R basics for beginners

Basque Center for Applied Mathematics

Dae-Jin Lee

`http://idaejin.github.io/bcam-courses`

Why R ?

- **R** is a command-driven statistical package.
- The most important reasons to use **R** are:
  - **R** is free and multiplatform (Windows/Linux/MACos)
  - **R** allows you to do all the statistical tests/models/analysis you need :)
  - Excellent graphics and programming capabilities
  - Growing community of users and developers
  - Lots of online resources
  - An user-friendly interface is RStudio
    `http://www.rstudio.com/`

Statistical features

- Graphical Techniques (Exploratory Data Analysis)
- Linear and non-linear modeling (linear regression, non-parametric regression, smoothing, etc . . .)
- Classical statistical tests
- Time-series analysis
- Econometrics
- Classification and clustering (data mining, machine learning)
- Optimization and Mathematical Programming
- Bayesian inference etc . . . .

Visit `http://cran.r-project.org/web/views` or `http://stackoverflow.com/questions/tagged/r`

Start with **R**

- Get current working directory

```
getwd()
```

- Set working directory

```
setwd("/Users/dlee")
```

Install and load an **R** library

```
install.packages("DAAG") # (Data Analysis And Graphics)
```

- Once installed the package, load it

```
library(DAAG) # or require(DAAG)
```

Data Import

- Several formats are available (`.txt`, `.csv`, `.xls`, `.xlsx`, `SAS`, `Stata`, etc. . .)
- Some **R** libraries to import data are

```
library(gdata)
library(foreign)
```

- Read data from a `.txt` or `.csv` files (*e.g.:* created in `Rintro1.R`)

```
mydata1 = read.table("cardata.txt")
mydata2 = read.csv("cardata.csv")
```

- Other formats `.xls` and `.xlsx`

```
# read in the worksheet named mysheet
mydata <- read.xlsx("myexcel.xlsx", sheetName = "mysheet")
```

- Minitab, SPSS, SAS or Stata

```
library(foreign)
mydata = read.mtp("mydata.mtp")  # Minitab
mydata = read.spss("myfile", to.data.frame=TRUE) # SPSS
mydata = read.dta("mydata.dta") # Stata
```

- Or

```
library(Hmisc)
mydata = spss.get("mydata.por", use.value.labels=TRUE)  # S
```

Exporting data

- There are numerous methods for exporting **R** objects into other formats . For SPSS, SAS and Stata. you will need to load the `foreign` packages. For Excel, you will need the `xlsx` package.
- Tab delimited text file

```
write.table(mydata, "mydata.txt", sep="\t")
```

- Excel spreadsheet

```
library(xlsx)
write.xlsx(mydata, "mydata.xlsx")
```

Data vectors

- Download `R code` here
- Create a vector of weights and heights

```
weight<-c(60,72,57,90,95,72) # function c is used to concat
class(weight)
```

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

```
height<-c(1.75,1.80,1.65,1.90,1.74,1.91)
```

- calculate Body Mass Index

```
bmi<- weight/height^2
bmi
```

```
## [1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.7363
```

Basic statistics

- mean, median, st dev, variance

```
mean(weight)
median(weight)
sd(weight)
var(weight)
```

- summarize data

```
summary(weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   57.00   63.00   72.00   74.33   85.50   95.00
```

- or

```
min(weight)
max(weight)
range(weight)
quantile(weight)
sum(weight)
length(weight)
```

Character vectors and factor variables

```
subject <- c("John","Peter","Chris","Tony","Mary","Jane")
sex <- c("MALE","MALE","MALE","MALE","FEMALE","FEMALE")
class(subject)
```

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

```
table(sex)
```

```
## sex
## FEMALE   MALE
##      2      4
```

Data frames

```
Dat <- data.frame(subject,sex,weight,height)
# add bmi to Dat
Dat$bmi <- bmi  # or Dat$bmi <- weight/height^2
class(Dat)
```

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

```
str(Dat) # display object structure
```

```
## 'data.frame':    6 obs. of  5 variables:
##  $ subject: Factor w/ 6 levels "Chris","Jane",..: 3 5 1
##  $ sex    : Factor w/ 2 levels "FEMALE","MALE": 2 2 2 2
##  $ weight : num  60 72 57 90 95 72
##  $ height : num  1.75 1.8 1.65 1.9 1.74 1.91
##  $ bmi    : num  19.6 22.2 20.9 24.9 31.4 ...
```

```
# Change rownames
rownames(Dat)<-c("A","B","C","D","E","F")


# Access to data frame elements (similar to a matrix)
```

Working with data frames

Example: Analyze data by groups

- Obtain the mean weight, heigth and bmi means by FEMALES and MALES:

1. Select each group and compute the mean

```
Dat[sex=="MALE",]
Dat[sex=="FEMALE",]

mean(Dat[sex=="MALE",3])  # weight average of MALEs
mean(Dat[sex=="MALE","weight"])
```

2. Use apply by columns

```
apply(Dat[sex=="FEMALE",3:5],2,mean)
apply(Dat[sex=="MALE",3:5],2,mean)

# we can use apply with our own function
apply(Dat[sex=="FEMALE",3:5],2,function(x){x+2})
```

Logical vectors

- Choose individuals with BMI>22

```
bmi
bmi>22
as.numeric(bmi>22) # convert a logical condition to a numer
which(bmi>22)  # gives the position of bmi for which bmi>22
```

- Which are between 20 and 25?

```
bmi > 20 & bmi < 25
which(bmi > 20 & bmi < 25)
```

Working with vectors

- Concatenate

```
x <- c(2, 3, 5, 2, 7, 1)
y <- c(10, 15, 12)
z <- c(x,y)  # concatenates x and y
```

- list two vectors

```
zz <- list(x,y) # create a list
unlist(zz) # unlist the list converting it to a concatenate
```

```
## [1]  2  3  5  2  7  1 10 15 12
```

- subset of vectors

```
x[c(1,3,4)]
```

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

```
x[-c(2,6)] # negative subscripts omit the chosen elements
```

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

- Sequences

Matrices and arrays

```
x<- 1:12
x
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```
dim(x)<-c(3,4)  # 3 rows and 4 columns

X <- matrix(1:12,nrow=3,byrow=TRUE)
X <- matrix(1:12,nrow=3,byrow=FALSE)

# rownames, colnames

rownames(X) <- c("A","B","C")
colnames(X) <- LETTERS[4:7]
colnames(X) <- month.abb[4:7]
```

- Column/Row bind operations `cbind()`, `rbind()`

```
Y <- matrix(0.1*(1:12),3,4)

cbind(X,Y)  # bind column-wise
```

Factors

```
gender<-c(rep("female",691),rep("male",692))
class(gender)
```

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

```
# change vector to factor (i.e. a category)
gender<- factor(gender)
levels(gender)
```

```
## [1] "female" "male"
```

```
summary(gender)
```

```
## female   male
##    691    692
```

```
table(gender)
```

```
## gender
## female   male
##    691    692
```

Indexing vector with logicals

```
a <- c(1,2,3,4,5)
b <- c(TRUE,FALSE,FALSE,TRUE,FALSE)

max(a[b])

## [1] 4

sum(a[b])

## [1] 5
```

Missing values (NA)

```
a <- c(1,2,3,4,NA)
sum(a)
```

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

```
sum(a,na.rm=TRUE)
```

```
## [1] 10
```

```
a <- c(1,2,3,4,NA)
is.na(a)
```

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

Working with data frames

- A data frame is used for storing data tables. It is a list of vectors of equal length.

```
mtcars
?mtcars        # or help(mtcars)
```

- look at the first rows

```
head(mtcars)
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1
```

- Structure of the data frame

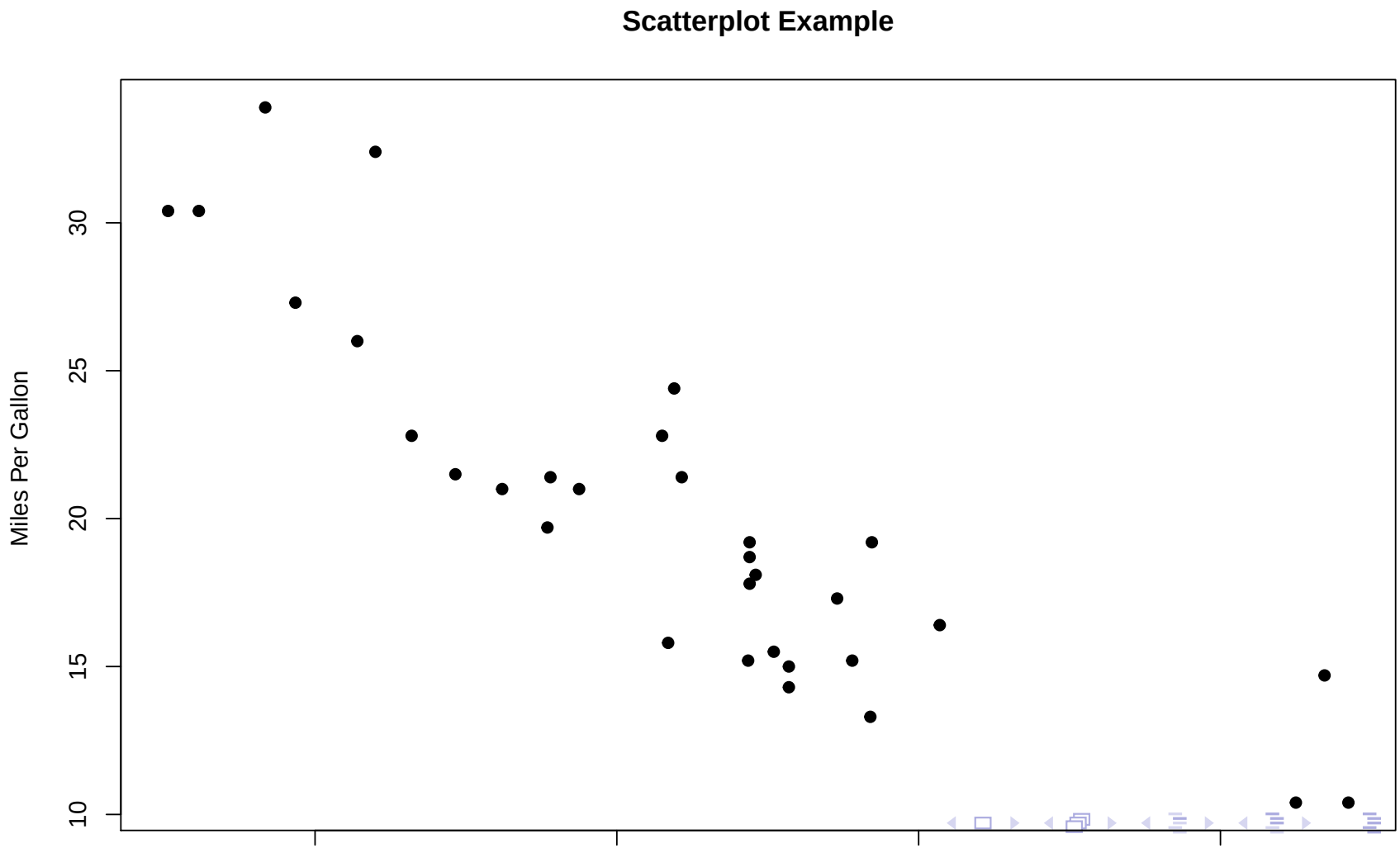
```
str(mtcars) # display the structure of the data frame
```

```
## 'data.frame':    32 obs. of  11 variables:
```

Plotting

- Scatterplot

```
attach(mtcars)
plot(wt, mpg, main="Scatterplot Example",
    xlab="Car Weight ", ylab="Miles Per Gallon ", pch=19)
```



Exercises

1. The data.frame `VADeaths` contains the death rates per 1000 in Virginia (US) in 1940
   - The death rates are measured per 1000 population per year. They are cross-classified by age group (rows) and population group (columns). The age groups are: 50–54, 55–59, 60–64, 65–69, 70–74 and the population groups are Rural/Male, Rural/Female, Urban/Male and Urban/Female.

```
VADeaths
```

```
##       Rural Male Rural Female Urban Male Urban Female
## 50-54       11.7          8.7       15.4          8.4
## 55-59       18.1         11.7       24.3         13.6
## 60-64       26.9         20.3       37.0         19.3
## 65-69       41.0         30.9       54.6         35.1
## 70-74       66.0         54.3       71.1         50.0
```

   - Compute the mean for each age group.
     - **Result:**

```
##   50-54   55-59   60-64   65-69   70-74
```