Data scraping/ reformatting example

Samantha Morrison and Gabriella Silva

September 15, 2017

This example used code and analysis from http://sharpsightlabs.com/blog/shipping-analysis-r-data-wrangling/

In this example, they looked at a Wikipedia dataset (https://en.wikipedia.org/wiki/List_of_busiest_container_ports) that had shipping volumes from various ports. (Below is an example of the dataset they wanted to scrape from the Wiki page)

Rank	Port	Jurisdiction	2015[1]	2014[2]	2013[3]	2012[4]	2011[5]	2010[6]	2009[7]	2008[8]	2007[9]	2006[10]	2005[11]	2004[12]
1	Shanghai	China	36,516	35,268	33,617	32,529	31,700	29,069	25,002	27,980	26,150	21,710	18,084	14,557
2	Singapore	Singapore	30,922	33,869	32,240	31,649	29,937	28,431	25,866	29,918	27,932	24,792	23,192	21,329
3	Shenzhen	China	24,142	23,798	23,280	22,940	22,570	22,510	18,250	21,414	21,099	18,469	16,197	13,615
4	Ningbo-Zhoushan	China	20,636	19,450	17,351	16,670	14,686	13,144	10,502	11,226	9,349	7,068	5,208	4,006
5	Hong Kong	Hong Kong SAR	20,073	22,374	22,352	23,117	24,384	23,532	20,983	24,248	23,881	23,539	22,427	21,984
6	Busan	South Korea	19,469	18,423	17,690	17,046	16,185	14,157	11,954	13,425	13,270	12,039	11,843	11,430
7	Qingdao	China	17,323	16,624	15,520	14,503	13,020	12,012	10,260	10,320	9,462	7,702	6,307	5,140
8	Guangzhou	China	17,097	16,160	15,309	14,744	14,400	12,550	11,190	11,001	9,200	6,600	4,685	3,308
9	Jebel Ali (Dubai)	United Arab Emirates	15,585	14,750	13,641	13,270	13,000	11,600	11,124	11,827	10,653	8,923	7,619	6,429
10	Tianjin	China	13,881	14,050	13,010	12,300	11,500	10,080	8,700	8,500	7,103	5,950	4,801	3,814
11	Rotterdam	Netherlands	12,235	12,453	11,621	11,866	11,877	11,146	9,743	10,784	10,791	9,655	9,287	8,281
12	Port Klang	Malaysia	11,887	10,736	10,350	10,000	9,604	8,870	7,309	7,970	7,120	6,326	5,544	5,244
13	Kaohsiung	Taiwan	10,264	10,593	9,938	9,781	9,636	8,872	8,581	9,677	10,257	9,775	9,471	9,714
14	Antwerp	Belgium	9,654	9,136	8,578	8,635	8,664	8,468	7,309	8,663	8,176	7,019	6,482	6,064

Figure 1:

This example required the following packages:

```
# packages needed
#install.packages("tidyverse")
#install.packages("stringr")
#install.packages("forcats")
#install.packages("ggmap")
#install.packages("rvest")



library(tidyverse)
library(stringr)
library(forcats)
library(ggmap)
library(rvest)
```

Reading in Data

Next we need to scrape the data from wikipedia:

```r
# scrape data from wikipedia
html.world_ports <- read_html("https://en.wikipedia.org/wiki/List_of_busiest_container_ports")

df.world_ports <- html_table(html_nodes(html.world_ports, "table")[[1]], fill = TRUE)
```

We look at the dataframe to see what we imported.

```r
# look at dataframe

glimpse(df.world_ports)
```

```
## Observations: 50
## Variables: 15
## $ Rank         <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15...
## $ Port         <chr> "Shanghai", "Singapore", "Shenzhen", "Ningbo-Zhou...
## $ Jurisdiction <chr> "China", "Singapore", "China", "China", "Hong Kon...
## $ 2015[1]      <chr> "36,516", "30,922", "24,142", "20,636", "20,073",...
## $ 2014[2]      <chr> "35,268", "33,869", "23,798", "19,450", "22,374",...
## $ 2013[3]      <chr> "33,617", "32,240", "23,280", "17,351", "22,352",...
## $ 2012[4]      <chr> "32,529", "31,649", "22,940", "16,670", "23,117",...
## $ 2011[5]      <chr> "31,700", "29,937", "22,570", "14,686", "24,384",...
## $ 2010[6]      <chr> "29,069", "28,431", "22,510", "13,144", "23,532",...
## $ 2009[7]      <chr> "25,002", "25,866", "18,250", "10,502", "20,983",...
## $ 2008[8]      <chr> "27,980", "29,918", "21,414", "11,226", "24,248",...
## $ 2007[9]      <chr> "26,150", "27,932", "21,099", "9,349", "23,881", ...
## $ 2006[10]     <chr> "21,710", "24,792", "18,469", "7,068", "23,539", ...
## $ 2005[11]     <chr> "18,084", "23,192", "16,197", "5,208", "22,427", ...
## $ 2004[12]     <chr> "14,557", "21,329", "13,615", "4,006", "21,984", ...
```

```r
head(df.world_ports)
```

```
##   Rank            Port  Jurisdiction 2015[1] 2014[2] 2013[3] 2012[4]
## 1    1        Shanghai         China  36,516  35,268  33,617  32,529
## 2    2       Singapore     Singapore  30,922  33,869  32,240  31,649
## 3    3        Shenzhen         China  24,142  23,798  23,280  22,940
## 4    4 Ningbo-Zhoushan         China  20,636  19,450  17,351  16,670
## 5    5       Hong Kong Hong Kong SAR  20,073  22,374  22,352  23,117
## 6    6           Busan   South Korea  19,469  18,423  17,690  17,046
##   2011[5] 2010[6] 2009[7] 2008[8] 2007[9] 2006[10] 2005[11] 2004[12]
## 1  31,700  29,069  25,002  27,980  26,150   21,710   18,084   14,557
## 2  29,937  28,431  25,866  29,918  27,932   24,792   23,192   21,329
## 3  22,570  22,510  18,250  21,414  21,099   18,469   16,197   13,615
## 4  14,686  13,144  10,502  11,226   9,349    7,068    5,208    4,006
## 5  24,384  23,532  20,983  24,248  23,881   23,539   22,427   21,984
## 6  16,185  14,157  11,954  13,425  13,270   12,039   11,843   11,430
```

We notice that most of these variables are read in as character (chr), so we will need to changes these to numeric or factor later on.

Adding to the dataset

Since we see we have a dataset on ports, we might be interested in getting latitudes/longitudes associated with each port name. We use the ggmaps::geocode() function to retrieve these latitudes/longitudes.

```r
# getting latitudes/longitudes

geocodes.world_ports <- geocode(df.world_ports$Port) #give it place names

# add latitudes/longitudes to the dataset (w cbind())
df.world_ports <- cbind(df.world_ports, geocodes.world_ports)

head(df.world_ports) # look at new dataset
```

```
##   Rank            Port  Jurisdiction 2015[1] 2014[2] 2013[3] 2012[4]
## 1    1        Shanghai         China  36,516  35,268  33,617  32,529
## 2    2       Singapore     Singapore  30,922  33,869  32,240  31,649
## 3    3        Shenzhen         China  24,142  23,798  23,280  22,940
## 4    4 Ningbo-Zhoushan         China  20,636  19,450  17,351  16,670
## 5    5       Hong Kong Hong Kong SAR  20,073  22,374  22,352  23,117
## 6    6           Busan   South Korea  19,469  18,423  17,690  17,046
##   2011[5] 2010[6] 2009[7] 2008[8] 2007[9] 2006[10] 2005[11] 2004[12]
## 1  31,700  29,069  25,002  27,980  26,150   21,710   18,084   14,557
## 2  29,937  28,431  25,866  29,918  27,932   24,792   23,192   21,329
## 3  22,570  22,510  18,250  21,414  21,099   18,469   16,197   13,615
## 4  14,686  13,144  10,502  11,226   9,349    7,068    5,208    4,006
## 5  24,384  23,532  20,983  24,248  23,881   23,539   22,427   21,984
## 6  16,185  14,157  11,954  13,425  13,270   12,039   11,843   11,430
##        lon       lat
## 1       NA        NA
## 2 103.8198  1.352083
## 3 114.0579 22.543096
## 4       NA        NA
## 5       NA        NA
## 6 129.0756 35.179554
```

Some of the ports are missing latitude/longitude data which we could manually look up and add in. We will not do this for this exercise.

Reformatting the dataset

Next, we want to consider the form of the dataset. We need to convert some variables to factors such as Port and Jurisdiction. We can do this using base R or using dplyr.

```r
# change some variables to factors

df.world_ports$Jurisdiction <-as.factor(str_trim(
  df.world_ports$Jurisdiction))

df.world_ports$Port <- as.factor(df.world_ports$Port)
```

We might be interested in looking at **year** as a variable, in which case we would want one variable for year (currently there are variables corresponding to all years).

In this case, we want to put the dataset in the long form where year is a variable and a new variable volume is used to contain the information currently under 2005, 2004 etc.

To do this we can use the tidyr package.

```r
# reshape the dataset to long format
df.world_ports <- gather(df.world_ports, year,volume,4:15)



# check the results
# head(df.world_ports)
# levels(as.factor(df.world_ports$year))
# levels(df.world_ports$Jurisdiction)
# levels(df.world_ports$Port)
# names(df.world_ports)
```

Again we see that year is coded as a character when we want it as a factor variable and the volume variable is a character but we want it as a numeric variable. We can recode these variables as follows:

We need to the stringr package to change variable volume to numeric.

```r
#year
df.world_ports$year <- as.factor(df.world_ports$year)

#volume

#("" are coded as NA)
df.world_ports$volume <- as.numeric(str_replace(df.world_ports$volume, ",",""))
```

```
## Warning: NAs introduced by coercion
```

(We skipped showing the code for renaming the year variables to remove the [] in the names)

Now that we have cleaned up most of the data and formatted it properly, we may want to find the ranks for the shipping volumes for each port for every year. The current Rank variable in the dataset has just the ranks for 2014 (so it is repeated for the years that are not 2014) so we can just remove this variable.

Then we can calculate the ranks we want (ranks for each year) using some dplyr functions.

```r
# remove the old 'rank' variable
df.world_ports <- select(df.world_ports, -Rank)



# new ranks
df.world_ports <- df.world_ports %>%
                    group_by(year) %>%
                    mutate(Rank = min_rank(desc(volume))) %>%
                    ungroup()



head(df.world_ports)
```

```
## # A tibble: 6 × 7
##              Port  Jurisdiction      lon       lat    year volume  Rank
##            <fctr>        <fctr>    <dbl>     <dbl>  <fctr>  <dbl> <int>
## 1        Shanghai         China       NA        NA 2015[1]  36516     1
## 2       Singapore     Singapore 103.8198  1.352083 2015[1]  30922     2
## 3        Shenzhen         China 114.0579 22.543096 2015[1]  24142     3
## 4 Ningbo-Zhoushan         China       NA        NA 2015[1]  20636     4
## 5       Hong Kong Hong Kong SAR       NA        NA 2015[1]  20073     5
## 6           Busan   South Korea 129.0756 35.179554 2015[1]  19469     6
```

Group Work

1. Practice Data Scraping

Use the rvest packages to read in the Superbowl dataset. The url we want to read in data from is http://espn.go.com/nfl/superbowl/history/winners

You can leave the dataset in table form:

```
##                              X1            X2
## 1 Super Bowl Winners and Results          <NA>
## 2                           NO.          DATE
## 3                             I Jan. 15, 1967
## 4                            II Jan. 14, 1968
## 5                           III Jan. 12, 1969
## 6                            IV Jan. 11, 1970
##                             X3                            X4
## 1                         <NA>                         <NA>
## 2                         SITE                       RESULT
## 3 Los Angeles Memorial Coliseum  Green Bay 35, Kansas City 10
## 4           Orange Bowl (Miami)      Green Bay 33, Oakland 14
## 5           Orange Bowl (Miami) New York Jets 16, Baltimore 7
## 6  Tulane Stadium (New Orleans)   Kansas City 23, Minnesota 7
```

Figure 2:

Hint: Use the read_html, html_nodes, html_table functions

(Do not just look up this exact example online, you should be able to generally summarize what each function does)

2. Practice using dplyr to clean up datasets

Rewrite the following functions from the above example using the dpylr command: mutate

A)

```
df.world_ports$Jurisdiction <-as.factor(str_trim(
  df.world_ports$Jurisdiction))

df.world_ports$Port <- as.factor(df.world_ports$Port)
```

B)

```
df.world_ports$year <- as.factor(df.world_ports$year)
```

C)

```
#("" are coded as NA)
df.world_ports$volume <- as.numeric(str_replace(df.world_ports$volume, ",",""))
```