**Prática 1 - Uma breve introdução ao R**

O R é uma linguagem de programação e também um ambiente computacional, onde é possível realizar análises estatísticas e gráficas. Ele surgiu como um "dialeto" da linguagem S, desenvolvida nos Laboratórios AT&T Bell por John Chambers e outros. Entretanto, o R possui um grande diferencial em relação aos pacotes estatísticos tradicionais: é uma linguagem de código aberto (qualquer um pode escrever ou modificar programas nele) e disponível gratuitamente para o mundo todo. Devido a isto, e ao fato de ser uma linguagem extremamente flexível e uma ferramenta gráfica poderosa, possui centenas de contribuintes e se tornou a língua franca da computação estatística, especialmente no meio acadêmico. Técnicas estatísticas de ponta tendem a ser implementadas primeiro no R.

Assim, o R pode ser um poderoso aliado no desenvolvimento da sua pesquisa científica, entretanto, não é um programa muito amigável, necessitando de tempo e dedicação para seu aprendizado.

O R é um pouco diferente dos pacotes estatísticos tradicionais. O ambiente original não possui um menu de análises estatísticas e não é possível realizar as análises desejadas utilizando o mouse. Todos os comandos desejados devem ser digitados (ou copiados e colados) em uma linha de comando, identificada com o símbolo ">".

Para facilitar nossa aula prática, colocamos em vermelho os comandos que vamos utilizar hoje. Portanto, ao longo da aula, as expressões em vermelho devem ser copiadas na linha de comando do R.

Para começar, vamos ver que o R pode funcionar como uma calculadora, provavelmente, a mais sofisticada que você já teve! Então, vamos realizar algumas operações aritméticas.

```
8+4
8-4
8*4
8/4
8^4
8**4
```

12/2+4

12/(2+4)

Agora, digite suas próprias expressões.

**Utilizando as funções do R**

As funções disponíveis no R têm a forma:

função (<argumentos obrigatórios>, <argumentos opcionais>)

Alguns exemplos:

```
sqrt(16)        # raiz quadrada
abs(-21.6)      # valor absoluto
exp(19.5)       # exponencial
log(10)         # logaritmo
```

A esta altura, você já deve ter notado que qualquer coisa digitada depois de um # na linha de comando é entendida pelo R como um comentário, e ignorada.

Para entender como uma determinada função funciona, utilize o help, por exemplo:

help(sqrt)

help(abs)

ou na janela Ajuda, clique em "Ajuda Html"

**Construindo objetos**

Agora vamos criar um objeto chamado x, colocando diferentes coisas nesse objeto:

Um número:

x <- 7

x

O símbolo <- significa que o objeto à esquerda recebe os elementos à direita e pode ser lido como "recebe". No exemplo acima, o objeto x recebe o valor 7.

Uma palavra:

```
x <- "casa"
x
```

Um vetor:

```
x <- c(1,2,3,4,5)          # c é uma abreviação para "concatenar"
x
```

Um vetor de nomes:

```
 x <- c("Ana", "Marcela", "Beatriz")
 x
```

Outros vetores:

```
x <- seq(from=1, to=10, by=1)
x

x<-1:10
x

x<-rep(7, 4)   # repete o valor 7, 4 vezes
x

x<-c(rep(3,5),rep(1,4))
x

x<-rep(seq(2, 20, by=2), 2)          # repete o padrão 2 4  ... 20 duas vezes

cores<-c("amarelo","verde","azul","vermelho")
```

```
cores
```

Selecionado elementos de um vetor

Para selecionar o elemento na posição 3:

```
cores[3]
```

Para selecionar os elementos nas posições 1, 3 e 4:

```
cores[c(1,3,4)]
```

Para selecionar todos os elementos do vetor, menos o segundo:

```
cores[-2]
```

Para criar um vetor (ou um banco de dados) a partir de outros vetores

```
celsius <- 25:30
celsius
fahrenheit <- 9/5*celsius+32
fahrenheit
conversion <- data.frame(Celsius=celsius, Fahrenheit=fahrenheit)
print(conversion)
```

Para criar uma matriz:

matrix(valores, nlinhas, ncolunas, byrow = FALSE)

Por exemplo:

```
x<-seq(1:15)
y <- matrix(x, ncol=5)
y
```

Para saber a dimensão da matriz:

```
dim(y)
```

Para vizualizar o elemento da segunda linha, terceira coluna de x ...

```
y[2,3]
```

ou as três primeiras colunas da matriz ...

```
y[, 1:3]
```

Para vizualizar a segunda linha:

```
y[2,]
```

Excluindo uma coluna de uma matriz:

```
y[,-3]
```

Adicionado uma coluna a matriz...

```
z<-c(16, 17, 18)
y<-cbind(y,z)
```

... ou uma linha...

```
z<-c(0,1,2)
rbind(y,z)
```

Produzindo algumas estatísticas descritivas a partir de um objeto:

```
tempo <- c(32, 31, 28, 32, 33, 39, 31, 30, 36)
mean(tempo)
median(tempo)
var(tempo)
sd(tempo)
length(tempo)
```

Armazenando essas medidas em objetos no R para poder utilizá-las posteriormente:

```
media.tempo <- mean(tempo)
```

```
var.tempo <- var(tempo)
dp.tempo <- sd(tempo)
n.tempo <- length(tempo)

media.tempo
var.tempo
dp.tempo
n.tempo
```

**Utilizando o *help* do R**

Antes de tentar procedimentos mais sofisticados, convém investir um pouco de tempo explorando o *help* do R. Isto é indispensável, se você se tornar um usuário do R. A boa notícia é que o esforço gasto com essa tarefa renderá bons lucros a médio prazo!

Descubra o que faz a função *quantile:*

Para exibir a documentação para a função *quantile*:

```
help(quantile)
```

Listando todas as funções que têm a palavra *quantile* em seu título:

```
apropos("quantile ")
```

Listando todas as funções cujos nomes incluem a palavra *quantile* em seu título ou como em seu conteúdo:

```
help.search("quantile")
```

Para ver exemplos de usos possíveis para a função *quantile*:

```
example(quantile)
```

Outras funções interessantes:

Para calcular áreas sobre as curvas Normal, t-Student, Qui-quadrado e F.

```
pnorm(1)                    # P(Z <= 1)
pnorm(-1)                   # P(Z <= -1)
pnorm(1) - pnorm(-1)        # P(-1<= Z <= 1)

pt(1,8)                     # P(t(8) <= 1)
pt(-1,8)                    # P(t(8) <=  -1)
pt(1,8) - pt(-1,8)          # P(-1<= t(8) <= 1)

pchisq(4, 10)               # P(χ2 (8) <= 4)
pchisq(2, 10)               # P(χ2 (8) <= 2)
pchisq(4, 10) - pchisq(2, 10)   # P(4 <= χ2 (8) <= 2)

pf(4, 9, 12)                # P(F (9, 12) <= 4)
pf(2, 9, 12)                # P(F (9, 12) <= 2)
pf(4, 9, 12) - pf(2, 9, 12) # P(2 <= F (9, 12) <= 4)
```

Para determinar o valor nas distribuições Normal, t-Student, Qui-quadrado e F que deixa uma determinada área à sua esquerda:

```
qnorm(0.975)                # z tal que P(Z<=z) = 0.975
qnorm(0.50)                 # z tal que P(Z<=z) = 0.50
qnorm(0.025)                # z tal que P(Z<=z) = 0.025

qt(0.975, 8)                # t tal que P(T<=t) = 0.975
qt(0.50, 8)                 # t tal que P(T<=t) = 0.50
qt(0.025, 8)                # t tal que P(T<=t) = 0.025

qchisq(0.975, 10)           # c tal que P(χ2 <= c) = 0.975
qchisq(0.50, 10)            # c tal que P(χ2 <= c) = 0.50
```

```
qchisq(0.025, 10)          # c tal que P(χ2 <= c) = 0.025

qf(0.975, 9, 12)           # f tal que P(F (9, 12) <= f) = 0.975
qf(0.50, 9, 12)            # f tal que P(F (9, 12) <= f) = 0.50
qf(0.025, 9, 12)           # f tal que P(F (9, 12) <= f) = 0.025
```

Para construir um intervalo de confiança para a variável tempo, do tipo $\overline{X} \pm t_{\gamma} \frac{S}{\sqrt{n}}$ :

```
ic.inf <- media.tempo - qt(0.975,8)*dp.tempo/sqrt(n.tempo)
ic.sup <- media.tempo + qt(0.975,8)*dp.tempo/sqrt(n.tempo)
ic.inf; ic.sup
```

Carregue a biblioteca epiDisplay e descubra o que fazem os comandos abaixo:

```
library(epiDisplay)

ci.numeric(tempo, alpha=0.05)

ci.numeric(media.tempo, sds= dp.tempo, n=n.tempo, alpha=0.05)
```