Handout 1

Student von Student III

In this handout, we will learn the first steps of any data analysis: reading in the data and summarizing it. Also, throughout the semester, we will be looking only at various subsets of the data. We will learn here how to select data by the subset.

Topics and Concepts Covered

- Basic operations on vectors
- Loading in data
- The structure and format of R

R Commands Covered

- Creating a vector using `c` and accessing its elements using `[` and `]`
- Basic operations on vectors: `length`, `mean`, `median`, `min`, `max`, `range`, `sum`, `prod`, `log`
- Reading data using `read.csv(..., header = TRUE)` and `read.table(..., header = TRUE)`
- Summarizing the data using `head`, `summary`, and `names`
- Accessing help files using `?` and `??`
- Using `$` to extract columns from a data frame
- Using brackets with two arguments, `[row, column]` to recover elements of a data frame

Before beginning this handout Do not forget to make a new folder for this assignment and set your working directory!

Working with Vectors

Before beginning an example from a recent paper in political science, we are going to continue with the data on turnout from the info session. The data is reprinted below.

Actual versus Projected Turnout, 2008 to 2012, based on Exits				
	2008 Actual	2012 projected	2012 actual	Difference
White	97,677,100	99,139,605	93,033,145	(6,106,459)
Black	16,564,353	17,455,952	16,797,651	(658,301)
Hispanic	11,174,365	13,134,504	12,921,270	(213,234)
Asian	2,629,262	3,300,869	3,230,318	(70,551)
Other	3,418,041	3,910,201	2,584,254	(1,325,947)

Figure 1: Turnout data from Trende

Creating Vectors

A vector is simply a collection of numbers or strings. Vectors are constructed in R from using the `c` function, which is used to *combine* objects into a single vector. We are going to create a set of vectors, one for each column in Trende's figure.

```
actual.2008 <- c(97677100, 16564353, 11174365, 2629262, 3418041)
projected.2012 <- c(99139605, 17455952, 13134504, 3300869, 3910201)
actual.2012 <- c(93033145, 16797651, 12921270, 3230318, 2584254)

difference <- actual.2012 - projected.2012
difference
```

```
[1] -6106460  -658301  -213234   -70551 -1325947
```

We can recall any element of a vector by using brackets.

```
actual.2008[2]
```

```
[1] 16564353
```

```
actual.2012[1] - projected.2012[1]
```

```
[1] -6106460
```

R allows us to perform any number of operations on a vector:

```
length(actual.2008) # Number of elements in a vector
```

```
[1] 5
```

```
mean(actual.2008) # Average of the elements
```

```
[1] 26292624
```

```
median(actual.2008) # Middle element, when sorted
```

```
[1] 11174365
```

```
min(actual.2008) # Smallest element
```

```
[1] 2629262
```

```
max(actual.2008) # Largest element
```

```
[1] 97677100
```

```
range(actual.2008) #Range of a vector
```

```
[1]  2629262 97677100
```

```
sum(actual.2008) # Adds all of the numbers in a vector
```

```
[1] 131463121
```

```
prod(actual.2008) # Multiplies all of the numbers in a vector
```

```
[1] 1.624805e+35
```

```
log(actual.2008)
```

```
[1] 18.39718 16.62276 16.22913 14.78221 15.04458
```

Coding Tip

If you want to access RStudio's help functions, click on the 'Help' tab in the lower right hand box, and type in your question. If you want to access help files from the command line, type in `?command`, e.g. `?sum` to learn about the `sum` function. If you do not remember the exact name of a command, type `??summar` and R will use 'fuzzy matching' to suggest some commands you might be
looking for.

Giving Names to a Vector

Next, we give names to a vector:

```
names(actual.2008) # No names yet
```

```
NULL
```

```
names(actual.2008) <- c("White", "Black", "Hispanic", "Asian", "Other")

actual.2008
```

```
   White    Black Hispanic    Asian    Other
97677100 16564353 11174365  2629262  3418041
```

```
actual.2008["White"]
```

```
   White
97677100
```

Example: Disaster Relief Aid and Support for the Incumbent President.

In this section, we analyze the relationship between disaster relief aid and support for the incumbent President's party, from 1988-2004. Political economists have long theorized that incumbent political leaders may 'buy' votes, through dispensing aid to sub-national political units in order to shore up electoral support.

Healy and Malhotra (2009) examined whether this effect is present in the contemporary United States. We are going to conduct an abridged version of their study, though the basic findings will be similar.

The authors explored the relationship between county-level support for the incumbent President's party and disaster aid disbursed to the county. Each observation is a county in the United States, observed in the four years before five consecutive elections (1988, 1992, 1996, 2000, and 2004). Like the authors, we are interested in characterizing a causal relationship between disaster aid disbursement and support for the incumbent party's candidate in the election.

The dataset `disasteraid.csv` is available as a comma-separated and tab-delimited file in the `data` folder. Comma separated files have the suffix `.csv`, while tab-delimited files often have the suffix `.tsv` or `.txt`.

The data contains the following variables:

Name	Description
`fips`	An identifier for each county. This is the level of government that received aid.
`year`	The year for which the variables are observed.
`incum_vote`	The percent of the vote received by the incumbent's party for that county in that election.

Name	Description
`prev_incum`	The percent of the vote received by the incumbent's party in the previous election.
`all_current_irelief`	A measure of disaster aid relief received, per capita, in the county.

Reading Data into R

First, we must load in the data. Here the data is in the `data` folder net to this file. Sometimes you will need to download it from Blackboard or another website.

When we read in a file we will follow a *three-step procedure*:

1. Read it in
2. Check the first five rows
3. Summarize the data.

This process will ensure that we have loaded in the data without error.
The structure is given below.

Reading in a tab-delimited `.tsv` file.

```
Data.disaster <- read.table("data/disasteraid.tsv", header = TRUE)
head(Data.disaster)
```

```
  fips year incum_vote prev_incum all_current_irelief
1 1001 1988   67.12975   70.06797            0.000000
2 1001 1992   55.92000   67.12975            0.473335
3 1001 1996   32.52000   30.92000            0.000000
4 1001 2000   28.72000   32.52000            0.000000
5 1001 2004   75.67000   69.69000            4.257831
6 1003 1988   72.84960   75.54547            2.486306
```

```
summary(Data.disaster)
```

```
      fips            year        incum_vote      prev_incum
 Min.   : 1001   Min.   :1988   Min.   : 6.83   Min.   : 6.84
 1st Qu.:19041   1st Qu.:1992   1st Qu.:38.02   1st Qu.:41.91
 Median :29211   Median :1996   Median :46.98   Median :52.00
 Mean   :30676   Mean   :1996   Mean   :47.94   Mean   :51.76
 3rd Qu.:46009   3rd Qu.:2000   3rd Qu.:57.48   3rd Qu.:61.92
 Max.   :56045   Max.   :2004   Max.   :97.97   Max.   :97.97
 all_current_irelief
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.9958
 3rd Qu.:1.3547
 Max.   :9.6895
```

Reading in a comma-separated `.csv` file.

```
Data.disaster <- read.csv("data/disasteraid.csv", header = TRUE)
head(Data.disaster)
```

```
  fips year incum_vote prev_incum all_current_irelief
1 1001 1988   67.12975   70.06797            0.000000
2 1001 1992   55.92000   67.12975            0.473335
3 1001 1996   32.52000   30.92000            0.000000
4 1001 2000   28.72000   32.52000            0.000000
5 1001 2004   75.67000   69.69000            4.257831
6 1003 1988   72.84960   75.54547            2.486306
```

```
summary(Data.disaster)
```

```
      fips            year        incum_vote      prev_incum
 Min.   : 1001   Min.   :1988   Min.   : 6.83   Min.   : 6.84
 1st Qu.:19041   1st Qu.:1992   1st Qu.:38.02   1st Qu.:41.91
 Median :29211   Median :1996   Median :46.98   Median :52.00
 Mean   :30676   Mean   :1996   Mean   :47.94   Mean   :51.76
 3rd Qu.:46009   3rd Qu.:2000   3rd Qu.:57.48   3rd Qu.:61.92
 Max.   :56045   Max.   :2004   Max.   :97.97   Max.   :97.97
 all_current_irelief
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.9958
 3rd Qu.:1.3547
 Max.   :9.6895
```

Notes

We set the parameter `header` to `TRUE` to let R know that the first row of the file should be used to name each column. To see what happens if we get this wrong, try

```
Data.disaster.wrong <- read.table("data/disasteraid.tsv") # Wrong!
head(Data.disaster.wrong)
```

```
    V1   V2          V3          V4                  V5
1 fips year  incum_vote  prev_incum all_current_irelief
2 1001 1988 67.12974548 70.06797028                   0
3 1001 1992 55.91999817 67.12974548         0.473334968
4 1001 1996 32.52000046 30.92000008                   0
5 1001 2000 28.71999931 32.52000046                   0
6 1001 2004 75.66999817 69.69000244          4.25783062
```

Notice how the first row of the data is the column names, while the column names are simply `V1` through `V5`.

We will use lower case names for vectors and variables. We will use capital letters for data frames, like `Data.disaster`.

When you read in a data frame, it shows up in the upper right hand box of RStudio.

Attributes of Data Frames

So far, we have been looking at a single data frame, `Data.disaster`. A `data.frame` contains one column for each variable, and one row for each observation. Below are some basic operations on data frames.

We can get the names of the columns of the data frames:

```
names(Data.disaster)
```

```
[1] "fips"                "year"                "incum_vote"
[4] "prev_incum"          "all_current_irelief"
```

The function `dim` returns a vector with two elements. The first is the number of rows of a data frame, the second is the number of columns.

```
dim(Data.disaster)
```

```
[1] 15561     5
```

The functions `nrow` and `ncol` return the same information as `dim`.

```
nrow(Data.disaster)
```

```
[1] 15561
```

```
ncol(Data.disaster)
```

```
[1] 5
```

We can extract columns of a data frame several different ways. For example, we can use `$` to extract the column directly by name and work with it.

```
mean(Data.disaster$incum_vote)
```

```
[1] 47.94061
```

or we can also *take a copy* of the column and work with this new variable

```
incum <- Data.disaster$incum_vote
mean(incum)
```

```
[1] 47.94061
```

The Structure of R

So far, we have encountered the following elements when working with R:

- The **script**. This should contain all of the commands necessary to replicate your analysis, as well as comments explaining the code.
- The **working directory**. This is the directory from where R will look for data, and to which it will save any objects.
- The **workspace**. This is the set of all objects in your current R session.

The work space will include variables, vectors, data frames, and a history of all commands you've run in this session. When you close RStudio, you will be asked if you want to save your work space.
In general, you will *not* want to. We save scripts, not the work space.
(You can change the Preference settings inside RStudio if you decide you never want to be asked).

Precept Questions

In 2007 a federal judge ordered New York City (NYC) to overhaul its stop-and-frisk program link. The program allowed police in NYC to temporarily detain and search pedestrians. For an overview of the program, see the first few pages of Gelman, Fagan, and Kiss (2007).

Concerns arose that blacks and Hispanics were being stopped at higher rates than whites. We are going to load in and look at some of the stop-and-frisk data. The data is large, with 532,911 observations and 101 variables. We will be looking at a subset of this data. You will find the data in the `data` folder. It is called `saf_subset.tsv`, so the path from the folder where this file is to the data is `data/saf_subset.tsv`.

Please answer the following questions and submit your code following the directions in the syllabus.

Question 1

Read the data into a data frame named `SAF`.

```
## My code here!
```

Question 2

How many rows and how many columns are in the subset of the data?

Question 3

What year is the data from?

Question 4

What is the earliest date for a stop?

Question 5

The latest?

Question 6

What are the youngest and oldest ages?

Question 7

Assuming that NYC cops are arresting neither infants nor Yoda, what could be going on here instead?

My explanation here

Often, when you use data for your research, variable names and descriptions will be found in a code book. The code book for the full data set can be found here.

Question 8

How many different categories of race are considered?

Question 9

Is the mean statistic returned from `summary` useful in characterizing the "average race" of those stopped and frisked?

Question 10

Try the command `table(SAF$race)`.

What do you think the command `table()` does?

Question 11

What does the function `getwd` do? When might this be helpful?

References

Gelman, Andrew, Jeffrey Fagan, and Alex Kiss. 2007. "An Analysis of the New York City Police Department's 'Stop-and-Frisk' Policy in the Context of Claims of Racial Bias." *Journal of the American Statistical Association* 102 (479): 813–23. doi:10.1198/016214506000001040.

Healy, Andrew, and Neil Malhotra. 2009. "Myopic Voters and Natural Disaster Policy." *American Political Science Review* 103 (03): 387–406. doi:10.1017/S0003055409990104.