Part 5: Working with data

Dr. Nguyen Quang Huy

September 16, 2020

Data types in R

- Objects in R can be of different types: numeric number, character, matrix, function ...
- The function "class" determines the type of an object

```
class(lm)
```

```
## [1] "function"
```

```
M<-matrix(0,2,2)
class(M)
```

```
## [1] "matrix" "array"
```

```
class("Actuary")
```

```
## [1] "character"
```

Data frames in R

- The most common way of storing data sets in R.
- Data frames are a special case of lists. Lists contain different data types of different lengths.
- We can think of data frame as an excel table where rows represent observations and columns represents variables.
- Data frames is an object containing different data types (variables).

```
library(dslabs) #package "dslabs" contains data sets we need
data("murders") #load data set "murders" from dslabs
class(murders)
```

```
## [1] "data.frame"
```

Data frames in R

```
str(murders) # show the structure of "murders" data set
```

```
## 'data.frame':    51 obs. of  5 variables:
##  $ state     : chr  "Alabama" "Alaska" "Arizona" "Arkansas"
##  $ abb       : chr  "AL" "AK" "AZ" "AR" ...
##  $ region    : Factor w/ 4 levels "Northeast","South",..: 2
##  $ population: num  4779736 710231 6392017 2915918 3725395
##  $ total     : num  135 19 232 93 1257 ...
```

Data frames in R

```
head(murders) # show the head of "murders" data set
```

```
##        state abb region population total
## 1    Alabama  AL  South    4779736   135
## 2     Alaska  AK   West     710231    19
## 3    Arizona  AZ   West    6392017   232
## 4   Arkansas  AR  South    2915918    93
## 5 California  CA   West   37253956  1257
## 6   Colorado  CO   West    5029196    65
```

Data frames

To access the different variables of a data frames, we use the accessor "$".

```
class(murders$state)
```

```
## [1] "character"
```

```
murders$state
```

```
##  [1] "Alabama"              "Alaska"               "Arizona
##  [4] "Arkansas"             "California"           "Colorad
##  [7] "Connecticut"          "Delaware"             "Distric
## [10] "Florida"              "Georgia"              "Hawaii"
## [13] "Idaho"                "Illinois"             "Indiana
## [16] "Iowa"                 "Kansas"               "Kentuck
## [19] "Louisiana"            "Maine"                "Marylan
## [22] "Massachusetts"        "Michigan"             "Minneso
## [25] "Mississippi"          "Missouri"             "Montana
## [28] "Nebraska"             "Nevada"               "New Ham
```

Data frames

The accessor "$" preserves the order of the rows in the data sets

```
class(murders$population)
```

```
## [1] "numeric"
```

```
pop<-murders$population
length(pop)
```

```
## [1] 51
```

```
pop[1:5]
```

```
## [1]  4779736   710231  6392017  2915918 37253956
```

Creating data frames

Using **data.frame** to create your own data frame:

```
dat<-data.frame(ID = paste("SV",1:5),
                names = c("You", "Me", "Him", "Her", "John"),
                grades = c(5.5, 1.5, 10.0, 9.0, 7.6),
                result = c(TRUE, FALSE,TRUE, TRUE, TRUE))
dat
```

```
##      ID names grades result
## 1 SV 1   You    5.5   TRUE
## 2 SV 2    Me    1.5  FALSE
## 3 SV 3   Him   10.0   TRUE
## 4 SV 4   Her    9.0   TRUE
## 5 SV 5  John    7.6   TRUE
```

WARNING: by default **data.frame** turns characters into factors.

Tibbles

Thay could use *tibbles* as an alternative to traditional *data.frame*

- They often use the term *tibble* and *data frame* interchangeably
- **Tibbles** are data frames, but they improve some behaviours to make it is easier to work with large datasets.
- To work with *tibbles*, they use the **tibble** package which is included in **tidyverse** package

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.3     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1
```

Tibbles

We can create a tibble from individual vectors with *tibble()*

```
tib<-tibble(
  a = c(1,2,3),
  b = c("X","Y","Z"),
  c = 2
)
tib
```

```
## # A tibble: 3 x 3
##       a b           c
##   <dbl> <chr> <dbl>
## 1     1 X           2
## 2     2 Y           2
## 3     3 Z           2
```

Tibbles

To convert a *data.frame* to *tibble*

```
class(iris)
```

```
## [1] "data.frame"
```

```
tib<-as.tibble(iris)
tib
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Specie
##           <dbl>       <dbl>        <dbl>       <dbl> <fct>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            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           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
```

Tibbles and data.frame

There are main differences in the usage of a *tibble* vs a *data.frame*:

1. In printing:
   - Tibble shows only the first 10 rows, and all the columns that fit on screen.
   - Each column of printed *tibble* reports its type, a feature borrowed from *str()*
2. *tibbles* never change the type of the inputs (e.g. never converts strings to factors), it never changes the names of variables, and it never creates row names.
3. A *tibbles* can have column names that are not valid R variable names.

Data frames

There is a column called regions (which state is in which region). It would be a character ("Northeast", "South", ...) but R says that it is "factor".

- The regions are categorial, there are four categories.
- R stores the levels as intergers.
- Integers are smaller memory-wise than characters.

```
class(murders$region)
```

```
## [1] "factor"
```

```
levels(murders$region)
```

```
## [1] "Northeast"     "South"         "North Central" "West"
```

```
as.character(murders$region)[1:5]
```

```
## [1] "South" "West"  "West"  "South" "West"
```

Data frames

Example 1: Load the **movielens** dataset.

- How many rows are in the dataset; how many variables are in the dataset ?
- What is the variable type of **title**, of **genres** ?
- How many levels are in the **genres** variable ?

Data frames

Example 1: Load the **movielens** dataset.

- How many rows are in the dataset; how many variables are in the dataset ?
- What is the variable type of **title**, of **genres** ?
- How many levels are in the **genres** variable ?

```
data("movielens")
str(movielens)
```

```
## 'data.frame':    100004 obs. of  7 variables:
##  $ movieId  : int  31 1029 1061 1129 1172 1263 1287 1293 13
##  $ title    : chr  "Dangerous Minds" "Dumbo" "Sleepers" "Es
##  $ year     : int  1995 1941 1996 1981 1989 1978 1959 1982
##  $ genres   : Factor w/ 901 levels "(no genres listed)",..:
##  $ userId   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ rating   : num  2 5 3 3 2 4 2 2 2 3 5 2
```

Data frames: Loading data from other sources

- Setting the directory to the folder where you located your datasets.
- Using one of the following function to read your datasets

Function	Format	Explanation	Package
read.table	txt	white space separated values	base
read.csv	csv	comma separated values	base
read_table	txt	white space separated values	readr
read_csv	csv	comma separated values	readr
read_excel	xls, xlsx	auto detect the format	readxl
read_xls	xls	original format	readxl
read_xlsx	xlsx	new format	readxl

Data frame: Loading data from other sources

```
setwd("C:/Users/AD/Desktop/Tex file/Thu latex/Introduction to
# Loading price & volume data on HOSE from 28/07/2000
stock<-read.csv("CafeF.HSX.Upto20102020.csv")
# View the data structure
str(stock)
#Rename the dataset
names(stock)<-c("Ticker","Day","Open",
                "High", "Low", "Close", "Volume")
```

How many stocks/etf/indexes are listed in "Ticker" variable ?

Data frame: Loading data from other sources

```
setwd("C:/Users/AD/Desktop/Tex file/Thu latex/Introduction to
# Loading price & volume data on HOSE from 28/07/2000
stock<-read.csv("CafeF.HSX.Upto20102020.csv")
# View the data structure
str(stock)
#Rename the dataset
names(stock)<-c("Ticker","Day","Open",
                "High", "Low", "Close", "Volume")
```

How many stocks/etf/indexes are listed in "Ticker" variable ?

```
length(unique(stock$Ticker))
```

```
stock$Ticker<-as.factor(stock$Ticker)
str(stock)
```

Data frames

Vector coercion:

- When an entry does NOT match the expected, R tries to guess what the codes meant before throwing in an error.
- This is the major difference between R and other languages.
- This can also lead to confusion.

```
x<-1:2
y<-c("a","b")
z<-c(x,y)
z
```

```
## [1] "1" "2" "a" "b"
```

```
class(z)
```

```
## [1] "character"
```

Data frames

- **as.numeric** function tries to convert other data types into numeric variables
- **as.character** function tries to convert other data types into character variables

When R fails to coerce something, it will return **NA**

```
z
```

```
## [1] "1" "2" "a" "b"
```

```
as.numeric(z)*2
```

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

```
## [1]  2  4 NA NA
```

Indexing with logicals

- R provides a powerful way of indexing vector/array (as mentioned in part I)
- We can subset a vector based on properties of another vectors.

```
x<-c(1,2,3,4,5)
y<-c(TRUE,FALSE,FALSE,FALSE,FALSE)
x[y]
```

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

```
safe<-murders$total/murders$population<=2/10^5
favo<-murders$region=="South"
murders$state[safe&favo]
```

```
## [1] "West Virginia"
```

Indexing with logicals

Example: Extracting trading volume of "VNM" from the "stock" dataset.

```
setwd("C:/Users/AD/Desktop/Tex file/Thu latex/Introduction to
stock<-read.csv("CafeF.HSX.Upto20102020.csv")
names(stock)<-c("Ticker","Day","Open", "High",
                "Low", "Close", "Volume")
```

```
rows<-stock$Ticker=="VNM"
cols<-names(stock)=="Volume"
VNM<-stock[rows,cols]
str(VNM)
```

```
##  int [1:3675] 109350 157840 81400 66000 57220 18750 25630 2
```

Indexing function "which"

The function which tells us which entries of a logical vector are TRUE

```
which(c(1,2,3,2,1)==1)
```

```
## [1] 1 5
```

```
which(murders$state == "California") #Index cua bang Cali
```

```
## [1] 5
```

```
which.max(murders$total)
```

```
## [1] 5
```

```
which.min(murders$total/murders$population)
```

```
## [1] 46
```

Indexing function "which"

Example: Extracting trading volume of "VNM" from the "stock" dataset.

```
rows<-which(stock$Ticker=="VNM")
cols<-which(names(stock)=="Volume")
VNM<-stock[rows,cols]
str(VNM)
```

```
##  int [1:3675] 109350 157840 81400 66000 57220 18750 25630 2
```

Indexing functions “match”

Function “match”: return the index needed to access several entries in a vector.

```
x<-c(1,2,3,4,5,4,3,2,1)
match(c(2,5,6),x)
```

```
## [1]  2  5 NA
```

```
match(c("Arizona","Hawaii","Hanoi"),murders$state)
```

```
## [1]  3 12 NA
```

```
match(c("Arizona","Hawaii","Hanoi","Hawaii"),murders$state)
```

```
## [1]  3 12 NA 12
```

Indexing functions "%in%"

"%in% operator": give us information about whether or not each elements of a first vector is in a second vector

```
c(1,3,5) %in% 1:4
```

```
## [1]  TRUE  TRUE FALSE
```

```
c("Arizona","Hawaii","Hanoi") %in% murders$state
```

```
## [1]  TRUE  TRUE FALSE
```

Sorting a vector - "sort" function

The function **sort** sorts a vector in increasing order.

```
sort(murders$total)[1:5] # increasing order
```

```
## [1] 2 4 5 5 7
```

```
sort(murders$total,decreasing = TRUE)[1:5] #decreasing order
```

```
## [1] 1257  805  669  517  457
```

The function **sort** does not give us information that which state has the lowest or highest number of gun murders. We use the function **order** for this purpose

```
order(murders$total)[1:5] # return indices of 5 smallest valu
```

```
## [1] 46 35 30 51 12
```

```
murders$total[order(murders$total)[1]]
```

Sorting a vector - "order" function

```
order(murders$total)[1:5] # return indices of 5 smallest valu
```

```
## [1] 46 35 30 51 12
```

```
murders$total[order(murders$total)[1]]
```

```
## [1] 2
```

Give the names of 3 states having the highest number of gun murders ?

Sorting a vector - "order" function

```
order(murders$total)[1:5] # return indices of 5 smallest valu
```

```
## [1] 46 35 30 51 12
```

```
murders$total[order(murders$total)[1]]
```

```
## [1] 2
```

Give the names of 3 states having the highest number of gun murders ?

```
indices<-order(murders$total,decreasing=TRUE)[1:3]
murders$state[indices]
```

```
## [1] "California" "Texas"      "Florida"
```

Sorting a vector - "rank" function

rank is also an important function in sorting a vector.

```
x<-c(17,13,7,19,11)
rank(x)
```

```
## [1] 4 3 1 5 2
```

$\rightarrow$ If $y < -rank(x)$, $y[i]$ is the rank of $x[i]$ in the vector x.

The **murders rate** is calculated by the murder number divided by the state's population. Give the names of 3 states having the highest murder rates?

```
indices<-order(murders$total/murders$population,decreasing=TRU
murders$state[indices]
```

```
## [1] "District of Columbia" "Louisiana"                "Missouri
```

Sorting a vector

IMPORTANT:

y<-sort(x)	y[1] is the smallest value in **x** y[2] is the 2^{nd} smallest value in **x** ...
y<-sort(x, decreasing = T)	y[1] is the largest value in **x** y[2] is the 2^{nd} largest value in **x** ...
y<-order(x)	y[1] is the index of the smallest value in **x** y[2] is the index of the 2^{nd} smallest value in **x** ...
y<-order(x, decreasing=T)	y[1] is the index of the largest value in **x** y[2] is the index of the 2^{nd} largest value in **x** ...
y<-rank(x)	y[i] is the rank of x[i] in **x (increasing order)**

Data manipulation

Package **"dplyr"** introduces functions that perform the most common operations in data manipulation.

```
#package "dplyr" is included in "tidyverse"
library(tidyverse)
```

Data manipulation - “mutate”

The function “mutate” is used to add a column in a dataframe.

```
dat<-murders
# adding a column name "rate" in murders data
dat<-mutate(dat,rate=total/population*10^6)
head(dat)
```

```
##        state abb region population total     rate
## 1    Alabama  AL  South    4779736   135 28.24424
## 2     Alaska  AK   West     710231    19 26.75186
## 3    Arizona  AZ   West    6392017   232 36.29527
## 4   Arkansas  AR  South    2915918    93 31.89390
## 5 California  CA   West   37253956  1257 33.74138
## 6   Colorado  CO   West    5029196    65 12.92453
```

Data manipulation - “mutate”

Example: Add a column named “Trading_value” to stock data, which is calculated by:

$$TradingValue = Close * Volume$$

Data manipulation - "mutate"

Example: Add a column named "Trading_value" to stock data, which is calculated by:

$$TradingValue = Close * Volume$$

```
stock<-mutate(stock,Trading_value = Close * Volume)
str(stock)
```

```
## 'data.frame':    874212 obs. of  8 variables:
##  $ Ticker       : chr  "SAM" "REE" "SAM" "REE" ...
##  $ Day          : int  20000728 20000728 20000731 20000731
##  $ Open         : num  2.28 1.57 2.31 1.6 2.35 ...
##  $ High         : num  2.28 1.57 2.31 1.6 2.35 ...
##  $ Low          : num  2.28 1.57 2.31 1.6 2.35 ...
##  $ Close        : num  2.28 1.57 2.31 1.6 2.35 ...
##  $ Volume       : int  3200 1000 10000 300 200 100 100 190
##  $ Trading_value: num  7308 1574 23106 481 470 ...
```

Data manipulation - "filter"

The function "filter" is used to subset a dataframe

```
# subsetting data with filter
dat<-filter(dat,rate<=15) #murder rate < 15/1000.000
head(dat)
```

```
##       state abb         region population total      rate
## 1  Colorado  CO           West    5029196    65 12.924531
## 2    Hawaii  HI           West    1360301     7  5.145920
## 3     Idaho  ID           West    1567582    12  7.655102
## 4      Iowa  IA North Central    3046355    21  6.893484
## 5     Maine  ME      Northeast    1328361    11  8.280881
## 6 Minnesota  MN North Central    5303925    53  9.992600
```

Data manipulation - “filter”

Tạo ra một dữ liệu tên là stock1 từ stock mà cột ticker là “VNM”

Data manipulation - "filter"

Tạo ra một dữ liệu tên là stock1 từ stock mà cột ticker là "VNM"

```
# subsetting data with filter
stock1<-filter(stock,Ticker=="VNM")
head(stock1)
```

```
##   Ticker      Day   Open   High    Low  Close Volume Tradi
## 1    VNM 20060119 2.7719 2.7719 2.7719 2.7749 109350     30
## 2    VNM 20060120 2.8765 2.8765 2.8242 2.8273 157840     44
## 3    VNM 20060123 2.8242 2.8242 2.7458 2.7487  81400     22
## 4    VNM 20060124 2.7458 2.7458 2.7196 2.7225  66000     17
## 5    VNM 20060125 2.7196 2.7719 2.7196 2.7749  57220     15
## 6    VNM 20060126 2.7981 2.7981 2.7981 2.8011  18750      5
```

Data manipulation - "select"

The function "select" is used to select several column of a dataframe and assign them to a new dataframe

```
# selecting columns
newdat<-select(dat,state, region, rate)
head(newdat)
```

```
##       state         region      rate
## 1  Colorado           West 12.924531
## 2    Hawaii           West  5.145920
## 3     Idaho           West  7.655102
## 4      Iowa North Central  6.893484
## 5     Maine      Northeast  8.280881
## 6 Minnesota North Central  9.992600
```

Data manipulation - “select”

Tạo ra một dữ liệu tên là stock2 từ stock1 chỉ bao gồm các cột “Day”, “Close”, “Volume” và “Trading_value”

Data manipulation - "select"

Tạo ra một dữ liệu tên là stock2 từ stock1 chỉ bao gồm các cột "Day", "Close", "Volume" và "Trading_value"

```
# selecting columns
stock2<-select(stock1,Day, Close, Volume, Trading_value)
head(stock2)
```

```
##        Day  Close Volume Trading_value
## 1 20060119 2.7749 109350     303435.32
## 2 20060120 2.8273 157840     446261.03
## 3 20060123 2.7487  81400     223744.18
## 4 20060124 2.7225  66000     179685.00
## 5 20060125 2.7749  57220     158779.78
## 6 20060126 2.8011  18750      52520.62
```

Data manipulation - "arrange"

The function "arrange" is used to sort a dataframe by a prespecified column.

```
#dat<-arrange(murders,total) #increasing order
dat<-arrange(murders,desc(total)) #decreasing order
head(dat)
```

```
##           state abb         region population total
## 1    California  CA           West   37253956  1257
## 2         Texas  TX          South   25145561   805
## 3       Florida  FL          South   19687653   669
## 4      New York  NY      Northeast   19378102   517
## 5  Pennsylvania  PA      Northeast   12702379   457
## 6      Michigan  MI North Central    9883640   413
```

Data manipulation - “arrange”

```
dat<-arrange(murders,region,desc(total)) #decreasing order
head(dat)
```

```
##           state abb    region population total
## 1      New York  NY Northeast   19378102   517
## 2  Pennsylvania  PA Northeast   12702379   457
## 3    New Jersey  NJ Northeast    8791894   246
## 4 Massachusetts  MA Northeast    6547629   118
## 5   Connecticut  CT Northeast    3574097    97
## 6  Rhode Island  RI Northeast    1052567    16
```

Data manipulation - “arrange”

Tìm 5 ngày mà cổ phiếu VNM có giá trị giao dịch lớn nhất

Data manipulation - “arrange”

Tìm 5 ngày mà cổ phiếu VNM có giá trị giao dịch lớn nhất

```
stock3<-arrange(stock2,desc(Trading_value)) #decreasing order
head(stock3)
```

```
##        Day    Close  Volume Trading_value
## 1 20171110 107.1466 9270450     993297198
## 2 20200929 109.2000 5191680     566931456
## 3 20160916  82.8198 6624630     548650532
## 4 20171113 113.7431 4495140     511291159
## 5 20161130  82.8198 5410310     448080792
## 6 20160907  87.6209 4166500     365072480
```

Data manipulation - "pipes %>%"

The **pipes** operation (for professionals only :D)

```
dat<-murders
dat %>% mutate(rate=total/population*10^6) %>%
   filter(rate<=15) %>%
   select(state, region, rate)
```

```
##                state         region      rate
## 1           Colorado           West 12.924531
## 2             Hawaii           West  5.145920
## 3              Idaho           West  7.655102
## 4               Iowa North Central  6.893484
## 5              Maine      Northeast  8.280881
## 6          Minnesota North Central  9.992600
## 7            Montana           West 12.128379
## 8      New Hampshire      Northeast  3.798036
## 9       North Dakota North Central  5.947151
```

Data manipulation - trump_tweets dataset

Example 1 Loading dataset "trump_tweets" from "dslabs" library.

- How many variables and observations in the dataset?
- Which two tweets have the highest favorite_count? When did he post these tweets?
- Calculate the average number of favorite count of each tweet before and after Nov 08, 206 ? What do you think about the result?

Data manipulation - trump_tweets dataset

How many variables and observations in the dataset?

```
mydat<-trump_tweets
str(mydat)
```

```
## 'data.frame':    20761 obs. of  8 variables:
##  $ source                 : chr  "Twitter Web Client" "Twit
##  $ id_str                 : chr  "6971079756" "6312794445"
##  $ text                   : chr  "From Donald Trump: Wishir
##  $ created_at             : POSIXct, format: "2009-12-23 12
##  $ retweet_count          : int  28 33 13 5 7 4 2 4 1 22 ..
##  $ in_reply_to_user_id_str: chr  NA NA NA NA ...
##  $ favorite_count         : int  12 6 11 3 6 5 2 10 4 30 ..
##  $ is_retweet             : logi  FALSE FALSE FALSE FALSE F
```

Data manipulation - trump_tweets dataset

Which two tweets have the highest favorite_count? When did he post these tweets?

Data manipulation - trump_tweets dataset

Which two tweets have the highest favorite_count? When did he post these tweets?

```
indices<-order(mydat$favorite_count,decreasing=TRUE)[1:5]
mydat$text[indices[1]]
```

```
## [1] "Such a beautiful and important evening! The forgotten
```

```
mydat$created_at[indices[1]]
```

```
## [1] "2016-11-09 06:36:58 EST"
```

```
mydat$text[indices[2]]
```

```
## [1] "Why would Kim Jong-un insult me by calling me \"old,\'
```

```
mydat$created_at[indices[2]]
```

```
## [1] "2017-11-11 19:48:01 EST"
```

Data manipulation - trump_tweets dataset

Calculate the average number of favorite count of each tweet before and after Nov 08, 206

Data manipulation - trump_tweets dataset

Calculate the average number of favorite count of each tweet before and after Nov 08, 206

```
mytime<-as.POSIXct("2016-11-08 00:00:00")
indices<-as.numeric(mydat$created_at)>=as.numeric(mytime)
mean(mydat$favorite_count[indices])#after
```

```
## [1] 84197.31
```

```
#median(mydat$favorite_count[indices],0.1)#after
indices<-as.numeric(mydat$created_at)<as.numeric(mytime)
mean(mydat$favorite_count[indices])#before
```

```
## [1] 3973.184
```

```
#median(mydat$favorite_count[indices],0.1)#before
```

Summarizing data with "group_by"

```
murders %>% mutate(rate = total/population*10^6) %>%
  group_by(region) %>%
  summarize(mean_rate = mean(rate))
```

```
## # A tibble: 4 x 2
##   region        mean_rate
##   <fct>             <dbl>
## 1 Northeast          18.5
## 2 South              44.2
## 3 North Central      21.8
## 4 West               18.3
```

Summarizing data with "group_by"

```
murders %>% mutate(rate = total/population*10^6) %>%
  group_by(region) %>%
  summarize(mean_rate = mean(rate),sd_rate =sd(rate),
            min_rate=min(rate), max_rate = max(rate))
```

```
## # A tibble: 4 x 5
##   region        mean_rate sd_rate min_rate max_rate
##   <fct>             <dbl>   <dbl>    <dbl>    <dbl>
## 1 Northeast          18.5    11.7     3.20     36.0
## 2 South              44.2    33.7    14.6     165.
## 3 North Central      21.8    14.4     5.95     53.6
## 4 West               18.3    11.7     5.15     36.3
```

Summarizing data with “group_by”

- Tạo thêm 1 cột tên là "Year" trong dataset stock2 chứa là năm của giao dịch (sử dụng hàm "substr").
- Hãy cho biết giá trị giao dịch trung bình của cổ phiếu VNM qua các năm.

Summarizing data with “group_by”

- Tạo thêm 1 cột tên là "Year" trong dataset stock2 chứa là năm của giao dịch (sử dụng hàm "substr").
- Hãy cho biết giá trị giao dịch trung bình của cổ phiếu VNM qua các năm.

```
stock2<-mutate(stock2,year = substr(Day,1,4))
str(stock2)
```

```
## 'data.frame':    3675 obs. of  5 variables:
##  $ Day          : int  20060119 20060120 20060123 20060124
##  $ Close        : num  2.77 2.83 2.75 2.72 2.77 ...
##  $ Volume       : int  109350 157840 81400 66000 57220 1875
##  $ Trading_value: num  303435 446261 223744 179685 158780 .
##  $ year         : chr  "2006" "2006" "2006" "2006" ...
```

Summarizing data with “group_by”

Hãy cho biết giá trị giao dịch trung bình của cổ phiếu VNM qua các năm.

Summarizing data with "group_by"

Hãy cho biết giá trị giao dịch trung bình của cổ phiếu VNM qua các năm.

```
stock2%>%group_by(year)%>%
  summarise(avg_trading_value=mean(Trading_value))
```

```
## # A tibble: 15 x 2
##    year  avg_trading_value
##    <chr>             <dbl>
##  1 2006              863809.
##  2 2007             1955468.
##  3 2008              853602.
##  4 2009             2134191.
##  5 2010             1689381.
##  6 2011             1037074.
##  7 2012             1939773.
##  8 2013             8712007.
##  9 2014             7946537.
```

Summarizing data with “group_by”

How many tweets Trump posted in each year from (2009 to 2016)?

Summarizing data with "group_by"

How many tweets Trump posted in each year from (2009 to 2016)?

```
dat<-trump_tweets
dat<-mutate(dat,year=substr(created_at,1,4))
newdat<-dat%>%group_by(year)%>%
  summarise(tweets_number = length(created_at))
newdat
```

```
## # A tibble: 10 x 2
##    year  tweets_number
##    <chr>         <int>
##  1 2009             43
##  2 2010            139
##  3 2011            749
##  4 2012           3206
##  5 2013           5616
##  6 2014           2319
```

Exercise

- 1. Create a data frame named VNM for "VNM" stock, with 3 columns "Day", "Close" and "Volume".
- 2. Function $movavg(x, k)$ in package "pracma" calculates the simple moving average of length k of vector x. Your work: adding 4 columns to VNM dataset, named MA5, MA25, MA80 and MA250, which are the moving averages of lengths 5, 25, 80 and 250 of the Close price of VNM stocks, respectively.
- 3. Average trading value is a measurement of the liquidity. List 5 tickers with the highest average trading value and 5 tickers with the lowest average trading value in **September 2020**

Exercise solution

Question 1: Create a data frame named VNM for "VNM" stock, with 3 columns "Day", "Close" and "Volume".

Exercise solution

Question 1: Create a data frame named VNM for "VNM" stock, with 3 columns "Day", "Close" and "Volume".

```
setwd("C:/Users/AD/Desktop/Tex file/Thu latex/Introduction to
stock<-read.csv("CafeF.HSX.Upto20102020.csv")
names(stock)<-c("Ticker","Day","Open",
                "High", "Low", "Close", "Volume")
stock$Ticker<-as.factor(stock$Ticker)
VNM<-stock%>%
  filter(Ticker=="VNM")%>%
  select("Day","Close","Volume")
```

Exercise solution

Question 2: Adding the moving average of close price

Exercise solution

Question 2: Adding the moving average of close price

```
library(pracma)
VNM<-VNM%>%
  mutate(MA5=movavg(Close,5))%>%
  mutate(MA25=movavg(Close,25))%>%
  mutate(MA80=movavg(Close,80))%>%
  mutate(MA250=movavg(Close,250))
str(VNM)
```

```
## 'data.frame':    3675 obs. of  7 variables:
##  $ Day   : int  20060119 20060120 20060123 20060124 200601
##  $ Close : num  2.77 2.83 2.75 2.72 2.77 ...
##  $ Volume: int  109350 157840 81400 66000 57220 18750 2563
##  $ MA5   : num  2.77 2.8 2.78 2.77 2.77 ...
##  $ MA25  : num  2.77 2.8 2.78 2.77 2.77 ...
##  $ MA80  : num  2.77 2.8 2.78 2.77 2.77 ...
```

Exercise solution

Question 3: The most liquid stocks in September 2020.

Exercise solution

Question 3: The most liquid stocks in September 2020.

```
#filter data for Sep 2020
dat<-stock%>%filter((Day<=20200930)&(Day>=20200901))%>%
  #add colume trading_value
  mutate(trading_value=Close*Volume)%>%
  group_by(Ticker)%>%
  summarise(avg_trading_value=mean(trading_value))%>%
  arrange(desc(avg_trading_value))
dat[1:5,]
```

```
## # A tibble: 5 x 2
##   Ticker      avg_trading_value
##   <fct>                   <dbl>
## 1 VNAll-INDEX     232812545044.
## 2 HPG                359737253.
## 3 STB                204476739.
```

Data tidying - gather()

spread() and *gather()* help us reshape the layout of our data to place variables in columns and observations in rows.

- *gather()* collects a set of column names and places them into a single "key" column

```
dat<-data.frame("Country" = c("Vietnam","Lao", "Cambonia"),
                "2020" = c(5.5, 2.3,1.4),
                "2021" = c(6.7, 3.5, 2.3),
                "2022" = c(6.9, 3.9, 4.5),
                "id" = c("A","B","C"))
names(dat)<-c("Country","2020","2021","2022","id")
dat%>%gather(key = "year", value = "value", c(2:4))
```

```
##     Country id year value
## 1   Vietnam  A 2020   5.5
## 2       Lao  B 2020   2.3
## 3 Cambonia  C 2020   1.4
```

Data tidying - gather()

gather(dat, key = "name_1", value = "name_2", c(2:4))

other_col_1	col_1	col_2	col_3	other_col_2
	1	4	7	
	2	5	8	
	3	6	9	

other_col_1	other_col_2	name_1	name_2
		col_1	*1*
		col_2	*2*
		col_3	*3*
		col_1	4
		col_2	5
		col_3	6
		col_1	7
		col_2	8
		col_3	9

Data tidying - spread()

- *spread()* does the reverse of *gather()*

```
library(dslabs)
dat<-gapminder%>%filter(year %in% c(1990,2000,2010),
                        country %in% c("Vietnam", "China"))%>%
  select(country, year, population, region)
#View(dat)
dat%>%spread(year, population)
```

```
##   country             region       1990       2000       20
## 1   China        Eastern Asia 1154605773 1269974572 1340968
## 2 Vietnam South-Eastern Asia   68209604   80285563   88357
```

Data tidying - spread()

What is going on when we use *spread()* as follows (how)

```
dat<-gapminder%>%filter(year %in% c(1990,2000,2010),
                        country %in% c("Vietnam", "China"))%>%
  select(country, year, population, fertility)
#View(dat)
dat%>%spread(year, population)
```

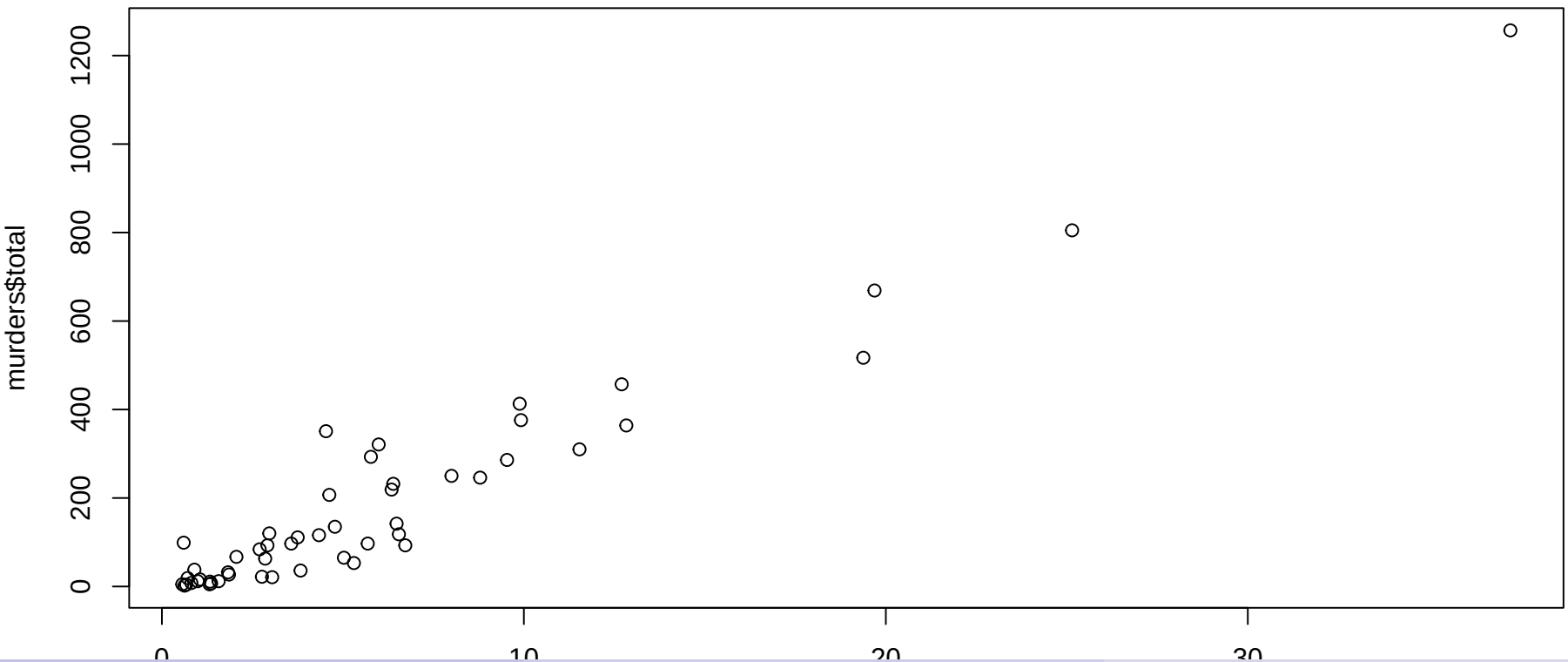
```
##   country fertility       1990       2000       2010
## 1   China      1.45         NA 1269974572         NA
## 2   China      1.54         NA         NA 1340968737
## 3   China      2.43 1154605773         NA         NA
## 4 Vietnam      1.82         NA         NA   88357775
## 5 Vietnam      1.98         NA   80285563         NA
## 6 Vietnam      3.56   68209604         NA         NA
```

Data tidying -

Basic plot with R

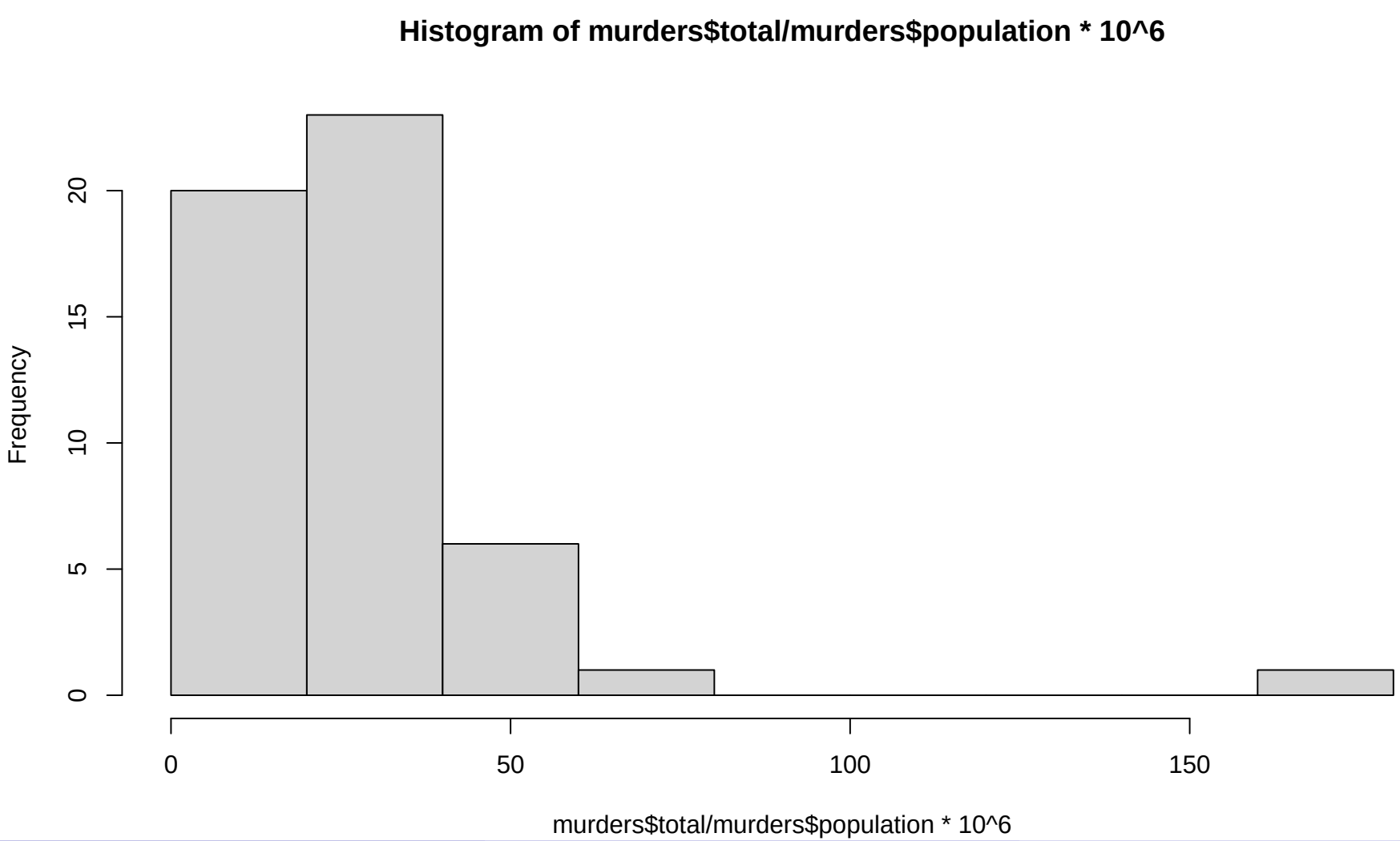
Exploratory data visualization is the main strength of R. Excel is more easier than R in creating plot but it is less flexible.

```
plot(murders$population/10^6,murders$total)
```



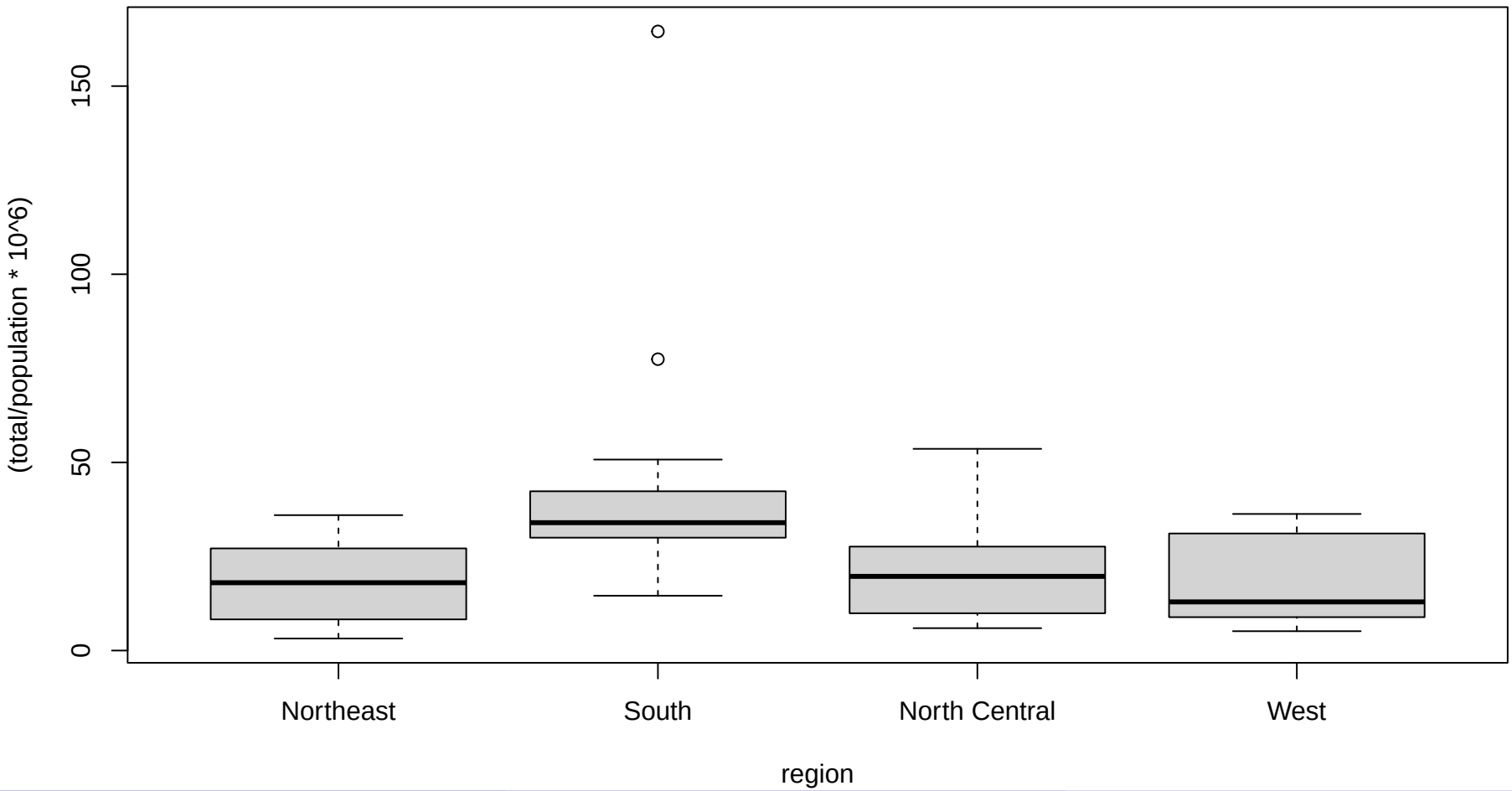
Basic plot with R

```
hist(murders$total/murders$population*10^6)
```



Basic plot with R

```
boxplot((total/population*10^6)~region,data=murders)
```



Example

Load the **dslabs** dataset **gapminder**

- Adding a new column, named GDP_per_capita, which is calculated by dividing the GDP of a country by its population ?
- Is there any difference in GDP_per_capita between the continents in 1960?
- Is there any difference in infant_mortality between the continents in 2000?
- What relationship do you see from the scatterplot of life_expectancy versus GDP_per_capita in 2010?
- What relationship do you see from the scatterplot of fertility versus GDP_per_capita in 2010?
- Which are 10 countries have the largest GDP in 1960?

Example

Adding a new column, named GDP_per_capita, which is calculated by dividing the GDP of a country by its population

Example

Adding a new column, named GDP_per_capita, which is calculated by dividing the GDP of a country by its population

```
dat<-gapminder
dat<-mutate(dat,GDP_per_capita=gdp/population)
str(dat)
```

```
## 'data.frame':    10545 obs. of  10 variables:
##  $ country          : Factor w/ 185 levels "Albania","Algeri
##  $ year             : int  1960 1960 1960 1960 1960 1960 196
##  $ infant_mortality: num  115.4 148.2 208 NA 59.9 ...
##  $ life_expectancy : num  62.9 47.5 36 63 65.4 ...
##  $ fertility        : num  6.19 7.65 7.32 4.43 3.11 4.55 4.8
##  $ population       : num  1636054 11124892 5270844 54681 20
##  $ gdp              : num  NA 1.38e+10 NA NA 1.08e+11 ...
##  $ continent        : Factor w/ 5 levels "Africa","Americas'
##  $ region           : Factor w/ 22 levels "Australia and New
```

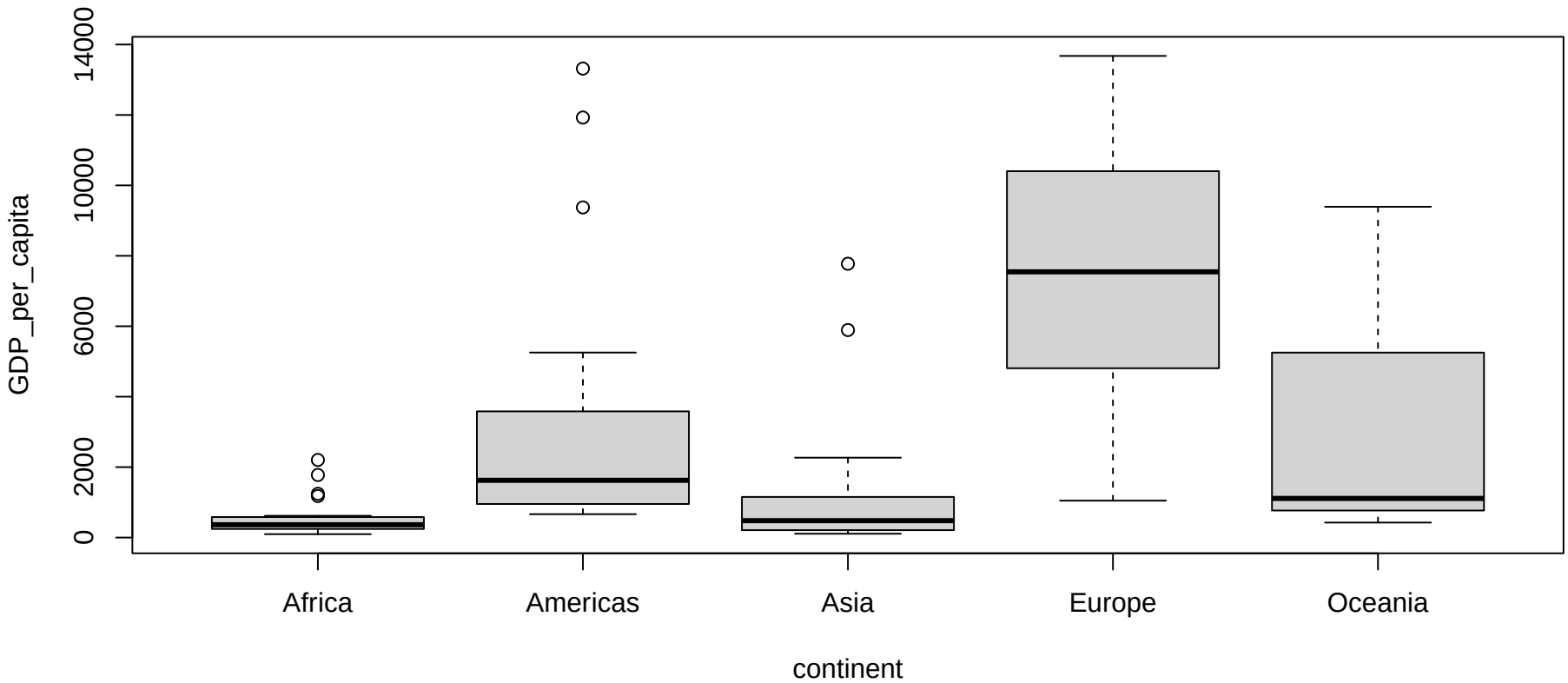
Example

Is there any difference in GDP_per_capita between the continents in 1960?

Example

Is there any difference in GDP_per_capita between the continents in 1960?

```
newdat<-filter(dat,year==1960)
boxplot(GDP_per_capita~continent,data=newdat)
```



Example

Is there any difference in infant_mortality between the continents in 2000?

Example

Is there any difference in infant_mortality between the continents in 2000?

```
newdat<-filter(dat,year==2000)
boxplot(infant_mortality~continent,data=newdat)
```