R Basics

author: Andrew Flowers date: Wed., May 4, 2016 autosize: true font-family: ‘Helvetica’ autosize: TRUE

Goals for R Learning Group

- Hands-on
- Complete beginners welcome
- No dumb questions
- 1 hour or less
- Gradually learn more R

Outline for today’s session

- Important notes
- Review
- Working directories
- Data types **(+ vectors and lists)**
- Installing and loading packages
- Importing data into RStudio
- Understanding your data
- Useful overview functions
- Selecting columns with the “$” operator
- Subsetting
- Useful analysis functions
- Missing values
- Dplyr?

Important notes (1/2)

R has a weird-looking “assignment operator”: <-

```
x <- 1
x
```

```
[1] 1
```

```
x = 2
x
```

```
[1] 2
```

- While the “=” symbol works, please use “<-”
- Short-cut for Mac (two keys): “alt/option” + “-”

To test equality, or select a variable, use “==”

```
1 == 2
1 == 1
```

Important notes (2/2)

Commenting

- Use the # sign for commenting
- Commenting is text or code that you do NOT want R to run

```
# This will not print
# print("hello world")
```

```
# This will print
print("hello world")
```

```
[1] "hello world"
```

- Make sure to comment you code so you can understand it later

Tab completion

- When typing, hit “Tab” and RStudio will suggest completed commands for you

The RStudio console is not responding. What do I do?

- Hit the “Esc” key

```
# Example of getting stuck
print("hello world"
```

Working directories (1/2)

“We all need a place to call home”

- Your working directory is the folder location out of which you’re working
- Use the getwd() command to display your current working directory

```
getwd()
```

```
[1] "/Users/flowersa/repos/r-learning/sessions/r-basics"
```

- Note: R commands often have a parenthesis “()”" where you input arguments

Make sure you've set your working directory correctly

- Use the setwd() command to set your working directory

```
setwd("/Users/flowersa/repos/r-learning")
```

```
setwd("/Users/flowersa/repos/r-learning/sessions/intro-to-R-and-RStudio/")
```

Working directories (2/2)

Display the files in your current working directory

- Use the list.files() command

```
list.files()
```

```
[1] "police_killings.csv" "r-basics-figure"     "r-basics.Rpres"
```

- Side note: dir() command does the same thing

```
dir()
```

```
[1] "police_killings.csv" "r-basics-figure"     "r-basics.Rpres"
```

Review of data types

- Four most common "atomic" data types:
- Note: use the typeof() function to find the data type

1.) Numerics (real numbers)

```
x <- 1
x
```

```
[1] 1
```

2.) Integers

```
y <- 1L
y
```

```
[1] 1
```

3.) Characters (strings)

```
a <- 'hello'
a
```

```
[1] "hello"
```

```
# Don't forget the quotes!
```

4.) Logicals (booleans)

```
b <- FALSE # Or you can just capitalize the first letter: T or F
b
```

```
[1] FALSE
```

Vectors and Lists

- Vectors and lists are collections of data, but they differ
- Vecotrs are homogeneous (all of the same data type)
- To make a vector, use the c() function

```
num_vector <- c(1, 2, 3)
num_vector
```

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

```
char_vector <- c("a", "b", "c")
char_vector
```

```
[1] "a" "b" "c"
```

- Lists are hetergenous (can contain different data types)
- To make a list, use the list() function

```
ex_list <- list(1, "a", TRUE)
ex_list
```

Installing and Loading Packages

- Let's load the following package: readr
- To install an R package:
- Use install.packages() command **with package name in quotes**

```
install.packages("readr")
```

- To load an R package:
- Use require() or library() command **don't need to put the package name in quotes, but you can**

```r
require("readr")
# or
library("readr")
```

Police killings data

Police killings data set

- Used by Ben in a story last June
- GitHub repo: https://github.com/fivethirtyeight/data/tree/master/police-killings

To load the csv file, use the read_csv() function from the readr package

```r
police_killings <- read_csv("police_killings.csv")
```

Lots of other great functions in readr package

Functions for understanding your data (1/2)

- str() means "structure" – gives you a description of the variables

```r
str(police_killings)
```

- dim() tells you the dimensions of your data

```r
dim(police_killings)
```

- ncol() and nrow() return how many columns and rows are in the DataFrame

```r
ncol(police_killings)
nrow(police_killings)
```

- length() returns the length of a vector (or the columns of a DataFrame)

```r
length(police_killings)
sample_vector <- c('a', 'b', 'c', 'd')
length(sample_vector)
```

Functions for understanding your data (2/2)

- head() and tail() show you the top and bottom rows (deafult is 5)

```
head(police_killings)
tail(police_killings, 2)
```

- names() produces a list of your DataFrame's columns

```
names(police_killings)
```

- summary() produces a statistical summary of each variable

```
summary(police_killings)
```

- View() displays the data in the RStudio viewer (can also click on data in Environment tab)

```
View(police_killings)
```

Selecting columns in a data frame

How do you pick out a specific column of a DataFrame?

- Use the "$" operator
- If your data is DataFrame, and the column is Column, then select it like this: DataFrame$Column
- For instance:

```
police_killings$name
```

- To understand the structure of the "name" column, do this:

```
str(police_killings$name)
```

```
 chr [1:467] "A'donte Washington" "Aaron Rutledge" "Aaron Siler" ...
```

- To create a summary of the county_income column, do this:

```
summary(police_killings$county_income)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  22540   43800   50860   52530   56830  110300 
```

Useful analysis functions

- table() give you a the breakdown for a variable

```
table(police_killings$state)
```

```
AK AL AR AZ CA CO CT DC DE FL GA HI IA ID IL IN KS KY LA MA MD ME MI MN MO
 2  8  4 25 74 12  1  1  2 29 16  4  2  4 11  8  6  7 11  5 10  1  9  6 10
MS MT NC NE NH NJ NM NV NY OH OK OR PA SC TN TX UT VA WA WI WV WY
 6  2 10  6  1 11  5  3 14 10 22  8  7  9  6 46  5  9 11  5  2  1
```

- sort() does just that (default is decreasing = F)

```
head(sort(police_killings$county_income))
tail(sort(police_killings$county_income))
head(sort(police_killings$county_income, decreasing=T))
```

- unique() returns the unique values of a variable

```
unique(police_killings$lawenforcementagency)
```

Subsetting data (1/3)

What is subsetting?

- When you want to see specific rows or columns in a data frame
- Cells in a DataFrames are accessed through bracket notation:
- **DataFrame**[**rows(s), column(s)**]
- Let's start with a vector example

```
x <- c("a", "b", "c", "d")

x[1]
x[2]
x[1:2]
x[c(4, 2, 1, 3)]
```

Subsetting data (2/3)

- Now, let's apply it to our police data
- To see the first row:

```
police_killings[1,]
# or
View(police_killings[1,])
```

- To see the second column (which is age):

```
police_killings[,2]
# or
View(police_killings[,2]) # This will NOT work, for complicated reasons we'll learn later
View(as.data.frame(police_killings[,2]))
```

- Don’t forget the comma “,” when subsetting a Dataframe!!!
- Think in terms of rows and columns Subsetting data (3/3) ==
- So, for instance, let’s look at all police killings in New York state
- To find how to code for New York state, let’s run table() or summary() on that column

```
table(police_killings$state)
# or
str(police_killings$state)
```

- Using bracket notation AND the full DataFrame$Column identifier, try this:

```
police_killings[police_killings$state=="NY",]
```

- Creating new data frames

```
# We can save this subsetted data as a new DataFrame
ny_police_killings <- police_killings[police_killings$state=="NY",]
```

- How many police killings were in NY state in 2015?

```
dim(ny_police_killings)
```

```
[1] 14 34
```

```
# Answer: 14
```

We will learn easier and much more flexible data munging next week (dplyr!)

Missing values

- is.na() function evalues for missing values

```
is.na(police_killings$county_bucket)

sum(is.na(police_killings$county_bucket)) # sum() treats TRUE = 1, FALSE = 0

# Now use that TRUE/FALSE vector to subset the DataFrame
police_killings[is.na(police_killings$county_bucket),]
```