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 re- search 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.
  - 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
  - Free as in speech and free as in beer
  - Stata is between $55/yr 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, R-bloggers)
- You can combine documents and r code together (rmarkdown or knitr)
  - For example, this presentation is written in rmarkdown

Disadvantages of R

- Some feel like it's not as intuitive as Stata or other stats programs
- Uses lots of memory
- If you know another programming language (C, Python, etc), R's syntax is very odd
- Since anyone can write a package, some of the code (and help files) are difficult to follow

R

- Go to 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 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
  - 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
- 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(5, 0, 1)
```

```
## [1] -2.1313857  1.5995918  0.3049912 -0.8190170  0.4947
```

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=FALSE)
mymatrix
```

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

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

```
## [1] 1.5
```

data.frame

```
mydata <- data.frame(x=c(1,2,3), y=c(5,11,4))
mydata
```

```
##   x  y
## 1 1  5
## 2 2 11
## 3 3  4
```

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

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

list

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

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

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]  0.2659956 -1.2093433 -0.3415457 -0.6611788 -0.665
##  [7]  1.2709624 -1.4753229 -0.2055798  0.2476187
```

Further resources

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