Lab 7: R Programming

In this lab we will learn basic R programming. As all of you already have completed a course on programming, I will give a quick overview to R programming.

Environments

Look at the following commands:

```
a <- 10
b <- "Lab7"
ls()
```

```
## [1] "a" "b"
```

`ls()` shows all the available variable in *Global Environment*. There are also packages environments. R follows a rule to access a requested object from the environment. First R looks for the object in global environment; and then in the next available environment. List of all available environment for the current R session can be found using `search()`.

```
search()
```

```
## [1] ".GlobalEnv"        "package:stats"     "package:graphics"
## [4] "package:grDevices" "package:utils"     "package:datasets"
## [7] "package:methods"   "Autoloads"         "package:base"
```

All objects in a particular environment can be found using `ls(package_name)`. For example:

```
ls('package:graphics')
```

```
##  [1] "abline"          "arrows"          "assocplot"       "axis"
##  [5] "Axis"            "axis.Date"       "axis.POSIXct"    "axTicks"
##  [9] "barplot"         "barplot.default" "box"             "boxplot"
## [13] "boxplot.default" "boxplot.matrix"  "bxp"             "cdplot"
## [17] "clip"            "close.screen"    "co.intervals"    "contour"
## [21] "contour.default" "coplot"          "curve"           "dotchart"
## [25] "erase.screen"    "filled.contour"  "fourfoldplot"    "frame"
## [29] "grconvertX"      "grconvertY"      "grid"            "hist"
## [33] "hist.default"    "identify"        "image"           "image.default"
## [37] "layout"          "layout.show"     "lcm"             "legend"
## [41] "lines"           "lines.default"   "locator"         "matlines"
## [45] "matplot"         "matpoints"       "mosaicplot"      "mtext"
## [49] "pairs"           "pairs.default"   "panel.smooth"    "par"
## [53] "persp"           "pie"             "plot"            "plot.default"
## [57] "plot.design"     "plot.function"   "plot.new"        "plot.window"
## [61] "plot.xy"         "points"          "points.default"  "polygon"
## [65] "polypath"        "rasterImage"     "rect"            "rug"
## [69] "screen"          "segments"        "smoothScatter"   "spineplot"
## [73] "split.screen"    "stars"           "stem"            "strheight"
## [77] "stripchart"      "strwidth"        "sunflowerplot"   "symbols"
## [81] "text"            "text.default"    "title"           "xinch"
## [85] "xspline"         "xyinch"          "yinch"
```

If you load an external package, it can be seen in available environment:

```
library("ggplot2")
search()
```

```
##  [1] ".GlobalEnv"        "package:ggplot2"   "package:stats"
##  [4] "package:graphics"  "package:grDevices" "package:utils"
##  [7] "package:datasets"  "package:methods"   "Autoloads"
## [10] "package:base"
```

You can also see the source environment of an object:

```
environment(points)
```

```
## <environment: namespace:graphics>
```

```
environment(ggplot)
```

```
## <environment: namespace:ggplot2>
```

Conditions and loops

No need of any introduction to this as you have already seen looping in c programming. We will only see the format.

if else stateement

- if stand alone

```
if (condition) {
  statements
}
```

- if with else

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

Let us see some examples:

```
a<-4
if (3<4){
  a <- 5
}
a
```

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

```
a<-4
if (1==0){
  a <- 5
}
a
```

```
## [1] 4
```

```
value <- 4.5
if (value^2<1) {
```

```
  value <- 1
} else {
  value <- value+1
}
value
```

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

What if `value` is a vector? Let us examine such cases:

```
value <- c(0.5,4.5)
if (value^2<1) {
  value <- 1
} else {
  value <- value+1
}
```

```
## Warning in if (value^2 < 1) {: the condition has length > 1 and only the first
## element will be used
```

```
value
```

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

We can use `ifelse` function to this case. The format is:

```
ifelse(test=conditions,yes=statement,no=statement)
```

```
value <- c(0.5,4.5)
value <- ifelse (test=value^2<1, yes=1, no=value+1)
value
```

```
## [1] 1.0 5.5
```

for loop

```
for(loopindex in loopvector){
  statement
}
```

```
for(myitem in c(-1,4,3)){
    cat("loop starts here\n")
    cat("myitem value is",myitem,"\n")
    cat("loop ends here\n\n")
}
```

```
## loop starts here
## myitem value is -1
## loop ends here
##
## loop starts here
## myitem value is 4
## loop ends here
##
## loop starts here
## myitem value is 3
## loop ends here
```

Problem 1. Find the sum $51^4 + 52^4 + \ldots 100^4$ using for loop in R.

```r
sum <- 0 # Note that 0 is the additive identity.
for(i in 51:100){
  sum <- sum+i^4
}
cat("The sum is",sum,".")
```

```
## The sum is 1984666665 .
```

Problem 2. Find first 100 numbers of the Recaman's sequence. Recaman's sequence is defined as following: $x_0 = 0$ and for $n > 0$,

$$x_n = \begin{cases} x_{n-1} - n & \text{if } x_{n-1} - n >= 0, \text{ and } x_k \neq x_{n-1} - n \ \forall \ k < n \\ x_{n-1} + n & otherwise \end{cases}$$

```r
xseq <- 0
for (i in 1:99){
  if (all(xseq[1:i]!=xseq[i]-i) & xseq[i]-i>=0){
    xseq[i+1]=xseq[i]-i
  } else {
    xseq[i+1]=xseq[i]+i
  }
}
cat("required numbers are:\n")
```

```
## required numbers are:
```

```r
xseq
```

```
##   [1]   0   1   3   6   2   7  13  20  12  21  11  22  10  23   9  24   8  25
##  [19]  43  62  42  63  41  18  42  17  43  16  44  15  45  14  46  79 113  78
##  [37] 114  77  39  78  38  79  37  80  36  81  35  82  34  83  33  84  32  85
##  [55]  31  86  30  87  29  88  28  89  27  90  26  91 157 224 156 225 155 226
##  [73] 154 227 153 228 152  75 153  74 154  73 155  72 156  71 157  70 158  69
##  [91] 159  68 160  67 161  66 162  65 163  64
```

Problem 3. (exercise) Find the first 100 Fibonacci numbers using for loop in R.

Problem 4. (exercise) Write a R program to find the roots of a quadratic equation.

Problem 5. (exercise) Write a R program to find the following sum using for loop

$$\sum_{k=1}^{1000} \frac{1}{k}.$$

}

while loop

```r
while(loopcondition){
  staements
}
```

Problem 6. Find the following sum:

$$\sum_{k=1}^{N} \frac{1}{3^{k-1}},$$

where N is the smallest natural number such that the N^{th} term of the series is less than 10^{-6}.

```r
k <- 1
sum <- 0
nth.term <- 1
while(nth.term>1e-6){
   k <- k+1
  nth.term <- 1/(3^(k-1))
  sum <- sum+nth.term
}
cat("The value of the sum is:",sum,"\n")
```

```
## The value of the sum is: 0.4999997
```

```r
cat("N=", k,"\n")
```

```
## N= 14
```

```r
cat("The value of the N-th term :", nth.term)
```

```
## The value of the N-th term : 6.272255e-07
```

Defining functions

```r
functionname <- function(arg1,arg2,arg3,...){
  statements
  return(returnobject)
}
```

Let us write a simple one:

```r
func.sum <- function(a,b){
  s <- a+b
  return(s)
}
func.sum(1,2)
```

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