# Session 2 - Introducing R 2

*Farokh Kakaei*

*January 2015*

## Contents

# 1 Subsetting

- Accessing values to a specific portion of a data structure (vectors, matrices, lists, data frames)

## 1.1 Vectors

- Subsetting vectors using an integer index (vectors are one based, i.e. indices start from 1)

```
# make randomness reproducible

# creating and showing base vector
vec<-sample(1:100, 15, replace=FALSE)
vec
```

```
##  [1] 53 51 50 84  1 75  5 31 21 27 20 56 85 97  4
```

```
# retrieving the fifth element
vec[5]
```

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

```
# assigning 500 to the fifth element
vec[5]<-500
vec
```

```
##  [1]  53  51  50  84 500  75   5  31  21  27  20  56  85  97   4
```

- Negative index

```
# creating and showing the base vector
vec<-sample(1:100, 15, replace=FALSE)
vec
```

```
##  [1] 86 83 91 82 66 14 64 65 97  2 94 60 88 74 77
```

```
# retrieving all elements except the fifth one
vec[-5]
```

```
##  [1] 86 83 91 82 14 64 65 97  2 94 60 88 74 77
```

```
# assigning 500 to all elements except the fifth one
vec[-5]<-500
vec
```

```
##  [1] 500 500 500 500  66 500 500 500 500 500 500 500 500 500 500
```

- Subsetting vectors using an integer index vector(a vector of indices)

```r
vec<-sample(1:100, 15, replace=FALSE)
vec
```

```
##  [1]  3 46 71 18 37 42 39 50 70 51 20 35 44 80  8
```

```r
vec[6:10]
```

```
## [1] 42 39 50 70 51
```

```r
vec2<-vec
vec3<-vec
vec2[6:10]<-500
vec3[6:10]<-c(500, 600, 700, 800, 1000)
vec2
```

```
##  [1]   3  46  71  18  37 500 500 500 500 500  20  35  44  80   8
```

```r
vec3
```

```
##  [1]    3   46   71   18   37  500  600  700  800 1000   20   35   44   80
## [15]    8
```

```r
vec[seq(1,15, by=3)]
```

```
## [1]  3 18 39 51 44
```

```r
vec2<-vec
vec3<-vec
vec2[seq(1,15, by=3)]<-100
vec3[seq(1,15, by=3)]<-c(100, 200, 300, 400, 500)
vec2
```

```
##  [1] 100  46  71 100  37  42 100  50  70 100  20  35 100  80   8
```

```r
vec3
```

```
##  [1] 100  46  71 200  37  42 300  50  70 400  20  35 500  80   8
```

```r
vec[c(1,3,7, 13)]
```

```
## [1]  3 71 39 44
```

```r
vec2<-vec
vec3<-vec
vec2[c(1,3,7, 13)]<-200
vec3[c(1,3,7, 13)]<-c(200, 300, 400, 500)
vec2
```

```
##  [1] 200  46 200  18  37  42 200  50  70  51  20  35 200  80   8
```

```
vec3
```

```
##  [1] 200  46 300  18  37  42 400  50  70  51  20  35 500  80   8
```

```
vec[rep(c(1,2), 2)]
```

```
## [1]  3 46  3 46
```

```
vec2<-vec
vec3<-vec
vec2[rep(c(1,2), 2)]<-300
vec3[rep(c(1,2), 2)]<-c(300, 400, 500, 600)
vec2
```

```
##  [1] 300 300  71  18  37  42  39  50  70  51  20  35  44  80   8
```

```
vec3
```

```
##  [1] 500 600  71  18  37  42  39  50  70  51  20  35  44  80   8
```

```
iv<-sample(1:15, 6)
iv
```

```
## [1]  7 15  4  3 11 14
```

```
vec[iv]
```

```
## [1] 39  8 18 71 20 80
```

```
vec2<-vec
vec3<-vec
vec2[iv]<-400
vec3[iv]<-c(400, 500, 600, 700, 800, 900)
vec2
```

```
##  [1]   3  46 400 400  37  42 400  50  70  51 400  35  44 400 400
```

```
vec3
```

```
##  [1]   3  46 700 600  37  42 400  50  70  51 800  35  44 900 500
```

- Negative index vectors

```
vec<-sample(1:100, 15, replace=FALSE)
vec
```

```
##  [1]  9 67 21 23 30 66 50 85 73 86 20 65 41  2 24
```

```
vec[-10:-6]
```

```
##  [1]  9 67 21 23 30 20 65 41  2 24
```

```
vec2<-vec
vec2[-10:-6]<-100
vec2
```

```
##  [1] 100 100 100 100 100  66  50  85  73  86 100 100 100 100 100
```

```
vec[-(6:10)]
```

```
##  [1]  9 67 21 23 30 20 65 41  2 24
```

```
vec2<-vec
vec2[-(6:10)]<-200
vec2
```

```
##  [1] 200 200 200 200 200  66  50  85  73  86 200 200 200 200 200
```

```
vec[-c(1,3,7, 13)]
```

```
##  [1] 67 23 30 66 85 73 86 20 65  2 24
```

```
vec2<-vec
vec2[-c(1,3,7, 13)]<-300
vec2
```

```
##  [1]   9 300  21 300 300 300  50 300 300 300 300 300  41 300 300
```

```
iv<-sample(1:15, 6)
iv
```

```
## [1]  4  8  9 11 12  5
```

```
vec[-iv]
```

```
## [1]  9 67 21 66 50 86 41  2 24
```

```
vec2<-vec
vec2[-iv]<-400
vec2
```

```
##  [1] 400 400 400  23  30 400 400  85  73 400  20  65 400 400 400
```

- Using name indices

```r
vec1<-c(1, 2,3 ,4, 5)
names(vec1)<-c("A", "B", "C", "D", "E")
vec1
```

```
## A B C D E
## 1 2 3 4 5
```

```r
vec2<-c(a=1, b=2, c=3, d=4, e=5)
vec2
```

```
## a b c d e
## 1 2 3 4 5
```

```r
vec1["B"]
```

```
## B
## 2
```

```r
vec2["c"]
```

```
## c
## 3
```

```r
vec1[c("B", "A", "C")]
```

```
## B A C
## 2 1 3
```

```r
vec2[c("a", "e", "b")]
```

```
## a e b
## 1 5 2
```

- Using logical vectors: accessing values meeting some condition(s)

```r
# creating and showing the base vector
vec<-sample(seq(-100,100), 20)
vec
```

```
##  [1]  46  76 -96 -87  15  71 -95  21  28  49  44  -3   1  68  10   7  89
## [18]  24 -63 -33
```

```r
# retrieve all the values lower than zero
vec[vec<0]
```

```
## [1] -96 -87 -95  -3 -63 -33
```

```r
# change all the values lower than zero to 200
vec2<-vec
vec2[vec2<0] <- 200
vec
```

```
##  [1]  46  76 -96 -87  15  71 -95  21  28  49  44  -3   1  68  10   7  89
## [18]  24 -63 -33
```

```r
vec2
```

```
##  [1]  46  76 200 200  15  71 200  21  28  49  44 200   1  68  10   7  89
## [18]  24 200 200
```

```r
# retrieve all the values greater than zero and lower than 50
vec[vec>0 & vec<50]
```

```
##  [1] 46 15 21 28 49 44  1 10  7 24
```

```r
# change all the values greater than zero and lower than 50 to 300
vec2<-vec
vec2[vec>0 & vec<50] <- 300
vec
```

```
##  [1]  46  76 -96 -87  15  71 -95  21  28  49  44  -3   1  68  10   7  89
## [18]  24 -63 -33
```

```r
vec2
```

```
##  [1] 300  76 -96 -87 300  71 -95 300 300 300 300  -3 300  68 300 300  89
## [18] 300 -63 -33
```

```r
# retrieve all the even values
vec[(vec %% 2)==0]
```

```
## [1]  46  76 -96  28  44  68  10  24
```

```r
# retrieve all the values greater than mean
mean(vec)
```

```
## [1] 8.6
```

```r
vec[vec > mean(vec)]
```

```
##  [1] 46 76 15 71 21 28 49 44 68 10 89 24
```

```r
# retrieve all the values which their distance from
# mean is lower than one standard deviation
mean(vec)
```

```
## [1] 8.6
```

```r
sd(vec)
```

```
## [1] 56.84179
```

```r
vec[abs(vec-mean(vec)) < sd(vec)]
```

```
##  [1]  46  15  21  28  49  44  -3   1  10   7  24 -33
```

```r
# correct character values
x<-c("Bill", "Mike", "Bill", "John", "Mike", "Bill")
x
```

```
## [1] "Bill" "Mike" "Bill" "John" "Mike" "Bill"
```

```r
x[x=="Bill"]<-"William"
x
```

```
## [1] "William" "Mike"    "William" "John"    "Mike"    "William"
```

## 1.2 Matrices and Arrays

- Retrieving entire rows or columns

```r
# creating and showing base matrix
mat<-matrix(sample(1:50, 24), 4, 6)
mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   37   42   33   16   28    8
## [2,]   19   13   20    9   31   46
## [3,]    3   18   11   48   24    7
## [4,]   49   26   10   40   12   45
```

```r
# retrieving row 3
mat[3,]
```

```
## [1]  3 18 11 48 24  7
```

```r
# retrieving all rows but 3
mat[-3,]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   37   42   33   16   28    8
## [2,]   19   13   20    9   31   46
## [3,]   49   26   10   40   12   45
```

```r
# retrieving rows 1 and 3
mat[c(1,3),]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   37   42   33   16   28    8
## [2,]    3   18   11   48   24    7
```

```r
# retrieving all rows but 1 and 3
mat[-c(1,3),]
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]   19   13   20    9   31   46
## [2,]   49   26   10   40   12   45
```

```r
# retrieving column 3
mat[,3]
```

```
## [1] 33 20 11 10
```

```r
# retrieving all columns but 3
mat[,-3]
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   37   42   16   28    8
## [2,]   19   13    9   31   46
## [3,]    3   18   48   24    7
## [4,]   49   26   40   12   45
```

```r
# retrieving columns 3 and 6
mat[,c(3,6)]
```

```
##      [,1] [,2]
## [1,]   33    8
## [2,]   20   46
## [3,]   11    7
## [4,]   10   45
```

```r
# retrieving all columns but 3 and 6
mat[, -c(3,6)]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   37   42   16   28
## [2,]   19   13    9   31
## [3,]    3   18   48   24
## [4,]   49   26   40   12
```

- Using integer indices (matrix rows and columns are one based, i.e. indices start from 1)

```r
# creating and showing base matrix
mat<-matrix(sample(1:20, 12), 3, 4)
mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   17    1   10    4
## [2,]    2   14    3    8
## [3,]    5    7   16   13
```

```r
# retrieving the element at row 2 and column 3
mat[2,3]
```

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

```r
# assigning 50 to the element at row 2 and column 3
mat[2,3]<-50
mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   17    1   10    4
## [2,]    2   14   50    8
## [3,]    5    7   16   13
```

- Using integer index vectors

```r
# creating and showing base matrix
mat<-matrix(sample(1:30, 12), 3, 4)
mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   22   28   24   30
## [2,]    1   16   29   23
## [3,]   20   27    3   10
```

```r
# retrieving the elements at rows 1 and 2  and columns 2 and 3
mat[1:2,2:3]
```

```
##      [,1] [,2]
## [1,]   28   24
## [2,]   16   29
```

```r
# assigning a submatrix to the elements at rows 1 and 2  and columns 2 and 3
submat<-matrix(c(50, 51,52,53), 2, 2)
mat[1:2,2:3]<-submat
mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   22   50   52   30
## [2,]    1   51   53   23
## [3,]   20   27    3   10
```

- Using an index matrix

```
# creating and showing base matrix
mat<-matrix(sample(1:50, 24), 4, 6)
mat
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    2   37   36   39   21   14
## [2,]   28   13   44   24   29   18
## [3,]    6   27   15    3   32   11
## [4,]   12   17   10   26   49    5
```

```
# retrieving the elements at rows 1 and 2  and columns 2 and 3
im<-matrix(c(1,2,2,4,4,6), 3,2)
im
```

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

```
mat[im]
```

```
## [1] 39 24 18
```

- Retrieving vectors as Matrices (drop=FALSE)

```
mat<-matrix(sample(1:50, 12), 3, 4)
mat
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   36   34    4   22
## [2,]   35   23   17   20
## [3,]   16   40    8   48
```

```
# the result of the following command is a vector (oflength one)
mat[2,3]
```

```
## [1] 17
```

```
# the result of the following command is a vector (of length four)
mat[3,]
```

```
## [1] 16 40  8 48
```

```
# the result of the following command is a matrix (one by one)
mat[2,3, drop=FALSE]
```

```
##      [,1]
## [1,]   17
```

```r
# the result of the following command is a matrix (one by four)
mat[3, , drop=FALSE]
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   16   40    8   48
```

## 1.3 Data Frames

- Retireving rows using row indices

```r
# building the sample data frame
id<-1:10
name<-sample(LETTERS, 10, replace=TRUE)
listening<-sample(seq(6,7, by=0.5), 10, replace=TRUE)
reading<-listening + sample(seq(-1,1, 0.5),1)
writing<-listening + sample(seq(-1,1, 0.5),1)
speaking<-listening + sample(seq(-1,1, 0.5),1)
ielts<-data.frame(id, name, listening, reading, writing, speaking)
ielts
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 3   3    G       6.0     6.5     5.0      6.5
## 4   4    K       7.0     7.5     6.0      7.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```r
# retireving row number 4
ielts[4, ]
```

```
##   id name listening reading writing speaking
## 4  4    K         7     7.5       6      7.5
```

```r
# retireving rows 4,6,9
ielts[c(4,6,9), ]
```

```
##   id name listening reading writing speaking
## 4  4    K       7.0     7.5     6.0      7.5
## 6  6    V       6.5     7.0     5.5      7.0
## 9  9    N       6.5     7.0     5.5      7.0
```

```r
# retireving rows 1 to 5
ielts[1:5, ]
```

```
##   id name listening reading writing speaking
## 1  1    H       7.0     7.5     6.0      7.5
## 2  2    E       6.5     7.0     5.5      7.0
## 3  3    G       6.0     6.5     5.0      6.5
## 4  4    K       7.0     7.5     6.0      7.5
## 5  5    T       6.5     7.0     5.5      7.0
```

```
# retireving all rows but 4
ielts[-4, ]
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 3   3    G       6.0     6.5     5.0      6.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```
# retireving all rows but 4,6, and 9
ielts[-c(4,6,9), ]
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 3   3    G       6.0     6.5     5.0      6.5
## 5   5    T       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```
# retireving all rows but 1 to 5
ielts[-(1:5), ]
```

```
##    id name listening reading writing speaking
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

- Retireving columns using column indices

```
# showing the sample data frame again
ielts
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
```

```
## 3   3    G       6.0     6.5     5.0      6.5
## 4   4    K       7.0     7.5     6.0      7.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```
# retireving coulmn 2 (name)
ielts[, 2]
```

```
##  [1] H E G K T V E J N B
## Levels: B E G H J K N T V
```

```
# retireving coulmns 2 (name) and 3 (listening)
ielts[, c(2,3)]
```

```
##    name listening
## 1     H       7.0
## 2     E       6.5
## 3     G       6.0
## 4     K       7.0
## 5     T       6.5
## 6     V       6.5
## 7     E       6.0
## 8     J       6.5
## 9     N       6.5
## 10    B       6.5
```

```
# retireving coulmns 3 to 6
ielts[, 3:6]
```

```
##    listening reading writing speaking
## 1        7.0     7.5     6.0      7.5
## 2        6.5     7.0     5.5      7.0
## 3        6.0     6.5     5.0      6.5
## 4        7.0     7.5     6.0      7.5
## 5        6.5     7.0     5.5      7.0
## 6        6.5     7.0     5.5      7.0
## 7        6.0     6.5     5.0      6.5
## 8        6.5     7.0     5.5      7.0
## 9        6.5     7.0     5.5      7.0
## 10       6.5     7.0     5.5      7.0
```

```
# retireving all coulmns but 2 (name)
ielts[, -2]
```

```
##    id listening reading writing speaking
## 1   1       7.0     7.5     6.0      7.5
## 2   2       6.5     7.0     5.5      7.0
```

```
## 3   3       6.0     6.5     5.0      6.5
## 4   4       7.0     7.5     6.0      7.5
## 5   5       6.5     7.0     5.5      7.0
## 6   6       6.5     7.0     5.5      7.0
## 7   7       6.0     6.5     5.0      6.5
## 8   8       6.5     7.0     5.5      7.0
## 9   9       6.5     7.0     5.5      7.0
## 10 10       6.5     7.0     5.5      7.0
```

```
# retireving all columns but 2 (name) and 3 (listening)
ielts[, -c(2,3)]
```

```
##    id reading writing speaking
## 1   1     7.5     6.0      7.5
## 2   2     7.0     5.5      7.0
## 3   3     6.5     5.0      6.5
## 4   4     7.5     6.0      7.5
## 5   5     7.0     5.5      7.0
## 6   6     7.0     5.5      7.0
## 7   7     6.5     5.0      6.5
## 8   8     7.0     5.5      7.0
## 9   9     7.0     5.5      7.0
## 10 10     7.0     5.5      7.0
```

```
# retireving all coulmns but 3 to 6
ielts[, -(3:6)]
```

```
##    id name
## 1   1    H
## 2   2    E
## 3   3    G
## 4   4    K
## 5   5    T
## 6   6    V
## 7   7    E
## 8   8    J
## 9   9    N
## 10 10    B
```

- Retireving columns using column names

```
# showing the sample data frame again
ielts
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 3   3    G       6.0     6.5     5.0      6.5
## 4   4    K       7.0     7.5     6.0      7.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
```

```
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```
# retireving coulmn "name"
ielts[, "name"]
```

```
##  [1] H E G K T V E J N B
## Levels: B E G H J K N T V
```

```
# retireving coulmns "name" and "listening"
ielts[, c("name" , "listening")]
```

```
##    name listening
## 1     H       7.0
## 2     E       6.5
## 3     G       6.0
## 4     K       7.0
## 5     T       6.5
## 6     V       6.5
## 7     E       6.0
## 8     J       6.5
## 9     N       6.5
## 10    B       6.5
```

- Retireving columns using `<data frame name>$<column name>` (the most convinient way of retrieving a single column)

```
# showing the sample data frame again
ielts
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 3   3    G       6.0     6.5     5.0      6.5
## 4   4    K       7.0     7.5     6.0      7.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```
# retireving coulmn "name"
ielts$name
```

```
##  [1] H E G K T V E J N B
## Levels: B E G H J K N T V
```

```r
# retireving coulmn "listening"
ielts$listening
```

```
##  [1] 7.0 6.5 6.0 7.0 6.5 6.5 6.0 6.5 6.5 6.5
```

```r
# compute mean of writing column
mean(ielts$writing)
```

```
## [1] 5.5
```

- Retireving data based on conditions

```r
# showing the sample data frame again
ielts
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 3   3    G       6.0     6.5     5.0      6.5
## 4   4    K       7.0     7.5     6.0      7.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```r
# retireving rows of data where writing is >= 6.5
ielts[ielts$writing >= 6.5 , ]
```

```
## [1] id        name      listening reading   writing   speaking
## <0 rows> (or 0-length row.names)
```

```r
# retireving rows of data where writing is >= 6.5 and speaking is >=6
ielts[ielts$writing >= 6.5 & ielts$speaking >= 6, ]
```

```
## [1] id        name      listening reading   writing   speaking
## <0 rows> (or 0-length row.names)
```

```r
# retireving rows of data where writing is >= 6.5 or speaking is >=6
ielts[ielts$writing >= 6.5 | ielts$speaking >= 6, ]
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 3   3    G       6.0     6.5     5.0      6.5
## 4   4    K       7.0     7.5     6.0      7.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 7   7    E       6.0     6.5     5.0      6.5
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

```r
# retireving just name column where writing is >= 6.5
ielts[ielts$writing >= 6.5 , "name"]
```

```
## factor(0)
## Levels: B E G H J K N T V
```

```r
# or
ielts[ielts$writing >= 6.5 , 2]
```

```
## factor(0)
## Levels: B E G H J K N T V
```

```r
# or
ielts[ielts$writing >= 6.5 , ]$name
```

```
## factor(0)
## Levels: B E G H J K N T V
```

```r
# retireving id and name columns where speaking > 6
ielts[ielts$speaking > 6 , c("id", "name")]
```

```
##    id name
## 1   1    H
## 2   2    E
## 3   3    G
## 4   4    K
## 5   5    T
## 6   6    V
## 7   7    E
## 8   8    J
## 9   9    N
## 10 10    B
```

```r
# retireving all data where speaking is >= average of speakings
ielts[ielts$speaking >= mean(ielts$speaking) , ]
```

```
##    id name listening reading writing speaking
## 1   1    H       7.0     7.5     6.0      7.5
## 2   2    E       6.5     7.0     5.5      7.0
## 4   4    K       7.0     7.5     6.0      7.5
## 5   5    T       6.5     7.0     5.5      7.0
## 6   6    V       6.5     7.0     5.5      7.0
## 8   8    J       6.5     7.0     5.5      7.0
## 9   9    N       6.5     7.0     5.5      7.0
## 10 10    B       6.5     7.0     5.5      7.0
```

# 2 Input/Output data

- Almost all of the statistical data of the real world projects are saved in files.
- They should be read and loaded to memory to be processed.
- The results of statistical analyses have to be written to files too.

## 2.1 Reading tabular data

- Tabular data are ideal to be loaded as data frames.
- Columns are separated from each other by some separator characters (whitespace, tab, comma, . . . )
- Each line represents a row of data.
- Reading data frames using `read.table()`
  - default separator is whitespace

```
# assuming that the file "ielts.txt" is in the working directory

# read the entire data
ielts2<-read.table("ielts.txt", header=TRUE)
ielts2
```

```
##    id name listening reading writing speaking
## 1   1    N       6.0     5.5     7.0      5.0
## 2   2    Z       7.0     6.5     8.0      6.0
## 3   3    X       6.5     6.0     7.5      5.5
## 4   4    N       6.5     6.0     7.5      5.5
## 5   5    I       6.5     6.0     7.5      5.5
## 6   6    T       6.5     6.0     7.5      5.5
## 7   7    I       7.0     6.5     8.0      6.0
## 8   8    I       6.0     5.5     7.0      5.0
## 9   9    U       7.0     6.5     8.0      6.0
## 10 10    A       7.0     6.5     8.0      6.0
```

```
# read the first 6 rows
ielts3<-read.table("ielts.txt", header=TRUE, nrows=6)
ielts3
```

```
##   id name listening reading writing speaking
## 1  1    N       6.0     5.5     7.0      5.0
## 2  2    Z       7.0     6.5     8.0      6.0
## 3  3    X       6.5     6.0     7.5      5.5
## 4  4    N       6.5     6.0     7.5      5.5
## 5  5    I       6.5     6.0     7.5      5.5
## 6  6    T       6.5     6.0     7.5      5.5
```

- Reading data frames using `read.csv()`
  - csv stands for Comma Separated Values
  - much like `read.table`
  - default separator is comma

```
# assuming that the file "ielts.csv" is in the working directory

# read the entire data
ielts2<-read.csv("ielts.csv", header=TRUE)
ielts2
```

```
##     X id name listening reading writing speaking
## 1   1  1    K       6.5     7.0     5.5      6.5
```

```
## 2   2  2    Y       6.0     6.5     5.0      6.0
## 3   3  3    X       7.0     7.5     6.0      7.0
## 4   4  4    A       6.0     6.5     5.0      6.0
## 5   5  5    Q       6.0     6.5     5.0      6.0
## 6   6  6    X       6.5     7.0     5.5      6.5
## 7   7  7    T       6.0     6.5     5.0      6.0
## 8   8  8    A       6.0     6.5     5.0      6.0
## 9   9  9    J       7.0     7.5     6.0      7.0
## 10 10 10    T       6.5     7.0     5.5      6.5
```

```
# read the first 6 rows
ielts3<-read.csv("ielts.csv", header=TRUE, nrows=6)
ielts3
```

```
##   X id name listening reading writing speaking
## 1 1  1    K       6.5     7.0     5.5      6.5
## 2 2  2    Y       6.0     6.5     5.0      6.0
## 3 3  3    X       7.0     7.5     6.0      7.0
## 4 4  4    A       6.0     6.5     5.0      6.0
## 5 5  5    Q       6.0     6.5     5.0      6.0
## 6 6  6    X       6.5     7.0     5.5      6.5
```

## 2.2 Reading text files

- Use `readLines()` to read all or some lines from a text file

```
# assuming that the file "myfile.txt" is in the working directory

# reading all the file
lines<-readLines("myfile.txt")
lines
```

```
##  [1] "We assume no responsibility for errors or omissions."
##  [2] "Damages resulting from the use of the information contained herein."
##  [3] "Problems specific to the modern age. Problems of the nuclear age. The Victorian age."
##  [4] "It takes ages to cook. I've been waiting for ages."
##  [5] "It's been ages since we last spoke. It's been an age since we last spoke."
##  [6] "Her back went bent with age. This cheese improves with age. This wine improves with age."
##  [7] "His temper hasn't improved with age."
##  [8] "Act you age, please."
##  [9] "He was prosecuted for having sex with a girl who was under age."
## [10] "In this age, it can sometimes seem like every system is connected to every other system."
## [11] "Will you be delivering services or consuming them?"
## [12] "It is a key part of all modern, public-facing applications."
## [13] "This book is here to help you navigate your way along the road ahead."
## [14] "You will see how to devise great solutions."
## [15] "This book has you covered, from technical details to the big picture."
## [16] "PHP has always taken on the mission to solve the web problem."
## [17] "Her voice took on a troubled tone."
```

```r
# reading first 4 lines
lines<-readLines("myfile.txt", 4)
lines
```

```
## [1] "We assume no responsibility for errors or omissions."
## [2] "Damages resulting from the use of the information contained herein."
## [3] "Problems specific to the modern age. Problems of the nuclear age. The Victorian age."
## [4] "It takes ages to cook. I've been waiting for ages."
```

## 2.3 Reading web pages

- `readLines()` can be used to read all or some lines from a web page

```r
# reading all the web page
lines<-readLines("http://www.google.com")
```

```
## Warning in readLines("http://www.google.com"): incomplete final line found
## on 'http://www.google.com'
```

```r
lines
```

```
## [1] "<!doctype html><html itemscope=\"\" itemtype=\"http://schema.org/WebPage\" lang=\"en-IR\"><head
## [2] "function _gjh(){!_gjuc()&&window.google&&google.x&&google.x({id:\"GJH\"},function(){google.nav&&
## [3] "if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}"
## [4] "}"
## [5] "})();</script><div id=\"mngb\">   <div id=gbar><nobr><b class=gb1>Search</b> <a class=gb1 href=
## [6] "a.i.Z,window.gbar.elr&&a.i.$(window.gbar.elr()),window.gbar.elc&&window.gbar.elc(a.i.$),a.i.G(!
## [7] "});})();</script> </div> </span><br clear=\"all\" id=\"lgpd\"><div id=\"lga\"><img alt=\"Google
## [8] "</script></div></body></html>"
```

```r
# reading first 4 lines
lines<-readLines("http://www.google.com", 4)
lines
```

```
## [1] "<!doctype html><html itemscope=\"\" itemtype=\"http://schema.org/WebPage\" lang=\"en-IR\"><head
## [2] "function _gjh(){!_gjuc()&&window.google&&google.x&&google.x({id:\"GJH\"},function(){google.nav&&
## [3] "if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}"
## [4] "}"
```

## 2.4 Writing tabular data

- Writing data frames using `write.table()`
  - default separator is whitespace

```r
# write the entire data
write.table(ielts, "ielts.data.txt", row.names=FALSE)
```

- Writing data frames using `write.csv()`
  - default separator is comma

```
# write the entire data
write.table(ielts, "ielts.data.csv", row.names=FALSE)
```

## 2.5 Writing to text files

- Use `writeLines()` to write text vectors to a text file

```
lines<-c("line 1", "line 2", "line 3")

# writing to file
writeLines(lines, "myTextFile.txt")

# reading file
readLines("myTextFile.txt")
```

```
## [1] "line 1" "line 2" "line 3"
```

## 2.6 Output to the screen

- Object's name followed by Enter

```
# show value of a vector
v<-sample(1:10, 5)
v
```

```
## [1] 7 9 6 4 8
```

```
# show value of a matrix
m<-matrix(sample(1:10, 9), 3, 3)
m
```

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

```
# show value of a data frame
df<-data.frame(a=sample(1:10, 4), b=sample(1:10, 4),
               c=sample(c("AB", "CD", "EF"), 4, replace=TRUE))
df
```

```
##    a  b  c
## 1 10  6 CD
## 2  4  8 CD
## 3  5  3 AB
## 4  8 10 CD
```

```r
# show contents of a function
mySquare<-function(x){
  return(x^2)
}
mySquare
```

```
## function(x){
##   return(x^2)
## }
```

- Using `print()` function

```r
# show value of a vector
v<-sample(1:10, 5)
print(v)
```

```
## [1]  7  6  8  2 10
```

```r
# show value of a matrix
m<-matrix(sample(1:10, 9), 3, 3)
print(m)
```

```
##      [,1] [,2] [,3]
## [1,]    6    7    9
## [2,]    2    4    1
## [3,]   10    5    3
```

```r
# show value of a data frame
df<-data.frame(a=sample(1:10, 4), b=sample(1:10, 4),
               c=sample(c("AB", "CD", "EF"), 4, replace=TRUE))
print(df)
```

```
##   a  b  c
## 1 4  4 AB
## 2 3 10 EF
## 3 9  1 AB
## 4 5  6 EF
```

```r
# show contents of a function
mySquare<-function(x){
  return(x^2)
}
print(mySquare)
```

```
## function(x){
##   return(x^2)
## }
```

- Using `cat()` function to concatenate and print values

- \n means newline
- \t means tab
- \\ means backslash \
- \' means ASCII apostrophe '
- \" means ASCII quotation mark "
- \‘ means ASCII grave accent (backtick) ‘

```
item<-50
cat("Item no :", item)
```

```
## Item no : 50
```

```
result<-matrix(sample(1:10, 9), 3, 3)
cat("The result is \n", result)
```

```
## The result is
##  2 10 1 6 3 5 8 4 7
```

```
v<-c(sample(1:10), 3)
cat("1:", v[1], "\t 2:", v[2], "\t 3:", v[3])
```

```
## 1: 10     2: 5    3: 1
```

```
cat("Let me introduce \"Tom Hanks\"")
```

```
## Let me introduce "Tom Hanks"
```

# 3 Basic R Functions

## 3.1 Exploratory Data Analysis Functions

- Useful functions for briefly exploring data structure and value of objects
- Use `head()` and `tail()` function to view respectively the first and last parts of vectors, matrices, data frames, ....

```
vec<-sample(1:1000, 500)
mat<-matrix(sample(1:10000, 1200), 200, 6)
df<-data.frame(a=vec, b=log(vec), c=sqrt(vec), e=sample(LETTERS, 500, replace=TRUE),
               stringsAsFactors=FALSE)

head(vec)                       # returns the first 10 elements
```

```
## [1] 768 615 828  49 914 902
```

```
head(vec, 15)                   # returns the first 15 elements
```

```
##  [1] 768 615 828  49 914 902 272 756 156 412 829 550 413 938 298
```

```r
tail(vec)                          # returns the last 10 elements
```

```
## [1] 765 352 986 845 389 646
```

```r
tail(vec, 8)                       # returns the last 8 elements
```

```
## [1] 454 256 765 352 986 845 389 646
```

```r
head(mat)                          # returns the first 10 rows
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]  431 4843 5076 8941 2192 5269
## [2,] 4730  868 8117 4159 9946 6526
## [3,] 7794 4513 4733 7150 9391 4050
## [4,] 2186 6568 8506 4407 9970 2229
## [5,] 7594 1726 8461 8205  388  705
## [6,] 4844 1230 7111 1860 6720 5328
```

```r
head(mat, 15)                      # returns the first 15 rows
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6]
##  [1,]  431 4843 5076 8941 2192 5269
##  [2,] 4730  868 8117 4159 9946 6526
##  [3,] 7794 4513 4733 7150 9391 4050
##  [4,] 2186 6568 8506 4407 9970 2229
##  [5,] 7594 1726 8461 8205  388  705
##  [6,] 4844 1230 7111 1860 6720 5328
##  [7,] 6400  890 5187 7696 8881 1617
##  [8,] 4249 7996 4570  837 6657 1858
##  [9,] 3460 4122 3659 7109  386 1778
## [10,] 8380 5363 2716 7679 2776 8691
## [11,] 9727 3145 5410 7166 8262 3987
## [12,] 4713 9094 3696 6818 7253 4028
## [13,] 5081 1795 1091 4255 9989 8508
## [14,] 8809 6937 4162 4731 2448 5746
## [15,] 8541 5047 8648 5446 8447 4492
```

```r
tail(mat)                          # returns the last 10 rows
```

```
##        [,1] [,2] [,3] [,4] [,5] [,6]
## [195,] 7539 4549 1732 7815 6290 5108
## [196,] 5179 3393 1419 3312 9226  762
## [197,] 2446 9184 6042 5596 8507 3528
## [198,] 6509 3439 3802 9583 3849 8416
## [199,] 9884 4219 5828 9565 4874 2382
## [200,] 9623 6947 7549 6906 7020 9538
```

```r
tail(mat, 8)                       # returns the last 8 rows
```

```
##        [,1] [,2] [,3] [,4] [,5] [,6]
## [193,] 1456 8582 5715 9121 6510 3334
## [194,] 7733 8278 7126 5740 3506 1462
## [195,] 7539 4549 1732 7815 6290 5108
## [196,] 5179 3393 1419 3312 9226  762
## [197,] 2446 9184 6042 5596 8507 3528
## [198,] 6509 3439 3802 9583 3849 8416
## [199,] 9884 4219 5828 9565 4874 2382
## [200,] 9623 6947 7549 6906 7020 9538
```

```
head(df)                        # returns the first 10 rows
```

```
##     a        b        c e
## 1 768 6.643790 27.71281 J
## 2 615 6.421622 24.79919 B
## 3 828 6.719013 28.77499 E
## 4  49 3.891820  7.00000 U
## 5 914 6.817831 30.23243 I
## 6 902 6.804615 30.03331 L
```

```
head(df, 15)                    # returns the first 15 rows
```

```
##      a        b        c e
## 1  768 6.643790 27.71281 J
## 2  615 6.421622 24.79919 B
## 3  828 6.719013 28.77499 E
## 4   49 3.891820  7.00000 U
## 5  914 6.817831 30.23243 I
## 6  902 6.804615 30.03331 L
## 7  272 5.605802 16.49242 K
## 8  756 6.628041 27.49545 F
## 9  156 5.049856 12.49000 T
## 10 412 6.021023 20.29778 D
## 11 829 6.720220 28.79236 F
## 12 550 6.309918 23.45208 Y
## 13 413 6.023448 20.32240 Q
## 14 938 6.843750 30.62679 E
## 15 298 5.697093 17.26268 I
```

```
tail(df)                        # returns the last 10 rows
```

```
##       a        b        c e
## 495 765 6.639876 27.65863 N
## 496 352 5.863631 18.76166 L
## 497 986 6.893656 31.40064 W
## 498 845 6.739337 29.06888 E
## 499 389 5.963579 19.72308 G
## 500 646 6.470800 25.41653 X
```

```
tail(df, 8)                     # returns the last 8 rows
```

```
##       a        b        c e
## 493 454 6.118097 21.30728 Z
## 494 256 5.545177 16.00000 S
## 495 765 6.639876 27.65863 N
## 496 352 5.863631 18.76166 L
## 497 986 6.893656 31.40064 W
## 498 845 6.739337 29.06888 E
## 499 389 5.963579 19.72308 G
## 500 646 6.470800 25.41653 X
```

- `table()` function returns the frequency of elements of a vector

```
vec1<-sample(1:10, 20, replace=TRUE)
vec1
```

```
##  [1]  5  1  2  6  2  2  8  5  8  5  9  4 10 10  8  8  4  8  9  4
```

```
table(vec)
```

```
## vec
##   2   5   7   8  14  16  17  18  19  23  25  28  29  30  33  35  36  39
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  44  47  48  49  52  56  58  59  60  62  65  67  70  73  75  79  80
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  81  84  85  87  89  94  97 100 103 105 106 113 115 117 118 119 120 124
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 126 131 132 134 139 140 141 143 145 146 148 149 151 152 154 155 156 158
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 162 164 165 166 167 169 170 171 172 176 184 185 187 188 189 191 192 193
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 196 197 198 200 201 205 207 208 209 211 212 215 219 223 225 227 229 232
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 233 234 235 238 239 241 244 245 248 249 251 252 255 256 257 260 261 262
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 269 271 272 276 277 278 280 281 282 283 285 286 288 289 290 291 298 299
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 303 304 306 309 311 312 313 314 315 317 322 326 328 329 331 334 337 338
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 341 342 343 344 345 346 349 351 352 354 356 357 360 364 365 367 371 372
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 381 382 383 384 385 388 389 392 395 396 397 398 399 402 404 405 408 409
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 410 412 413 414 415 419 420 421 427 428 429 431 438 439 440 441 443 445
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 446 447 449 450 451 452 453 454 456 457 459 460 461 462 464 465 467 469
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 470 473 474 479 480 481 482 484 486 487 488 489 490 491 492 496 497 502
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 503 504 506 508 509 514 516 517 518 522 523 525 526 527 529 531 532 535
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 537 539 541 542 543 549 550 552 553 555 558 562 564 565 568 569 574 575
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 576 578 580 581 582 583 584 585 587 588 591 592 593 596 600 603 604 607
```

```
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 609 612 613 615 616 617 622 625 628 634 638 639 641 644 646 647 648 651
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 652 653 654 659 664 665 667 669 672 674 676 678 680 686 687 695 697 700
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 702 703 707 709 711 714 715 718 720 725 726 730 733 734 736 738 739 740
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 741 742 743 744 745 746 748 750 754 756 757 759 760 761 762 765 766 768
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 770 772 775 776 777 779 781 784 786 787 789 790 791 795 797 799 802 804
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 805 810 813 815 816 820 821 824 826 828 829 830 831 833 834 836 840 841
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 842 844 845 846 849 850 851 852 853 854 855 857 858 860 862 865 867 868
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 872 874 878 879 880 882 889 890 891 894 895 901 902 903 904 906 907 911
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 912 914 918 919 921 924 925 927 929 931 933 934 938 942 946 949 951 953
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 956 957 958 959 962 963 965 967 968 970 971 972 974 975 976 978 979 980
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 982 983 984 985 986 987 988 989 991 994 996 997 998 999
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

```
vec2<-c(TRUE, FALSE, FALSE, FALSE, TRUE)
vec2
```

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

```
table(vec2)
```

```
## vec2
## FALSE  TRUE
##     3     2
```

- Use `str()` function to investigate briefly the internal structure of an object

```
vec<-sample(1:1000, 500)
mat<-matrix(sample(1:10000, 1200), 200, 6)
df<-data.frame(a=vec, b=log(vec), c=sqrt(vec), e=sample(LETTERS, 500, replace=TRUE),
               stringsAsFactors=FALSE)

str(vec)
```

```
##  int [1:500] 179 940 309 963 339 862 970 65 432 139 ...
```

```
str(mat)
```

```
##  int [1:200, 1:6] 1000 1845 6183 7578 2683 9969 654 2696 7344 3190 ...
```

```r
str(df)
```

```
## 'data.frame':    500 obs. of  4 variables:
##  $ a: int  179 940 309 963 339 862 970 65 432 139 ...
##  $ b: num  5.19 6.85 5.73 6.87 5.83 ...
##  $ c: num  13.4 30.7 17.6 31 18.4 ...
##  $ e: chr  "O" "J" "J" "B" ...
```

```r
str(sd)                # structure of sd() function
```

```
## function (x, na.rm = FALSE)
```

## 3.2 Factors

- Used to represent categorical data, for example:
  - Gender: Male, Female
  - Week Days: Sunday, Monday, Tuseday, Wednesday, Thursday, Friday
  - Status: Successful, Failed
- Use `factor()` function to create factor vectors of other data types

```r
gender<-c("Male", "Female", "Male", "Male", "Female", "Female")

f<-factor(gender)
f
```

```
## [1] Male   Female Male   Male   Female Female
## Levels: Female Male
```

```r
str(f)
```

```
##  Factor w/ 2 levels "Female","Male": 2 1 2 2 1 1
```

```r
table(f)
```

```
## f
## Female   Male
##      3      3
```

```r
mode(f)
```

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

- Use `as.factor()` function to coerce other data types to factors

```r
vec1<-sample(1:5, 10, replace=TRUE)
as.factor(vec1)
```

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

```r
vec2<-c("A", "B", "A", "C", "C", "A")
as.factor(vec2)
```

```
## [1] A B A C C A
## Levels: A B C
```