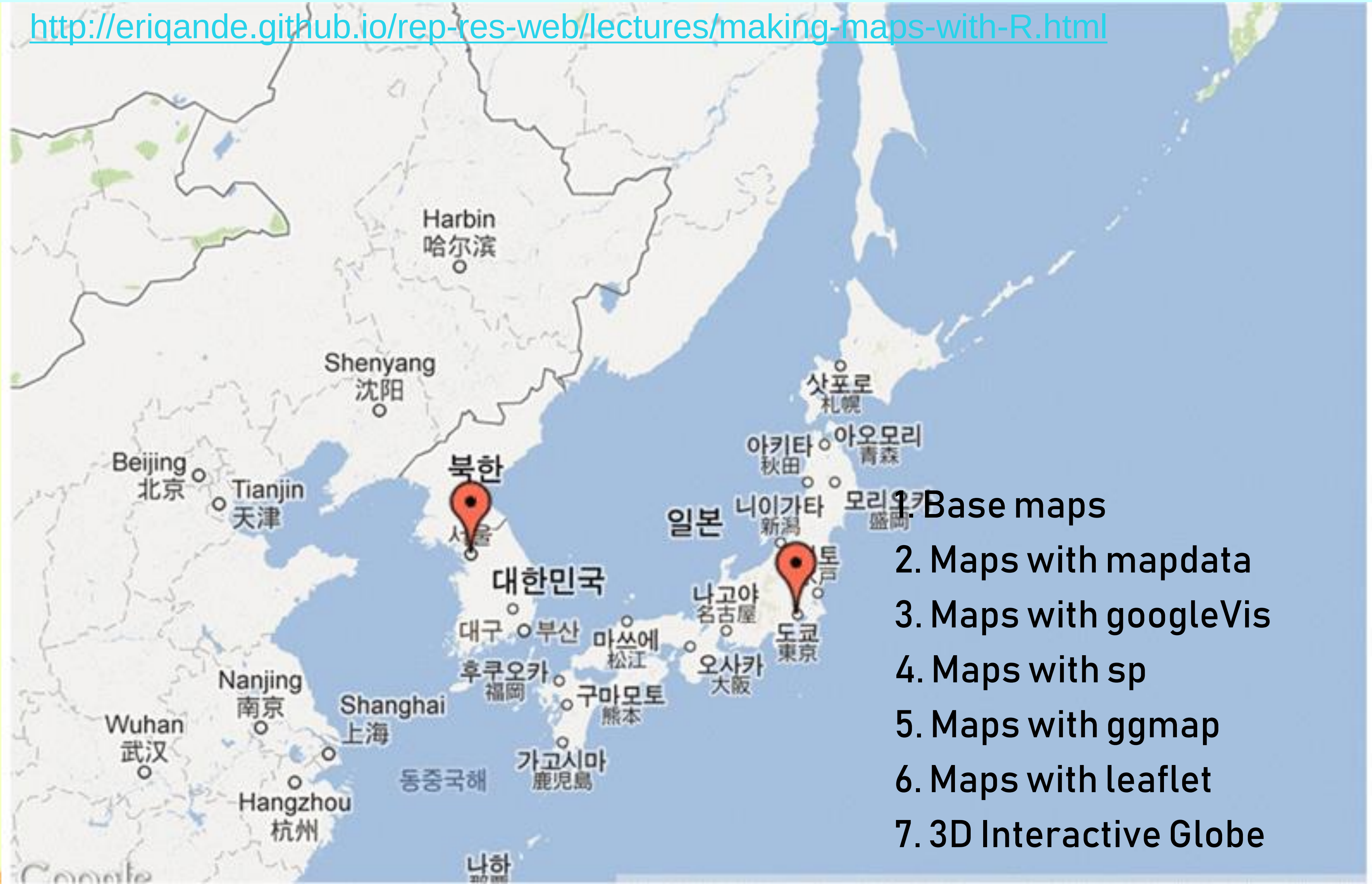
Maps in R

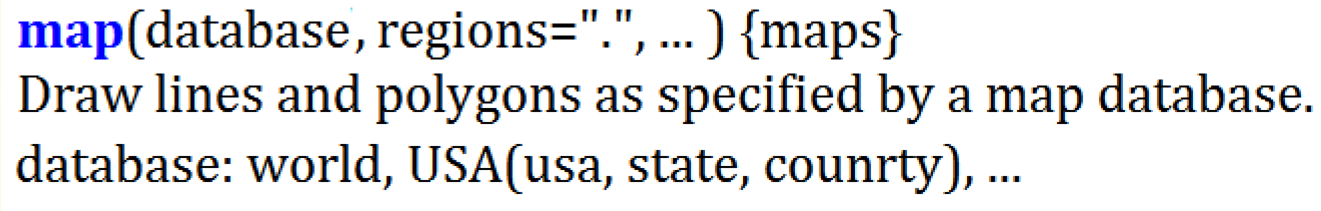
http://eriqande.github.io/rep-res-web/lectures/making-maps-with-R.html



1. Base maps
2. Maps with mapdata
3. Maps with googleVis
4. Maps with sp
5. Maps with ggmap
6. Maps with leaflet
7. 3D Interactive Globe

1. Base maps

The **maps** package provides a means of constructing **base maps** for plotting the locations of points, which can be decorated with text, symbols, and so on.



map(database, regions=".", ...) {maps}

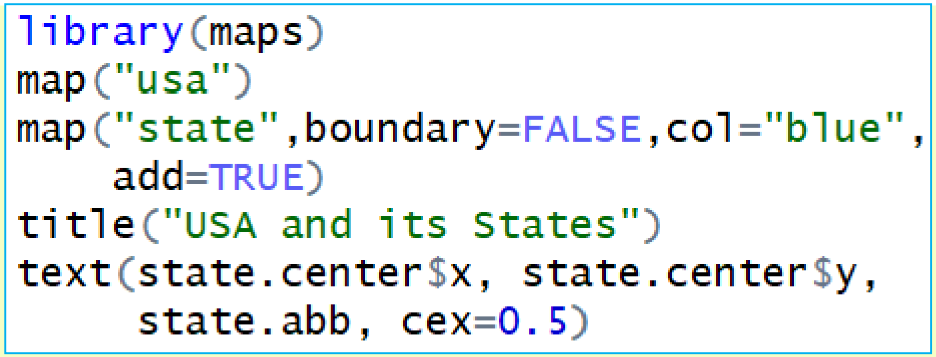
Draw lines and polygons as specified by a map database.

database: world, USA(usa, state, counrty), ...

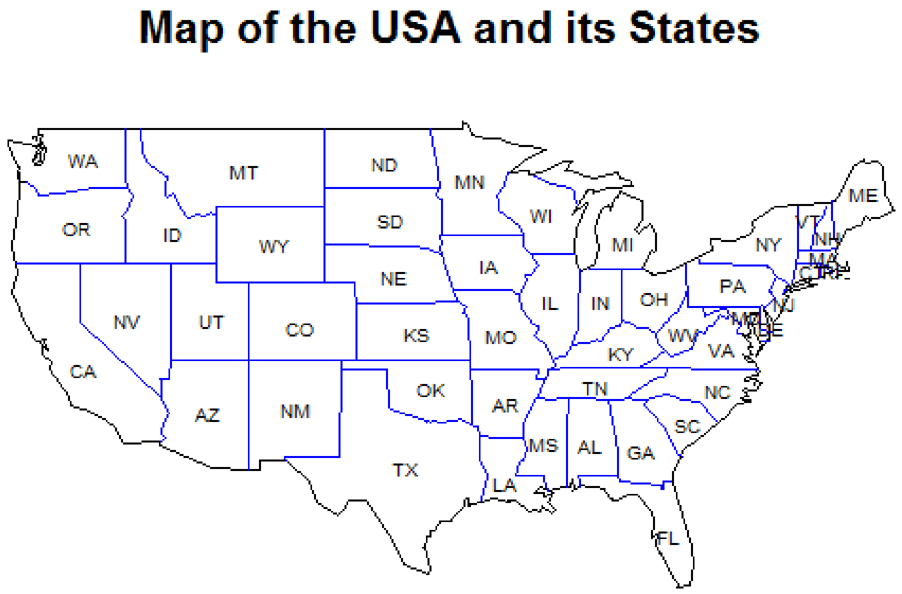
Example codes:

```
map('usa')        # national boundaries
map('county', 'new jersey')  # county map of New Jersey
map('state', region = c('new york', 'new jersey', 'penn')) # map of three states
```

[Example 1] Map of the USA and its States



```
library(maps)
map("usa")
map("state",boundary=FALSE,col="blue",
    add=TRUE)
title("USA and its States")
text(state.center$x, state.center$y,
     state.abb, cex=0.5)
```



Map of the USA and its States

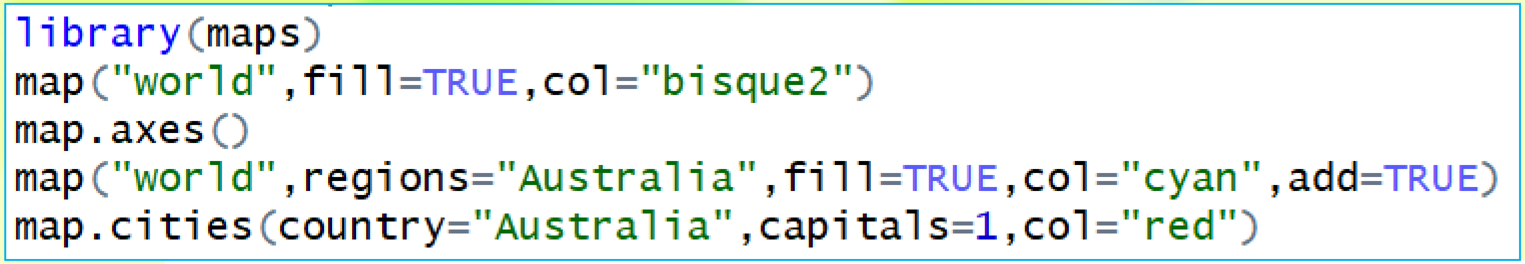
map.axes(...) {maps}

Draws a set of axes on an existing map.

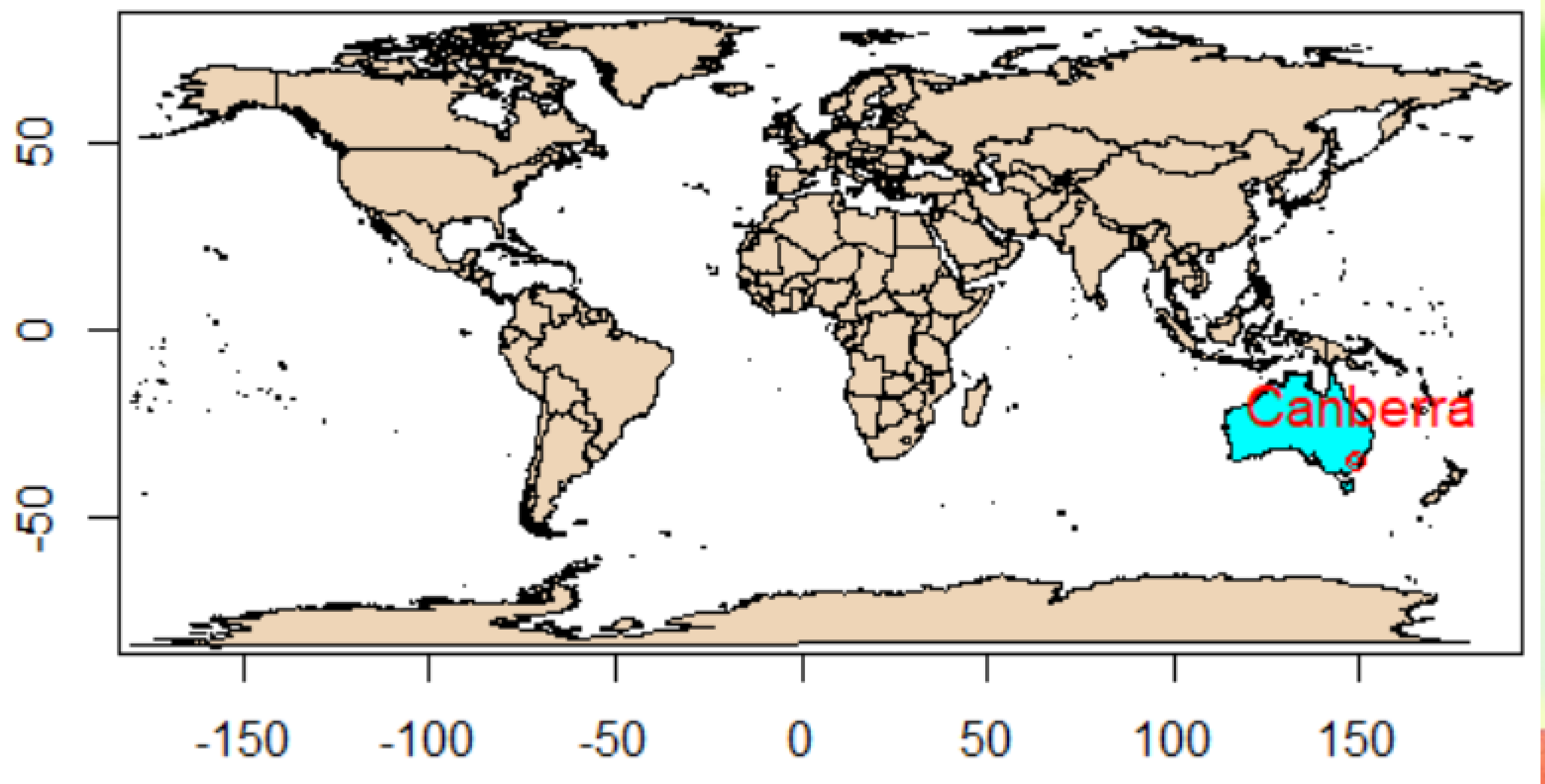
map.cities(x=world.cities, country, ...) {maps}

Adds city locations and (optionally) names to an existing map using a specified database.

[Example 2] World Map

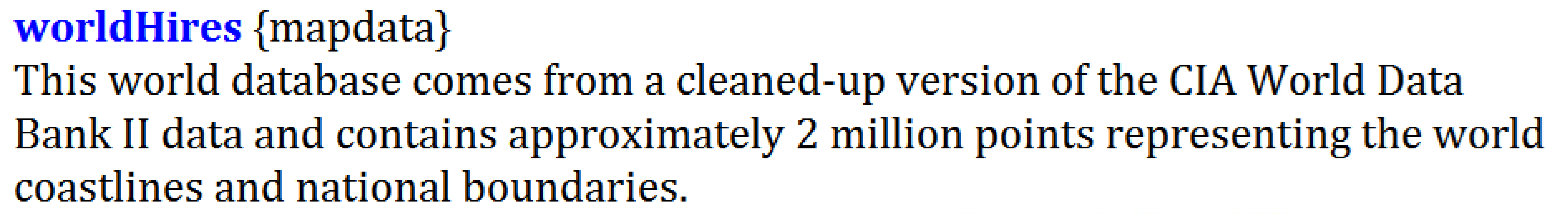


```
library(maps)
map("world",fill=TRUE,col="bisque2")
map.axes()
map("world",regions="Australia",fill=TRUE,col="cyan",add=TRUE)
map.cities(country="Australia",capitals=1,col="red")
```



2. Maps with mapdata

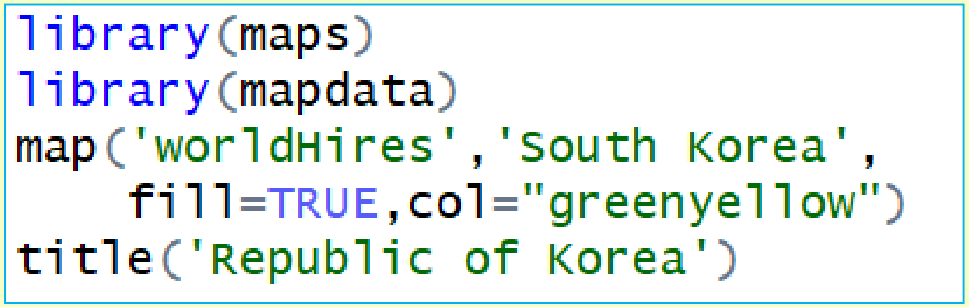
Supplement to maps package, providing the **larger** and **higher-resolution** databases.



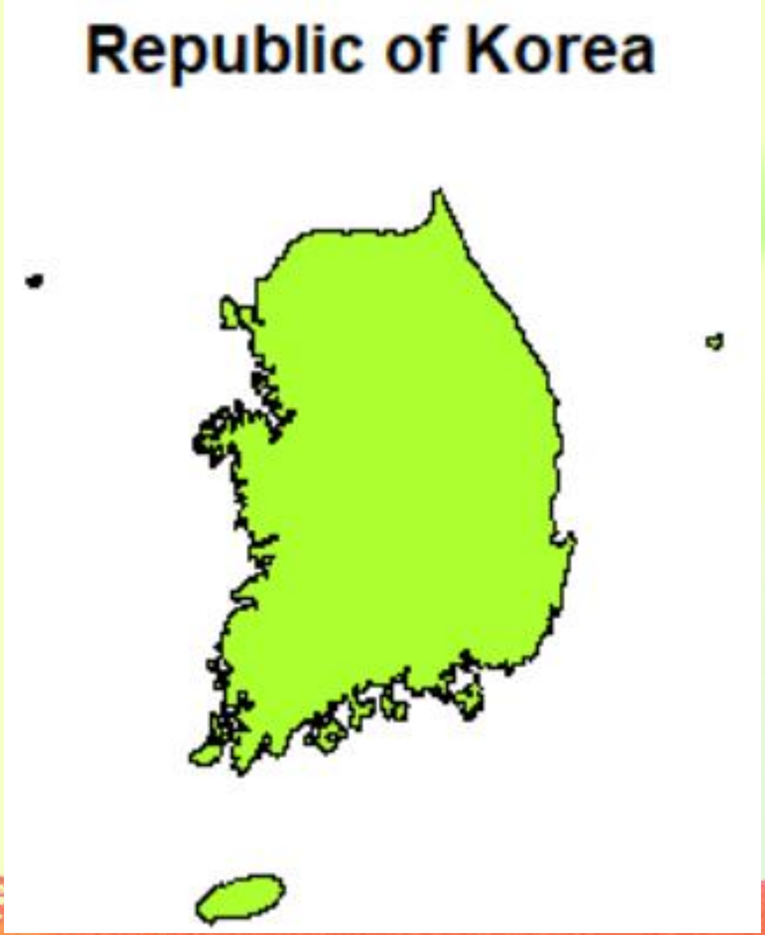
worldHires {mapdata}

This world database comes from a cleaned-up version of the CIA World Data Bank II data and contains approximately 2 million points representing the world coastlines and national boundaries.

[Example 1] Republic of Korea

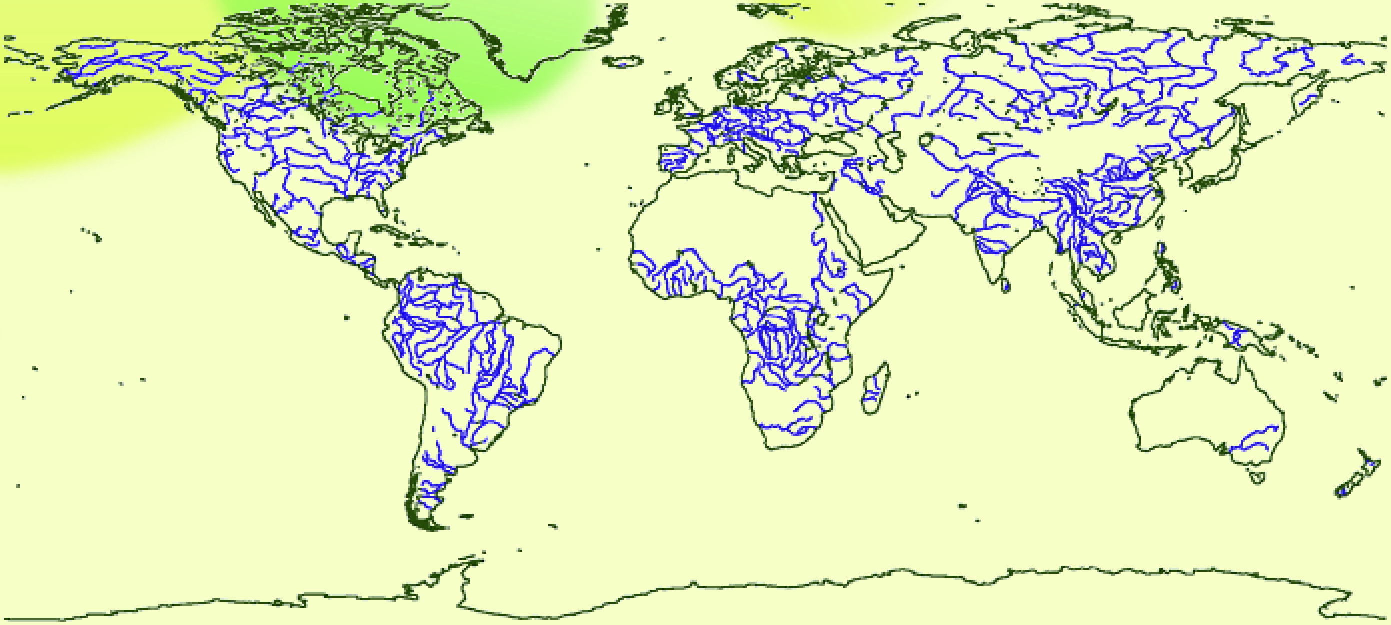


```
library(maps)
library(mapdata)
map('worldHires','South Korea',
    fill=TRUE,col="greenyellow")
title('Republic of Korea')
```



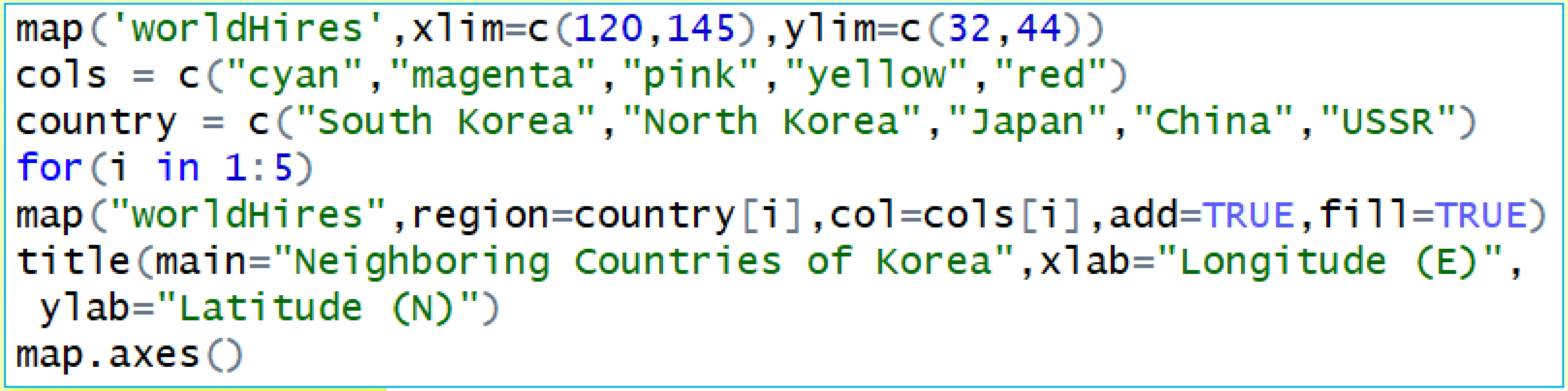
The CIA World Data Bank II

[*Gorny and Carter*, 1987]

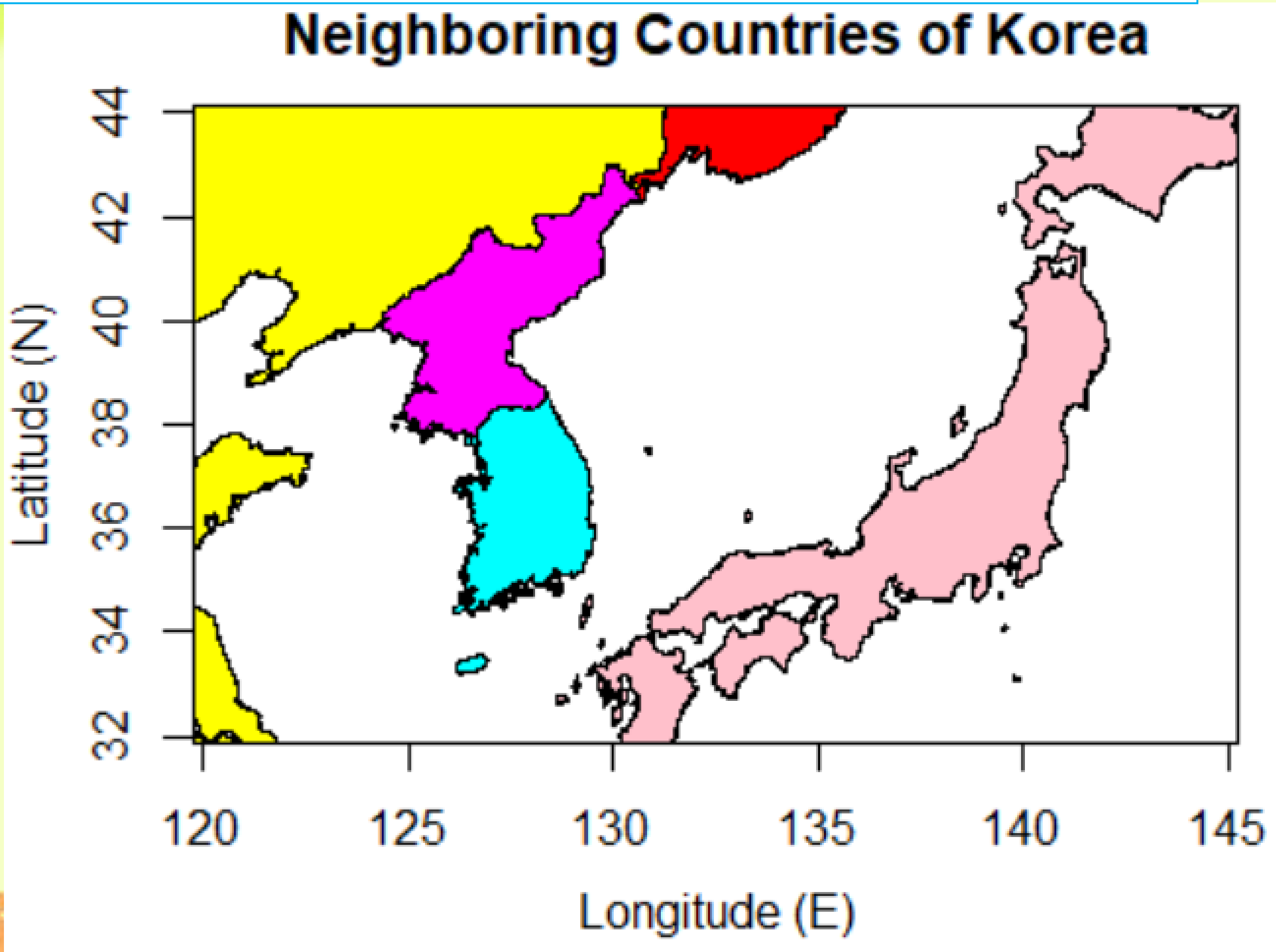


The CIA World DataBank II is a collection of world map data, consisting of vector descriptions of land outlines, rivers, and political boundaries. It was created by the U.S. government in the 1980s.

[Example 2] Neighboring Countries of Korea



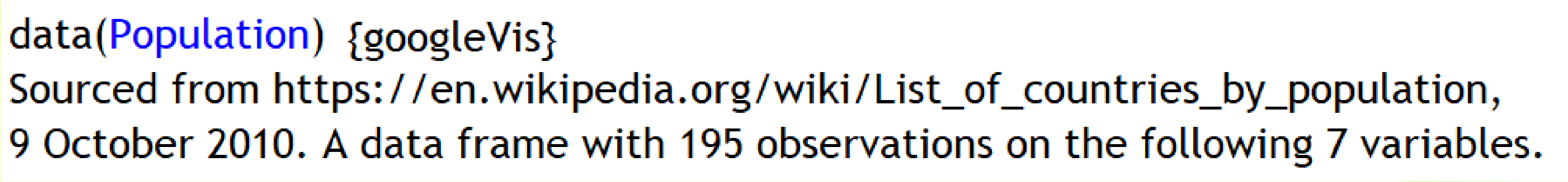
```
map('worldHires',xlim=c(120,145),ylim=c(32,44))
cols = c("cyan","magenta","pink","yellow","red")
country = c("South Korea","North Korea","Japan","China","USSR")
for(i in 1:5)
map("worldHires",region=country[i],col=cols[i],add=TRUE,fill=TRUE)
title(main="Neighboring Countries of Korea",xlab="Longitude (E)",
 ylab="Latitude (N)")
map.axes()
```



3. Maps with googleVis

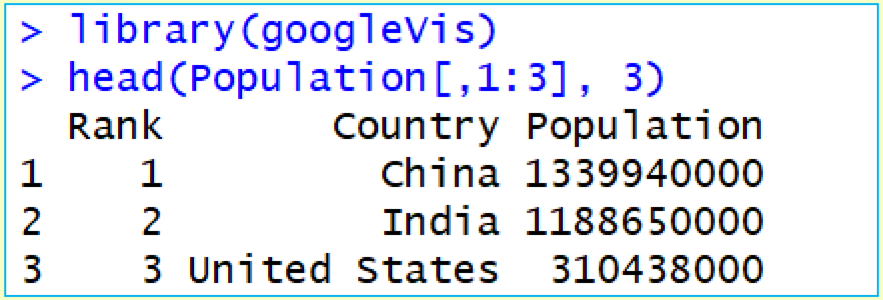
The functions of the gogleVis package allow the user to visualize data stored in R data frames with Google Charts.

[Example 1] World population in millions

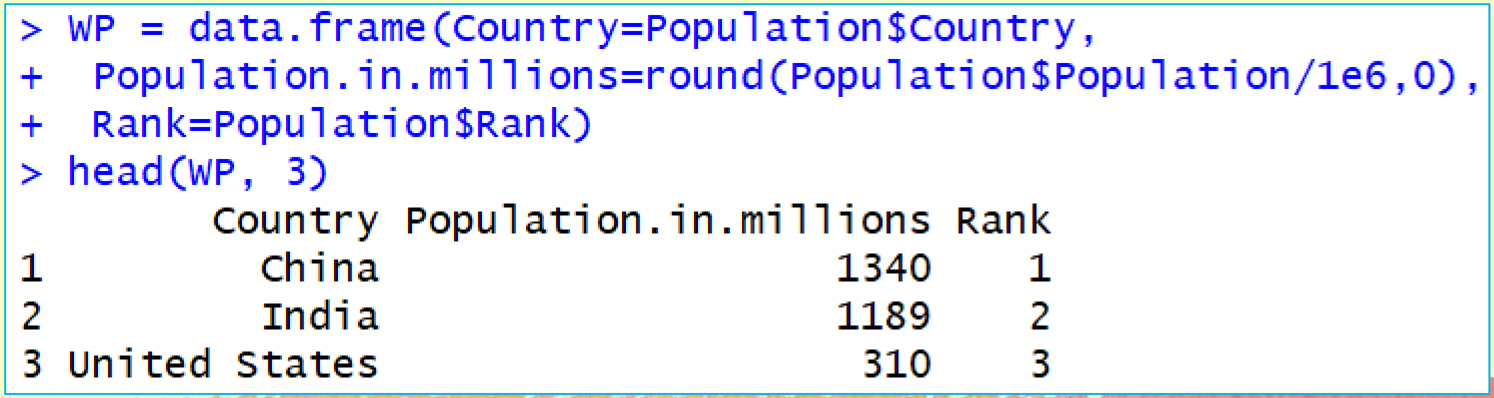


data(Population) {googleVis}

Sourced from https://en.wikipedia.org/wiki/List_of_countries_by_population, 9 October 2010. A data frame with 195 observations on the following 7 variables.



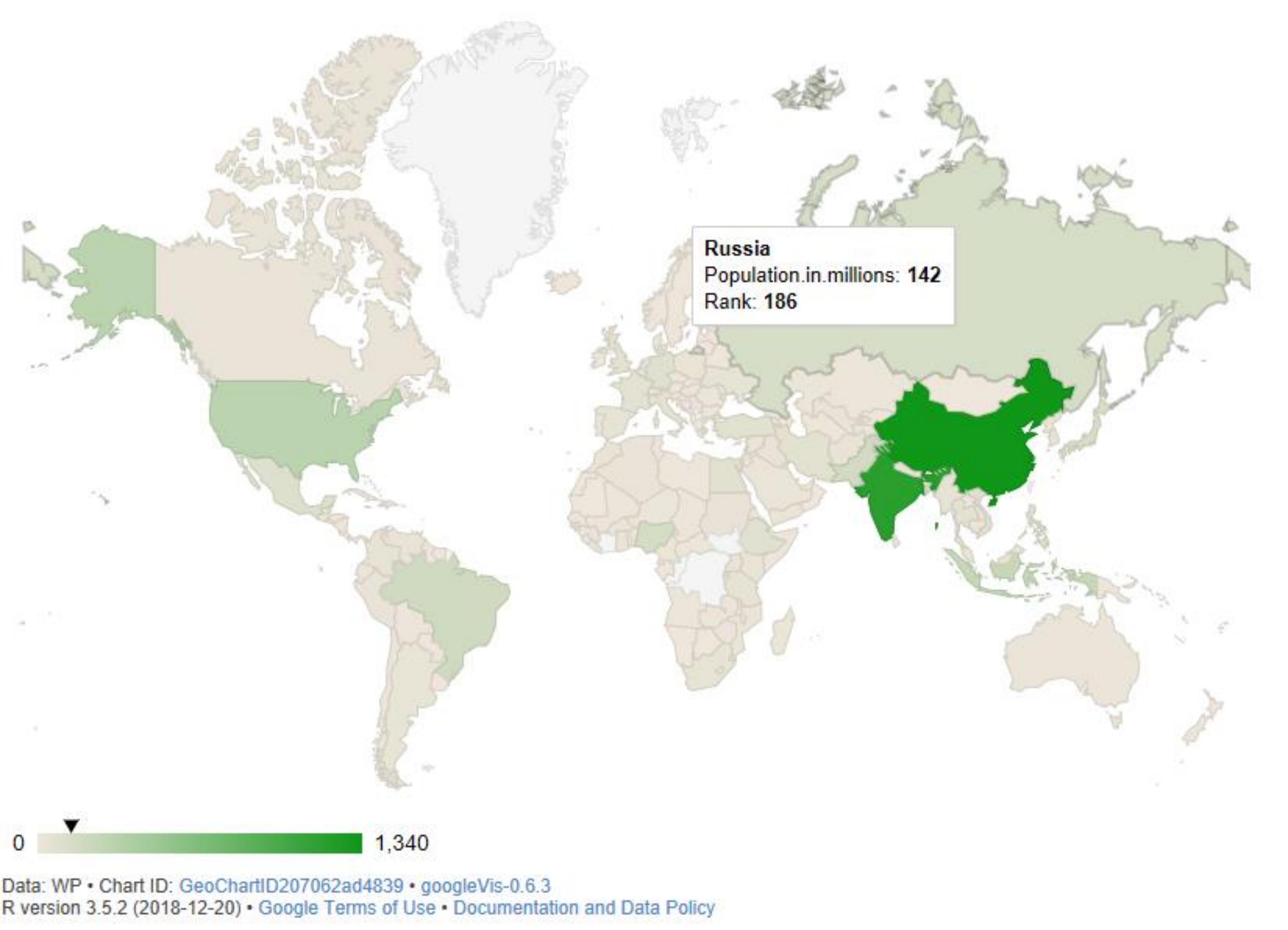
```
> library(googleVis)
> head(Population[,1:3], 3)
  Rank       Country Population
1    1         China 1339940000
2    2         India 1188650000
3    3 United States  310438000
```



```
> WP = data.frame(Country=Population$Country,
+  Population.in.millions=round(Population$Population/1e6,0),
+  Rank=Population$Rank)
> head(WP, 3)
        Country Population.in.millions Rank
1         China                   1340    1
2         India                   1189    2
3 United States                    310    3
```

3. Maps with googleVis

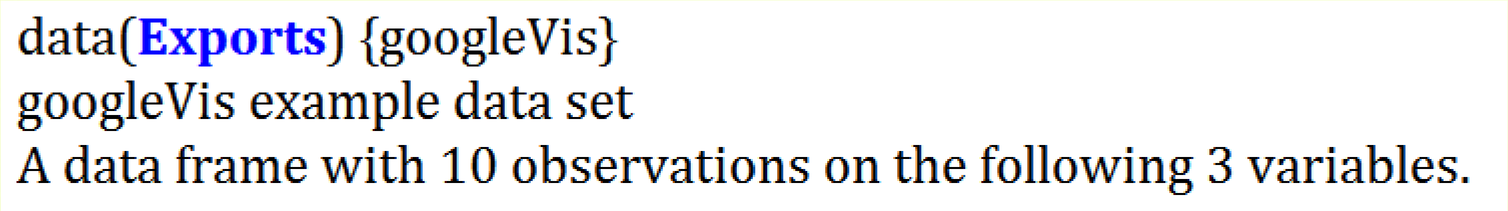
```
G1 <- gvisGeoChart(WP,"Country","Population.in.millions","Rank",
 options=list(dataMode="regions",width=800,height=600))
plot(G1)
```



Russia
Population.in.millions: **142**
Rank: **186**

Data: WP • Chart ID: GeoChartID207062ad4839 • googleVis-0.6.3
R version 3.5.2 (2018-12-20) • Google Terms of Use • Documentation and Data Policy

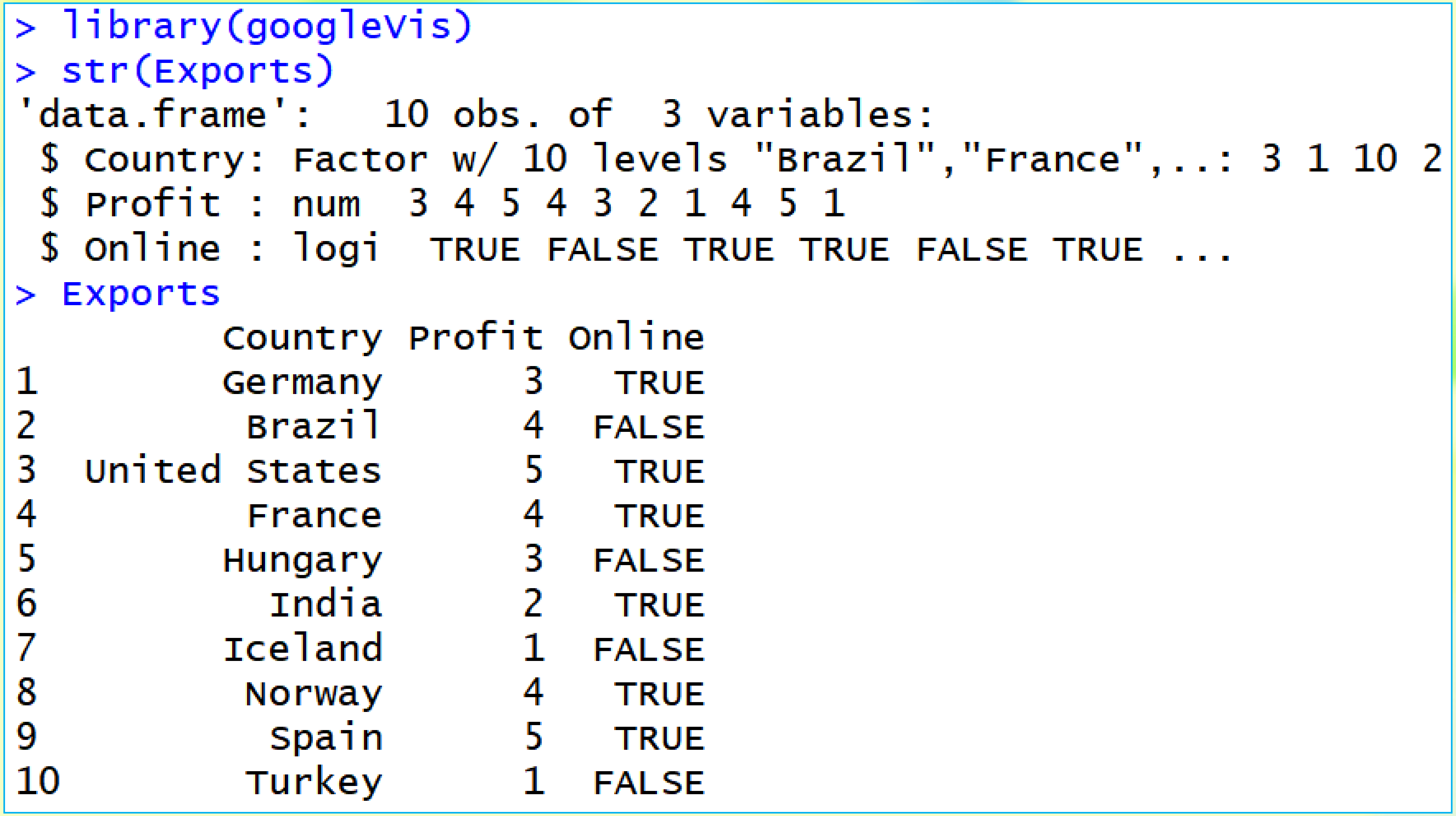
[Example 2] World profit map and table



data(**Exports**) {googleVis}

googleVis example data set

A data frame with 10 observations on the following 3 variables.



```
> library(googleVis)
> str(Exports)
'data.frame':	10 obs. of  3 variables:
 $ Country: Factor w/ 10 levels "Brazil","France",..: 3 1 10 2
 $ Profit : num  3 4 5 4 3 2 1 4 5 1
 $ Online : logi  TRUE FALSE TRUE TRUE FALSE TRUE ...
> Exports
         Country Profit Online
1        Germany      3   TRUE
2         Brazil      4  FALSE
3  United States      5   TRUE
4         France      4   TRUE
5        Hungary      3  FALSE
6          India      2   TRUE
7        Iceland      1  FALSE
8         Norway      4   TRUE
9          Spain      5   TRUE
10        Turkey      1  FALSE
```

[Example 2] World profit map and table

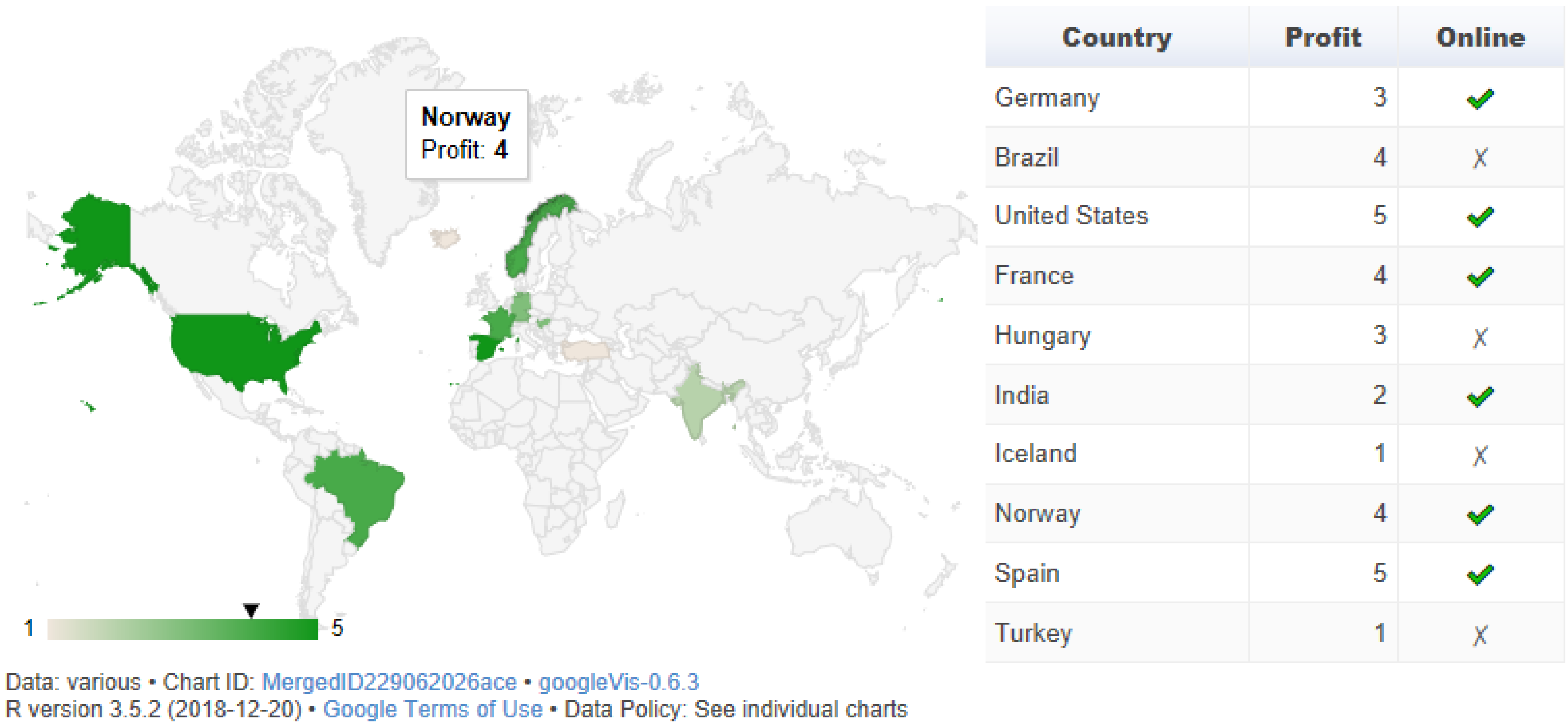
gvisTable(data, options=list(), ...) {googleVis}

Google Table Chart with R

gvisMerge(x, y, ...) {googleVis}

Merge two googleVis charts into one gvis-object

```
GM <- gvisGeoChart(Exports, "Country", "Profit",
                   options=list(width=500,height=340))
GT <- gvisTable(Exports,options=list(width=300,height=340))
G2 <- gvisMerge(GM,GT,horizontal=TRUE)
plot(G2)
```



Country	Profit	Online
Germany	3	✓
Brazil	4	✗
United States	5	✓
France	4	✓
Hungary	3	✗
India	2	✓
Iceland	1	✗
Norway	4	✓
Spain	5	✓
Turkey	1	✗

Data: various • Chart ID: MergedID229062026ace • googleVis-0.6.3
R version 3.5.2 (2018-12-20) • Google Terms of Use • Data Policy: See individual charts

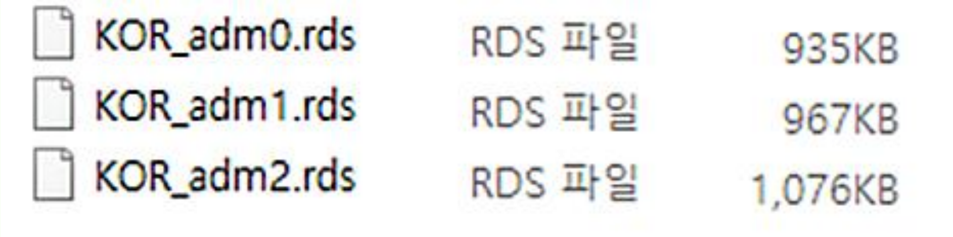
4. Maps with sp

GADM is a spatial database of the location of the world's administrative boundaries.

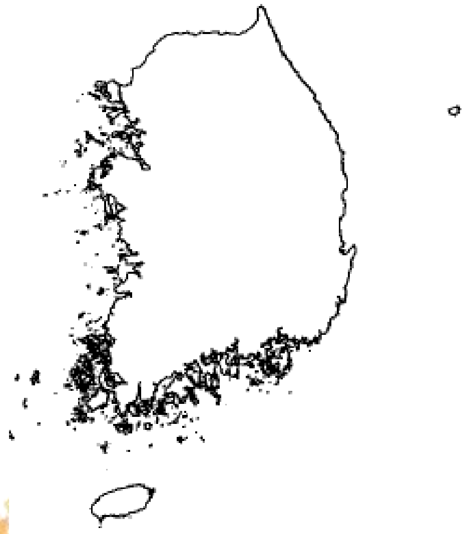
The map information is available as native R objects that can be plotted directly with the **spplot** function (**sp** package).

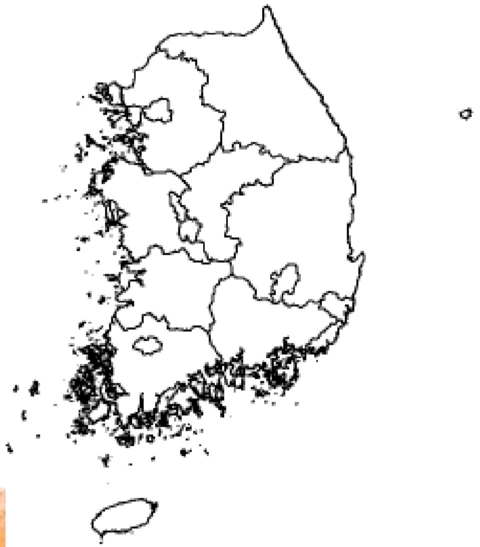
A "**R SpatialPolygonsDataFrame**" (.rds) file can be used in R.

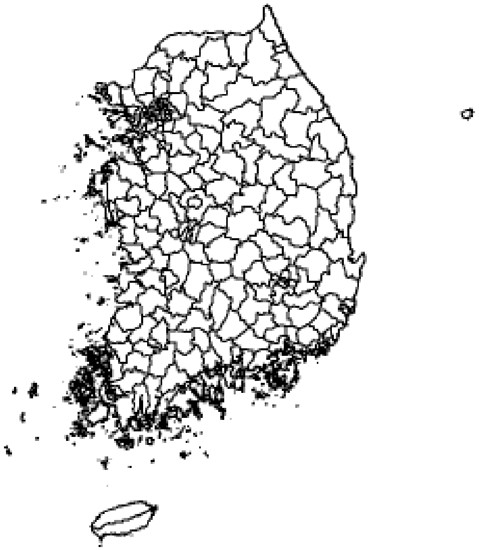
Map data of the “Republic of Korea”:



KOR_adm0.rds	RDS 파일	935KB
KOR_adm1.rds	RDS 파일	967KB
KOR_adm2.rds	RDS 파일	1,076KB





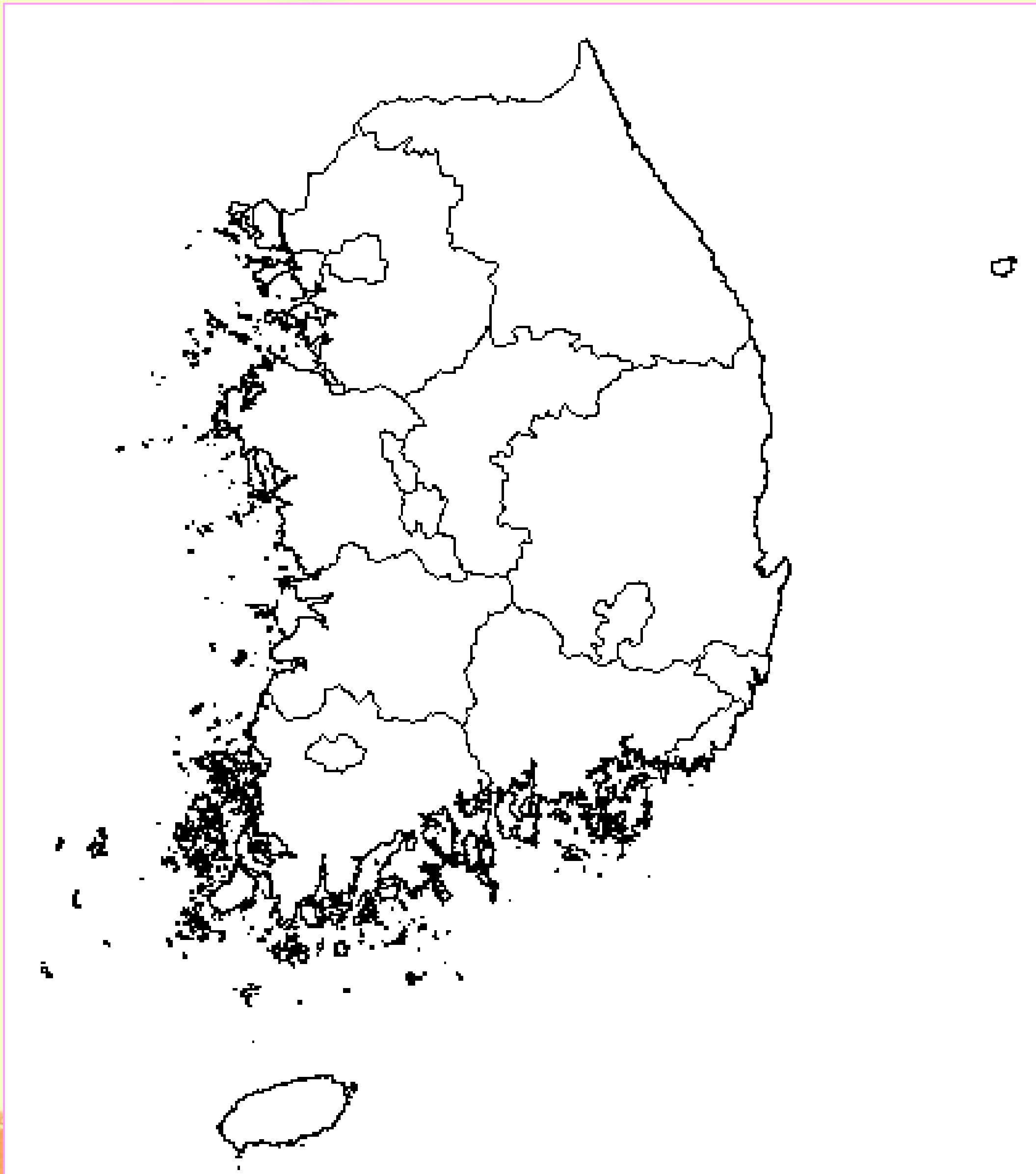


[Example 1] Administrative Divisions of South Korea

saveRDS(object, file, ...) / **readRDS**(file)

Functions to write a single R object to a file, and to restore it.

```
#(1) map without names
library(sp)
gadm <- readRDS("KOR_adm1.rds")
plot(gadm)
```

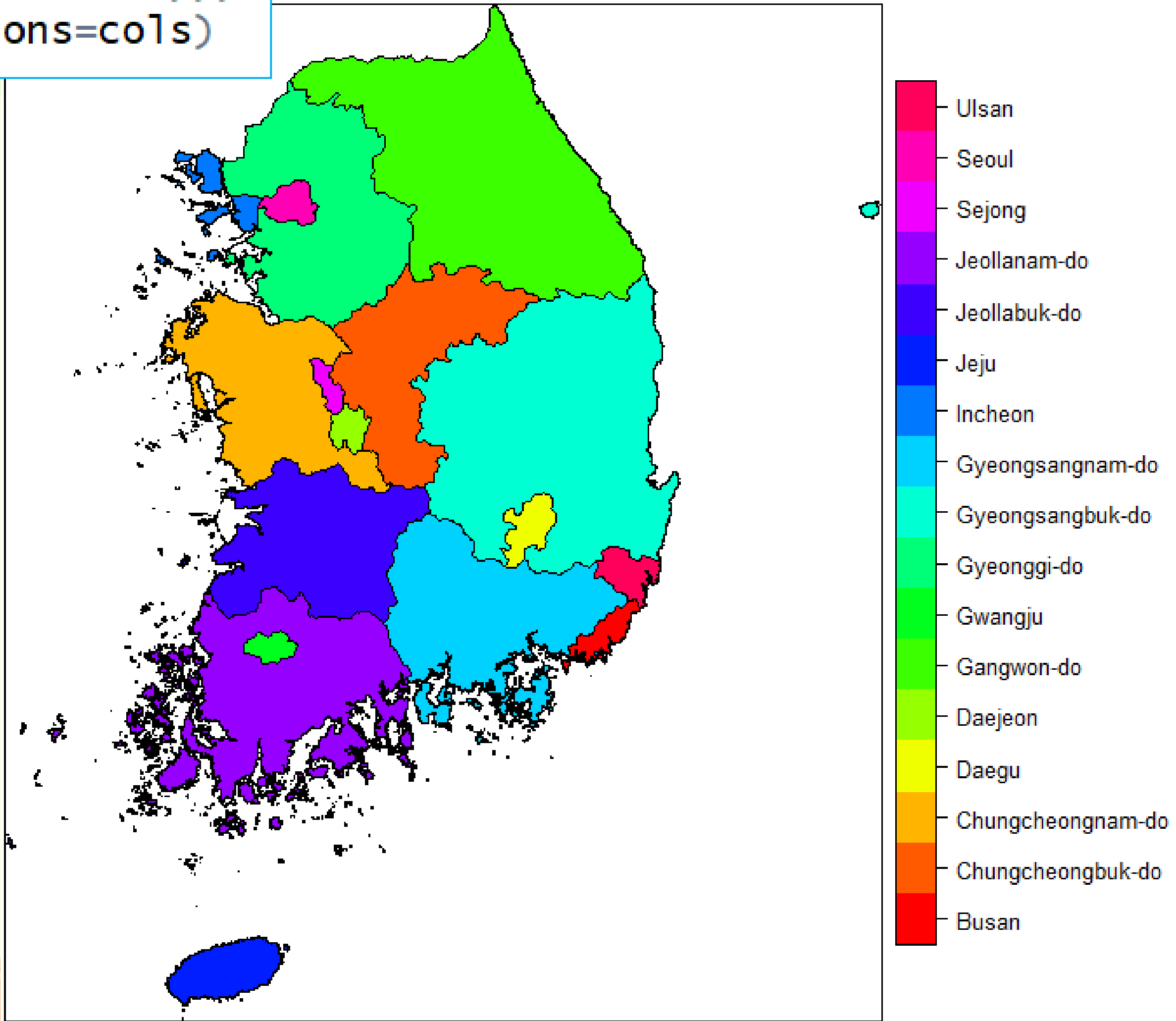


[Example 1] Administrative Divisions of South Korea

spplot(obj, ...) {sp}

Plot method for spatial data with attributes.

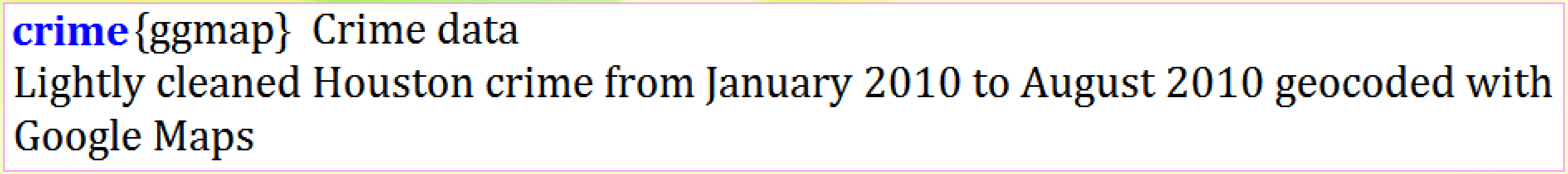
```
#(2) map with names
rname <- gadm$NAME_1
gadm$rname <- as.factor(rname)
cols=rainbow(length(levels(gadm$rname)))
spplot(gadm, "rname", col.regions=cols)
```



5. Maps with ggmap

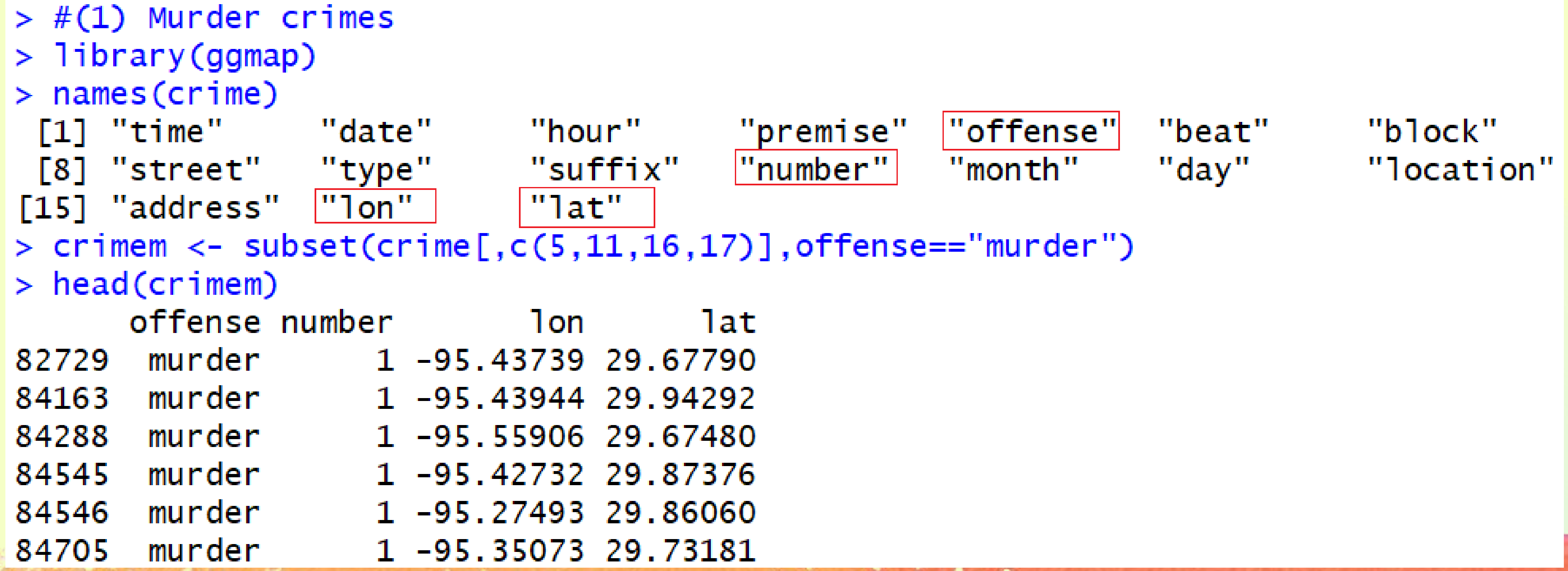
The **ggmap** package enables geographic visualization by combining the spatial information of static maps from **Google Maps**, **OpenStreetMap**, **Stamen Maps** or **CloudMade Maps** with the layered grammar of graphics implementation of ggplot2.

Example: Crime in Houston



crime {ggmap} Crime data

Lightly cleaned Houston crime from January 2010 to August 2010 geocoded with Google Maps



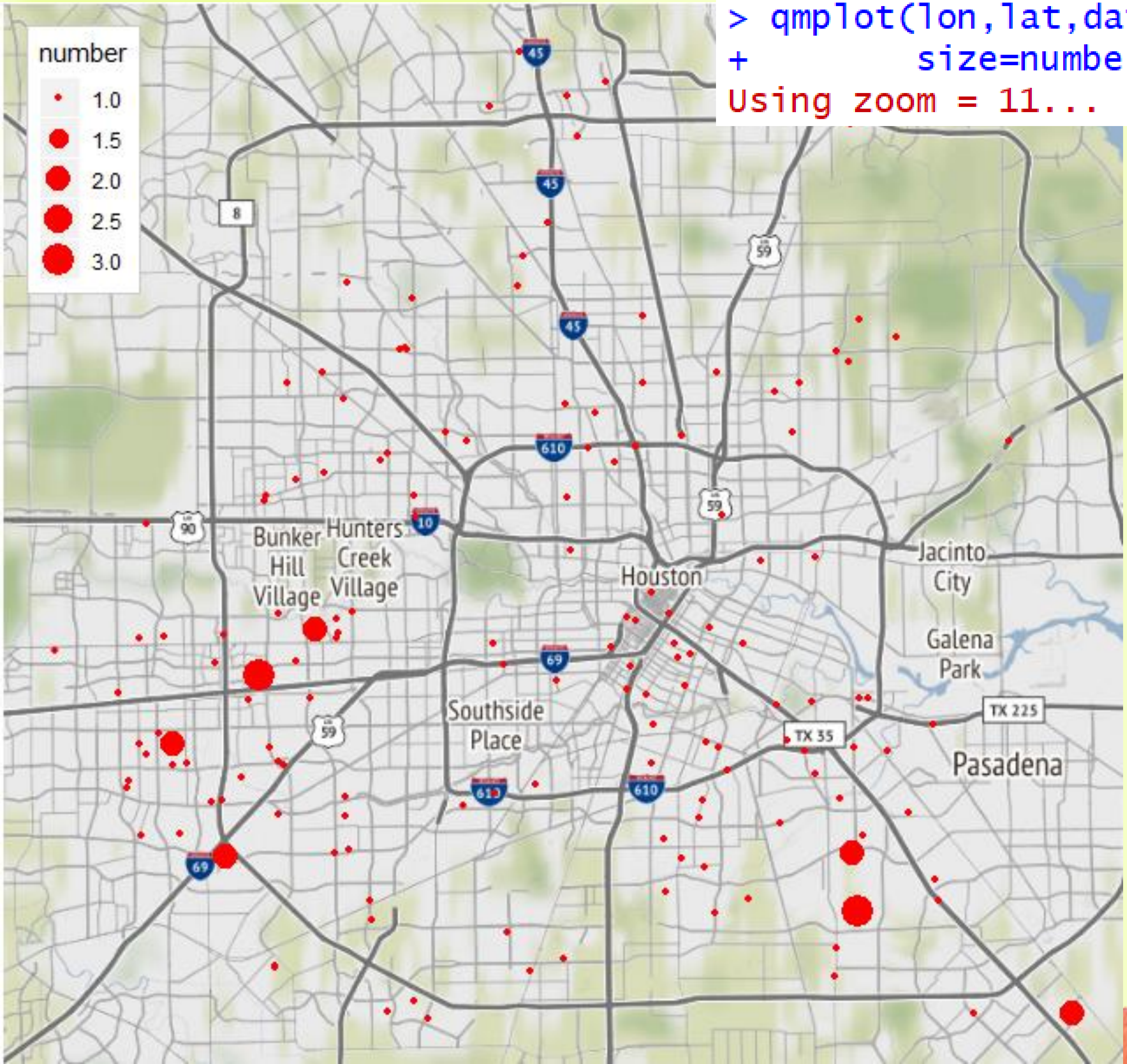
```
> #(1) Murder crimes
> library(ggmap)
> names(crime)
 [1] "time"     "date"     "hour"     "premise"  "offense"  "beat"     "block"   
 [8] "street"   "type"     "suffix"   "number"   "month"    "day"      "location"
[15] "address"  "lon"      "lat"     
> crimem <- subset(crime[,c(5,11,16,17)],offense=="murder")
> head(crimem)
      offense number       lon      lat
82729  murder      1 -95.43739 29.67790
84163  murder      1 -95.43944 29.94292
84288  murder      1 -95.55906 29.67480
84545  murder      1 -95.42732 29.87376
84546  murder      1 -95.27493 29.86060
84705  murder      1 -95.35073 29.73181
```

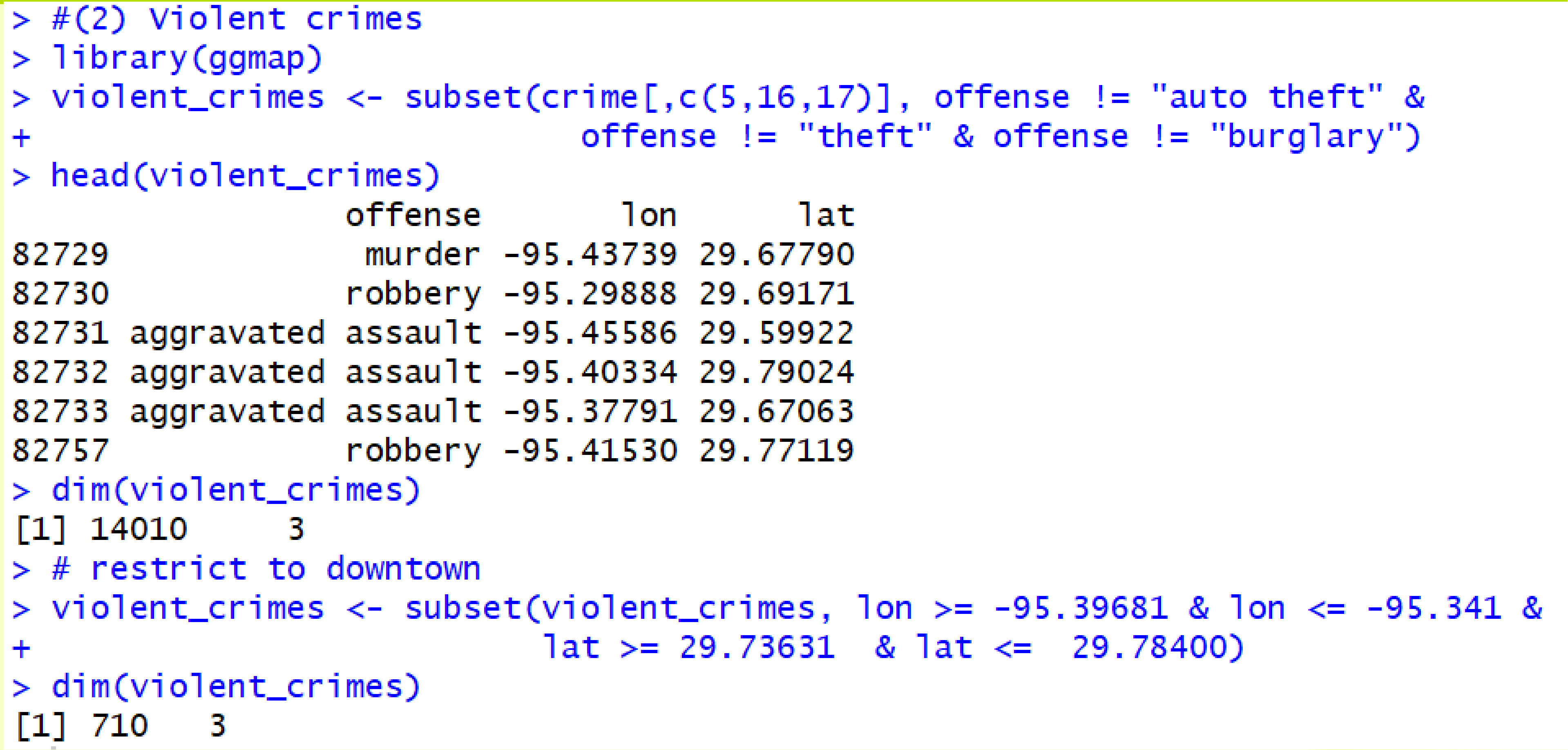
qmplot(x, y, ..., data, ...) {ggmap}

Quick map plot

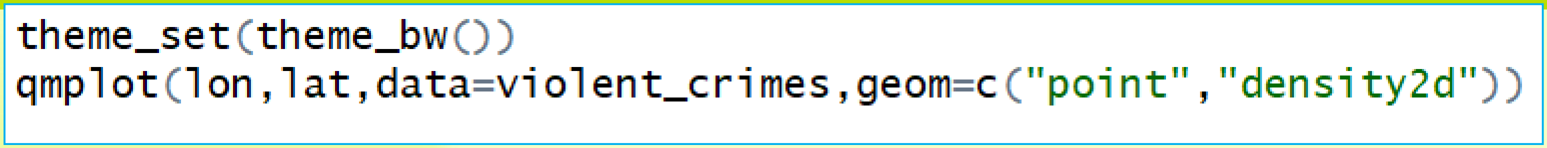
```
> table(crimem$number)

  1   2   3 
150   5   2 
> qmplot(lon,lat,data=crimem,colour=I('red'),
+        size=number,legend="topleft")
Using zoom = 11...
```

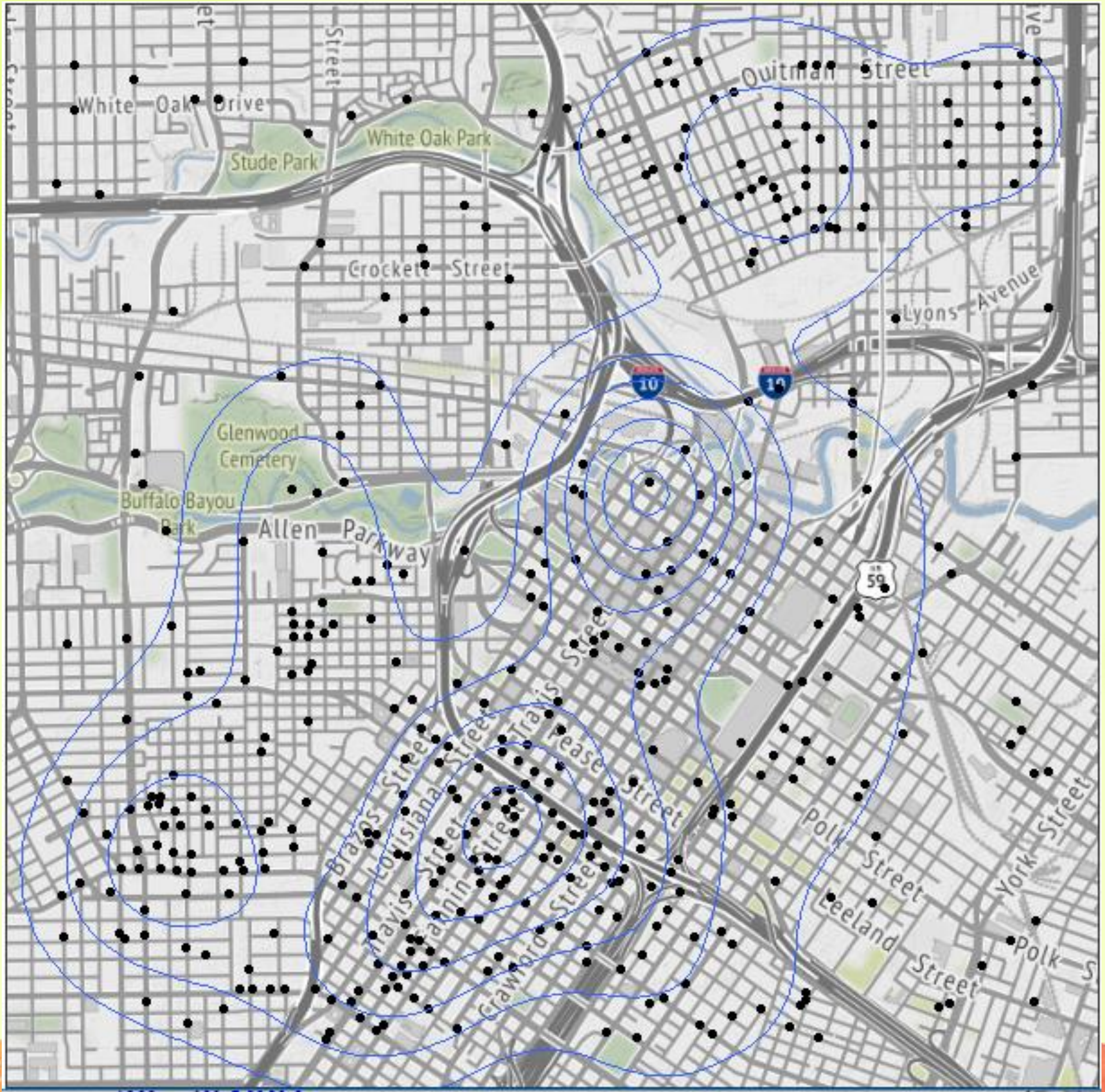




```
> #(2) Violent crimes
> library(ggmap)
> violent_crimes <- subset(crime[,c(5,16,17)], offense != "auto theft" &
+                             offense != "theft" & offense != "burglary")
> head(violent_crimes)
                 offense       lon      lat
82729             murder -95.43739 29.67790
82730            robbery -95.29888 29.69171
82731 aggravated assault -95.45586 29.59922
82732 aggravated assault -95.40334 29.79024
82733 aggravated assault -95.37791 29.67063
82757            robbery -95.41530 29.77119
> dim(violent_crimes)
[1] 14010     3
> # restrict to downtown
> violent_crimes <- subset(violent_crimes, lon >= -95.39681 & lon <= -95.341 &
+                           lat >= 29.73631  & lat <=  29.78400)
> dim(violent_crimes)
[1] 710   3
```



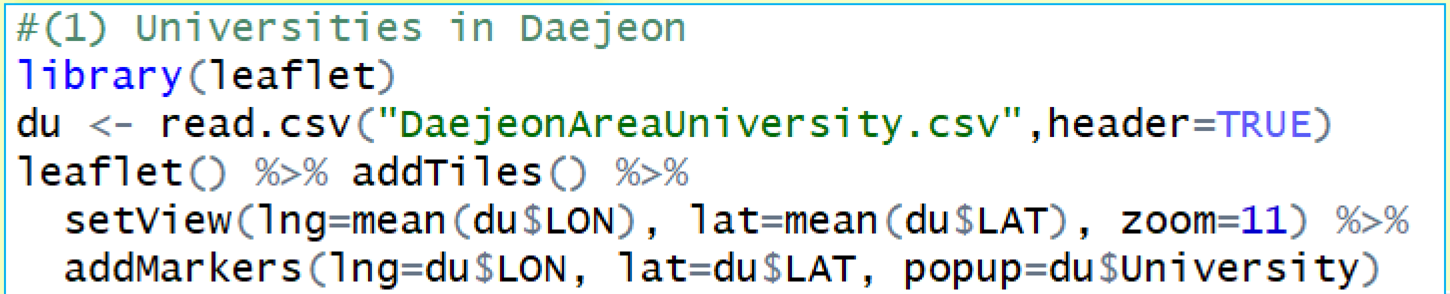
```
theme_set(theme_bw())
qmplot(lon,lat,data=violent_crimes,geom=c("point","density2d"))
```



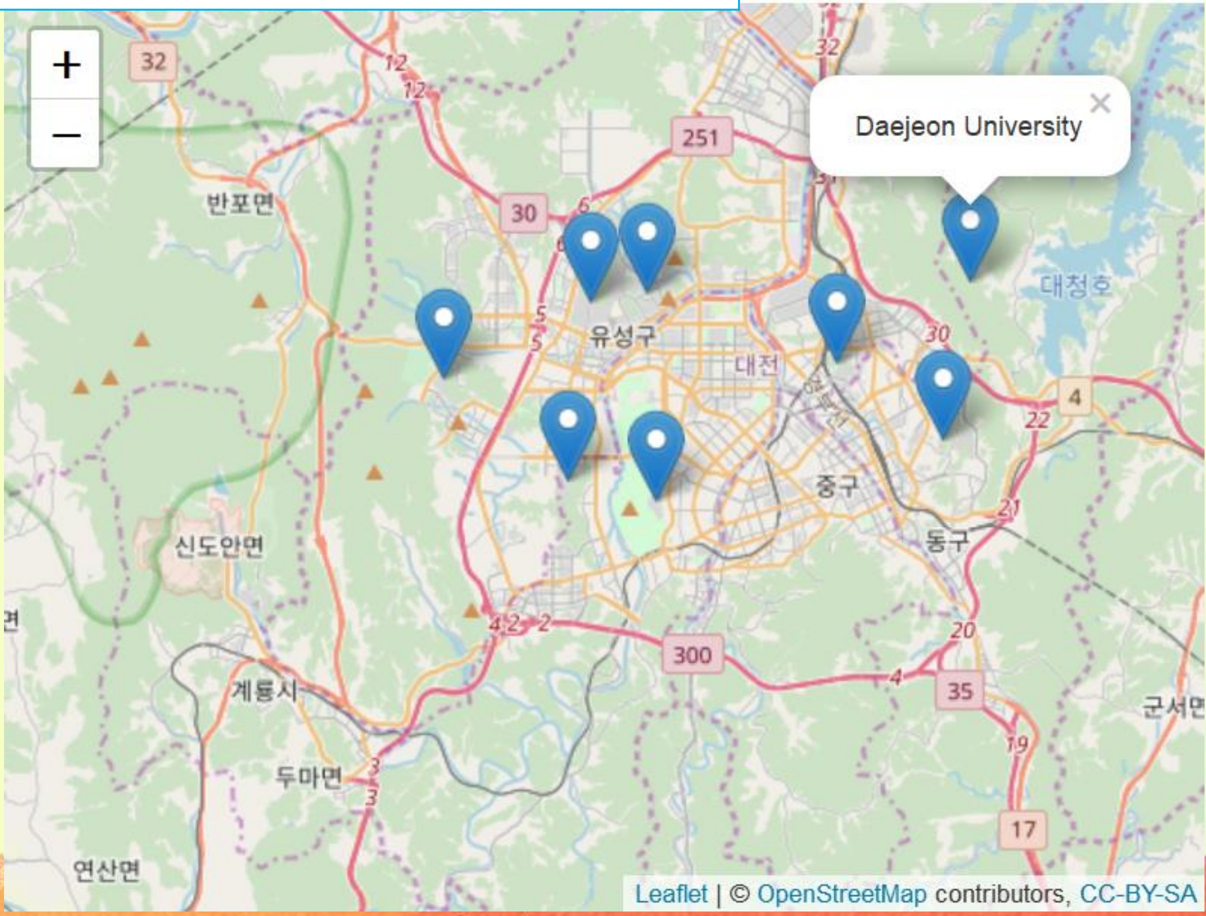
6. Maps with leflet

leaflet(data, ...) {leaflet}

This function creates a Leaflet map widget using htmlwidgets.



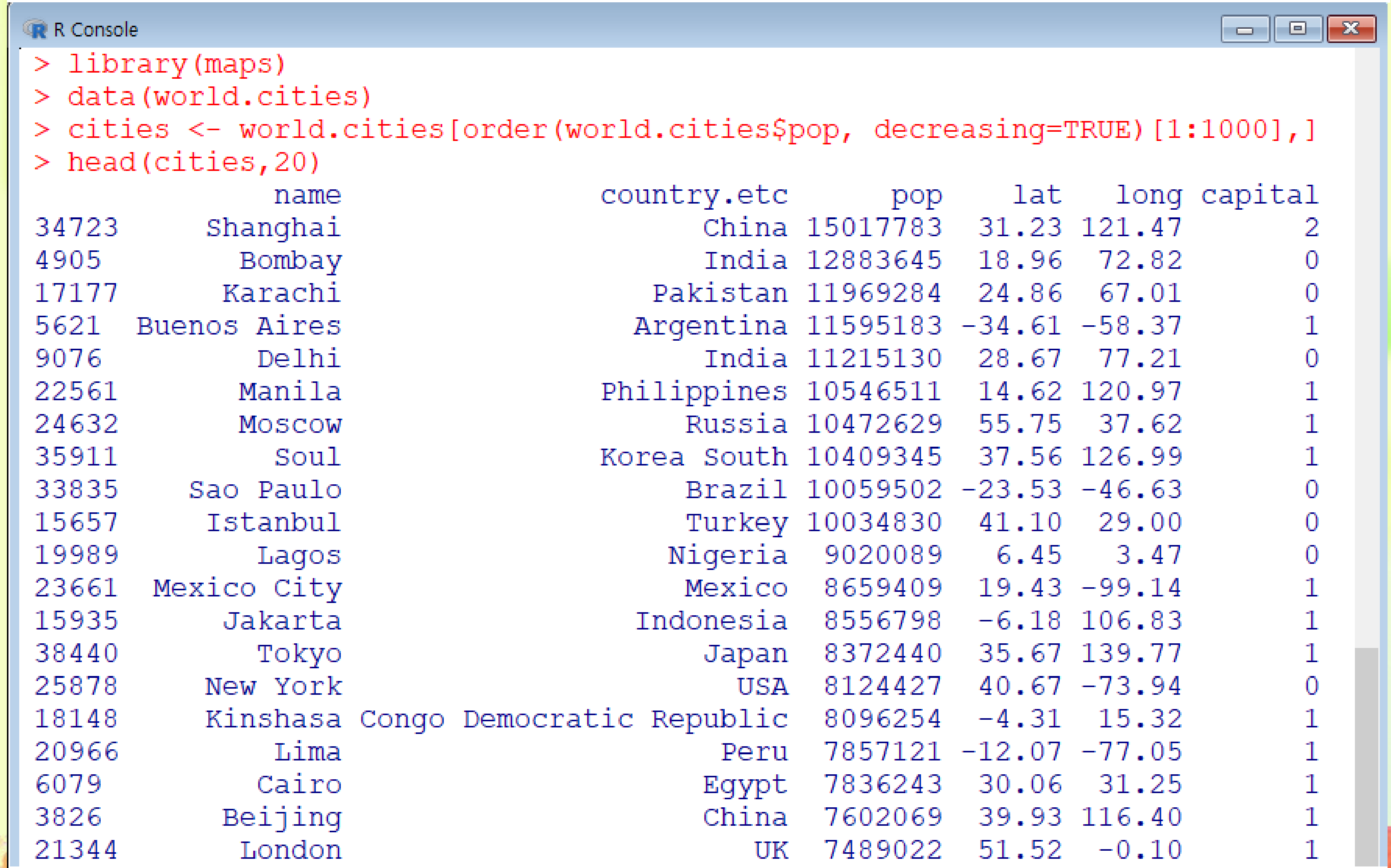
```
#(1) Universities in Daejeon
library(leaflet)
du <- read.csv("DaejeonAreaUniversity.csv",header=TRUE)
leaflet() %>% addTiles() %>%
  setView(lng=mean(du$LON), lat=mean(du$LAT), zoom=11) %>%
  addMarkers(lng=du$LON, lat=du$LAT, popup=du$University)
```



7. 3D Interactive Globe

data(**world.cities**) {maps} Database of world cities

This database is primarily of world cities of population greater than about 40,000. Also included are capital cities of any population size, and many smaller towns.



```
> library(maps)
> data(world.cities)
> cities <- world.cities[order(world.cities$pop, decreasing=TRUE)[1:1000],]
> head(cities,20)
              name                     country.etc      pop    lat   long capital
34723     Shanghai                           China 15017783  31.23 121.47       2
4905        Bombay                           India 12883645  18.96  72.82       0
17177      Karachi                        Pakistan 11969284  24.86  67.01       0
5621  Buenos Aires                       Argentina 11595183 -34.61 -58.37       1
9076         Delhi                           India 11215130  28.67  77.21       0
22561       Manila                     Philippines 10546511  14.62 120.97       1
24632       Moscow                          Russia 10472629  55.75  37.62       1
35911         Soul                     Korea South 10409345  37.56 126.99       1
33835    Sao Paulo                          Brazil 10059502 -23.53 -46.63       0
15657     Istanbul                          Turkey 10034830  41.10  29.00       0
19989        Lagos                         Nigeria  9020089   6.45   3.47       0
23661  Mexico City                          Mexico  8659409  19.43 -99.14       1
15935      Jakarta                       Indonesia  8556798  -6.18 106.83       1
38440        Tokyo                           Japan  8372440  35.67 139.77       1
25878     New York                             USA  8124427  40.67 -73.94       0
18148     Kinshasa Congo Democratic Republic  8096254  -4.31  15.32       1
20966         Lima                            Peru  7857121 -12.07 -77.05       1
6079         Cairo                           Egypt  7836243  30.06  31.25       1
3826       Beijing                           China  7602069  39.93 116.40       1
21344       London                              UK  7489022  51.52  -0.10       1
```

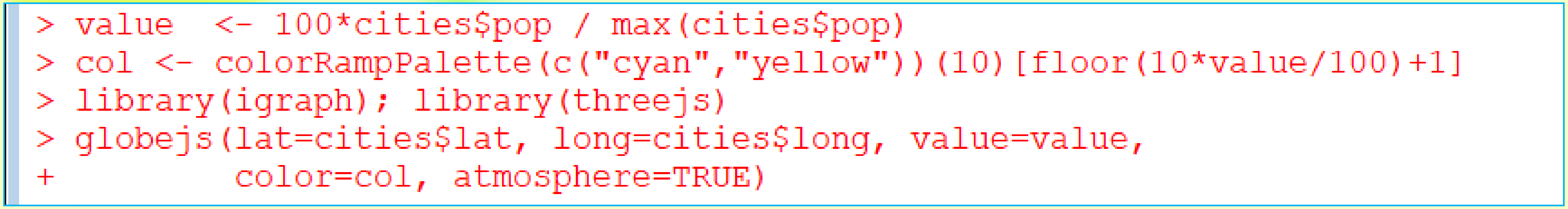
colorRampPalette(colors, ...) {grDevices} Color interpolation

These functions return functions that interpolate a set of given colors to create new color palettes (like topo.colors), functions that map the interval [0, 1] to colors (like grey).

globejs(img, lat, long, value, color, ...) {threejs} Plot Data on 3D Globes

Plot points, arcs and images on a globe in 3D using Three.js.

The globe can be **rotated** and **zoomed.**



```
> value  <- 100*cities$pop / max(cities$pop)
> col <- colorRampPalette(c("cyan","yellow"))(10)[floor(10*value/100)+1]
> library(igraph); library(threejs)
> globejs(lat=cities$lat, long=cities$long, value=value,
+          color=col, atmosphere=TRUE)
```

3D Interactive Visualization: