R basics

Basic commands

Packages

Packages provide supplement to the Built-in functions of R. Check the list of installed packages:

```
library()
```

Let us, for example, install the AER (applied econometrics with R) package and the ISLR2 package. The ISLR2 package comes with datasets used by the textbook. The dependencies=NA option specifies that if the package depends for its operation on other packages, these should be installed as well (if they have not already been installed). Setting dependencies=TRUE installs all packages that depend on the package.

To get an overview of an installed package:

```
help(package="ISLR2")
```

Working directory

Get working directory:

```
getwd()
```

```
## [1] "/Users/junma/OneDrive/Statistical Learning/Labs/introduction"
```

Set working directory:

```
?setwd()
```

Or in R Studio, use

- Session->Set Working Directory, or
- Tools->Global Options.

Vectors and matrices

Generate a vector:

```
x<-c(1,2,3)
x
```

```
## [1] 1 2 3
```

```
typeof(x)
```

```
## [1] "double"
```

Check the length"

```
length(x)
```

```
## [1] 3
```

```r
x<-c("No","Yes")
x
```

```
## [1] "No"  "Yes"
```

```r
typeof(x)
```

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

Generate a matrix:

```r
X<-matrix(c(1,2,3,4),ncol=2)
X
```

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

```r
typeof(X)
```

```
## [1] "double"
```

Note: `R` fills in a matrix on a column-by-column basis.

Add a vector as another column:

```r
x=c(5,6)
Y=cbind(X,x)
Y
```

```
##          x
## [1,] 1 3 5
## [2,] 2 4 6
```

Add a vector as another row:

```r
y=c(7,8,9)
rbind(Y,y)
```

```
##       x
##   1 3 5
##   2 4 6
## y 7 8 9
```

The `ls()` function allows us to look at a list of all of the objects, such as data and functions, that we have saved so far.

```r
ls()
```

```
## [1] "x" "X" "y" "Y"
```

The `rm()` function can be used to delete any that we don't want.

```r
ls()
```

```
## [1] "x" "X" "y" "Y"
```

```r
rm(x)
ls()
```

```
## [1] "X" "y" "Y"
```

It's also possible to remove all objects at once:

```
rm(list = ls())
```

Random matrix: generate eight independent $N(0,1)$ random variables arranged in 4 columns:

```
X=matrix(rnorm(8),ncol=4)
X
```

```
##            [,1]       [,2]       [,3]       [,4]
## [1,] -0.8467866 -0.1569592 -0.9882767 -0.3729088
## [2,]  0.8649702  1.0736745  0.6354491  0.5291373
```

Choose the mean and the standard deviation:

```
X=matrix(rnorm(8,mean=1,sd=.1),ncol=2)
X
```

```
##           [,1]      [,2]
## [1,] 1.1254492 0.9283011
## [2,] 1.0908651 1.1868552
## [3,] 1.0219595 1.1450776
## [4,] 0.8481265 1.0817983
```

Picking specific elements:

```
X[1,2]
```

```
## [1] 0.9283011
```

Pick an entire column (first column):

```
X[,1]
```

```
## [1] 1.1254492 1.0908651 1.0219595 0.8481265
```

Pick an entire row:

```
X[1,]
```

```
## [1] 1.1254492 0.9283011
```

Pick rows 3 & 4:

```
X[c(3,4),]
```

```
##           [,1]     [,2]
## [1,] 1.0219595 1.145078
## [2,] 0.8481265 1.081798
```

Sequences:

```
?seq
x=seq(1,10,by=2)
x
```

```
## [1] 1 3 5 7 9
```

Matrix algebra operations:

```
X=matrix(seq(-1,-4,by=-1),ncol=2)
Y=matrix(seq(1,4),ncol=2)
X
```

```
##      [,1] [,2]
## [1,]   -1   -3
## [2,]   -2   -4
```

```
Y
```

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

Matrix addition:

```
X+Y
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

Matrix product:

```
X%*%Y
```

```
##      [,1] [,2]
## [1,]   -7  -15
## [2,]  -10  -22
```

Transpose:

```
t(X)
```

```
##      [,1] [,2]
## [1,]   -1   -2
## [2,]   -3   -4
```

Element-by-element operations:

```
sqrt(Y)
```

```
##          [,1]     [,2]
## [1,] 1.000000 1.732051
## [2,] 1.414214 2.000000
```

```
X*Y
```

```
##      [,1] [,2]
## [1,]   -1   -9
## [2,]   -4  -16
```

```
1/Y
```

```
##      [,1]      [,2]
## [1,]  1.0 0.3333333
## [2,]  0.5 0.2500000
```

```
Y^X
```

```
##      [,1]       [,2]
## [1,] 1.00 0.03703704
## [2,] 0.25 0.00390625
```

Working with data

Data frames

The basic object that is used by `R` to store data is a data frame: tabular data consisting of rows (observations) and columns (variables).

```
x=c(1,2,3,4)
y=c("male","male","female","female")
X=cbind(x,y)
```

When combining x and y in a matrix, x is converted into characters:

```
X
```

```
##      x   y
## [1,] "1" "male"
## [2,] "2" "male"
## [3,] "3" "female"
## [4,] "4" "female"
```

```
typeof(X)
```

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

Data frames can have variables (columns) of different types. There are relationships between the columns: each row is an observation.

```
Data=data.frame(years=x,gender=as.factor(y))
typeof(Data)
```

```
## [1] "list"
```

```
Data
```

```
##   years gender
## 1     1   male
## 2     2   male
## 3     3 female
## 4     4 female
```

Note that gender is now a factor! (Factors are variables that take on limited number of values. They are used to categorize data by levels. Can be integers or characters.)

```
class(Data$years)
```

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

```
class(Data$gender)
```

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

The `summary()` and `names()` commands on Data:

```
names(Data)
```

```
## [1] "years"  "gender"
```

```
summary(Data)
```

```
##      years          gender
##  Min.   :1.00   female:2
##  1st Qu.:1.75   male  :2
##  Median :2.50
##  Mean   :2.50
##  3rd Qu.:3.25
##  Max.   :4.00
```

Load data

Data can be loaded from external files using:

- `read.table()`
- `read.csv()`
- `read.xlsx()`

We load data from a text file, `Auto.data`:

```
Auto <- read.table("Auto.data")
```

Once the data has been loaded, the `View()` function can be used to view it in a spreadsheet-like window. The `head()` function can also be used to view the first few rows of the data.

```
View(Auto)
head(Auto)
```

```
##     V1        V2           V3         V4     V5           V6   V7     V8
## 1  mpg cylinders displacement horsepower weight acceleration year origin
## 2 18.0         8        307.0      130.0  3504.         12.0   70      1
## 3 15.0         8        350.0      165.0  3693.         11.5   70      1
## 4 18.0         8        318.0      150.0  3436.         11.0   70      1
## 5 16.0         8        304.0      150.0  3433.         12.0   70      1
## 6 17.0         8        302.0      140.0  3449.         10.5   70      1
##                          V9
## 1                      name
## 2 chevrolet chevelle malibu
## 3         buick skylark 320
## 4        plymouth satellite
## 5             amc rebel sst
## 6               ford torino
```

Using the option `header = T` (or `header = TRUE`) in the `read.table()` function tells `R` that the first line of the file contains the variable names, and using the option `na.strings` tells `R` that any time it sees a particular character or set of characters (such as a question mark), it should be treated as a missing element of the data matrix. The `stringsAsFactors = T` argument tells `R` that any variable containing character strings should be interpreted as a qualitative variable, and that each distinct character string represents a distinct level for that qualitative variable.

```
Auto <- read.table("Auto.data", header = T, na.strings = "?", stringsAsFactors = T)
View(Auto)
```

An easy way to load data from Excel into `R` is to save it as a csv (comma-separated values) file, and then use the `read.csv()` function.

```
Auto <- read.csv("Auto.csv", na.strings = "?", stringsAsFactors = T)
View(Auto)
dim(Auto)
```

```
## [1] 397   9
```

The `dim()` function tells us that the data has 397 observations, or rows, and nine variables, or columns:

```
dim(Auto)
```

```
## [1] 397   9
```

There are various ways to deal with the missing data. In this case, only five of the rows contain missing observations, and so we choose to use the `na.omit()` function to simply remove these rows.

```
Auto <- na.omit(Auto)
dim(Auto)
```

```
## [1] 392   9
```

Once the data are loaded correctly, we can use `names()` to check the variable names.

```
names(Auto)
```

```
## [1] "mpg"          "cylinders"    "displacement" "horsepower"   "weight"
## [6] "acceleration" "year"         "origin"       "name"
```

Many R packages come with imported data sets. Package `ISLR2` contains data `Boston` on housing values in Boston area:

```
library(ISLR2)
?Boston
```

Quick inspection of the data:

```
summary(Boston)
```

```
##       crim                zn             indus            chas
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08205   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       nox               rm             age              dis
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       rad              tax           ptratio          lstat
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   : 1.73
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.: 6.95
##  Median : 5.000   Median :330.0   Median :19.05   Median :11.36
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :12.65
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:16.95
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :37.97
##       medv
##  Min.   : 5.00
##  1st Qu.:17.02
##  Median :21.20
##  Mean   :22.53
##  3rd Qu.:25.00
##  Max.   :50.00
```

The first 4 observations:

```
Boston[1:4,]
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
```

Also the first 4 observations:

```
head(Boston,4)
```

```
##      crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3  4.98 24.0
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8  9.14 21.6
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8  4.03 34.7
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7  2.94 33.4
```

The last 4 observations:

```
tail(Boston,4)
```

```
##        crim zn indus chas   nox    rm  age    dis rad tax ptratio lstat medv
## 503 0.04527  0 11.93    0 0.573 6.120 76.7 2.2875   1 273      21  9.08 20.6
## 504 0.06076  0 11.93    0 0.573 6.976 91.0 2.1675   1 273      21  5.64 23.9
## 505 0.10959  0 11.93    0 0.573 6.794 89.3 2.3889   1 273      21  6.48 22.0
## 506 0.04741  0 11.93    0 0.573 6.030 80.8 2.5050   1 273      21  7.88 11.9
```