Lab 1

R quick start [1]

- Assignment operator <- or =
- Case sensitive
- Indexes start at 1
- 2-D array notation m[1,2]
- 2-D array storage – column-major order
- Mixed container type – list
- Mechanism for external code packaging – library()
- Run mode – interactive, batch
- Comment symbol #

File “test.R” (version 1)

```
oddcount<-function(x) {
k<-0
for (n in x) {
    if (n%%2==1) #n is odd
      k<-k+1
}
return(k)
}
```

```
>source(“test.R”)  #load code from the file
>ls() #what objects we have
>class(oddcount)  #what kind of object is oddcount
>oddcount  #or print(oddcount)
```

```
>y<-c(3,6,2,8,9)  #concatenate function
>y
>y[3]
>y[1:4]
>y[c(1,3:5)]
>oddcount(y)
```

[1] Norman Martoff. Probability and Statistics for Data Science Math+R+Data, CRC Press, 2019.

File “test.R” (version 2)

```
    oddcount<-function(x) {
    x1<-(x%%2==1)  #x1 is a vector of TRUEs and FALSEs
    x2<-x[x1]
    return (length(x2))
    }
```

```
>oddcount(y[2:5])
```

File “test.R” (version 3)

```
    oddcount<-function(x) {
    length(x[x%%2==1])    #the last value computed is auto returned
    }
```

File “test.R” (version 4)

```
    oddcount<-function(x) sum(x%%2==1)
```

```
>which(y %% 2==1)  #which elements are odd
```

File “test.R” (version 5)

```
    oddcount<-function(x) {
    x1<-x[x%%2==1]
    return (list(odds=x1,numodds=length(x1)) )
    }
```

```
# rbind(), cbind() functions combine rows/columns of matrices

>m1<-rbind(2:3,c(4,5))

> m1<-rbind(m1,c(7,8))

>m2<-matrix(1:6,nrow=2)   #ncol=2

>ncol(m2)   #nrow(m2)

>m3<-m2[ ,2:3]

> m2[ ,2:3]<-cbind(c(8,9),c(10,11))


>m1*t(m2)

>2*m2  #recycling

>m1 %*% m2

>sum(m1)  #matrices are special cases of vectors
```

---

ifelse(boolean vectorexpression1, vectorexpression2, vectorexpression3) – each element of the result will be the corresponding element in vectorexpression2 or vectorexpression3, depending on whether the corresponding element in vectorexpression1 is TRUE or FALSE.

```
>ifelse(m2 %% 3==1,0,m2)


>x<-c(4,2,6)

>sort(x)

>sort(x,decreasing=TRUE)
```

The R list type

```
>g<-list(x=3:6,s= "hello")
>g$x   #g[[1]]

>g$s  #g[[2]]

> for (i in 1:length(g)) print(g[[i]])
```

An S3 object is simply a list, with a class name added as an attribute:

```
>j<-list(name="Joe", salary=3200,union=T)
>class(j)<- "employee"
>m<-list(name="Joe", salary=3200,union=F)
>class(m)<- "employee"

#print() is a generic function
print.employee<-function(w){
  cat(w$name,"\n")
  cat("salary",w$salary,"\n")
  cat("union member",w$union,"\n")
}

>print(j)  #or j

>rm(print.employee)   #remove function
>print(j)  #or j
```

Handling errors

```
for (p in 1:2) {
  for (q in 0:2) {
    tryCatch({
      if (q==0)
        stop("Something has occurred!",call.=TRUE) #call.=FALSE
      else
          w<-p/q
          print(w)
    }, error = function(e) {message("An error occurred:\n", e)
    })
}}
```

Data Frames

A data frame is similar to a matrix, except that it can mix data of different modes. One column may consist of integers, another of characters and so on. All columns must have the same length and within a column all elements must be of the same mode.

```
>?airquality

>names(airquality)

>head(airquality)   #tail(airquality,n=3L)

>airquality[5,3]  # airquality$Wind[5]

>nrow(airquality)  #ncol

>airquality$Celsius<-(5/9)*(airquality[,4]-32)  #add a column

>airquality<-airquality[,1:6]

>aqJune<-airquality[airquality$Month==6,]

>nrow(aqJune)

>names(aqJune)

>mean(aqJune$Temp)

>write.table(aqJune,"C:\\Users\\Desktop\\AQJune")  #write data frame to file

>aa<-read.table(header=T,"C:\\Users\\Desktop\\AQJune")  #read data frame from file
```

Variables/vectors generation

1) 1000 dice rolls

```
x=c(1,2,3,4,5,6)
p=c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
y<-sample(x,1000,replace=F,prob=p)
hist(y)
summary(y)
var(y)
```

2) Grades from the admission exam for a random sample of 100 candidates

```
x<- 1+rbinom(100,9,0.6)
hist(x)
```

```
summary(x)
var(x)
```

3) Distribution portfolio - for different parameters values

- **Binomial**

```
    rbinom(n, size, prob);  n= 500; size= 4; 5; prob= 0.2; 0.5; 0.7
```

- **Poisson**

```
    rpois(n, lambda);       n= 500; lambda= 0.5; 1; 5
```

- **Geometric**

```
    rgeom(n, prob);         n= 500; prob= 0.3; 0.7
```

- **Continuous uniform**

```
    runif(n, min, max)
```

- **Gamma**

```
    rgamma(n, shape, rate= 1, scale= 1/rate);    n= 500; (shape= 2; scale= 2);
                                                 (shape= 2; scale= 0.2);
                                                 (shape= 1; scale= 2);
                                                 (shape= 1; scale= 0.2);
                                                 (shape= 0.5; scale=0.2)

    x<-rgamma(1000,2,scale=3)

    hist(x,freq=F)

    y<-dgamma(x,2,scale=3)

    curve(dgamma(x,2,scale=3),min(x),max(x),add=T,col="red")
```

- **Normal**

```
    rnorm(n, mean, sd);     n= 100;1000;10000; mean 3;5; sd=0.7;2;9
```

- **CHI square (central)**

```
    rchisq(n, df, ncp= 0);  n= 500; df= 2;5;10
```

- **Multinomial**

```
        rmultinom(n, size, prob)

X=rmultinom(100, size = 12, prob=c(0.3,0.1,0.6))
a<-c(mean(X[1, ]),mean(X[2, ]),mean(X[3, ]))
cov(t(X))
```

- **Normal distribution**

```
        mvrnorm(n, mu, Sigma),
#n = the number of samples required
#mu = a vector giving the means of the variables
#Sigma = a positive-definite symmetric matrix specifying the covariance matrix of the variables.

library(MASS)
mu=c(0,2)
Sigma=matrix(c(10,3,3,2),2,2)
X<-mvrnorm(100,mu,Sigma)
X
plot(X[,1],X[,2],col="red")
```