Functions

Users may define their own functions and then use them the same way as built-in functions. Functions are objects, so may be seen, saved and loaded.

```
myNewFunction = function( arguments ){
                    do something
                    return(something)
                    }

square = function(x){
                    z<-x^2
                    return(z)
                }

> square(3)
> square(1:4)
> square
```

Write your own function “sumsquare” to compute a sum of squared values. Argument: values

```
sumsquare = function(x){ sum(square(x))}
```

Write your own function to compute standard deviation. Argument: values

Square root(sum of squared differences of each data point from the mean, divided by n-1)

```
stdev=function(x){
        a=sqrt(sum(square(x-mean(x)))/(length(x)-1))
        return(a)
        }
```

Function may contain more expressions, divided by separate lines or colons, within {}. They may have many arguments. Only one object is returned as a result by the function, to ensure it is the proper one, use return()

Instead of:

```
anotherFunction = function(x){z<-x^2+3;u=sqrt(z); u}
```

better:

```
anotherFunction = function(x){
    #this is an exemplary function
    #it does…
    #arguments:
    #values:
                        z<-x^2+3
                        u=sqrt(z)
                        return(u)
                    }
```

Functions may take many arguments. Everything initialized within a function call will remain there. Caution for identical/missing names.

```
test=function(x,y){

          z<-x*3+y

          u<-max(c(x,y,z))

          return(u)
        }

>z=123
>u=1
>x=2
>test(-3,8)
>z
>u
```

Functions may also plot, print out values etc.

```
myNewFunction = function( mean,stdev,noOfbins ){
   #draws 1000 time from a normal distribution
   #With given mean and stdev, plots a histogram
   #of those values, using a suggested number of
   bins
                  values=rnorm(1000,mean,stdev)
                  print(values)
                  hist(values,breaks=noOfbins)
                  }
```

print() is especially usefull as a simple way of debugging a function

Write your own function, which

a)Draws n times from a normal distribution, with given mean and sd

b) prints out a summary of values

c) plots a histogram of them

d) gives mean and stdev as output value.

```
testsample=function(n,mean,stdev){
       sample=rnorm(n,mean,stdev)
       print(summary(sample))
       hist(sample)
       res=c(mean(sample),sd(sample))
       return(res)
    }
```

Control flow

```
if (condition) {
        expression
        }

if (condition) {
        expression
        }
else {
        expression
    }
```

```
if (x==0) {
        x=x+0.1
        }

if (x>y) {
    print("X is bigger than y")
        }
else {
    print("X is not bigger than y"
    }
```

while (condition) {

expression

}

```
while (x>0) {

    print(x)
    x = x -  1

}
```

for (variable in vector){

expression

}

```
for (x in 1:10){

    print(x*10)

    }
```

Function computing area and perimeter of a figure, given that a figure is a circle or square. Takes name of the figure and radius/length of a side.

```
geom=function( a, figure){

        area="unknown"
        perim="unknown"

        if ( figure=="square"){
           area=a*a
           perim = 4*a
           }

        if(figure =="circle"){
           area=pi*(a^2)
           perim = 2*pi*a
           }
        res= c(area,perim)
        return(res)
    }
```

Write a function SQRT, which takes a square root of a number when it is >0 and square root of it's reciprocal when it is <0 (eg. SQRT(-4)=-2, SQRT(9)=3).

Write a function to get 1/x, which addds 0.001 to x when x==0.

Write a function which subtracts 2 from a number and prints the number out, until it is equal 0.

Write a function, which given two arguments, prints the first one the number of times specified by the second argument.

Write a function, which given a vector of words, a number and a number of repeats, combines them together and repeats:

fun(c("peach","lake","kinh"),3,2)
[1] "peach 3" "peach 3"
[1] "lake 3" "lake 3"
[1] "kinh 3" "kinh 3”

Write a function, which given a matrix, computes a mean of row means of it.