

Introduction to Social Data Analytics

Class 16

Arushi Kaushik

arkaushi@ucsd.edu

May 7, 2019

Today: Introduction to R and RStudio

By the end of today's lecture, you should be able to:

- ▶ Locate and identify the essential parts of the RStudio interface
- ▶ Create, edit, and save .R and .RData files
- ▶ Generate objects and differentiate between datasets, numbers, strings, and functions

RStudio Interface

The screenshot displays the RStudio interface with the following components:

- Editor window:** Contains R code:

```
1 a <- 25
2 b <- 3
3 a+b
```
- Environment / Variables:** Shows the current environment with variables:

Variable	Value
a	25
b	3
- Console window:** Shows the execution of the code:

```
> a <- 25
> b <- 3
> a+b
[1] 28
> |
```
- Help / Packages / Plots / Files:** Displays the documentation for the `read.table` function, including its description and usage.

Arithmetic Operations

R can be used as a calculator:

```
5 + 3
```

```
## [1] 8
```

```
5 / 3
```

```
## [1] 1.666667
```

```
5 ^ 3
```

```
## [1] 125
```

► The [1] is telling you the row number.

R is an “object-oriented” programming language

Objects, any pieces of information stored by R, can be:

- ▶ A dataset (e.g. resume)
- ▶ A subset of a dataset (e.g. just the even observations of resume)
- ▶ A number (e.g. $2\pi + 1$)
- ▶ A text string (e.g. “UCSD is awesome”)
- ▶ A function (e.g. a function that takes in x and gives you $x^2 + 8$)

Creating objects

R can store *objects* with a name of our choice. Use `<-` as an assignment operator for objects.

```
object_1 <- 5 + 3  
object_1
```

```
## [1] 8
```

If we assign a new value to the same object name, then we will overwrite this object (so be careful when doing so!)

```
object_1 <- 5 - 3  
object_1
```

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

Objects (cont.)

R can also represent other types of values as objects, such as strings of characters:

```
MySchool <- "UCSD"  
MySchool
```

```
## [1] "UCSD"
```

A *vector* represents a collection of information stored in a specific order

We use the function `c()`, which stands for “concatenate,” to enter a data vector (with commas separating elements of the vector):

```
vector.1 <- c(93, 92, 83, 99, 96, 97)
vector.1
```

```
## [1] 93 92 83 99 96 97
```

- Note: when creating a vector, R creates column vectors ($n \times 1$)

Vectors (cont.)

To access specific elements of a vector, we use square brackets `[]`. This is called *indexing*:

```
vector.1[2]
```

```
## [1] 92
```

```
vector.1[c(2, 4)]
```

```
## [1] 92 99
```

```
vector.1[-4]
```

```
## [1] 93 92 83 96 97
```

Vectors (cont.)

Since each element of this vector is a numeric value, we can apply arithmetic operations to it:

```
vector.1 * 1000
```

```
## [1] 93000 92000 83000 99000 96000 97000
```

Element-corresponding operations with vectors

```
vec1 <- c(1, 2, 3); vec2 <- c(3, 3, 3)  
vec1 + vec2
```

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

```
vec1 * vec2
```

```
## [1] 3 6 9
```

```
vec1 / vec2
```

```
## [1] 0.3333333 0.6666667 1.0000000
```

Functions

A *function* takes input object(s) and returns an output object. In R, a function generally runs as `funcname(input)`. Some basic functions useful for summarizing data include:

- ▶ `length()`: length of a vector (number of elements)
- ▶ `min()`: minimum value
- ▶ `max()`: maximum value
- ▶ `range()`: range of data
- ▶ `mean()`: mean
- ▶ `sum()`: sum

Try these with `vector.1`

Functions (cont.)

```
length(vector.1)
```

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

```
min(vector.1)
```

```
## [1] 83
```

```
max(vector.1)
```

```
## [1] 99
```

Functions (cont.)

```
range(vector.1)
```

```
## [1] 83 99
```

```
mean(vector.1)
```

```
## [1] 93.33333
```

```
sum(vector.1)
```

```
## [1] 560
```

Like Stata, we need to specify a working directory in R

- ▶ Use the function `setwd()` to change the working directory

```
setwd("path")
```

- ▶ Use the function `getwd()` to display the current working directory.

```
getwd()
```

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

Loading data from your working directory

- ▶ For CSV files:

```
resume <- read.csv("resume.csv")
```

- ▶ For *RData* files:

```
resume <- load("resume.RData")
```


Data Frames

A *data frame* is a collection of vectors, but we can think of it like an Excel spreadsheet. Useful functions for data frames include:

- ▶ `names()`: return a vector of variable names
- ▶ `nrow()`: return the number of rows
- ▶ `ncol()`: return the number of columns
- ▶ `dim()`: combine `ncol()` and `nrow()` into a vector
- ▶ `summary()`: produce a summary
- ▶ `head()`: displays the first six observations
- ▶ `tail()`: displays the last six observations

Load `resume.csv`, assign it to an object called `resume`, and try the above functions on this newly created data frame.

Data Frames (cont.)

```
names(resume)
```

```
## [1] "X"          "firstname" "sex"        "race"       "call"
```

```
nrow(resume)
```

```
## [1] 4870
```

```
ncol(resume)
```

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

Data Frames (cont.)

```
dim(resume)
```

```
## [1] 4870    5
```

```
summary(resume)
```

```
##           X           firstname           sex           race
## Min.      :    1   Tamika : 256   female:3746   black:2435
## 1st Qu.:1218   Anne   : 242   male  :1124   white:2435
## Median :2436   Allison: 232
## Mean    :2436   Latonya: 230
## 3rd Qu.:3653   Emily   : 227
## Max.    :4870   Latoya  : 226
##                (Other):3457
##
##      call
## Min.    :0.00000
```

Data Frames (cont.)

```
head(resume)
```

```
##      X  firstname      sex  race  call
## 1 1      Allison female white      0
## 2 2      Kristen female white      0
## 3 3      Lakisha female black      0
## 4 4      Latonya female black      0
## 5 5        Carrie female white      0
## 6 6          Jay   male white      0
```

```
tail(resume)
```

```
##              X  firstname      sex  race  call
## 4865 4865      Lakisha female black      0
## 4866 4866        Tamika female black      0
## 4867 4867         Ebony female black      0
```

Data Frames: using []

We can retrieve specified observations and variables using brackets [] with a comma in the form [rows, columns]:

```
resume[1:3, "firstname"]
```

```
## [1] Allison Kristen Lakisha
```

```
## 36 Levels: Aisha Allison Anne Brad Brendan Brett Carrie Dan
```

```
resume[1:3, 2]
```

```
## [1] Allison Kristen Lakisha
```

```
## 36 Levels: Aisha Allison Anne Brad Brendan Brett Carrie Dan
```

Observe that “firstname” is the second variable in the “resume” data frame.

Data Frames: using \$

The \$ operator is another way to access variables from a data frame:

```
head(resume$firstname, 3)
```

```
## [1] Allison Kristen Lakisha
```

```
## 36 Levels: Aisha Allison Anne Brad Brendan Brett Carrie Dan
```

Note: the “3” after the comma specifies how many observations to display.

Saving Objects

When you quit RStudio, you will be asked whether you would like to save the workspace. You should answer *no* to this in general: we only want to save what we want!

- ▶ To export CSV:

```
write.csv(resume, file = "resume.csv")
```

- ▶ To export *RData*:

```
save(resume, file = "resume.RData")
```

Go ahead and export your data frame as RData.

Here are the commands/operators we covered today:

- ▶ `<-`
- ▶ `c()`
- ▶ `vector[]`
- ▶ `length()`, `min()`, `max()`, `range()`, `mean()`, `sum()`
- ▶ `head()`, `tail()`
- ▶ `setwd()`, `getwd()`
- ▶ `read.csv()`, `load()`
- ▶ `names()`, `nrow()`, `ncol()`, `dim()`, `summary()`
- ▶ `write.csv()`, `save()`
- ▶ `$`