Introdução ao software R

O R foi criado originalmente por Ross Ihaka e por Robert Gentleman na universidade de Auckland, Nova Zelândia, e foi desenvolvido por um esforço colaborativo de pessoas em vários locais do mundo. Faz parte da filosofia do Projeto GNU e está disponível como Software Livre sob os termos da ”Licença Pública Geral do GNU” da Fundação do Software Livre (*Free Software Foundation's GNU General Public License*) na forma de código fonte. Ele compila e roda sobre umalarga variedade de plataformas UNIX e sistemas similares (incluindo FreeBSB e Linux), Windows e MacOS.

O R é ao mesmo tempo uma linguagem de programação e um ambiente para computação estatística e gráfica. Trata-se de uma linguagem de programação especializada em computação com dados.

Como instalar.

Para a instalação do R basta estar conectado a internet conectar-se ao site http://cran.r-project.org , em CRAN “Comprehensive R Archive Network” mais próximo, no caso de Viçosa: http://www.termix.ufv.br/CRAN . Dar um clique duplo no ícone de acordo com o sistema operacional do seu computador, posteriormente no ícone base, e depois no arquivo executável. E seguir a rotina de instalação. Após instalado deve-se iniciar o R e clicar na barra de ferramentas em: Packages UPDATE PACKEGES FROM CRAN Para receber as versões atualizadas dos principais pacotes.

Interface do R : R Console

> Prompt de comandos

+ esperando continuação de comandos

Os comandos são separados por ponto e vírgula (“;”) ou são inseridos em nova linha. Se ao terminar uma linha, o comando não está sintaticamente completo o R mostra o símbolo “+” que é o comando de continuação do comando inicial.

Faz Diferença entre maiúsculas e minúsculas

Os comandos podem ser digitados no próprio R console, e executados linha por linha usando “enter”. Ou eles podem ser digitados e salvos no editor de texto do R, local em que também podem ser executados e salvos. **É importante sempre salvar seus comandos no script, e nunca na área de trabalho do R.**

Para isso, abrir novo script no R, salvar esse script dentro de uma pasta e mudar o diretório do R.

Atribuição de valores

```
x <- 10            #x é a variável que recebe o valor 10;
0.56 -> x          #x é a variável que recebe o valor 0.56;
x = -8             #x é a variável que recebe o valor -8;
assign("x", 2i)    #x é a variável que recebe o imaginário 2i;
```

Tipos de dados

```
 #Numérico
 valor <- 605
 valor
[1] 605
>>
#Caracteres
 string <- "Olá, mundo!"
 string
[1] "Olá, mundo!"
>>
#Lógicos
 2 < 6
[1] TRUE
>>
#Números complexos
 nc <- 2 + 3i
 nc
[1] 2+3i
```

```
> mode(valor)
[1] "numeric"
> length(valor)
[1] 1
> mode(string)
[1] "character"
> length(string)
[1] 1
> mode(2<4)
[1] "logical"
> length(2<4)
[1] 1
> mode(nc)
[1] "complex"
> length(nc)
[1] 1
> mode(sin)
[1] "function"
```

Comandos auxiliares

Função	**Descrição**
ls() ou *objects*()	lista curta de variáveis definidas
ls.str()	lista detalhada de variáveis definidas
str(*x*)	ver informações detalhadas de *x*
ls.str(*ab*)	ver informações detalhadas sobre todas as variáveis com "*ab*" em seu nome
rm(*x*)	deletar variável *x*
rm(*x, y*)	deletar as variáveis *x* e *y*
rm(*list* = *ls*())	deletar todas as variáveis (limpar a workspace)
class(*x*)	ver que tipo de objeto é *x*
ctrl + *L*	no teclado, pressione "ctrl+L" para limpar a tela da console (mas não remove as variáveis da memória!)

Símbolo	**Descrição**
<	Menor

<=	Menor ou igual
>	Maior
>=	Maior ou igual
==	Igual (comparação)
! =	Diferente
&	AND
|	OR
!	NOT

Funções matemáticas simples

Função	**Descrição**
abs(*x*)	valor absoluto de *x*
log(*x, b*)	logaritmo de *x* com base *b*
log(*x*)	logaritmo natural de *x*
*log*10(*x*)	logaritmo de *x* com base 10
exp(*x*)	exponencial elevado a *x*
sin(*x*)	seno de *x*
cos(*x*)	cosseno de *x*
tan(*x*)	tangente de *x*
round(*x, digits* = *n*)	arredonda *x* com *n* decimais
ceiling(*x*)	arredondamento de *x* para o maior valor
floor(*x*)	arredondamento de *x* para o menor valor
length(*x*)	número de elementos do vetor *x*
sum(*x*)	soma dos elementos do vetor *x*
prod(*x*)	produto dos elementos do vetor *x*
max(*x*)	seleciona o maior elemento do vetor *x*
min(*x*)	seleciona o menor elemento do vetor *x*
range(*x*)	retorna o menor e o maior elemento do vetor *x*

```
#Exemplo:
> x = 30; y = 60
> (sin(x))^2 + (cos(x))^2
[1] 1
> round(tan(2*y),digits =3)
```

[1] 0.713
*floor(tan(2*y))*
[1] 0
*ceiling(tan(2*y))*
[1] 1

Números complexos
sqrt(-17)
[1] NaN
Warning message:In sqrt(-17) : NaNs produzidos
sqrt(-17+0i)
[1] 0+4.123106i

Vetores e matrizes
#Exemplos:
vec <- c(1, 4, 10.5, 54.48, 9, 10)
vec
[1] 1.00 4.00 10.50 54.48 9.00 10.00
vec2 <- (1:10)
> vec2
[1] 1 2 3 4 5 6 7 8 9 10
vec3 <- c((1:3),(3:1))
vec3
[1] 1 2 3 3 2 1
vec4 <- c(0, vec3, 0)
vec4
[1] 0 1 2 3 3 2 1 0
#vetor de "a" até "z"
seq(from= a, to= z)
#vetor de "a" até "z" com passo "n"
seq(from= a, to= z, by= n)
#vetor de "a" até "z" com "n" elementos
seq(from= a, to= z, length.out= n)
#Exemplos:
> seq(from=1, to=5)
[1] 1 2 3 4 5
> seq(from=1, to=5, by=0.5)

[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0

#MATRIZ sintaxe:
x <- matrix(data = dados, nrow = m, ncol = n, byrow = Q)
onde "m" é o número de linhas, "n" é o número de colunas e
se Q = 1 #ativa disposição por linhas
se Q = 0 #mantém disposição por colunas

#Exemplo:
A <- matrix(c(1:10),2,5,1) ## disposição por linhas
A
[,1] [,2] [,3] [,4] [,5]
[1,] 1 2 3 4 5
[2,] 6 7 8 9 10
disposição por colunas:
A <- matrix(c(1:10),2,5)
Ou A <- matrix(c(1:10),2,5,0)
Para selecionar um elemento de uma matriz utilizamos a indexação por colchetes
#Exemplos:
A[2,4]
[1] 9
A[2,4] - x[1,5]
[1] 0
A[2,]
[1] 6 7 8 9 10
> A[,2:4]
[,1] [,2] [,3]
[1,] 2 3 4
[2,] 7 8 9

Operações e funções com Matrizes

Função	*Descrição*
A _ B	*produto elemento a elemento de A e B*
A% _ %	*B produto matricial de A por B*
B = aperm(A)	*matriz transposta: B = At*
B = t(A)	*matriz transposta: B = At*

B = solve(A) matriz inversa: B = A−1
x = solve(A, b) resolve o sistema linear Ax = b
det(A) retorna o determinante de A
diag(v) retorna uma matriz diagonal onde o vetor v é a diagonal
diag(A) retorna um vetor que é a diagonal da matriz A
diag(n) sendo n um inteiro, retorna uma matriz identidade de ordem n
eigen(A) retorna os autovalores e autovetores de A
eigen(A)$ values retorna os autovalores de A
eigen(A)$ vectors retorna os autovetores de A

#Exemplos:
> B = t(A)
> B
[,1] [,2]
[1,] 1 6
[2,] 2 7
[3,] 3 8
[4,] 4 9
[5,] 5 10
> b= array(c(0,1,5),dim=c(3,1));
> C= matrix(c(c(1,1,0),c(0,1,4),c(0:2)),3,3,1);
> y = solve(C,b)
> y
[,1]
[1,] -9
[2,] 9
[3,] -2
> Cinv = solve(C)
> Cinv%%b*
[,1]
[1,] -9
[2,] 9
[3,] -2

#Exemplos:
D = t(A)
D
*A*B*
A%%B*

b= array(c(0,1,5),dim=c(3,1));
A= matrix(c(c(1,1,0),c(0,1,4),c(0:2)),3,3,1);
*x = solve(A,b) ## resolve o sistema A*x = b*
x
Cinv = solve(A) ## conferindo
Cinv%%b*

Os arrays são a generalização das matrizes para mais de duas dimensões. Um exemplo é o objeto Titanic,
com as seguintes dimensões:
> dim(Titanic)
[1] 4 2 2 2
> dimnames(Titanic)

Função "data.frame()"

Com a função data.frame reunimos vetores de mesmo comprimento em um só objeto:

funcionarios <-
data.frame(nome = c("João","Maria","José"),sexo = c("M", "F", "M"),salario = c(1000, 1200, 1300))

Exercícios

1) Operações matemáticas e atribuições simples
a) Atribua a um objeto r a raiz quadrada do número 10345.
b) Aproxime o valor de r:
b1) com duas casas decimais,
b2) para o inteiro inferior a r e;
b3) para o inteiro posterior a r.

2) a) Crie um vetor v1 pa
ra receber 6 números que você gostaria de apostar na mega sena.
b) Obtenha o somatório de todos os elementos desse vetor
c) Obtenha o produto de todos os elementos desse vetor.
d) Multiplique o vetor pelo número 10
e) Crie um segundo vetor que recebe os números das posições ímpares de v1 (atribuir de v1)

3)a) Importar para o R os dados Mit.
b) Atribua os dados a uma matriz
c)Obtenha uma outra matriz A com 5 linhas dos dados e 4 colunas.
d) Crie uma matriz B identidade de ordem compatível e multiplique pela matriz A
e) Crie uma matriz D diagonal de mesma ordem de B colocando números inteiros consecutivos a partir de 1
na diagonal. Multiplique D por A.
f) Obtenha a partir de A uma matriz com 2 linhas e 3 colunas.

Faça um gráfico de dispersão da primeira pela segunda coluna dos dados originais.

Entrada de Arquivos Externos

Entrada de Arquivos Externos

Depois de salvar o arquivo no diretório especificado, carregue o arquivo no console
do R.
> setwd('C:\\Rdados') #diretório C:\Rdados
 # Conferindo o diretório atualizado através do comando:
> getwd()
> dir() # verifica a presença de arquivos no diretório de trabalho
Em seguida, devemos dar o comando para que o R carregue o arquivo .csv ou .txt no console de trabalho. Para isso digite o seguinte comando:

```
dados  <-  read.table("arquivo.csv",header=T,sep=",",dec=".")
```

_ dados: é o objeto no qual os dados lidos serão reconhecidos pelo R;
_ read.table: função que lê o arquivo do tipo .csv ou txt
O parâmetro "header" nos permite indicar se o arquivo de dados (data.frame) tem ou
não o nome nas colunas (título) na primeira linha de dados. O parâmetro "sep" permite indicar o tipo de separador dos dados presentes no arquivo. Finalmente o parâmetro
"dec" permite indicar o caractere usado como separador de casas decimais dos números reais.

```
####  importando planilha csv
exemplo <- read.table("dadosfic.csv", head=T, sep=":", dec=",")
  exemplo


#####  importando planilha excel

require(xlsx)
library(xlsx)
## Loading required package: rJava
## Loading required package: xlsxjars
```

Quando o R não consegue conexão com internet para baixar os pacotes, usar a função ## setInternet2(TRUE)

setInternet2(use=TRUE)

dadosex <- read.xlsx("dadosEmpresas.xlsx", "PlanEmpresa")
print(dadosex)
salvar os dados com o nome dadosEmpresas no excel workbook e colocar
na planilha com nome PlanEmpresa

Outras sintaxes para carregar dados no console do R: "help(read.table)")

#Exemplo: importando dados de uma página na internet
dadosi <-
read.table("http://www.leg.ufpr.br/~paulojus/dados/gam01.txt")
attach(dadosi)
dim(dadosi)
Exemplo com dados
DadosEst <- read.table("estudios21.txt", h-=T)
DadosEst
Attach(DadosEst)
Mit <- edit(data.frame()) ### abre uma planilha para digitar tabela
Mit
Mit <- edit(data.frame(Mit)) ## abre novamente a planilha para editar os dados
A tabela a seguir refere-se a um motor de indução trifásico (MIT). Os dados nessa tabela são experimentais, e foram medidos o valor de corrente, a potência total e a velocidade para cada variação de tensão.
Tabela 3. Valores de corrente, potência e velocidade em função da tensão.
Tensão (V) Corrente (I) Potência (W1) Potência (W2) Velocidade (rmp)

264 6,13 920 560 1798
240 4,33 620 380 1798
220 3,40 450 260 1797
200 3,06 380 200 1797
180 2,66 280 145 1797
160 2,33 220 100 1796
140 2,06 175 65 1795
120 1,73 138 40 1794
100 1,13 100 20 1792
80 1,20 70 0 1790
60 1,03 50 0 1783
20 0,53 10 0 1770
10 0,56 10 0 0

Exportar dados do R para aarquivo txt ou csv.
write.table

>write.table(dados,file="C:\\Documents and Settings\\Convidado\\Desktop\\ dados.txt",sep=" ") <ENTER>
>write.table(dados,file="C:\\Documents and Settings\\Convidado\\Desktop\\ dados.CSV",sep=" ",dec=",") <ENTER>
>write.table(dados,file="C:\\Documents and Settings\\Convidado\\Desktop\\dados.xls") <ENTER>

Gráficos
#Exemplo:
> a <- 1:20
> b <- a^2
> plot(a,b)
> plot(a,b, type="l")

```
#Salvando gráficos automaticamente
>jpeg(file="figure.jpeg")  #figure é o nome do arquivo imagem
>plot(rnorm(10))  # gráfico que estou salvando
>dev.off()   #fecha a janela gráfica automaticamente

#Criando várias janelas gráficas
>plot(rnorm(10)) #plotando o primeiro gráfico
>windows()   #criação de uma nova janela gráfica
>plot(rnorm(20))  #plotando o segundo gráfico

#Personlalizando um gráfico
> par(mfrow=c(1,2))
> x<-1:10
> y<- c(2,5,9,6,7,8,4,1,3,10)
> x;y
[1] 1 2 3 4 5 6 7 8 9 10
[1] 2 5 9 6 7 8 4 1 3 10
> plot(x,y)
> plot (x,y, xlab="Eixo X", ylab="Eixo Y",
+ main="Personalizando um gráfico", xlim=c(0,10), ylim=c(0,10),
+ col="red", pch=22, bg="blue", tcl=0.4, las=1, cex=1.5, bty="l")

## Gráficos da estatística descritiva
> hist(dados,nclass=k,) #k é o número de classes do histograma
#Exemplo:
> rest <- c(96,96,102,102,102,104,104,108,
+ 126,126,128,128,140,156,160,160,164,170,
+ 115,121,118,142,145,145,149,112,152,144,
+ 122,121,133,134,109,108,107,148,162,96)
> par(mfrow=c(1,2))
> hist(rest,nclass=12)
> hist(rest,nclass=6)

x<-c(1.75,1.80,1.65,1.9,1.74,1.91)  ### alturas em cm
```

```
y<-c(60,72,57,90,95,72)   ### pesos em kg


f<-function(x) -46.34+67.35*x
plot(f)

plot(f,xlim=c(min(x),max(x)))

plot(f,xlim=c(min(x),max(x)),ylim=c(55,95), lwd=2)
points(x,y,pch=17)    # acrescenta pontos a um plot

plot(f,xlim=c(min(x),max(x)), ylim=c(50,100), lwd=2)
##xlim e ylim delimita o intervalo para os eixos x e y
respectivamente
points(x,y,pch=17)    # acrescenta pontos a um plot
text(1.7,90,expression(hat(y)*"=-46,34+67,35x")) # acrescenta
texto a um plot
text(1.7,87,expression(mu))
text(1.7,85,"regressão")


##   Outro modo  ##############

#####  Após digitar o comando, clicar sobre o gráfico para inserir o
texto.
text(locator(1),"regressão")


#aumentando o tamanho dos caracteres do texto

plot(f,xlim=c(min(x),max(x)), ylim=c(50,100), lwd=2)
points(x,y,pch=17)
text(1.7,90,expression(hat(y)*"=-46,34+67,35x"),cex=1.2)
## cex=1.2 aumenta o tamanho dos caractes

### mais de uma linha de texto
plot(f,xlim=c(min(x),max(x)), ylim=c(50,100), lwd=2)
```

```
points(x,y,pch=17)
text(c(1.7,1.7),c(90,85),c(expression(hat(y)*"=-46,34+67,35x"),
expression(R^2*"=19,18%")),cex=1.2)

### renomeando os eixos
plot(f,xlim=c(min(x),max(x)), ylim=c(50,100), lwd=2,
xlab="Altura",ylab="Peso",cex.lab=1.2)

points(x,y,pch=17)
text(c(1.7,1.7),c(90,85),c(expression(hat(y)*"=-46,34+67,35x"),
expression(R^2*"=19,18%")),cex=1.2)

#Gráfico de barras
 x <- c(1,2,3,4,5,6,7)
 barplot(x)
 barplot(euro,xlab="Euro conversions",col="red",
 legend.text="Valor da taxa")

materia   <-    c("matemática","geografia", "historia")
frequencia <-  c(12,4,6)
barplot(frequencia, names.arg = materia)
```

Matéria Favorita	Frequência
Matemática	12
Geografia	4
História	6
Português	7
Inglês	2
Biologia	9
Fisíca	17
Química	11
Total	68

```
#Box plot
 x = c(5,5,5,13,7,11,11,9,8,9)
```

```
 y = c(11,8,4,5,9,5,10,5,4,10)
 boxplot(x,y) #para plotar no mesmo gráfico (comparação)
 boxplot(x); boxplot(y) #para plotar em gráficos diferentes

## Usando dados do dataset R
 boxplot(count~spray,data=InsectSprays,xlab="Tipo de Spray",
 ylab="Contagem de Insetos",main="InsectSprays data",
 col="yellow")

#Gráfico de pizza
#pie(dados,opções)
 a<-c(0.12, 0.3, 0.26, 0.16, 0.04, 0.12)
 names(a)<-c("a","b","c","d","e","f")
 pie(a,col = c("red","blue","green","gray", "brown", "black"))



#Medidas de posição e dispersão
> x <- c(10, 14, 13, 15, 16, 18, 12)
> mean(x)
>g <- c(1,3,0,0,2,4,1,3,5,6)
> median(g)
> var(x)
>sd(x)
>sqrt(var(x))

#Exemplo Moda:
> y <- c(7,8,9,10,10,10,11,12)
> table(y)  ## ordena os dados e conta a frequencia
y
7 8 9 10 11 12
1 1 1 3 1 1
> subset(table(y),table(y)==max(table(y)))  calcula diretamente a
moda
10
3
```

#Quartis:
> z <- c(5,2,6,9,10,13,15)
> summary(z)
Min. 1st Qu. Median Mean 3rd Qu. Max.
2.000 5.500 9.000 8.571 11.500 15.000

Distribuições de probabilidade

IMPORTANTE 4 LETRAS

d: para calcular a densidade no ponto (probabilidade no ponto)
p : para calcular probabilidade acumulada até o ponto
q: para calcular o quantil q que tem probabilidade acumulada especificada
r: para gerar uma amostra aleatória da distribuição especificada.

Exemplo Distribuição binomial
dbinom( )
pbinom( )
qbinom( )
rbinom( )

#DISTRIBUIÇÃO BINOMIAL:
> x <- 0:6
> n <- 6
> p <- 0.5
> bino <- dbinom(x, n, p)
> bino
[1] 0.015625 0.09375 0.234375 0.312500 0.234375 0.093750 0.015625
> plot(x,bino,type="h",xlab="N_ de peças com perfeição",
+ ylab="Probabilidade",main="Distribuição binomial")

```
# Exemplo. Calcular probabilidade de x=3 com n =20 p=0.1
#Consultar help para probabilidade de menor ou igual

## DISTRIBUIÇÃO POISSON  ##
> x<-2
> lambda<-2.3
> #distribuição de Poisson com parâmetros x e lambda:
> dpois(x,lambda)

#DISTRIBUIÇÃO NORMAL: ## rnorm; qnomr; pnorm
> #Item a)
> 1-pnorm(179,170,6) #pnorm(x,média,desvio padrão)
[1] 0.0668072
> #Item b)
> qnorm(0.8, 170,6)
[1] 175.0497
> #Item c)
> curve(dnorm(x,170,6),170-3*6,170+3*6,xlab="Alturas (cm)",
+ ylab="Probabilidade de se encontrar a altura x",
+ main="Distribuição Normal")
> lines(c(179,179),c(0,0.022),col="red")
> lines(c(175.0497,175.0497),c(0,0.0465),col="blue")



#  Distribuição normal
#dnorm(x, mean = 0, sd = 1, log = FALSE)
#pnorm(q, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
#qnorm(p, mean = 0, sd = 1, lower.tail = TRUE, log.p = FALSE)
#rnorm(n, mean = 0, sd = 1)
help(rnorm)
# Exemplo: Se X~N(0,1) calcule:
# a) P(X < 0)
pnorm(0,mean=0,sd=1) ## ou pnorm(0)

# b) P(X > 0)
```

```
1-pnorm(0,0,1)
# c) P(X < -4)
pnorm(-4,0,1)
# d) P(X > 4)
1-pnorm(4,0,1)
# e) P(-1,96 < X < 1,96)

pnorm(1.96,0,1)-pnorm(-1.96,0,1)
# f) P(-2,0 < X < -1,0)
pnorm(-1,0,1)-pnorm(-2,0,1)

## Exemplo: Se X~N(75,100) (media = 75 e variancia = 100)
calcule:
# a) Determine os quartis desta distribuição
?rnorm

q1<-qnorm(0.25,75,10) # primeiro quartil
q2<-qnorm(0.50,75,10) # segundo quartil
q3<-qnorm(0.75,75,10) # terceiro quartil
# b) Determine o quantil 5% e 95%
qnorm(0.05,75,10)
qnorm(0.95,75,10)


# #########     ilustração gráfica da normal
par(mfrow=c(2,2))
##  no commando curve, x é um vetor genérico
curve(dnorm(x,0,1),from=-4, to=4,ylim=c(0,1),lwd=2) ##  gráfico
da densidade normal verdadeira
curve(dnorm(x,0,0.4),from=-4, to=4,add=T,col="blue",lwd=2)

curve(pnorm(x,0,1),from=-4, to=4,ylim=c(0,1),lwd=2)
curve(pnorm(x,0,0.4),from=-4, to=4,add=T,col="blue",lwd=2)

########      Exemplo. Fazer uma amostra de tamanho n = 1000
da distribuição normal N(75,100), fazer o
```

```
##### gráfico da densidade dessa amostra, comparar com a densidade normal de um vetor genérico x.
####   Fazer o gráfico da distribuição acumulada para essa amostra
A   <-  rnorm(10000,75,10)
plot(density(A) )
curve(dnorm(x,75,10),from=20, to=130, lwd=2)
plot(ecdf(A))


hist(y,freq=F)
curve(dnorm(x,75,10),from=40, to=100,add=T,col="blue",lwd=2)

################################
#########              Distribuição t de Student
#################
#dt(x, df, ncp, log = FALSE) # df graus de liberdade df=n-1
#pt(q, df, ncp, lower.tail = TRUE, log.p = FALSE)
#qt(p, df, ncp, lower.tail = TRUE, log.p = FALSE)
#rt(n, df, ncp)
?dt
par(mfrow=c(1,1))
curve(dnorm(x,0,1),from=-4, to=4,lwd=2)
curve(dt(x,5),from=-4, to=4,add=T,col="blue",lwd=2)
curve(dt(x,100),from=-4, to=4,add=T,col="red",lwd=2)

# Exemplo: X tem dist t de Student
# com n=10. Calcule:
#a) P(x<-1)
pt(-1,df=9)
# b) P(-1< X < 2)
pt(2,df=9)-pt(-1,df=9)

#########   Distribuição F      Sintaxe :  (  df(x, GL1,   GL2)
df
pf
rf
```

qf
help(df) df(x, df1, df2)
########## Distribuição quiquadrado Sintaxe : (dchisq(x, GL)

dchisq
pchisq
rchisq
qchisq
help(dchisq)

DISTRIBUIÇOES NO R

Distribuição/função	Função
beta	dbeta(n, shape1, shape2)
binomial	dbinom(n, size, prob)
binomial negativa	dnbinom(n, size, prob)
Cauchy	dcauchy(n, location=0, scale=1)
estatística de Wilcoxon's	dwilcox(nn, m, n, n), dsignrank(nn,n)
exponencial	dexp(n, rate=1)
Fischer-Snedecor(F)	df(n, df1, df2)
gamma	dgamma(n, shape, scale=1)
Gauss(normal)	dnorm(n, mean=0, sd=1)
geométrica	dgeom(n, prob)
hipergeométrica	dhyper(nn, m, n, k)
logística	dlogis(n, location=0, scale=1)
log-normal	dlnorm(n, meanlog=0, sdlog=1)
Poisson	dpois(n, lambda)
qui-quadrado	dchisq(n, df)
"Student" (t)	dt(n, df)
uniforme	dunif(n, min=0, max=1)
Weibull	dweibull(n, shape, scale=1)

```
#################################################################
# distribuições não definidas no r
## P(a < X < b) = integrate(f,a,b)
f<-function(x) (1/40)*((x/10)+1) # 0 < x < 20
f
curve(f,0,20)
```

```
# Calcule:
# a) P(0 < X < 20)
integrate(f,0,20)
# b) P(0 < X < 10)
integrate(f,0,10)
# c) P(5 < X < 15)
integrate(f,5,15)
# d) P(25 < X < 50) zero por definiçao
integrate(f,25,50)
curve(f,0,50)
P(25 < X < 50)=0
## e) O valor esperado e a variância da variável X.
g<- function(x) x*(1/40)*((x/10)+1)
g
Ex <- integrate(g,0,20) ### E(X)
Ex
h<- function(x) x^(2)*(1/40)*((x/10)+1)
h
Ex2 <- integrate(h,0,20)
Ex2

## variância
v  <- Ex2 - (Ex)^2
v<-166.6667-(11.6667)^2
v
##  Calculando a integral definida de uma função
f2   <-  function(x)  { 20000/x^3}    ###  define a funcao
plot(f2)  ###   faz o grafico
integrate(f2, 100, Inf)  ###  Calcula a integral da funcao no interval de 100  a infinito
f1  <-  function(x) 4*sqrt(1-x^2)
plot(f1)
integrate(f1,0,1)
f4  <-  function(x) 5*x^2*exp(-x)
integrate(f4,0,1)
plot(f4)

f3   <-  function(x) {3*x^3 + 5*x^2 - 7*x +11 }
plot(f3)
plot(f3, xlim = c(-10,10))  ###  faz o grafico alterando o intervalo x para -10 ate 10
###############   Calculando a derivada de uma função
##   função D  ou deriv
flinha   <-   D(expression(3*x^3 + 5*x^2 - 7*x +11), "x")
flinha
```

######## PREPARAÇÃO PARA ESTRUTURAS DE CONTROLE

```
y<-c(1,2,6,7,8)
x  <- c(3,2,7,5,4)
> x>y # Retorna TRUE para os maiores e FALSE para os menores
> x>=y
> x<y
> x==y # Retorna TRUE para os x que são iguais a y
> x!=y # Retorna TRUE para os x que são diferentes de y


####  A função which funciona como se fosse a pergunta: Quais?
 a<-c(2,4,6,8,10,12,14,16,18,20)


 a>10 # Retorna um vetor contendo TRUE se for maior e FALSE se for menor
 which(a>10) #"Quais valores de a são maiores que 10?" a resposta é a posição dos
valores (o sexto, o sétimo…)
 a[which(a>=14)] ## retorna os valores >= 14 e não a posição

##   Comando ifelse
###  o comando ifelse significa: se for isso, então faça aquilo, caso contrário, faça aquilo
outro.
###  uso do ifelse:
##   ifelse(aplicamos um teste, especificamos o valor caso a resposta seja verdade, e o
valor caso falso).

salarios<-c(1000, 400, 1200, 3500, 380, 3000, 855, 700, + 1500, 500) ##
ifelse(salarios<1000,"pouco","muito")   # Se o salário é menor que 1000, ####    seja
pouco, se for maior seja muito.

####   O comando for
##  O comando for é usado para fazer loopings, e funciona da seguinte maneira:
##  for(i in 1:n){comandos}


 resu<-numeric(0)
## Agora vamos usar o for para elevar i valores ao quadrado:
 for(i in 1:5){ resu[i]   <-   i^2  } # Fim do for (i)
 resu

plot(0:10,0:10, type="n")
for(i in 1:9){
     text(i,i, paste("Passo", i))
 }

 plot(0:10,0:10, type="n")
 for(i in 1:9){
       text(i,i, paste("Passo", i))
       Sys.sleep(1) ## retarda os passos em 1 segundo
 }
```

Vamos usar a função for para descobrir os 12 primeiros números da seqüência de Fibonacci (exemplo retirado e adaptado de: Braun & Murdoch 2007 pp, 48)

```
 Fibonacci<-numeric(0)
 Fibonacci[c(1,2)]<-1       # o 1° e 2° valores da seqüência devem ser = 1
 for (i in 3:12)
{
   Fibonacci[i]<-Fibonacci[i-2]+Fibonacci[i-1]
 }
 Fibonacci
```

Exercício) Modifique o código para que os valores sejam compostos pela diferença entre os dois valores imediatamente anteriores somada ao terceiro valor imediatamente anterior. Faça inicialmente com que a sequencia Fibonacci comece com 3 valores [1,1,1].

```
######   Criando um código ou escrevendo uma função
###gerar um código que escolhe números para a mega sena
 njogos<-20  # quantos jogos queremos produzir
 numeros<-matrix(NA,6,njogos)
 for(i in 1:njogos)
  {
    numeros[,i]<-sample(1:60,6)
 }
numeros

#####################  Transformar o código em função
 megasena  <-   function(njogos)
    {                        # cria a função com nome de megasena
      numeros  <-matrix(NA,6,njogos) # cria o arquivo que recebe os jogos
      for(i in 1:njogos)
        {
         numeros[,i]<-sample(1:60,6)
        }
    return(numeros)
  }
 megasena(10)
 megasena(20)
```

```
####   Uso do comando if
#Exemplo:
 x<-2
 z<-1
 if (x > 0) {
```

```
 cat("x é positivo!\n") #comando para escrever texto
 y <- z / x
 }else {
 cat("x não é positivo!\n")
 y <- z}
 y

#Exemplo:
 idade <- 60
 if (idade < 18) {
 cat("Idade menor que 18\n")
 } else if (idade < 35) {
 cat("Idade menor que 35\n")
 } else if (idade < 60) {
 cat("Idade menor que 65\n")
 } else {
 cat("Idade maior ou igual a 60. BEM-VINDO À TERCEIRA IDADE!\n")}


# Comando While Sintaxe:
##  while (condição)
##  {bloco de instruções}
##  enquanto a condição for verdadeira,  repetir o bloco de instruções do ciclo.

#  Exemplo:
#  x recebe um número aleatório de uma distribuição normal
   x <- rnorm(1)
   while (x < 1)
    {                          #enquanto x for menor que 1, faça:
       cat("x=", x, "\t")      #escreve o valor x menor que 1
       x <- rnorm(1)
       if(x>=1)
          {                          #condição para nova linha
            cat("\n")
          }
     }
```

OBSERVAÇÃO: se o primeiro número ”sorteado” pela função rnorm() for superior ou igual a 1, as instruções do ciclo não serão executadas.