

# Intro to R - the Language

J. Alexander Branham

Fall 2015

*If you wanted to do foundational research in statistics in the mid-twentieth century, you had to be bit of a mathematician, whether you wanted to or not. . . if you want to do statistical research at the turn of the twenty-first century, you have to be a computer programmer. - Andrew Gelman*

# 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.
  - From R's website

# What is R?

- R is a language and environment for statistical computing and graphics.
  - From R's website
- Note that it is a **language**

# What is R?

- R is a language and environment for statistical computing and graphics.
  - From R's website
- Note that it is a **language**
  - Those of you who have learned a foreign language know how slow and frustrating that process is. . .

# Advantages of R

- It's **FREE**, open-source, and available on every platform

# Advantages of R

- It's **FREE**, open-source, and available on every platform
  - Free as in speech and free as in beer



# Advantages of R

- It's **FREE**, open-source, and available on every platform
  - Free as in speech and free as in beer
  - Stata is between \$55/year and \$1000/year for students, depending on what you need

# Advantages of R

- It's **FREE**, open-source, and available on every platform
  - Free as in speech and free as in beer
  - Stata is between \$55/year and \$1000/year for students, depending on what you need
- Unparalleled in the number of statistical packages (groups of functions)

# Advantages of R

- It's **FREE**, open-source, and available on every platform
  - Free as in speech and free as in beer
  - Stata is between \$55/year and \$1000/year for students, depending on what you need
- Unparalleled in the number of statistical packages (groups of functions)
- Great community help (e.g. [stackoverflow.com](https://stackoverflow.com), R-bloggers)

# Advantages of R

- It's **FREE**, open-source, and available on every platform
  - Free as in speech and free as in beer
  - Stata is between \$55/year and \$1000/year for students, depending on what you need
- Unparalleled in the number of statistical packages (groups of functions)
- Great community help (e.g. [stackoverflow.com](https://stackoverflow.com), R-bloggers)
- You can combine documents and r code together (rmarkdown or knitr)

# Advantages of R

- It's **FREE**, open-source, and available on every platform
  - Free as in speech and free as in beer
  - Stata is between \$55/year and \$1000/year for students, depending on what you need
- Unparalleled in the number of statistical packages (groups of functions)
- Great community help (e.g. [stackoverflow.com](https://stackoverflow.com), R-bloggers)
- You can combine documents and r code together (rmarkdown or knitr)
  - For example, this presentation is written in rmarkdown

# Disadvantages of R

- vs Stata

# Disadvantages of R

- vs Stata
  - Some feel like it's not as intuitive as Stata or other stats programs

# Disadvantages of R

- vs Stata
  - Some feel like it's not as intuitive as Stata or other stats programs
  - Documentation is sometimes not as clear (but you can see functions. . . )



## Disadvantages of R

- vs Stata
  - Some feel like it's not as intuitive as Stata or other stats programs
  - Documentation is sometimes not as clear (but you can see functions. . . )
- vs other programming languages (e.g. Python, C++)

## Disadvantages of R

- vs Stata
  - Some feel like it's not as intuitive as Stata or other stats programs
  - Documentation is sometimes not as clear (but you can see functions. . . )
- vs other programming languages (e.g. Python, C++)
  - Uses lots of memory

## Disadvantages of R

- vs Stata
  - Some feel like it's not as intuitive as Stata or other stats programs
  - Documentation is sometimes not as clear (but you can see functions. . . )
- vs other programming languages (e.g. Python, C++)
  - Uses lots of memory
  - If you know another programming language (C, Python, etc), R's syntax is very odd

# R

- Go to [www.r-project.org](http://www.r-project.org), click “Download R,” select a mirror close to you, and select your operating system

# R

- Go to [www.r-project.org](http://www.r-project.org), click “Download R,” select a mirror close to you, and select your operating system
- R by itself is pretty dinky, so we like to use RStudio

# RStudio

- Go to [www.rstudio.com](http://www.rstudio.com) and download the appropriate desktop version of Rstudio (the free one)

# RStudio

- Go to [www.rstudio.com](http://www.rstudio.com) and download the appropriate desktop version of Rstudio (the free one)
- Install like normal

# RStudio

- Go to [www.rstudio.com](http://www.rstudio.com) and download the appropriate desktop version of Rstudio (the free one)
- Install like normal
- Open RStudio, click tools > global options and change “Save workspace to RData on exit” to Never



# RStudio Layout

- Bottom left: Console

# RStudio Layout

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

# RStudio Layout

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

# RStudio Layout

- 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

# RStudio Layout

- 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)

# RStudio Layout

- 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”

# RStudio Layout

- 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”

# RStudio Layout

- 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



# R Projects

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

# R Projects

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

# R Projects

- 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

# Calculator

- R is a great calculator

```
3 + 2
```

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

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

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

# 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)
```

# Basic Data Structures

R has four basic types of data: logical, numeric, integer, and character

TRUE ; FALSE

3

3L

"character"

# matrix

```
mymatrix <- matrix(c(1,2,3, 5,11,4), nrow=2, byrow=TRUE)
mymatrix
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    5   11    4
```

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

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



## data.frame

```
mydata <- data.frame(x=c(1,2,3), 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
```

# list

```
mylist <- list(amatrix = mymatrix,  
              adataframe = mydata)  
mylist
```

```
## $amatrix  
##      [,1] [,2] [,3]  
## [1,]    1    2    3  
## [2,]    5   11    4  
##  
## $adataframe  
##    x  y    z  
## 1 1  5 some  
## 2 2 11 fancy  
## 3 3  4 text
```

# If-else

If else statements say IF this is true, then do this. OTHERWISE, do that. There are two types in R: if, else (which work for scalars) and ifelse, which works on vectors

```
x <- 1:10
if(5>3){
  x+1
}
```

```
## [1] 2 3 4 5 6 7 8 9 10 11
```

```
ifelse(x<=5, x+1, x-4)
```

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

## For loops

For loops do something a specified number of times:

```
x <- numeric()
for(i in 1:10){
  x[i] <- rnorm(1)
}
x
```

```
## [1] -0.632140871 -0.593974814 1.218840487 -1.125583655
## [6] -0.514208444 -1.512096568 0.003036536 2.058991654
```

## Further resources

- Importing data into R

## Further resources

- Importing data into R
- Useful R packages

## Further resources

- Importing data into R
- Useful R packages
- R tutorials on lynda.com (free for UT students)