Data basics

Stat140-04

Shan Shan

Topic 1: Identify the W's: Online shopping

Suppose that the observational units in a study are the purchases that I made on amazon.com *in the past 12 months*. Identify each of the following as a categorical variable, a quantitative/numerical variable, or not a variable. If not a variable, briefly explain why.

1. Was the purchase shipped to me or to someone else?

2. Do I tend to spend more on purchases sent to others than on purchases sent to me?

3. How much did I spend on the purchase?

4. Did the purchase include at least one book?

5. What was the average price of these purchases?

Topic 2: Understanding the data tables

1. Iris [Adpated from OpenIntro Chapter 1] Sir Ronald Aylmer Fisher was an English statistician, evolutionary biologist, and geneticist who worked on a data set contained sepal length and width, and petal length and width from three species of iris flowers (setosa, versicolor and virginica). There were 50 flowers from each species in the data set. Below is the output of the `head` function.

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

1. What does the row of the table represent?

2. What does the columns of the table represent?

3. How many variables are there in the data table?

4. Is the variable 'Sepal.Length' categorical or quantitative?

5. Is the variable 'Species' categorical or quantitative?

Topic 3: Looking into a very large data set

In the era of big data, a data table might contain thousands of (or even millions of, if you work in fields like astrophysical science) of observational units or variables. In this section, you will learn how to investigate your data using the core functions of R.

1. Flights The Bureau of Transportation Statistics (BTS) is a statistical agency that is a part of the Research and Innovative Technology Administration (RITA). As its name implies, BTS collects and makes available transportation data, such as the flights data we will be working with in this lab. Now go to https://rdrr.io/snippets/ and copy the following code in the code snippet.

```
library(nycflights13)
```

I have preloaded the dataset `flights` here. `flights` is now a *data matrix*, with each row representing an *observation* and each column representing a *variable*. R calls this data format a **data frame**, which is a term that will be used throughout this module.

To take a look at the first few rows of the data set (by default, the first 6 rows), you can use the **head** function. It's good for getting a quick summary of what's in the data frame, but it will not tell you how many observational units there are. Run the following code to see the output of the **head** function.

```
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>
## 1  2013     1     1      517            515         2      830
## 2  2013     1     1      533            529         4      850
## 3  2013     1     1      542            540         2      923
## 4  2013     1     1      544            545        -1     1004
## 5  2013     1     1      554            600        -6      812
## 6  2013     1     1      554            558        -4      740
## # ... with 12 more variables: sched_arr_time <int>, arr_delay <dbl>,
## #   carrier <chr>, flight <int>, tailnum <chr>, origin <chr>, dest <chr>,
## #   air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
## #   time_hour <dttm>
```

The **str** function will print out some more detailed information about the data frame, including how many observational units and variables there are, and the type of each variable – but its output is a little less well organized.

```
str(flights)
```

```
## tibble [336,776 x 19] (S3: tbl_df/tbl/data.frame)
##  $ year          : int [1:336776] 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...
##  $ month         : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ day           : int [1:336776] 1 1 1 1 1 1 1 1 1 1 ...
##  $ dep_time      : int [1:336776] 517 533 542 544 554 554 555 557 557 558 ...
##  $ sched_dep_time: int [1:336776] 515 529 540 545 600 558 600 600 600 600 ...
##  $ dep_delay     : num [1:336776] 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...
##  $ arr_time      : int [1:336776] 830 850 923 1004 812 740 913 709 838 753 ...
##  $ sched_arr_time: int [1:336776] 819 830 850 1022 837 728 854 723 846 745 ...
```

```
##  $ arr_delay     : num [1:336776] 11 20 33 -18 -25 12 19 -14 -8 8 ...
##  $ carrier       : chr [1:336776] "UA" "UA" "AA" "B6" ...
##  $ flight        : int [1:336776] 1545 1714 1141 725 461 1696 507 5708 79 301 ...
##  $ tailnum       : chr [1:336776] "N14228" "N24211" "N619AA" "N804JB" ...
##  $ origin        : chr [1:336776] "EWR" "LGA" "JFK" "JFK" ...
##  $ dest          : chr [1:336776] "IAH" "IAH" "MIA" "BQN" ...
##  $ air_time      : num [1:336776] 227 227 160 183 116 150 158 53 140 138 ...
##  $ distance      : num [1:336776] 1400 1416 1089 1576 762 ...
##  $ hour          : num [1:336776] 5 5 5 5 6 5 6 6 6 6 ...
##  $ minute        : num [1:336776] 15 29 40 45 0 58 0 0 0 0 ...
##  $ time_hour     : POSIXct[1:336776], format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...
```

You can see the dimensions of this data frame by using **dim**, **nrow**, **ncol**. The **dim** function will tell you how many rows (i.e., how many observational units) and columns (i.e., how many variables) are in the data frame (in that order). The **nrow** function will tell you how many rows there are, and the **ncol** function will tell you how many columns there are.

```
dim(flights)
```

```
## [1] 336776     19
```

```
nrow(flights)
```

```
## [1] 336776
```

```
ncol(flights)
```

```
## [1] 19
```

The output indicates that there are 336776 rows and 19 columns.

You can see the names of the columns (or variables) by typing:

```
names(flights)
```

```
##  [1] "year"           "month"          "day"            "dep_time"
##  [5] "sched_dep_time" "dep_delay"      "arr_time"       "sched_arr_time"
##  [9] "arr_delay"      "carrier"        "flight"         "tailnum"
## [13] "origin"         "dest"           "air_time"       "distance"
## [17] "hour"           "minute"         "time_hour"
```

This returns the names of the variables in this data frame, e.g., `year`, `month`, `day`, `dep_time`, `arr_time`, `dep_delay`, `arr_delay`, `carrier`, `tailnum`, `flight`, `origin`, `dest`, `air_time`, ...

Let's start to examine the data a little more closely. We can access the data in a single column of a data frame separately using a command like

```
head(flights$origin)
```

```
## [1] "EWR" "LGA" "JFK" "JFK" "LGA" "EWR"
```

This command will only show the origin of the flights. What command would you use to extract just the destination of the flights? Write your code here.

The `flights` data frame is a massive trove of information. Let's think about some questions we might want to answer with these data:

- We might want to find out how delayed flights headed to a particular destination tend to be.
- We might want to evaluate how departure delays vary over months.
- Or we might want to determine which of the three major NYC airports has a better on time percentage for departing flights.

In the following weeks, you will learn statistical and data analysis skills to help you answer these questions.