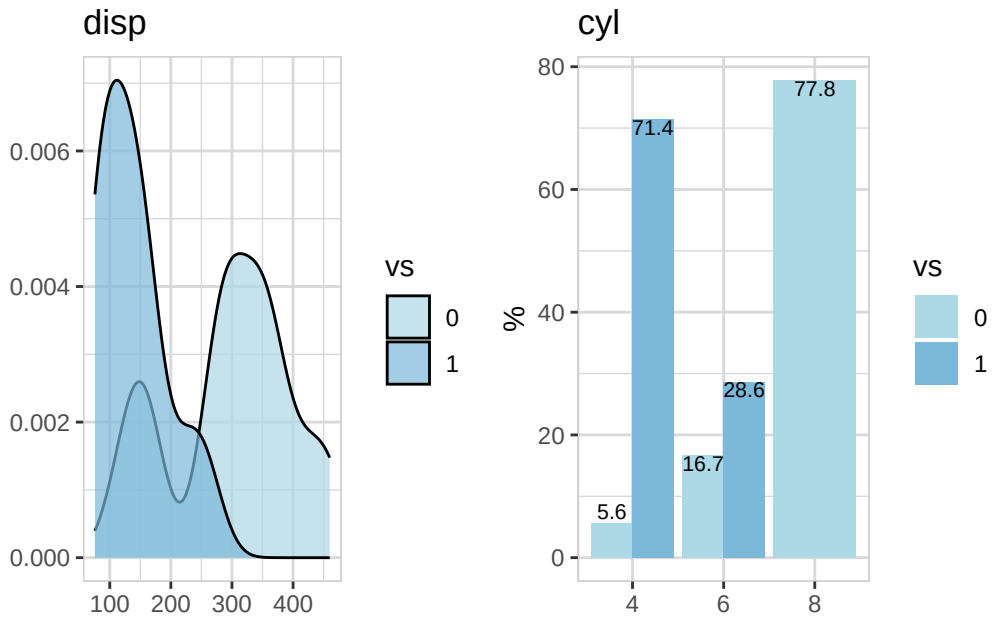
Works with Tidyverse!

```
explore(data)

data %>% explore()

describe(data)

data %>% describe()
```

Use `help(explore)` after loading the package for all subfunctions.

0.20 Combining Packages

```
mtcars %>%
  select(vs, disp, cyl) %>%
  explore_all(target = vs)
```



0.21 Base R Exploration Commands

`table()`

`print()`

`str()`

`summary()`

`length()`

`mean()`

`median()`