Lab sheet 2: R Programming

More on character variables

```r
hello <- "Hello World!"
nchar(hello)
```

```
## [1] 12
```

```r
a <- "0.1" # now try: a <- a+1
"alpha" == " alpha"
```

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

```r
c("alpha","beta","gamma")=="beta"
```

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

```r
"alpha"<="beta"
```

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

```r
str1=c("boring", "class", "this", "is")
str1
```

```
## [1] "boring" "class"  "this"   "is"
```

```r
cat(str1[3],str1[2],str1[4],"really", str1[1])
```

```
## this class is really boring
```

```r
paste(str1[3],str1[2],str1[4],"really", str1[1],sep="-")
```

```
## [1] "this-class-is-really-boring"
```

You can not use single backslash (/) and double inverted comma (“) inside string. The following can be done:

```r
cat("I really want a backslash: \\\nand a double quote: \"")
```

```
## I really want a backslash: \
## and a double quote: "
```

```r
str2 <- "I am a beginner in R!"
substr(x=str2,start=8,stop=15)
```

```
## [1] "beginner"
```

```r
substr(x=str2,start=8,stop=15) <-"PRO"
str2
```

```
## [1] "I am a PROinner in R!"
```

```r
# consider a string "Hello dear"
# replace the character 'e' in "Hello dear" with "E"
print(gsub("e", "E", "Hello dear") )
```

```
## [1] "HEllo dEar"
```

```r
# use parse to convert character variable to expression

x <- "sqrt(2)"
eval(x)
```

```
## [1] "sqrt(2)"
```

```r
xparse <- parse(text = x)
eval(xparse)
```

```
## [1] 1.414214
```

```r
# Upper case.
string1 <- toupper("I love Hyderabad")
string1
```

```
## [1] "I LOVE HYDERABAD"
```

```r
# Lower case.
string2 <- tolower("I love Hyderabad")
string2
```

```
## [1] "i love hyderabad"
```

left justifying

```r
x <- format("MU", width = 16, justify = "l")
print(x)
```

```
## [1] "MU              "
```

centering

```r
x <- format("MU", width = 16, justify = "c")
print(x)
```

```
## [1] "       MU       "
```

The stringr package

```r
# load the stringr package to replace a string
library("stringr")
str3 <- "I am a beginner in R!"
print(str_replace_all( str3,"beginner", "PRO") )
```

```
## [1] "I am a PRO in R!"
```

```r
x <- c("this", "is", "our", "third", "lab", "in", "this", "course")
str_length(x)
```

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

```r
str_c(x, collapse = ", ")
```

```
## [1] "this, is, our, third, lab, in, this, course"
```

```r
str_c(x, collapse = " ")
```

```
## [1] "this is our third lab in this course"
```

```r
str_sub(x, 1, 2)
```

```
## [1] "th" "is" "ou" "th" "la" "in" "th" "co"
```

```r
str_subset(x, "[aeiou]")
```

```
## [1] "this"   "is"     "our"    "third"  "lab"    "in"
## [7] "this"   "course"
```

```r
str_count(x, "[aeiou]")
```

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

```r
str_replace(x, "[aeiou]", "?")
```

```
## [1] "th?s"   "?s"     "?ur"    "th?rd"  "l?b"    "?n"
## [7] "th?s"   "c?urse"
```

The following two functions are widely used.

```r
str_extract(string, pattern)

str_extract_all(string, pattern, simplify = FALSE)
```

To know more about pattern see this Link. Some examples are:

```r
book_list <- c("Calculus", "Linear Algebra II", "Numerical Methods v2", "Real Analysis")
str_extract(book_list, "\\d")
```

```
## [1] NA  NA  "2" NA
```

```r
str_extract(book_list, "[[a-z][A-Z]]+")
```

```
## [1] "Calculus"  "Linear"    "Numerical" "Real"
```

```r
str_extract(book_list, "[a-z]{1,4}")
```

```
## [1] "alcu" "inea" "umer" "eal"
```

```r
str_extract(book_list, "\\b[a-z]{1,4}\\b")
```

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

```r
# Extract all matches
str_extract_all(book_list, "[[A-Z][a-z]]+")
```

```
## [[1]]
## [1] "Calculus"
##
## [[2]]
## [1] "Linear"  "Algebra" "II"
##
## [[3]]
## [1] "Numerical" "Methods"   "v"
##
## [[4]]
## [1] "Real"     "Analysis"
```

```
str_extract_all(book_list, "\\d")
```

```
## [[1]]
## character(0)
##
## [[2]]
## character(0)
##
## [[3]]
## [1] "2"
##
## [[4]]
## character(0)
```

```
str_extract_all(str_c(book_list, collapse = " "),"[[A-Z][a-z]]+")
```

```
## [[1]]
## [1] "Calculus"  "Linear"    "Algebra"   "II"        "Numerical"
## [6] "Methods"   "v"         "Real"      "Analysis"
```

```
# Rounded off.
x <- format(3.14159265358979323846, digits = 5)
x
```

```
## [1] "3.1416"
```

```
# in scientific notation format.
x <- format(3.14159265358979323846, scientific = TRUE)
print(x)
```

```
## [1] "3.141593e+00"
```

```
# Show at least 5 digits after decimal.
x <- format(3.14, nsmall = 5)
print(x)
```

```
## [1] "3.14000"
```

```
x <- format(007, width = 16)
print(x)
```

```
## [1] "               7"
```

Factors

```
first.name <- c("Priya","Aarav","Darpan","Bandita", "Anika",
                "Chaitanya","Shruti","Chandran", "Darsh","Evanshi")
gender.num <- c(1,0,0,1,1,0,1,0,0,1)
gender.str <- c("female","male","male","female","female","male",
             "female","male","male","female")
first.name[gender.num==0]
```

```
## [1] "Aarav"     "Darpan"    "Chaitanya" "Chandran"  "Darsh"
```

```
first.name[gender.str=="female"]
```

```
## [1] "Priya"   "Bandita" "Anika"   "Shruti"  "Evanshi"
```

```r
gender.num.fac <- factor(x=gender.num)
gender.num.fac
```

```
##  [1] 1 0 0 1 1 0 1 0 0 1
## Levels: 0 1
```

```r
gender.str.fac <- factor(x=gender.str)
gender.str.fac
```

```
##  [1] female male   male   female female male   female male  
##  [9] male   female
## Levels: female male
```

```r
levels(x=gender.str.fac)
```

```
## [1] "female" "male"
```

```r
mob.str=c("Feb","Jul","Jun","Sep","Dec","Aug","Jul","Mar","Jul","Mar")
mob.str.fac=factor(x=mob.str)
levels(mob.str.fac)
```

```
## [1] "Aug" "Dec" "Feb" "Jul" "Jun" "Mar" "Sep"
```

```r
ms <- c("Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec")

mob.fac <- factor(x=mob.str,levels=ms,ordered=TRUE)
mob.fac
```

```
##  [1] Feb Jul Jun Sep Dec Aug Jul Mar Jul Mar
## 12 Levels: Jan < Feb < Mar < Apr < May < Jun < Jul < ... < Dec
```

```r
mob.fac[2]<mob.fac[3]
```

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

```r
Y <- c(0.53,5.4,1.5,3.33,0.45,0.01,2,4.2,1.99,1.01)
br <- c(0,2,4,6)
cut(x=Y,breaks=br)
```

```
##  [1] (0,2] (4,6] (0,2] (2,4] (0,2] (0,2] (0,2] (4,6] (0,2] (0,2]
## Levels: (0,2] (2,4] (4,6]
```

```r
cut(x=Y,breaks=br,right=F)
```

```
##  [1] [0,2) [4,6) [0,2) [2,4) [0,2) [0,2) [2,4) [4,6) [0,2) [0,2)
## Levels: [0,2) [2,4) [4,6)
```

```r
cut(x=Y,breaks=br,right=F,include.lowest=T)
```

```
##  [1] [0,2) [4,6] [0,2) [2,4) [0,2) [0,2) [2,4) [4,6] [0,2) [0,2)
## Levels: [0,2) [2,4) [4,6]
```

```r
lab <- c("Small","Medium","Large")
cut(x=Y,breaks=br,right=F,include.lowest=T,labels=lab)
```

```
##  [1] Small  Large  Small  Medium Small  Small  Medium Large 
##  [9] Small  Small 
## Levels: Small Medium Large
```

List and data frames

Vectors, matrices, and arrays can store only one type of data (Numeric, logical or character). In some cases, in fact, in most cases, we need to store all kinds of data in a tabular form. We will explore two more data structures in this lab session: lists and data frames that allow one to store multiple types of data variables at a time.

List

```
my.list <- list(matrix(data=5:9,nrow=2,ncol=2),c(F,T,T,F,T),
                "hi, I am inside a list")
```

```
## Warning in matrix(data = 5:9, nrow = 2, ncol = 2): data length
## [5] is not a sub-multiple or multiple of the number of rows [2]
```

```
my.list
```

```
## [[1]]
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
##
## [[2]]
## [1] FALSE  TRUE  TRUE FALSE  TRUE
##
## [[3]]
## [1] "hi, I am inside a list"
```

```
length(x=my.list)
```

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

```
my.list[[3]]
```

```
## [1] "hi, I am inside a list"
```

```
my.list[[3]] <- paste(my.list[[3]],"and you too!")
my.list
```

```
## [[1]]
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
##
## [[2]]
## [1] FALSE  TRUE  TRUE FALSE  TRUE
##
## [[3]]
## [1] "hi, I am inside a list and you too!"
```

```
names(my.list) <- c("my.matrix","my.logical","my.string")
my.list$my.matrix
```

```
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
```

```
my.list1 <- list(var1=c(T,F,T,T),var2="Do you want to be a part of the Math Club?",
            var3=my.list$my.matrix)
```

```
names(my.list1)
```

```
## [1] "var1" "var2" "var3"
```

```
my.list1$inner.list <- list(my.vector=c(T,F,F,T),
                            my.string="Contact/Meet Faculty Incharge")
my.list1
```

```
## $var1
## [1]  TRUE FALSE  TRUE  TRUE
##
## $var2
## [1] "Do you want to be a part of the Math Club?"
##
## $var3
##      [,1] [,2]
## [1,]    5    7
## [2,]    6    8
##
## $inner.list
## $inner.list$my.vector
## [1]  TRUE FALSE FALSE  TRUE
##
## $inner.list$my.string
## [1] "Contact/Meet Faculty Incharge"
```

```
my.list1$inner.list$my.string
```

```
## [1] "Contact/Meet Faculty Incharge"
```

Data Frames

```
my_data <- data.frame(person=c("Ramesh","Priyanka","Vinodh","Ravi","Palak"),
age=c(22,19,17,19,20),
gender=factor(c("M","F","M","M","F")),
stringsAsFactors=FALSE
)
my_data
```

```
##     person age gender
## 1   Ramesh  22      M
## 2 Priyanka  19      F
## 3   Vinodh  17      M
## 4     Ravi  19      M
## 5    Palak  20      F
```

```
my_data[1,3]
```

```
## [1] M
## Levels: F M
```

```
my_data[2:4,1]
```

```
## [1] "Priyanka" "Vinodh"   "Ravi"
```

```
my_data$age
```

```
## [1] 22 19 17 19 20
```

```
dim(my_data)
```

```
## [1] 5 3
```

```
nrow(my_data)
```

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

```
ncol(my_data)
```

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

```
my_data$person
```

```
## [1] "Ramesh"   "Priyanka" "Vinodh"   "Ravi"     "Palak"
```

```
newrecord <- data.frame(person="Akansha",age=17,
gender=factor("F",levels=levels(my_data$gender)))
my_data <- rbind(my_data,newrecord)
my_data
```

```
##     person age gender
## 1   Ramesh  22      M
## 2 Priyanka  19      F
## 3   Vinodh  17      M
## 4     Ravi  19      M
## 5    Palak  20      F
## 6  Akansha  17      F
```

```
stream <- c("CM","CM","ME","ME","EEE","CM")
stream <- factor(x=stream,levels=c("CM","ME","EEE"))
my_data <- cbind(my_data,stream)
my_data
```

```
##     person age gender stream
## 1   Ramesh  22      M     CM
## 2 Priyanka  19      F     CM
## 3   Vinodh  17      M     ME
## 4     Ravi  19      M     ME
## 5    Palak  20      F    EEE
## 6  Akansha  17      F     CM
```

```
my_data[my_data$stream=="CM",]
```

```
##     person age gender stream
## 1   Ramesh  22      M     CM
## 2 Priyanka  19      F     CM
## 6  Akansha  17      F     CM
```

```
my_data[my_data$age>18,]
```

```
##     person age gender stream
## 1   Ramesh  22      M     CM
## 2 Priyanka  19      F     CM
## 4     Ravi  19      M     ME
## 5    Palak  20      F    EEE
```

```
sorted_data <-my_data[order(my_data$person,decreasing = FALSE),]
sorted_data
```

```
##     person age gender stream
## 6  Akansha  17      F     CM
## 5    Palak  20      F    EEE
## 2 Priyanka  19      F     CM
## 1   Ramesh  22      M     CM
## 4     Ravi  19      M     ME
## 3   Vinodh  17      M     ME
```

Special values

```
big_num <- 90000^100
big_num
```

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

```
-2*Inf
```

```
## [1] -Inf
```

```
1/Inf
```

```
## [1] 0
```

```
1+Inf
```

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

```
4/0
```

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

```
Inf/0
```

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

```
Inf+Inf
```

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

```
Inf-Inf
```

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

```
0/0
```

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

```
Inf/Inf
```

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

```
is.nan(2+6*(4-4)/0)
```

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

```
spcl <- c(NaN,54.3,-2,NaN,90094.123,-Inf,55)
is.infinite(spcl)
```

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

```
1>NaN
```

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

```
a <- c(0,5,NA,9)
a
```

```
## [1]  0  5 NA  9
```

```
a[6]
```

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

```
c(0,5,NULL,9)
```

```
## [1] 0 5 9
```

```
c(NA,NA,NA)
```

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

```
c(NULL,NULL,NULL)
```

```
## NULL
```

is and as dot functions

```
is.numeric(0.1)
```

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

```
is.character(0.1)
```

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

```
is.character("0.1")
```

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

```
as.numeric("0.1")+1
```

```
## [1] 1.1
```

```
is.integer(8.7)
```

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

```
as.character(0.1)
```

```
## [1] "0.1"
```

```
as.logical(c("1","0","1","0","0"))
```

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

```
as.logical(as.numeric(c("1","0","1","0","0")))
```

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

Reading and writing files

To work with available real data in most cases we need to read the data from a file, typically a '.csv' or '.txt' file. Before doing anything you need to know how the data are provided. Basically you need to know the following:

- Do you have a **header** for the data?
- How the columns are separated in each line? Commonly used delimiter are: semi colon(;), space or a comma(,)
- What character was used to specify the missing data?

I have the following data in a file named 'file.txt':

```
person age sex
Ramesh 22 M
Priyanka 19 F
Vinodh 17 M
Ravi 19 M
Rstudio 12 na
Palak 20 F
```

Note that I have a header to the file. Data are separated with space and `na` is used to denote missing data. Reading such file can be done easily as follows:

```
mydata <- read.table(file="file.txt",header=T, sep=" ",na.strings="na")
mydata[,1]
```

```
## [1] "Ramesh"   "Priyanka" "Vinodh"   "Ravi"     "Rstudio"
## [6] "Palak"
```

Now let us read a csv file:

```
my.csvtable <- read.csv(file="biostats.csv",sep=",",quote = "\"",header=T,stringsAsFactors=F)
my.csvtable
```

```
##    Name Sex Age Height Weight
## 1  Alex   M  41     74    170
## 2  Bert   M  42     68    166
## 3  Carl   M  32     70    155
## 4  Dave   M  39     72    167
## 5  Elly   F  30     66    124
## 6  Fran   F  33     66    115
## 7  Gwen   F  26     64    121
## 8  Hank   M  30     71    158
## 9  Ivan   M  53     72    175
## 10 Jake   M  32     69    143
## 11 Kate   F  47     69    139
## 12 Luke   M  34     72    163
## 13 Myra   F  23     62     98
## 14 Neil   M  36     75    160
## 15 Omar   M  38     70    145
## 16 Page   F  31     67    135
## 17 Quin   M  29     71    176
## 18 Ruth   F  28     65    131
```

```
class(my.csvtable)
```

```
## [1] "data.frame"
```

```
names(my.csvtable)
```

```
## [1] "Name"   "Sex"    "Age"    "Height" "Weight"
```

You can also read files from a website:

```
my.url <- "https://people.sc.fsu.edu/~jburkardt/data/csv/deniro.csv"
my.urldata <- read.csv(my.url)
```

Now let us write a data frame to a file.

```
my.csvtable <- my.csvtable[c(1,3,2,4,5)]
write.csv(x=my.csvtable,file="somenewfile.csv",
sep="-",row.names=F,quote = F,col.names = T,append = F)
```

Now look for the file `somenewfile.csv` in the current directory. Following is written inside the file:

```
Name,Age,Sex,Height,Weight
Alex,41, M,74,170
Bert,42, M,68,166
Carl,32, M,70,155
Dave,39, M,72,167
Elly,30, F,66,124
Fran,33, F,66,115
Gwen,26, F,64,121
Hank,30, M,71,158
Ivan,53, M,72,175
Jake,32, M,69,143
Kate,47, F,69,139
Luke,34, M,72,163
Myra,23, F,62,98
Neil,36, M,75,160
Omar,38, M,70,145
Page,31, F,67,135
Quin,29, M,71,176
Ruth,28, F,65,131
```