

# R BASICS

---

J. Alexander Branham

Fall 2016



# WHAT IS R?

---

# R IS A COMPUTER PROGRAMMING LANGUAGE

```
R version 3.3.0 (2016-05-03) -- "Supposedly Educational"  
Copyright (C) 2016 The R Foundation for Statistical Computing  
Platform: x86_64-pc-linux-gnu (64-bit)
```

```
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.
```

```
  Natural language support but running in an English locale
```

```
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.
```

```
> █
```

# WHAT IS R?

- R is a language and environment for statistical computing and graphics.

# WHAT IS R?

- R is a language and environment for statistical computing and graphics.
- Derived from S, designed at Bell Laboratories

# WHAT IS R?

- R is a language and environment for statistical computing and graphics.
- Derived from S, designed at Bell Laboratories
  - S first appeared in 1976!

# WHAT IS R?

- R is a language and environment for statistical computing and graphics.
- Derived from S, designed at Bell Laboratories
  - S first appeared in 1976!
- Full language, but flexible and (can be) simple



mtcars

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	car
##	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	
##	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	
##	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	
##	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	
##	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	
##	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	
##	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	
##	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	
##	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	
##	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	
##	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	
##	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	

```
mean(mtcars$mpg)
```

```
## [1] 20.09062
```

```
summary(mtcars$mpg)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    10.40   15.42   19.20   20.09   22.80   33.90
```

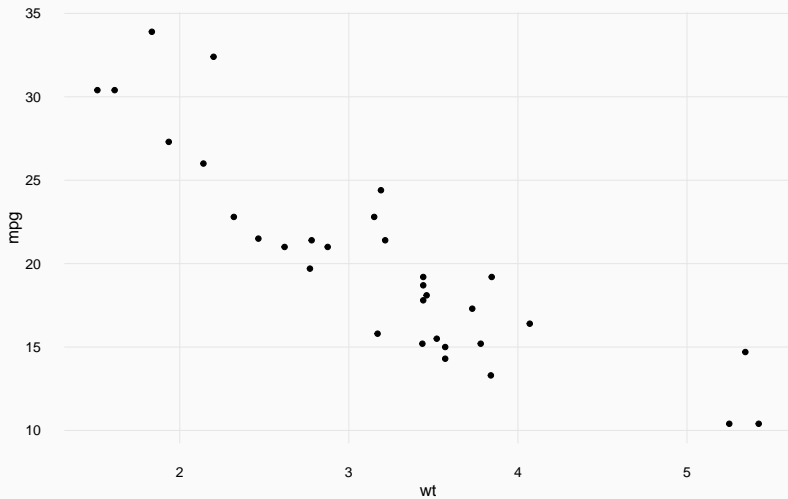
```
table(mtcars$cyl)
```

```
##
```

```
##  4  6  8
```

```
## 11  7 14
```

```
ggplot(mtcars, aes(wt, mpg)) + geom_point()
```



## ADVANTAGES OF R

---

- It's free, open-source, and available on nearly every platform

## ADVANTAGES OF R

- It's free, open-source, and available on nearly every platform
  - Stata, the main competitor with R in our department, is between \$55/year and \$235/year for students, depending on what you need

## ADVANTAGES OF R

- It's free, open-source, and available on nearly every platform
  - Stata, the main competitor with R in our department, is between \$55/year and \$235/year for students, depending on what you need
- Unparalleled in the number of packages (groups of functions)

## ADVANTAGES OF R

- It's free, open-source, and available on nearly every platform
  - Stata, the main competitor with R in our department, is between \$55/year and \$235/year for students, depending on what you need
- Unparalleled in the number of packages (groups of functions)
- More than 8,500 packages on CRAN as of June 2016,

## ADVANTAGES OF R

- It's free, open-source, and available on nearly every platform
  - Stata, the main competitor with R in our department, is between \$55/year and \$235/year for students, depending on what you need
- Unparalleled in the number of packages (groups of functions)
- More than 8,500 packages on CRAN as of June 2016,
- Great community help (e.g. #rstats on twitter, stackoverflow.com, R-bloggers)



## ADVANTAGES OF R

- It's free, open-source, and available on nearly every platform
  - Stata, the main competitor with R in our department, is between \$55/year and \$235/year for students, depending on what you need
- Unparalleled in the number of packages (groups of functions)
- More than 8,500 packages on CRAN as of June 2016,
- Great community help (e.g. #rstats on twitter, stackoverflow.com, R-bloggers)
  - `gov-r@utlists.utexas.edu` - departmental R listserv

## ADVANTAGES OF R

- It's free, open-source, and available on nearly every platform
  - Stata, the main competitor with R in our department, is between \$55/year and \$235/year for students, depending on what you need
- Unparalleled in the number of packages (groups of functions)
- More than 8,500 packages on CRAN as of June 2016,
- Great community help (e.g. #rstats on twitter, stackoverflow.com, R-bloggers)
  - `gov-r@utlists.utexas.edu` - departmental R listserv
- You can combine documents and r code together (rmarkdown or knitr)

## ADVANTAGES OF R

- It's free, open-source, and available on nearly every platform
  - Stata, the main competitor with R in our department, is between \$55/year and \$235/year for students, depending on what you need
- Unparalleled in the number of packages (groups of functions)
- More than 8,500 packages on CRAN as of June 2016,
- Great community help (e.g. #rstats on twitter, stackoverflow.com, R-bloggers)
  - `gov-r@utlists.utexas.edu` - departmental R listserv
- You can combine documents and r code together (rmarkdown or knitr)
  - For example, this presentation is written in rmarkdown

- No one likes using R in a terminal, so everyone uses an editor

- No one likes using R in a terminal, so everyone uses an editor
- Most popular is probably RStudio

- No one likes using R in a terminal, so everyone uses an editor
- Most popular is probably RStudio
- Emacs with ESS also very good choice

- Bottom left: Console

- Bottom left: Console
  - This is where you can enter R code to execute



- Bottom left: Console
  - This is where you can enter R code to execute
- Top left: editor window

- Bottom left: Console
  - This is where you can enter R code to execute
- Top left: editor window
  - You can have open R scripts or rmarkdown files here to edit

- Bottom left: Console
  - This is where you can enter R code to execute
- Top left: editor window
  - You can have open R scripts or rmarkdown files here to edit
- Top right: environment, history, (git)

- Bottom left: Console
  - This is where you can enter R code to execute
- Top left: editor window
  - You can have open R scripts or rmarkdown files here to edit
- Top right: environment, history, (git)
  - Environment will list everything that R is “remembering”

- Bottom left: Console
  - This is where you can enter R code to execute
- Top left: editor window
  - You can have open R scripts or rmarkdown files here to edit
- Top right: environment, history, (git)
  - Environment will list everything that R is “remembering”
  - History lists all commands entered in that “project”

- Bottom left: Console
  - This is where you can enter R code to execute
- Top left: editor window
  - You can have open R scripts or rmarkdown files here to edit
- Top right: environment, history, (git)
  - Environment will list everything that R is “remembering”
  - History lists all commands entered in that “project”
- Bottom right: files, plots, packages, help

- You can set up different “projects” from within Rstudio

- You can set up different “projects” from within Rstudio
- This will automatically change the working directory to where you put the project



- You can set up different “projects” from within Rstudio
- This will automatically change the working directory to where you put the project
- I use a project for each paper, for example

- What is a directory?

- What is a directory?
- What is a working directory?

- What is a directory?
- What is a working directory?
- Run `getwd()` in R. What does it return?

- What is a directory?
- What is a working directory?
- Run `getwd()` in R. What does it return?
  - This is where R will save/load files from

## USING R

---

R is a great calculator

```
3 + 2
```

```
## [1] 5
```

```
1.7729 ^ 4 * (1930 / 4)
```

```
## [1] 4766.881
```

# ALGEBRA & FUNCTIONS

R can do algebra and functions

```
a <- 1
```

```
b <- 2
```

```
a + b
```

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

```
A <- 3
```

```
a + b - A
```

```
## [1] 0
```

```
factorial(3)
```



## YOU TRY!

What will be the output of:

```
round(round(2.391) + 7.21)
```

# ASSIGNMENT

R uses `<-` for assignment.

```
a <- 4
```

```
a + 7
```

```
## [1] 11
```

```
a <- a + 2
```

```
a + 7
```

```
## [1] 13
```

`a` is now referred to as an “object.” Pretty much anything R remembers is an object.

# FUNCTIONS

Functions take arguments

```
myvector <- c(1, 5, 2, 7, 9, NA, 1)
mean(myvector, na.rm = TRUE)
```

```
## [1] 4.166667
```

```
?rnorm
```

```
rnorm(5, 0, 1)
```

## YOU TRY!

---

What's the square root of 3?

What's  $e^3$

# SCALARS, VECTORS, AND MATRICES

---

12

```
## [1] 12
```

Created with `c` (create) function:

```
c(1, 4, 2, -1)
```

```
## [1] 1 4 2 -1
```

created with the `matrix` function:

```
matrix(c(1, 2, 3, 4, 2, 1), nrow = 3)
```

```
##      [,1] [,2]  
## [1,]    1    4  
## [2,]    2    2  
## [3,]    3    1
```



## DATA TYPES

---

## YOU TRY!

What kinds of data exist here?

```
head(mtcars)
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
##	Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
##	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
##	Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
##	Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

We'll talk about four kinds of data

Something that is (TRUE) or isn't (FALSE)

```
4 == 6
```

```
## [1] FALSE
```

```
class(TRUE)
```

```
## [1] "logical"
```

A number

```
class(7.12)
```

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

```
a <- 2.192
```

```
class(a)
```

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

# CHARACTER

Character data is R's way of remembering letters

```
"My name is Alex"
```

```
## [1] "My name is Alex"
```

```
class("3.218")
```

```
## [1] "character"
```

```
class("TRUE")
```

```
## [1] "character"
```

Factors are R's way of remembering categories and labels

```
myfactor <- factor(c("a", "b", "c", "c", "b", "A"))  
myfactor
```

```
## [1] a b c c b A
```

```
## Levels: a A b c
```

## YOU TRY!

What class are the following:

```
x <- 3.18
```

```
"x"
```

```
x
```

Make a vector that contains a number, a letter, and FALSE. What's the class of the vector?



- Anything can get coerced to a character

- Anything can get coerced to a character
- Coercing logical to numeric means TRUE = 1, FALSE = 0

What type will result:

`c(5, "a")`

`c(TRUE, "FALSE")`

`TRUE + 7`

A matrix can have only one data type:

```
matrix(c(1, 2, 3, "a", "b", "c", TRUE, FALSE, TRUE), ncol = 3)
```

```
##      [,1] [,2] [,3]  
## [1,] "1"  "a"  "TRUE"  
## [2,] "2"  "b"  "FALSE"  
## [3,] "3"  "c"  "TRUE"
```

Lists and data frames allow for multiple data types

## LISTS

A list is a one-dimensional group of objects:

```
mylist <- list(1, "r", FALSE)  
class(mylist)
```

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

```
length(mylist)
```

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

Elements in a list can be anything - including vectors and lists:

```
mylist2 <- list(c(1, 2, 3), TRUE, list(c(4, 5, 6), FALSE))
```

```
## [[1]]  
## [1] 1 2 3  
##  
## [[2]]  
## [1] TRUE  
##  
## [[3]]  
## [[3]][[1]]  
## [1] 4 5 6  
##  
## [[3]][[2]]  
## [1] FALSE
```



# LISTS

Accessing elements of a list is a little weird:

```
mylist2[1]
```

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

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

```
mylist2[[1]]
```

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

## YOU TRY!

Return the vector (4, 5, 6) from this list:

```
mylist2 <- list(c(1, 2, 3), TRUE, list(c(4, 5, 6), FALSE))
```

## YOU TRY (ANSWER)

```
mylist2[[3]][[1]]
```

```
## [1] 4 5 6
```

Objects in lists can have names:

```
mylist2 <- list(element1 = c(1, 2, 3),  
               element2 = TRUE,  
               element3 = list(c(4, 5, 6), FALSE))
```

Note: since `mylist2` already exists, we could have modified the names directly by calling:

```
names(mylist2) <- c("element1", "element2", "element3")
```

Which make them easier to access:

```
mylist2$element3
```

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

```
## [1] 4 5 6
```

```
##
```

```
## [[2]]
```

```
## [1] FALSE
```

```
mydata <- data.frame(x = 1:3,  
                     y = c(5, 11, 4),  
                     z = c("some", "fancy", "text"))
```

mydata

```
##   x  y    z  
## 1 1  5  some  
## 2 2 11 fancy  
## 3 3  4  text
```

```
mean(mydata$x)
```

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

```
mean(mydata[, 1])
```

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

```
mean(mydata[, "x"])
```

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

## YOU TRY!

Make a data frame containing:

**var1** - integers from 1 through 10

**var2** - the sequence TRUE, FALSE repeated 5 times (hint: `?rep`)

What's the mean of **var1**? The mean of **var2**?



# TYPES AND DIMENSIONS

	Single type	Multiple types
1D	Vector	List
2D	Matrix	Data frame
nD	Array	

## PACKAGES

---

- Packages extend R's functionality

```
install.packages("ggplot2")
```

- Packages extend R's functionality
- CRAN hosts many packages and R can install packages from CRAN easily:

```
install.packages("ggplot2")
```