NC STATE UNIVERSITY

R Programming: Introduction to R

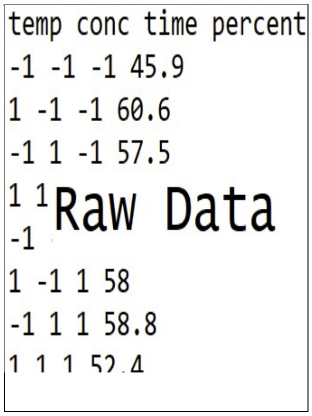
Justin Post

What is this course about?

Basic use of R for reading, manipulating, and plotting data!

What is this course about?

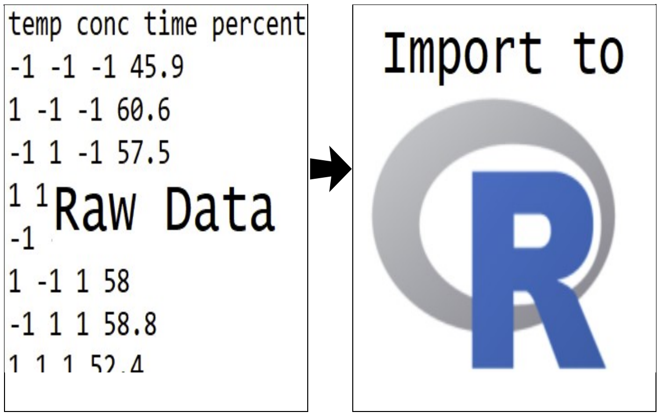
Basic use of R for reading, manipulating, and plotting data!



```
temp conc time percent
-1 -1 -1 45.9
1 -1 -1 60.6
-1 1 -1 57.5
1 1 Raw Data
-1
1 -1 1 58
-1 1 1 58.8
1 1 1 52.4
```

What is this course about?

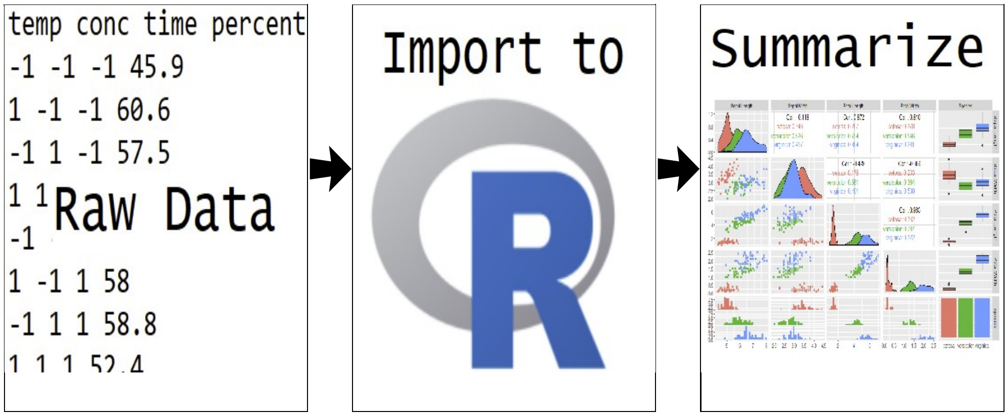
Basic use of R for reading, manipulating, and plotting data!



```
temp conc time percent
-1 -1 -1 45.9
1 -1 -1 60.6
-1 1 -1 57.5
1 1
-1
1 -1 1 58
-1 1 1 58.8
1 1 1 52.4
```

What is this course about?

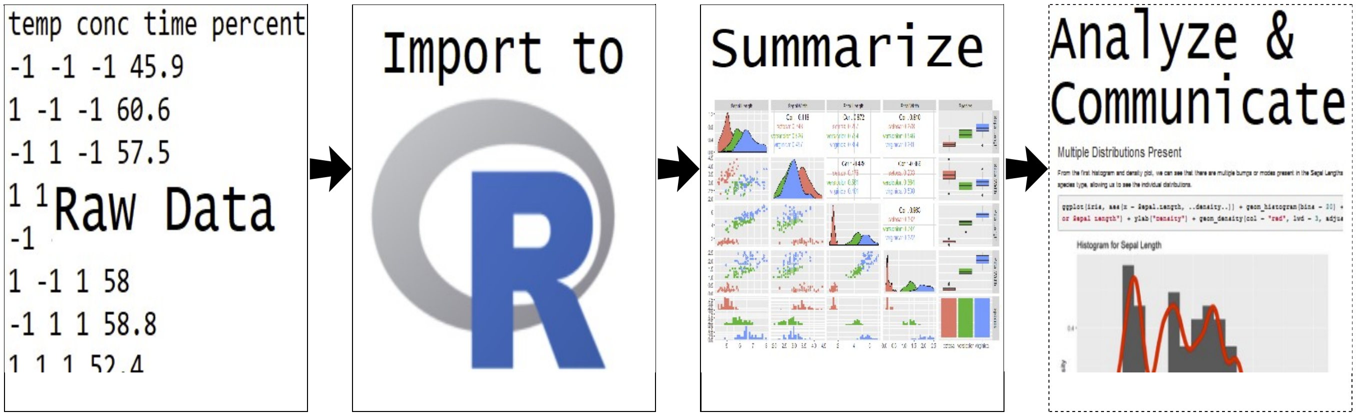
Basic use of R for reading, manipulating, and plotting data!



```
temp conc time percent
-1 -1 -1 45.9
1 -1 -1 60.6
-1 1 -1 57.5
1 1
-1
1 -1 1 58
-1 1 1 58.8
1 1 1 52.4
```

What is this course about?

Basic use of R for reading, manipulating, and plotting data!



```
temp conc time percent
-1 -1 -1 45.9
1 -1 -1 60.6
-1 1 -1 57.5
1 1
-1
1 -1 1 58
-1 1 1 58.8
1 1 1 52.4
```

Multiple Distributions Present

From the first histogram and density plot, we can see that there are multiple bumps or modes present in the Sepal Lengths
species type, allowing us to see the individual distributions.

```
ggplot(iris, aes(x = Sepal.Length, ..density..)) + geom_histogram(bins = 20) +
or Sepal Length") + ylab("Density") + geom_density(col = "red", lwd = 3, adju
```

What is this course about?

Basic use of R for reading, manipulating, and plotting data!

- **read and write basic R programs**
- import well formatted data into R
- do basic data manipulation in R
- produce common numerical and graphical summaries in R
- describe a use case of an analysis done in R

Where do we start?

- Install R/RStudio
    - Module 0!
- RStudio IDE (Integrated Development Environment)
- R Objects and Classes
- Data Objects & Basic Manipulation

RStudio IDE

In RStudio, four main ‘areas’

- Console (& Terminal)
- Scripting and Viewing Window
- Plots/Help (& Files/Packages)
- Environment (& Connections/Git)

Console

- Type code directly into the **console** for evaluation

```
#simple math operations
# <-- is a comment - code not evaluated
3 + 7
```

```
## [1] 10
```

```
10 * exp(3) #exp is exponential function
```

```
## [1] 200.8554
```

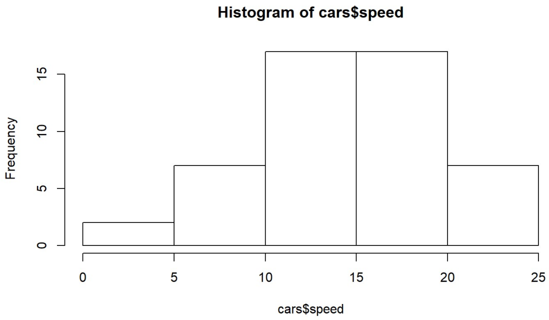
```
log(pi^2) #log is natural log by default
```

```
## [1] 2.28946
```

```
mean(cars$speed)
```

```
## [1] 15.4
```

```
hist(cars$speed)
```



Scripting and Viewing Window

- Usually want to keep code for later use!
- Write code in a ‘script’ and save script (or use markdown - covered later)
- From script can send code to console via:
    - “Run” button (runs current line)
    - CTRL+Enter (PC) or Command+Enter (MAC)
    - Highlight section and do above

Scripting and Viewing Window

- Go to file –> New File –> R Script
- Type `View(cars)` (note capital `V`)
- Type `plot(cars)`
- Submit to console using button or hot key

Plots/Help

- Created plots stored in `Plots` tab
  - Cycle through past plots
  - Easily save
- `Help` tab to learn about R functions
- Type `help(hist)` in the console

Environment

- Store **data/info/function/etc.** in R objects
- Create an R object via <– (recommended) or =

```
#save for later
avg <- (5 + 7 + 6) / 3
#call avg object
avg
```

```
## [1] 6
```

```
#strings (text) can be saved as well
words <- c("Hello there!", "How are you?")
words
```

```
## [1] "Hello there!" "How are you?"
```

Environment

- Look at all current objects with `ls()`

```
ls()
```

```
## [1] "avg"   "words"
```

- `rm()` to remove

```
rm(avg)
ls()
```

```
## [1] "words"
```

Environment

- Built-in objects exist like `letters` and `cars`

```
letters
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```
head(cars, n = 3)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
```

- `data()` shows available built-in datasets

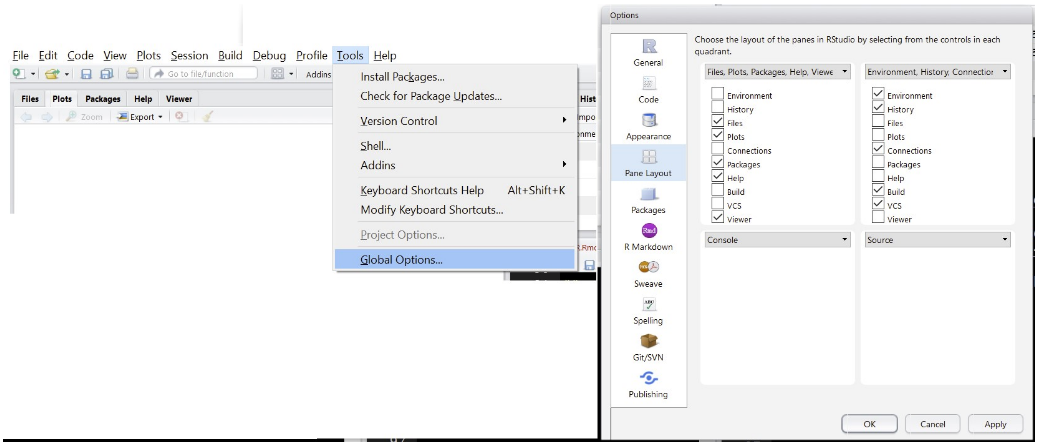
RStudio IDE

Four main ‘areas’

- Console (& Terminal)
- Scripting and Viewing Window
- Plots/Help (& Files/Packages)
- Environment (& Connections/Git)

RStudio IDE

To rearrange panes



Choose the layout of the panes in RStudio by selecting from the controls in each quadrant.

Files, Plots, Packages, Help, Viewe

- □ Environment
- □ History
- ☑ Files
- ☑ Plots
- □ Connections
- ☑ Packages
- ☑ Help
- □ Build
- □ VCS
- ☑ Viewer

Environment, History, Connectior

- ☑ Environment
- ☑ History
- □ Files
- □ Plots
- ☑ Connections
- □ Packages
- □ Help
- ☑ Build
- ☑ VCS
- □ Viewer

Console

Source

RStudio IDE

Other useful global options:

- Appearance
  - font size
  - theme
- Code
  - editing –> soft-wrap
  - display –> show whitespace

R Objects and Classes

- R has strong **O**bject **O**riented **P**rogramming (OOP) tools
- Object: data structure with attributes (class)
- Method: procedures (functions) act on object based on attributes

R Objects and Classes

- R has strong **O**bject **O**riented **P**rogramming (OOP) tools
- Object: data structure with attributes (class)
- Method: procedures (functions) act on object based on attributes
- R functions like `print()` or `plot()` act differently depending on object class

```
class(cars)
```

```
## [1] "data.frame"
```

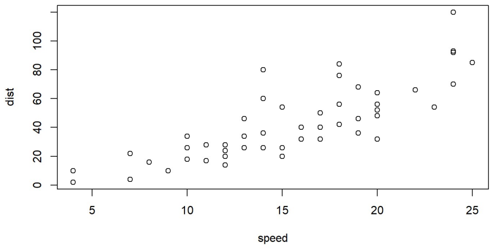
```
class(exp)
```

```
## [1] "function"
```

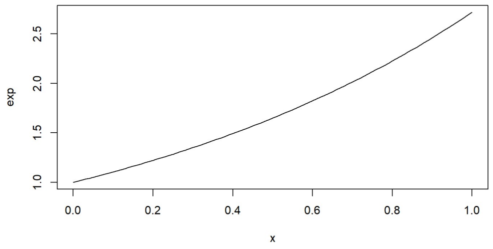
R Objects and Classes

- R has strong **O**bject **O**riented **P**rogramming (OOP) tools
- Object: data structure with attributes (often a ‘class’)
- Method: procedures (often ‘functions’) act on object based on attributes
- R functions like `print()` or `plot()` act differently depending on object class

```
plot(cars)
```



```
plot(exp)
```



R Objects and Classes

- Create an R object via <- (recommended) or =
    - allocates memory to object
    - object attributes usually depend on how you created it!

```
vec <- c(1, 4, 10)
class(vec)
```

```
## [1] "numeric"
```

```
fit <- lm(dist ~ speed, data = cars)
class(fit)
```

```
## [1] "lm"
```

Investigating Objects

Many functions to help understand an R Object

- `class()`
- describes the `class` attribute of an R object

```
class(cars)
```

```
## [1] "data.frame"
```

Investigating Objects

Many functions to help understand an R Object

- `typeof()`
- determines the (R internal) type or storage mode of any object

```
typeof(cars)
```

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

Investigating Objects

Many functions to help understand an R Object

- `str()`

- compactly displays the internal structure of an R object

```
str(cars)
```

```
## 'data.frame':    50 obs. of  2 variables:
##  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
##  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

Recap & What’s next?!

Create an R Object with `<-`

Many functions to help understand an R Object

- `class()`
- `typeof()`
- `str()`

Recap & What's next?!

Create an R Object with `<-`

Many functions to help understand an R Object

- `class()`
- `typeof()`
- `str()`

Common data structures

1. Atomic Vector (1d)
2. Matrix (2d)
3. Array (nd) (not covered)
4. Data Frame (2d)
5. List (1d)