# Data Extracting and Loading

**References**

[1] S. Munzert, C. Rubba, P. Meißner, and D. Nyhuis, Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining (Wiley, Chechester, West Sussex, UK) Chap. 9.
[2] J. Manuel and M. Reyes, Introduction to Data Science for Social and Policy Research: Collecting and Organizing Data with R and Python (Cambridge University Press, Cambridge, 2017) Chap. 4.

1. Data Import and Export
2. Extracting HTML Tables
3. Google Documents and Analytics

# 1. Data Import and Export

## (1) CSV File

CSV: a comma separated text format that is widely used

**read.csv**(file, header=TRUE, sep=",", ... )
**write.csv**(x, file="", row.names=TRUE, col.names=TRUE, ... )

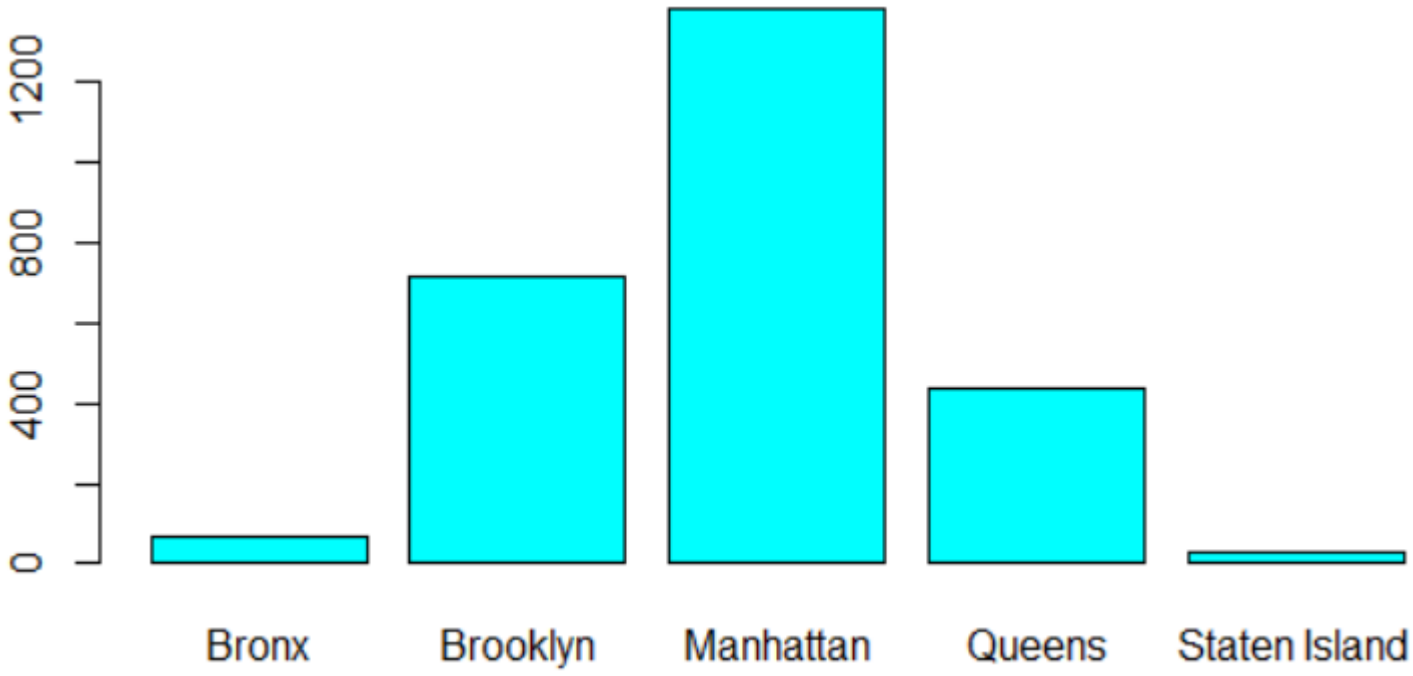
```
#(1) Loading Datasets from the Internet
web = "https://www.jaredlander.com/data/housing.csv"
house <- read.csv(web,header=TRUE)
str(house)
```

```
'data.frame':   2626 obs. of  13 variables:
 $ Neighborhood           : Factor w/ 151 levels "ALPHABET CITY",..: 45 45
 $ Building.Classification: Factor w/ 4 levels "R2-CONDOMINIUM",..: 3 2 4 2
 $ Total.Units            : int  42 78 500 282 239 133 109 107 247 121 ...
 $ Year.Built             : int  1920 1985 NA 1930 1985 1986 1985 1986 1987
 $ Gross.SqFt             : int  36500 126420 554174 249076 219495 139719 1
 $ Estimated.Gross.Income : int  1332615 6633257 17310000 11776313 10004582
 $ Gross.Income.per.SqFt  : num  36.5 52.5 31.2 47.3 45.6 ...
 $ Estimated.Expense      : int  342005 1762295 3543000 2784670 2783197 149
 $ Expense.per.SqFt       : num  9.37 13.94 6.39 11.18 12.68 ...
 $ Net.Operating.Income   : int  990610 4870962 13767000 8991643 7221385 36
 $ Full.Market.Value      : int  7300000 30690000 90970000 67556006 5432099...
 $ Market.Value.per.SqFt  : num  200 243 164 271 247 ...
 $ Boro                   : Factor w/ 5 levels "Bronx","Brooklyn",..: 3 3 3
```

```
# Save dataset to disk
write.csv(house,file="housing.csv",row.names=FALSE)
# Read dataset from disk
rhouse <- read.csv("housing.csv",header=TRUE)
names(rhouse)
```

'Neighborhood' 'Building.Classification' 'Total.Units' 'Year.Built' 'Gross.SqFt'
'Estimated.Gross.Income' 'Gross.Income.per.SqFt' 'Estimated.Expense' 'Expense.per.SqFt'
'Net.Operating.Income' 'Full.Market.Value' 'Market.Value.per.SqFt' 'Boro'

```
barplot(table(rhouse$Boro), col=5)
```



**download.file**(url, destfile, ... ) {utils}
The function download.file can be used to download a single file as described by url from the internet and store it in destfile. The url must start with a scheme such as http://, https://, ftp:// or file://.

```
# Download csv or txt file from internet
download.file('https://www.jaredlander.com/data/wine.csv',
              destfile='wine.csv')
wine <- read.csv('wine.csv', header=TRUE)
dim(wine)
```

```
178  14
```

```
names(wine)
```

```
'Cultivar'  'Alcohol'  'Malic.acid'  'Ash'  'Alcalinity.of.ash'  'Magnesium'
'Total.phenols'  'Flavanoids'  'Nonflavanoid.phenols'  'Proanthocyanins'
'Color.intensity'  'Hue'  'OD280.OD315.of.diluted.wines'  'Proline'
```

Reading the data with fread() is extremely quick compared to the read.csv().

**fread**(input, header, stringsAsFactors, ... ) {data.table}
Similar to read.table but faster and more convenient.

```
####### Read Big Data ######################
library(data.table)
file <- "flights_sep_oct15.csv"
system.time(test1 <- read.csv(file,header=TRUE))
system.time(test2 <- fread(file, stringsAsFactors=TRUE))
```

```
   user  system elapsed
  17.42    0.39   17.81

   user  system elapsed
   1.81    0.15    1.95
```

```
dim(test2)
```

```
951111  28
```

## (2) Text File

**read.table**(file, header=FALSE, ... ) / **write.table**(x, file, ... ) {utils}

```
url = "https://www.jaredlander.com/data/reactionFull.txt"
dat <- read.table(url, header=TRUE)
names(dat)
```

'ID' 'Test' 'Gender' 'Age' 'BMI' 'React' 'Regulate'

```
# Save dataset to disk
write.table(dat, file="relationFull.txt", row.names=FALSE)
# Read dataset from disk
rdat1 <- read.table("relationFull.txt",header=TRUE)
names(rdat1)
```

'ID' 'Test' 'Gender' 'Age' 'BMI' 'React' 'Regulate'

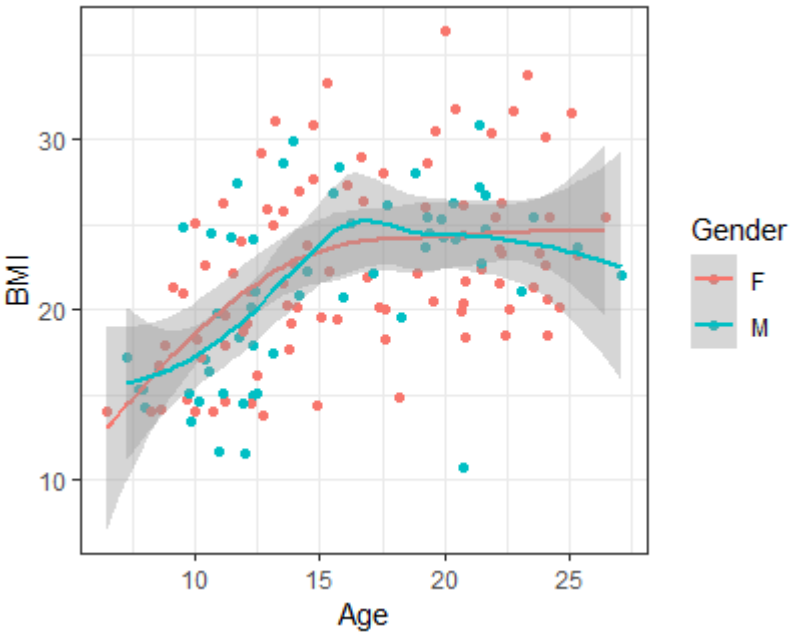
```
download.file('https://www.jaredlander.com/data/reaction.txt','reaction.txt')
rdat2 <- read.table("reaction.txt",header=TRUE)
head(rdat2,2)
```

| ID | Test | Age | Gender | BMI | React | Regulate |
|---|---|---|---|---|---|---|
| 1 | 1 | 9.69 | F | 14.71 | 4.17 | 3.15 |
| 1 | 2 | 12.28 | F | 14.55 | 3.89 | 2.55 |

```
rdat3 <- fread(url, stringsAsFactors=TRUE)
head(rdat3)
```

| ID | Test | Gender | Age | BMI | React | Regulate |
|---|---|---|---|---|---|---|
| 1 | 3 | F | 14.87 | 14.39 | 3.61 | 1.95 |
| 2 | 3 | F | 19.52 | 20.46 | -0.33 | -0.01 |
| 3 | 3 | F | 14.16 | 27.00 | 4.62 | 4.61 |
| 4 | 3 | M | 21.41 | 30.79 | 2.89 | 3.49 |
| 5 | 3 | M | 20.72 | 10.79 | 3.26 | 1.94 |
| 6 | 3 | M | 12.33 | 14.99 | 3.24 | 6.34 |

```
library(ggplot2)
ggplot(rdat3,aes(x=Age,y=BMI,color=Gender)) +
  geom_point() + stat_smooth() + theme_bw()
```



## (3) SPSS File

**read.spss**(file, use.value.labels=TRUE, to.data.frame=FALSE, . . .) {foreign}

| SPSS On-Line Training Workshop | |
|---|---|
| **Projects & Data Description** | **Data Download** |
| Cancer Data | Cancer.sav \|\| Cancer.xls |
| Cars Data | Cars.sav |

```
#data source:"http://calcnet.mth.cmich.edu/org/spss/Prjs_DataSets.htm"
library(foreign)
cancer <- read.spss(file="Cancer .sav",use.value.labels=TRUE,
                    to.data.frame=TRUE)
head(cancer)
```

| ID | TRT | AGE | WEIGHIN | STAGE | TOTALCIN | TOTALCW2 | TOTALCW4 | TOTALCW6 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 52 | 124.0 | 2 | 6 | 6 | 6 | 7 |
| 5 | 0 | 77 | 160.0 | 1 | 9 | 6 | 10 | 9 |
| 6 | 0 | 60 | 136.5 | 4 | 7 | 9 | 17 | 19 |
| 9 | 0 | 61 | 179.6 | 1 | 6 | 7 | 9 | 3 |
| 11 | 0 | 59 | 175.8 | 2 | 6 | 7 | 16 | 13 |
| 15 | 0 | 69 | 167.6 | 1 | 6 | 6 | 6 | 11 |

# 2. Web Data Extraction

## Web-related data:

- HTML (hypertext markup language): <table> format
- XML (extensible markup language) format, a tree-based document structure
- JSON (JavaScript Object Notation) is a data format that breaks the "rows-and-columns" paradigm
- Google spreadsheets: published as HTML
- API (Application programming interfaces)

## (1) Scraping HTML Tables with rvest

[Ref.] http://bradleyboehmke.github.io/2015/12/scraping-html-tables.html

HTML tables are contained within <table> tags.

In order to extract the tables from the BLS employment statistics webpage we first use the html_nodes() function to select the <table> nodes. In this example, html_nodes() captures 15 HTML tables.

```
> library(xml2); library(rvest)
> web <- read_html("http://www.bls.gov/web/empsit/cesbmart.htm")
> tbl <- html_nodes(web, "table")
> head(tbl,10)
```

```
{xml_nodeset (10)}
 [1] <table id="main-content-table"><tr>\n<td id="secondary-nav-td">\r\ ...
 [2] <table id="Table1" class="regular" cellspacing="0" cellpadding="0" ...
 [3] <table id="Table2" class="regular" cellspacing="0" cellpadding="0" ...
 [4] <table id="Table3" class="regular" cellspacing="0" cellpadding="0" ...
 [5] <table id="Table4" class="regular" cellspacing="0" cellpadding="0" ...
 [6] <table id="Table5" class="regular" cellspacing="0" cellpadding="0" ...
 [7] <table id="Table6" class="regular" cellspacing="0" cellpadding="0" ...
 [8] <table id="Table7" class="regular" cellspacing="0" cellpadding="0" ...
 [9] <table id="Table8" class="regular" cellspacing="0" cellpadding="0" ...
[10] <table id="Table9" class="regular">\n<caption><span class="tableTi ...
```

We want to parse the third table (Table 3) on the webpage:

- Table 3. Net birth/death estimates by industry supersector, April – December 2014 (in thousands)

# We can parse a html table into a data frame by using html_table().

```
> #Extract table 3
> tbl3 <- tbl %>% .[4] %>%
+   html_table(fill = TRUE)
> tbl3
```

```
[[1]]
                                                    CES Industry Code
1
                                                    CES Industry Code
2
                                                             00-000000
3
...
23                                                                0.4
24 Footnotes\r\n                  (1) With the 2018 benchmark, CES reconstructed seve
ral series. The effects of these reconstructions are included in this table. For m
ore information, see the Reconstructions section in the 2018 CES Benchmark Article
.\r\n                  (2) Absolute revision is less then 0.05 percent.
```

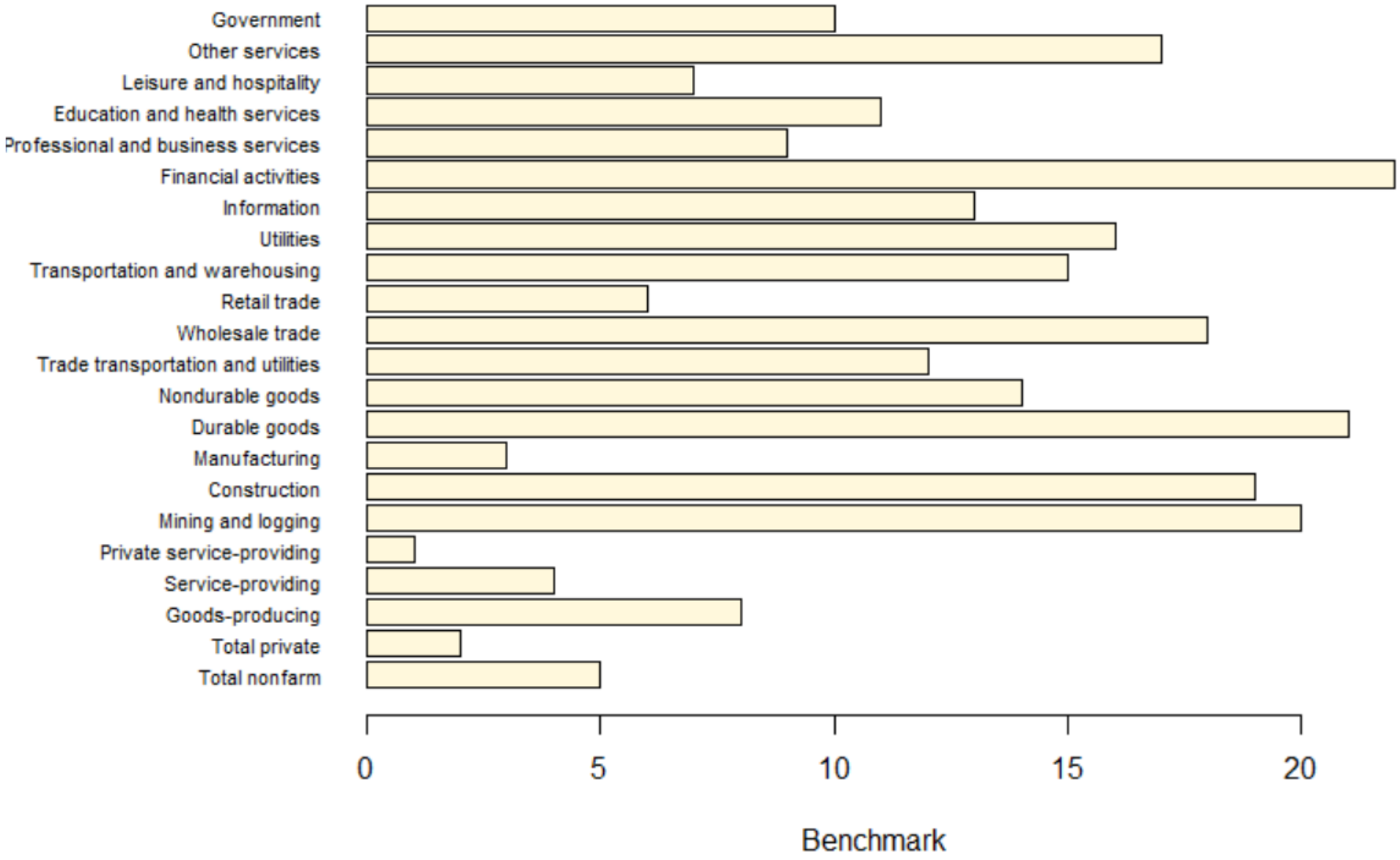
```
> #Remove rows 1 and 24 (headings and footnotes)
> table3 <- tbl3[[1]][-c(1,24),]
> head(table3,3)
```

| | CES Industry Code | CES Industry Title | Benchmark | Estimate(1) | Differences | Differences |
|---|---|---|---|---|---|---|
| 2 | 00-000000 | Total nonfarm | 147,368 | 147,384 | -16 | (2) |
| 3 | 05-000000 | Total private | 124,601 | 124,705 | -104 | -0.1 |
| 4 | 06-000000 | Goods-producing | 20,195 | 20,177 | 18 | 0.1 |

```
> #Remove all the commas(",")
> table3 = sapply(table3, FUN=function(x)
+   as.character(gsub(",","", as.character(x), fixed=TRUE)))
> #Rename table headings
> colnames(table3) <- c("CES_Code", "Ind_Title", "Benchmark",
+                       "Estimate", "Amt_Diff", "Pct_Diff")
> #Create data.frame
> table3_df = as.data.frame(table3)
> head(table3_df)
```

| | CES_Code | Ind_Title | Benchmark | Estimate | Amt_Diff | Pct_Diff |
|---|---|---|---|---|---|---|
| 1 | 00-000000 | Total nonfarm | 147368 | 147384 | -16 | (2) |
| 2 | 05-000000 | Total private | 124601 | 124705 | -104 | -0.1 |
| 3 | 06-000000 | Goods-producing | 20195 | 20177 | 18 | 0.1 |
| 4 | 07-000000 | Service-providing | 127173 | 127207 | -34 | (2) |
| 5 | 08-000000 | Private service-providing | 104406 | 104528 | -122 | -0.1 |
| 6 | 10-000000 | Mining and logging | 704 | 712 | -8 | -1.1 |

```
#Barplot of Benchmark variable
par(mar=c(4.3,10,1,1))
barplot(as.numeric(table3_df[,3]), names.arg=table3_df[,2],
        horiz=TRUE, col='cornsilk', las=1,
        cex.names=0.7, xlab='Benchmark')
```



# (2) Scraping HTML Tables with XML

apps.saferoutesinfo.org/legislation_funding/state_apportionment.cfm

The table below provides state-specific apportionment information related to funding provided through the SAFETEA-LU transportation legislation. (Please note: the National Center updates this table as necessary.)

## State Apportionment Table

| State | Actual 2005 | Actual 2006* | Actual 2007 | Actual 2008 | Actual 2009 | Actual 2010 | Actual 2011 | Actual 2012 | Total |
|---|---|---|---|---|---|---|---|---|---|
| Alabama | $1,000,000 | $1,313,659 | $1,767,375 | $2,199,717 | $2,738,816 | $2,738,816 | $2,994,316 | $2,556,869 | $17,309,568 |
| Alaska | $1,000,000 | $990,000 | $1,000,000 | $1,000,000 | $1,000,000 | $1,000,000 | $1,554,670 | $933,567 | $8,478,237 |
| Arizona | $1,000,000 | $1,557,644 | $2,228,590 | $2,896,828 | $3,612,384 | $3,612,384 | $3,733,355 | $3,372,404 | $22,013,589 |

**htmlParse**(file, ... ) {XML}

Parses an XML or HTML file or string containing XML/HTML content, and generates an R structure representing the XML/HTML tree.

```
> #Load HTML table
> library(XML)
> u="http://apps.saferoutesinfo.org/legislation_funding/state_apportionment.cfm"
> w <- htmlParse(u)
> class(w)
```

```
[1] "HTMLInternalDocument" "HTMLInternalDocument" "XMLInternalDocument"
[4] "XMLAbstractDocument"
```

**readHTMLTable**(doc, ... ) {XML} Read data from one or more HTML tables

This function provide somewhat robust methods for extracting data from HTML tables in an HTML document.

```
> w.table <- readHTMLTable(w, stringsAsFactors=FALSE)
> w.table
```

```
$`NULL`
   State\n\t\t\t  \t\t           Actual 2005\n\t\t\t  \t\t
1                        Alabama                                $1,000,000
2                         Alaska                                $1,000,000
3                        Arizona                                $1,000,000
4                       Arkansas                                $1,000,000
5                     California                                $1,000,000
6                       Colorado                                $1,000,000
7                    Connecticut                                $1,000,000
8                       Delaware                                $1,000,000
9                  Dist. of Col.                                $1,000,000
10                       Florida                                $1,000,000
...
41                     $1,186,047                                $1,584,924
42                       $990,000                                $1,000,000
43                     $1,596,222                                $2,158,074
44                     $7,009,094                                $9,408,067
45                       $990,000                                $1,063,690
46                       $990,000                                $1,000,000
47                     $2,024,830                                $2,717,436
48                     $1,694,515                                $2,271,034
49                       $990,000                                $1,000,000
50                     $1,554,314                                $2,048,636
51                       $990,000                                $1,000,000
52                    $96,030,000                              $122,000,000
```

The next steps involve processing the table into a usable format.

```
> # remove all the commas
> money <- sapply(w.table[[1]][,-1], FUN= function(x)
+   as.character(gsub(",", "", as.character(x), fixed=TRUE)))
> #remove all the leading $ signs
> money <- as.data.frame(substring(money,2), stringsAsFactors=FALSE)
> head(money,2)
```

```
  Actual 2005\n\t\t\t  \t\t           Actual 2006*\n\t\t\t  \t\t
1                             1000000                                1313659
2                             1000000                                 990000
...
```

```
> #rename the variables
> names(money) <- c("Actual2005","Actual2006","Actual2007","Actual2008",
+                   "Actual2009","Actual2010","Actual2011","Actual2012","Total")
> #move the state names to the first column
> money$State <- w.table[[1]][,1]
> money <- money[,c(10,1:9)]
> head(money,2)
```

```
    State Actual2005 Actual2006 Actual2007 Actual2008 Actual2009 Actual2010 Actual2011
1 Alabama    1000000    1313659    1767375    2199717    2738816    2738816    2994316
2  Alaska    1000000     990000    1000000    1000000    1000000    1000000    1554670
  Actual2012    Total
1    2556869 17309568
2     933567  8478237
```

```
> tail(money,2)
```

```
     State Actual2005 Actual2006 Actual2007 Actual2008 Actual2009 Actual2010 Actual2011
51 Wyoming    1000000     990000    1000000    1000000    1000000    1000000    1083988
52   Total   51000000   96030000  122000000  147000000  180000000  180000000  202439733
   Actual2012      Total
51     933567    8007555
52  168042127 1146511860
```

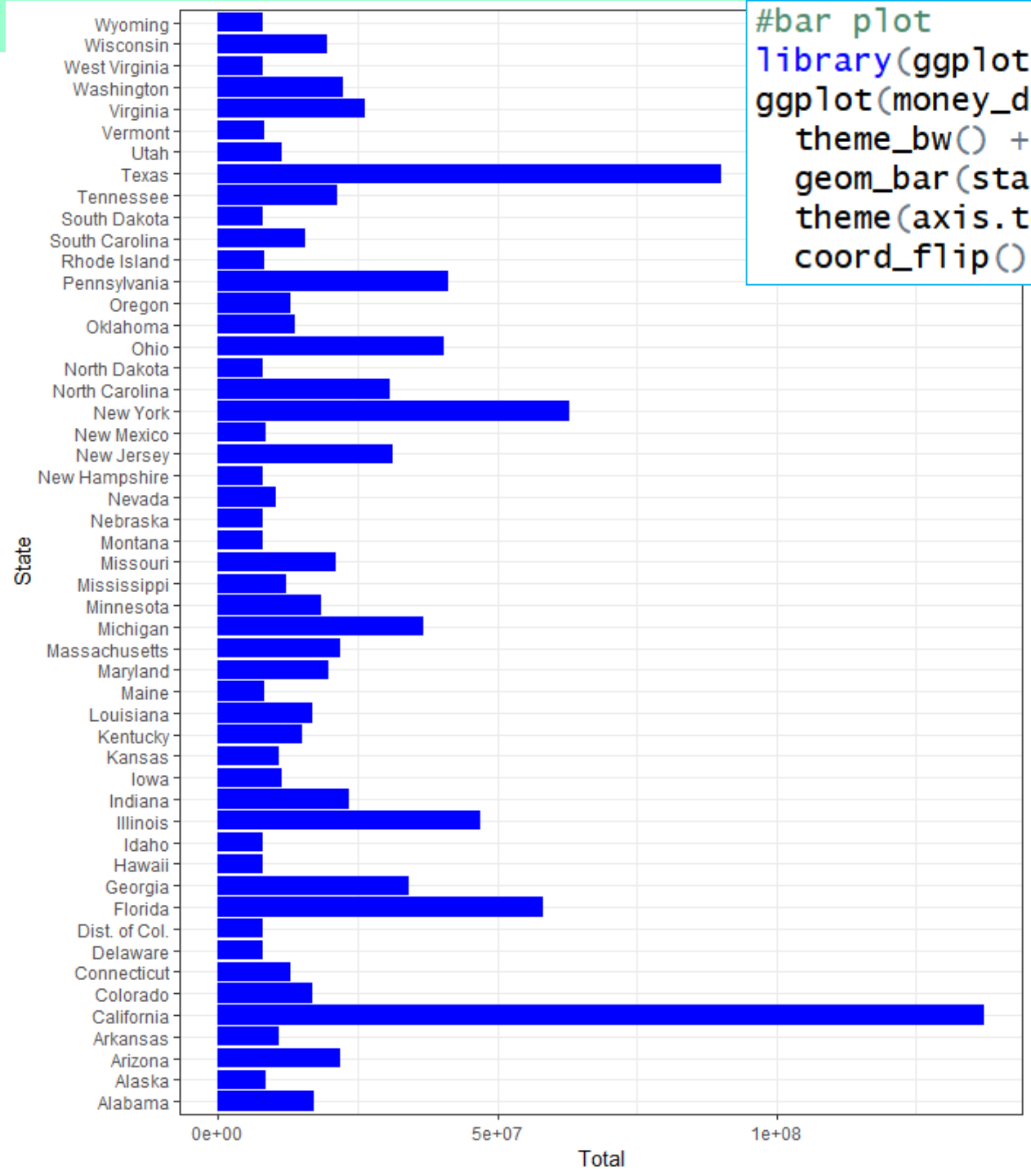
# Check data type and create a new data frame

```
> str(money)
'data.frame':   52 obs. of  10 variables:
 $ State     : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
 $ Actual2005: chr  "1000000" "1000000" "1000000" "1000000" ...
 $ Actual2006: chr  "1313659" "990000" "1557644" "990000" ...
 $ Actual2007: chr  "1767375" "1000000" "2228590" "1027338" ...
 $ Actual2008: chr  "2199717" "1000000" "2896828" "1297202" ...
 $ Actual2009: chr  "2738816" "1000000" "3612384" "1622447" ...
 $ Actual2010: chr  "2738816" "1000000" "3612384" "1622447" ...
 $ Actual2011: chr  "2994316" "1554670" "3733355" "1911273" ...
 $ Actual2012: chr  "2556869" "933567" "3372404" "1514664" ...
 $ Total     : chr  "17309568" "8478237" "22013589" "10985371" ...
```

```
> money[52,]
    State Actual2005 Actual2006 Actual2007 Actual2008 Actual2009 Actual2010
52 Total   51000000   96030000  122000000  147000000  180000000  180000000
   Actual2011 Actual2012      Total
52  202439733  168042127 1146511860
```

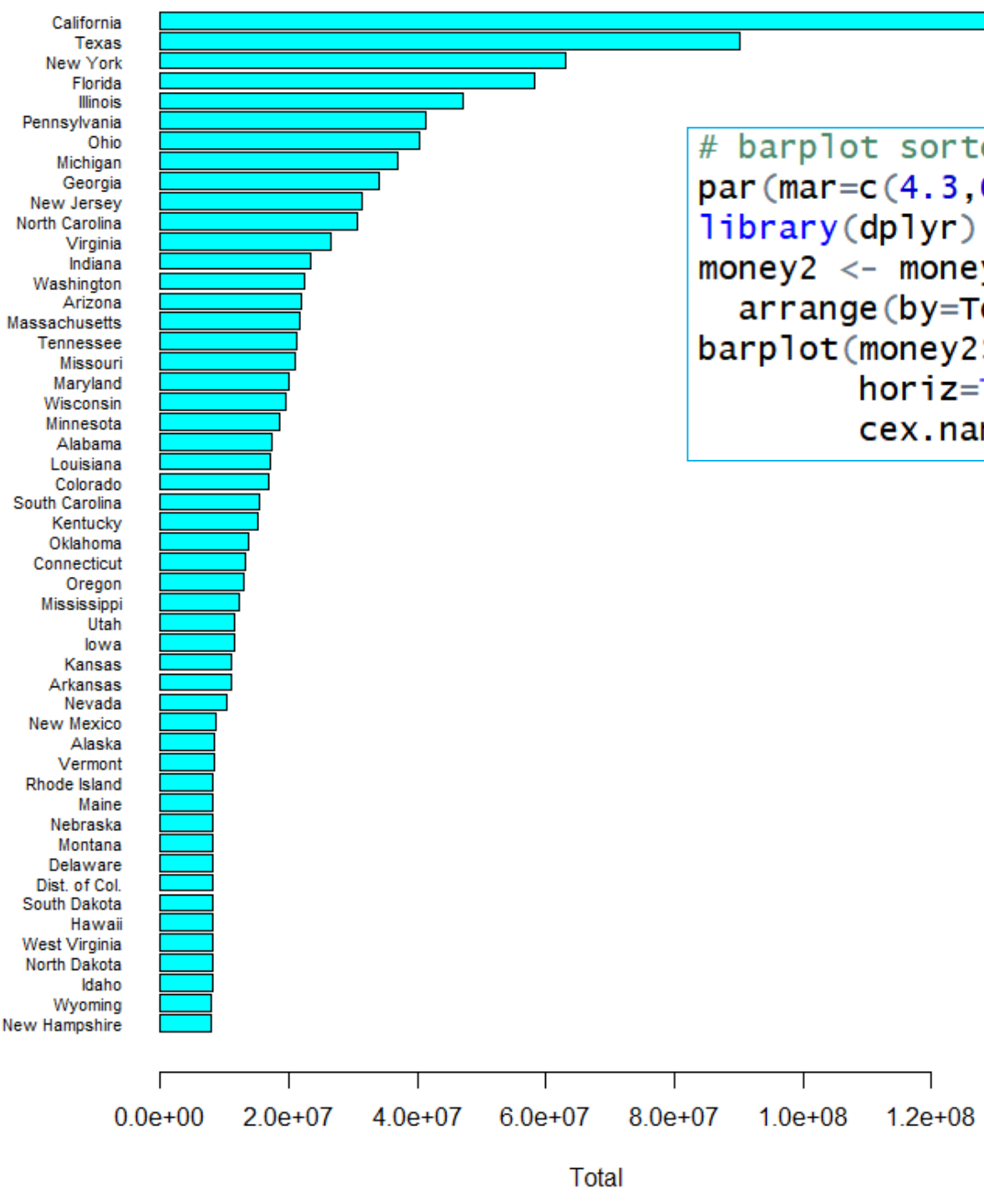
```
> #total(52 row) will be excluded
> money_dat <- data.frame(State=money[-52,1],
+                         Total=as.numeric(money[-52,10]))
> head(money_dat,4)
     State    Total
1  Alabama 17309568
2   Alaska  8478237
3  Arizona 22013589
4 Arkansas 10985371
```

```
> tail(money_dat,4)
           State    Total
48    Washington 22469209
49 West Virginia  8090697
50     Wisconsin 19526738
51       Wyoming  8007555
```




```r
#bar plot
library(ggplot2)
ggplot(money_dat,aes(State,Total)) +
  theme_bw() +
  geom_bar(stat="identity",fill="blue") +
  theme(axis.text.x=element_text()) +
  coord_flip()
```

```r
# barplot sorted by Total
par(mar=c(4.3,6,1,1))
library(dplyr)
money2 <- money_dat %>%
  arrange(by=Total)
barplot(money2$Total, names.arg=money2$State,
        horiz=TRUE, col='cyan', las=1,
        cex.names=0.7, xlab='Total')
```



# 3. Google Documents and Analytics

## (1) Google Search Trend in R

Google Trends shows the changes in the popularity of search terms over a given time.

**gtrends**(keyword, geo, time, gprop, ... ) {gtrendsR}
The gtrends default method performs a Google Trends query for the 'query' argument and session 'session'. Optional arguments for geolocation and category can also be supplied.

```
> ##(1) Online Google search trends
> library(gtrendsR)
> hs_trend <- gtrends(keyword=c("Happy","Sad"),
+                     geo=c("KR", "KR"),
+                     time="now 7-d")
> names(hs_trend)
```

```
[1] "interest_over_time"  "interest_by_country" "interest_by_region"
[4] "interest_by_dma"     "interest_by_city"    "related_topics"
[7] "related_queries"
```

```
> head(hs_trend$interest_over_time)
```

```
                 date hits keyword geo gprop category
1 2019-02-08 03:00:00    9   Happy  KR   web        0
2 2019-02-08 04:00:00   12   Happy  KR   web        0
3 2019-02-08 05:00:00    8   Happy  KR   web        0
4 2019-02-08 06:00:00   10   Happy  KR   web        0
5 2019-02-08 07:00:00   12   Happy  KR   web        0
6 2019-02-08 08:00:00    9   Happy  KR   web        0
```

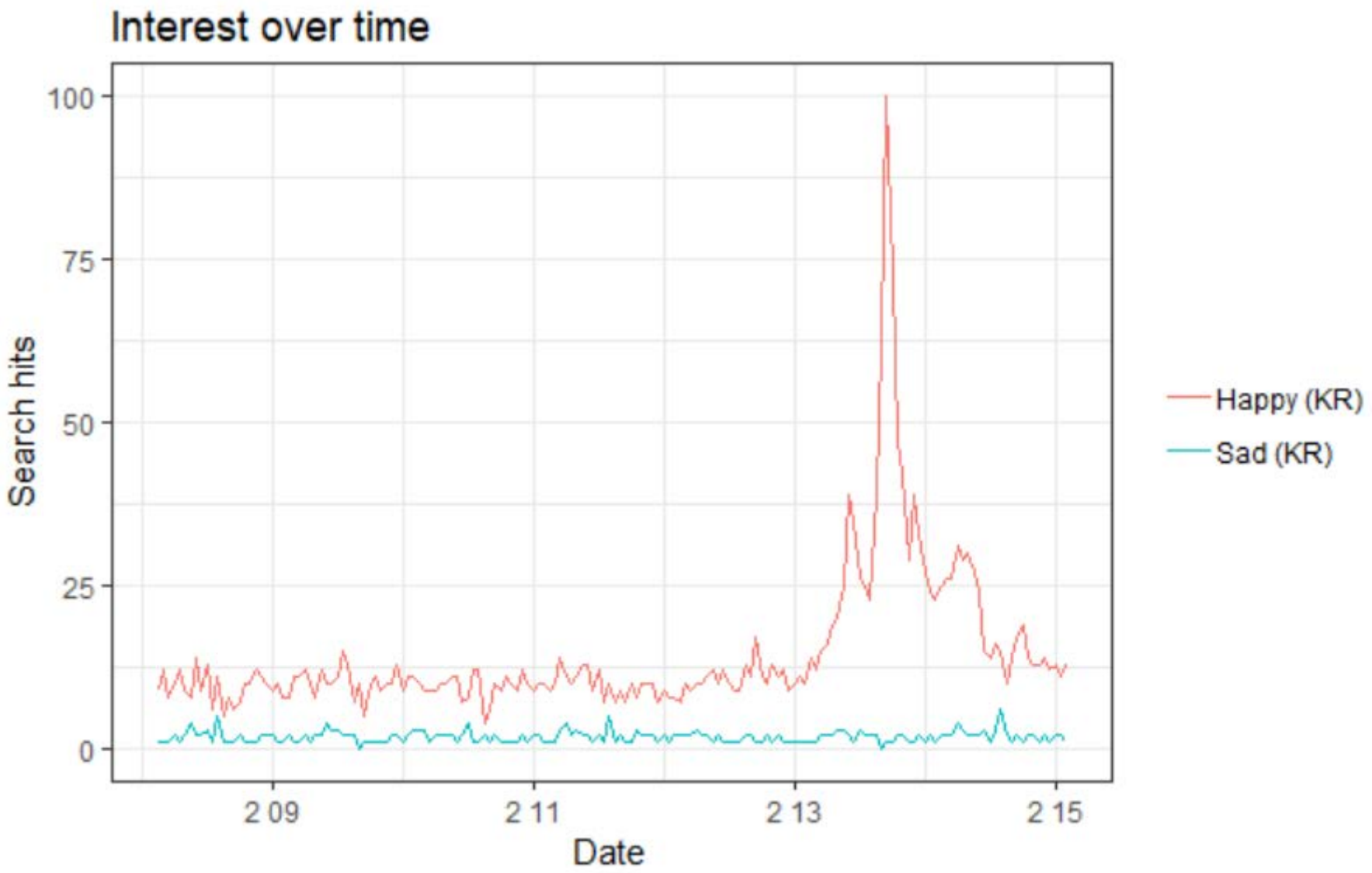
```
> ##(1) Online Google search trends
> library(gtrendsR)
> hs_trend <- gtrends(keyword=c("Happy","Sad"),
+                     geo=c("KR", "KR"),
+                     time="now 7-d")
> names(hs_trend)
```

```
head(hs_trend$interest_over_time)
head(hs_trend[[1]])
head(hs_trend[[3]]) # interest_by_region
head(hs_trend[[5]]) # interest_by_city
head(hs_trend[[7]]) # related_queries
plot(hs_trend)
```

**Arguments**

| | |
|---|---|
| keyword | A character vector with the actual Google Trends query keywords. Multiple keywords are possible using `gtrends(c("NHL","NBA", "MLB", "MLS"))`. |
| geo | A character vector denoting geographic regions for the query, default to "all" for global queries. Multiple regions are possible using `gtrends("NHL", c("CA", "US"))`. |
| time | A string specifying the time span of the query. Possible values are:<br>**"now 1-H"** Last hour<br>**"now 4-H"** Last four hours<br>**"now 1-d"** Last day<br>**"now 7-d"** Last seven days<br>**"today 1-m"** Past 30 days<br>**"today 3-m"** Past 90 days<br>**"today 12-m"** Past 12 months<br>**"today+5-y"** Last five years (default)<br>**"all"** Since the beginning of Google Trends (2004)<br>**"Y-m-d Y-m-d"** Time span between two dates (ex.: "2010-01-01 2010-04-03") |
| gprop | A character string defining the Google product for which the trend query if pre-formed. Valid options are:<br>• "web" (default)<br>• "news"<br>• "images"<br>• "froogle"<br>• "youtube" |

```
plot(hs_trend)
```

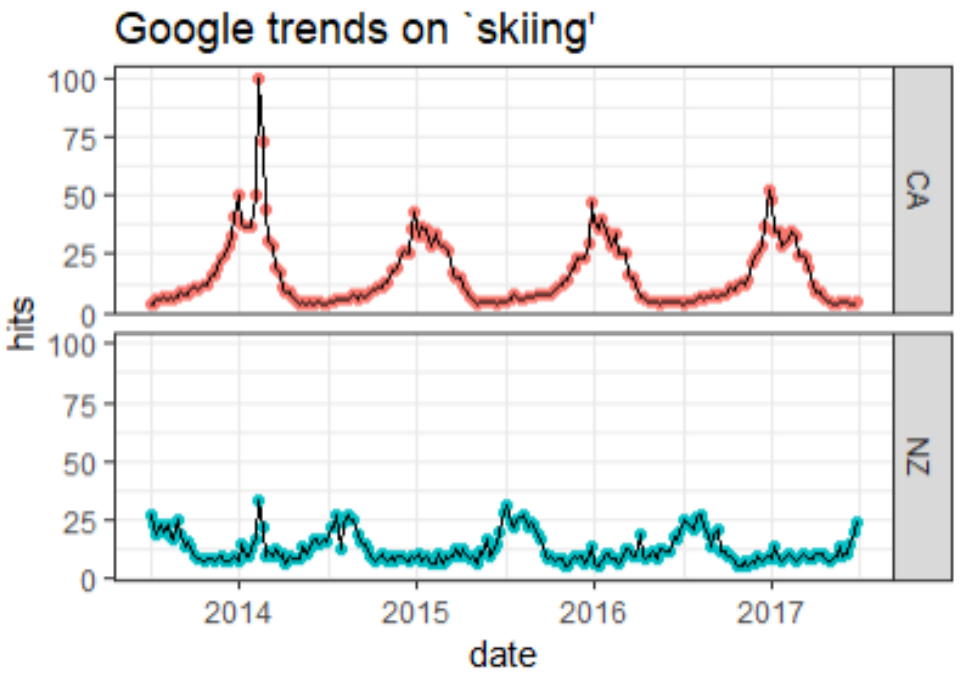


| | date | hits | keyword | geo | gprop | category |
|---|---|---|---|---|---|---|
| 1 | 2019-02-08 03:00:00 | 9 | Happy | KR | web | 0 |
| 2 | 2019-02-08 04:00:00 | 12 | Happy | KR | web | 0 |
| 3 | 2019-02-08 05:00:00 | 8 | Happy | KR | web | 0 |
| 4 | 2019-02-08 06:00:00 | 10 | Happy | KR | web | 0 |
| 5 | 2019-02-08 07:00:00 | 12 | Happy | KR | web | 0 |
| 6 | 2019-02-08 08:00:00 | 9 | Happy | KR | web | 0 |

reshape2

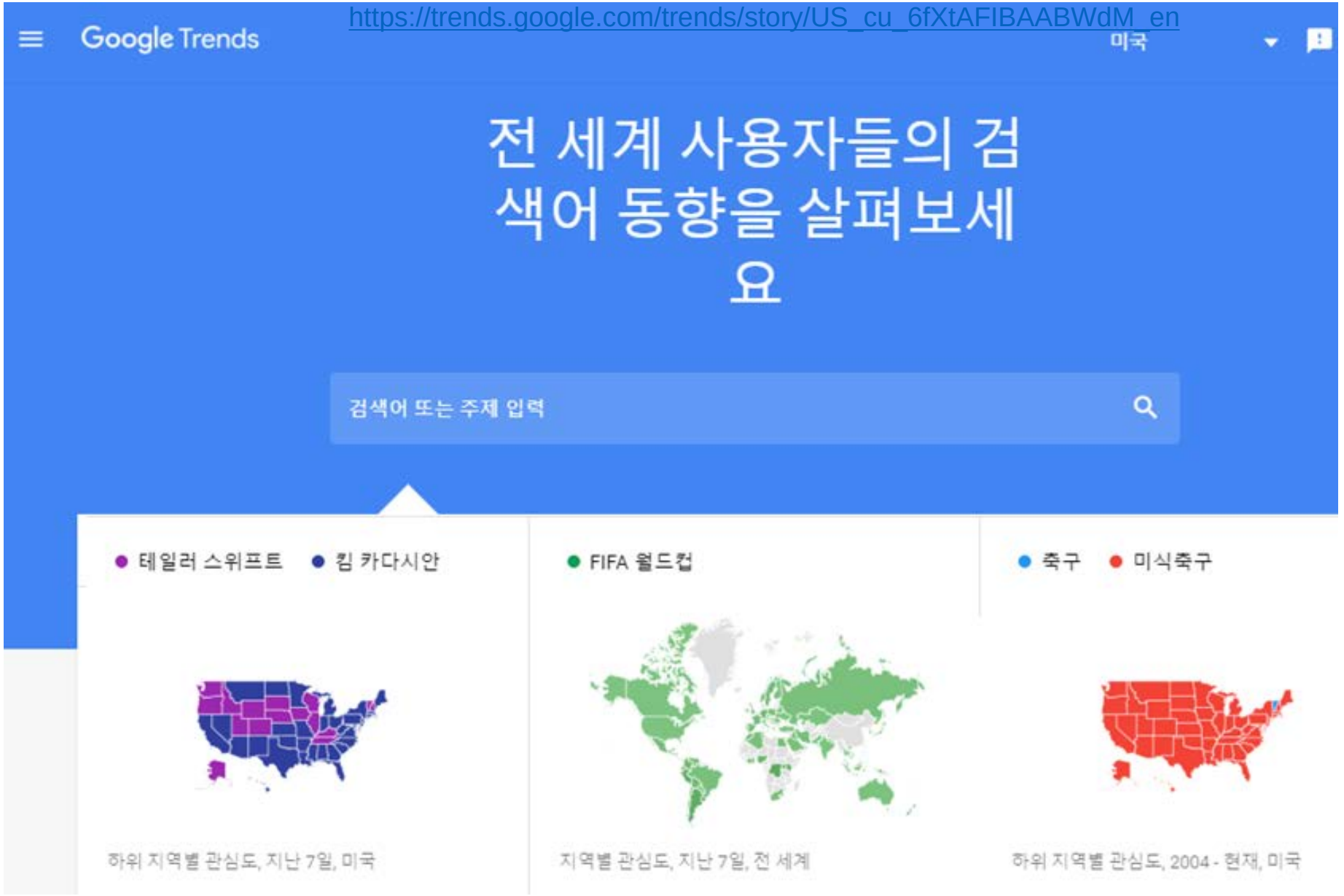
```
> library(reshape2)
> hs_trend2 = dcast(hs_trend[[1]], date ~ keyword + geo, value.var = "hits")
> rownames(hs_trend2) = hs_trend2$date
> hs_trend2$date = NULL
> head(hs_trend2)
                    Happy_KR Sad_KR
2019-02-27 00:00:00       64     11
2019-02-27 01:00:00       63     24
2019-02-27 02:00:00       64     24
2019-02-27 03:00:00       78     11
2019-02-27 04:00:00       66     11
2019-02-27 05:00:00       67     19
```

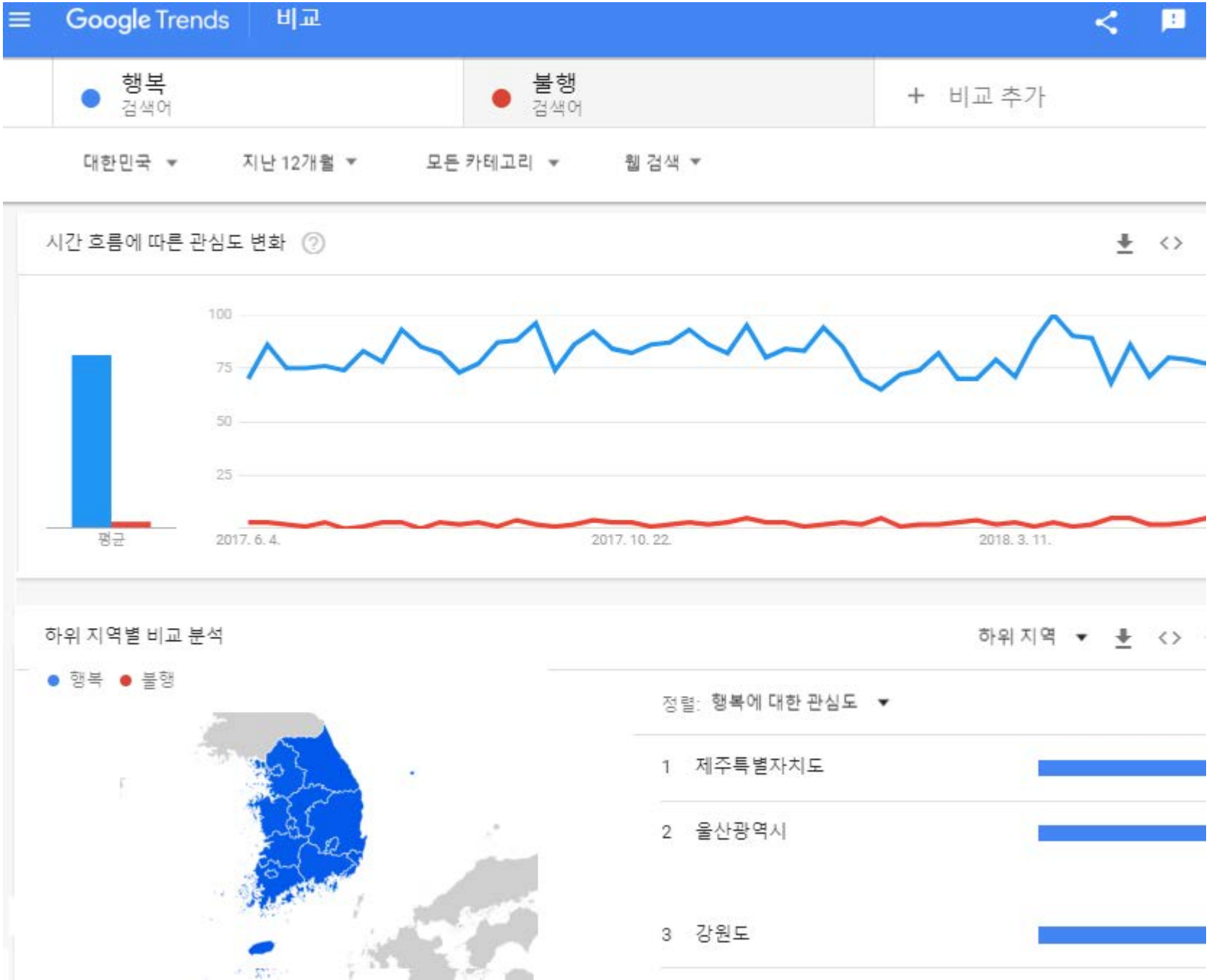
```
> google.trends = gtrends(c("skiing"), geo = c("CA", "NZ"),
+                         gprop = "web", time = "2013-06-30 2017-06-30")[[1]]
> head(google.trends)
        date hits keyword geo gprop category
1 2013-06-30    4  skiing  CA   web        0
2 2013-07-07    4  skiing  CA   web        0
3 2013-07-14    6  skiing  CA   web        0
4 2013-07-21    6  skiing  CA   web        0
5 2013-07-28    7  skiing  CA   web        0
6 2013-08-04    6  skiing  CA   web        0
> ggplot(google.trends, aes(x=date, y=hits, fill=geo )) +
+   geom_point(aes(color=geo)) +
+   geom_line() +
+   facet_grid( geo ~ .) +
+   ggtitle("Google trends on `skiing'") +
+   theme(legend.position = "none") +
+   theme_bw()
```



# (2) Google Trends in Google Web Page

Google web page https://trends.google.com/trends/?geo=US




Google Trends
비교
행복
검색어
불행
검색어
+ 비교 추가
대한민국
지난 12개월
모든 카테고리
웹 검색
시간 흐름에 따른 관심도 변화
100
75
50
25
평균
2017. 6. 4.
2017. 10. 22.
2018. 3. 11.
하위 지역별 비교 분석
하위 지역
행복
불행
정렬: 행복에 대한 관심도
1 제주특별자치도
2 울산광역시
3 강원도