Curso *Software* R

Disciplina: **Métodos Computacionais para Estatística II**
Professor: **Jony Arrais Pinto Junior**

Departamento de Estatística
Universidade Federal Fluminense

Sobre o *Software R*

- O R é uma **linguagem** para **manipular** objetos.
- Uma linguagem interativa, orientada a objetos ou funções.
- Os objetos podem ser conjunto de dados, vetores, funções, matrizes, etc.
- As manipulações podem ser cálculos, entrada e saída de dados, análises estatísticas e gráficos.

Departamento de Estatística UFF

Principais Características

- É gratuito.
- Diversas funções estatísicas disponíveis na versão básica.
- Grande variedade de pacotes/bibliotecas com funções específicas disponíveis.
- *Software open*.

R-studio: Download e Instalação

- O *RStudio* é um ambiente de desenvolvimento integrado aberto e *free* para o R.
  - Disponível no site: **www.rstudio.org**.
  - Clique no menu **download**.
  - Escolha *Download RStudio Desktop*.
  - Escolha o sistema operacional (Linux, Mac ou Windows).
  - Instale o programa a partir do arquivo salvo.

Conhecendo o *RStudio*

- O *RStudio* possui 4 janelas. A janela do canto superior esquerdo é um editor (como o Tinn-R), a do canto inferior esquerdo é o *console* do R, a do canto superior direito ficam o *workspace* e o history e a do canto inferior direito apresenta o *help*, os pacotes e os gráficos.

- Instalando e habilitando Pacotes/Bibliotecas
  - Na janela do canto inferior direito clique em *install packages*.
  - Digite o nome do pacote em *Packages*.
  - Para habilitar o pacote é necessário clicar no pacote na janela do canto inferior esquerdo ou usar o comando **require(nome do pacote)**

Departamento de Estatística UFF

Conhecendo o *RStudio*

- A janela do canto superior esquerdo é o lugar reservado para que você digite os seus comandos. Você poderá enviá-los para o console do R selecionando o texto e clicando no botão **Run**.
- Comando *help*
  - Na janela do canto inferior direito clique em *help* e digite a função dentro da caixa que possui uma lupa e aperte *enter*.
  - Pode-se utilizar também *help***(nome da função)** ou **?nome da função**

Exemplo

- Alguns comandos úteis.

Função	Descrição
setwd(" ")	muda o diretório do trabalho
getwd()	mostra o diretório do trabalho
ls()	lista o nome dos objetos criados na sessão atual
dir()	lista todos os arquivos na pasta de trabalho atual
search()	lista todos os pacotes carregados
rm()	remove o objeto entre parênteses
rm(list=ls(all=TRUE))	remone todos os objetos, limpando a memória
attach()	reconhece os objetos dentro de um *data frame*
detach()	função que desfaz o *attach*

Exemplo

- Operadores matemáticos úteis.

Função	Descrição
sqrt()	raiz quadrada
abs()	valor absoluto
exp()	exponencial
log()	logaritmo na base e
log10()	logaritmo na base 10
gamma(a)	função gama de a: $\Gamma(a) = (a-1)!$
beta(a, b)	função beta $\Gamma(a)\Gamma(b)/\Gamma(a+b)$
choose(n, k)	$n!/(k!(n-k)!)$

- Calcule: $\Gamma(5)^{1/2} - e^{-3}$.

Objetos

- **Vetores**: conjunto de elementos de uma mesma natureza.
- **Matrizes**: conjunto de elementos de uma mesma natureza organizado em linhas e colunas.
- **Array**: generaliza a ideia de matriz. Enquanto em uma matriz os elementos são organizados em duas dimensões (linhas e colunas), em um array os elementos podem ser organizados em um número arbitrário de dimensões.
- **Data frames**: similar como matrizes, porém diferentes colunas podem possuir elementos de natureza diferentes.
- **Listas**: generalizações de vetores, representa uma coleção de objetos.
- Valores faltantes e especiais:

  c(-1,0,1,NA)/0

  NA (*Not available*) e NaN (*Not a Number*)
  -Inf e Inf: menos e mais infinito

Departamento de Estatística UFF

Criando Vetores

- **c()** - concatenação. Ex: c(-1,0,1)
- **seq()** - sequência.
  seq(0, 1, length.out=11)
  seq(1, 9, by = 2)
  seq(10) # é o mesmo que 1:10
- **rep()** - replicação.
  rep(1:4, 2)
  rep(1:4, each = 2) # diferente do anterior.
  rep(1:4, c(2,2,2,2)) # mesmo que o secundo.
  rep(1:4, c(2,1,2,1))
  rep(1:4, each = 2, len = 4) # somente os 4 primeiros.
  rep(1:4, each = 2, len = 10) # 8 primeiros + os 2 primeiros repetidos.
  rep(1:4, each = 2, times = 3) # 3 replicações completas.

Criando Matrizes

- **matrix()**
  matrix(data = 1:6, nrow = 2, ncol = 3, byrow = FALSE,
  dimnames = NULL)

  matrix(1:6,2,3, byrow = TRUE,
  dimnames = list(c("L1","L2"),c("C1","C2","C3")))

- **rbind(),cbind()** - junta linhas/colunas a matrizes já existentes.

Criando Array e Data Frame

- array(1:24, c(2,4,3)) #objeto para "alimentar" a array e as dimensões
- **data.frame()**
  dados = data.frame(c(20,23,30),c("A","B","A"))
  names(dados)=c("idade","grupo")
  ou
  dados2 = data.frame(idade=c(20,23,30),grupo=c("A","B","A"))

Operacionalizando vetores, matrizes e data frames

- **length(v)** - comprimento do vetor v.
- **dim(M)** - dimensão da matriz/array M.
- **sort(v)** - ordena o v vetor em ordem crescente.
- **rank(v)** - retorna as posições que cada elemento ocupa no vetor v ordenado.
- **round(v)** - arredonda o vetor v com quantas casas decimais desejadas.
- **max(v)/min(v)** - retorna o valor máximo/mínimo de v.
- **which.max(v)/which(min(v)** - retorna a posição em que o valor máximo/mínimo se encontra.
- **v[i]** - retorna os elentos de v que se encontra na posição i.
- **v[-i]** - retorna os elentos de v com excessão daquele que se encontra na posição i.

Departamento de Estatística UFF

Operacionalizando vetores, matrizes e data frames

- **solve(M)** - retorna a inversa da matriz M.
- **det(M)** - retorna o determinante da matriz M.
- acessando os elementos do objeto:
  M[i,] # acessa a linha i da matriz M
  M[,j] # acessa a coluna j da matriz M
  M[i,j] # acessa a célula na linha i e na coluna j da matriz M
- editando o objeto:
  M[i,j]=3 # substitui o elemento contido na linha i e coluna j de M
  M[,j:l]=0 # atribui o valor 0 a todos o elementos da colunas de j a l.

Operacionalizando data frames

- dados1=data.frame(id=c(1:10),
  idade=c(16,18,19,20,22,21,23,16,17,18),
  genero=c("M","F","M","F","M","F","M","F","M","F"))
- dados2=data.frame(id=c(1:10),
  altura=c(1.66,1.82,1.9,1.52,2,1.76,1.63,1.6,1.7,1.8))
- dados3=data.frame(id=c(3,4,5,8,9,11,22),
  peso=c(61,82,91,100,62,71,92))
- Acessar as variáveis do objeto dados1:
  idade # vai dar erro!!!
  dados1$idade # acessa a variável idade de dados1

  attach(dados1)
  idade # os dois comandos também acessam a variável idade de dados1

Operacionalizando data frames

- *Merge* data frames
  - dad1=merge(dados1,dados2,by="id")
  - dad2=merge(dados1,dados3,by="id")
  - dad2.1=merge(dados1,dados3,by="id",all=TRUE)
  - dad2.2=merge(dados1,dados3,by="id",all.x=TRUE)
  - dad2.3=merge(dados1,dados3,by="id",all.y=TRUE)

Criando Listas

- **list()**
  lista = list(x = sample(1:5, 20, rep=T),y = rep(letters[1:5], 3),
  z = sample(c(0,1,2), 20, rep=T))
- acessando os elementos da lista
  lista[["x"]]
  lista[[1]]
  lista$x

Manipulando listas

- **length(lista)** - número de objeto na lista.
- **names(lista)** - retorna os nomes dos objetos na lista e serve para renomear os objetos.
- **c(lista1,lista2)** - concatena duas listas.
- **append(lista1,lista2,after=2)** - acrescenta o conteúdo da lista2 após a segunda componente da lista1

Departamento de Estatística UFF

Leitura de dados externos

- Comandos de leitura:
  - **read.table**
    - lê dados em arquivos de texto formatados (em grades regulares - arquivos de texto).
    - permite especificar um argumento de cabeçalho, separadores, como lidar com valores em falta e não preenchidos ou linhas em branco.
    - é adequado para as pequenas e médias séries de dados; não é indicado grandes matrizes de dados
    - Ex: read.table("caminho e nome do arquivo",header=T,dec = ".")
  - **read.csv**
    - CSV (*comma separeted values*)
    - lê dados em arquivos com variáveis separadas por vírgula
    - muito útil para dados salvos através do Excel
    - Ex: read.csv("caminho e nome do arquivo")

Leitura de dados externos

- Pacote foreign:
  - **read.dta** - leitura de arquivos em STATA
  - **read.spss** - leitura de arquivos em SPSS
  - Ler arquivo minitab, SAS, Systat, dBase entre outros!
  - **write.dta** - escreve arquivos em STATA
  - **write.table** - escreve arquivos em txt
  - **write.csv** - escrevearquivos em csv.

Departamento de Estatística UFF

Leitura de dados externos

```
dad1=read.table("banco1.txt")

dad1=read.table("banco1.txt",header=TRUE)
str(dad1)

dad1=read.table("banco1.txt",header=TRUE,dec=",")
str(dad1)

dad2=read.dta("basemae.dta")
str(dad)

dad3=read.spss("Dados.sav",to.data.frame = TRUE)
str(dad2)
```

Criando novas variáveis

```
# Três maneiras de efetuar a mesma operação
dados=data.frame(x1=c(1,2,3),x2=c(3,4,5))

Primeira
dados$soma2 = dados$x1 + dados$x2
dados$media2 = (dados$x1 + dados$x2)/2

Segunda
attach(dados)
dados$soma = x1 + x2
dados$media = (x1 + x2)/2
detach(dados)

Terceira
dados2 = transform( dados, soma = x1 + x2, media = (x1 + x2)/2 )
```

Renomeando variáveis

Pacote - reshape

Mudar o nome da variável x1 para y1.

```
rename(dados,c(x1="y1"))
```

Mudar o nome da variável x1 para y1 e x2 para y2 no objeto dados.

```
dados=rename(dados,c(x1="y1",x2="y2"))
```

Comando ddply, substring e paste

- **substring** - extrai parte de uma string
- **paste** - concatena vetores em uma string
- Pacote plyr:
  - **ddply** - aplica funções segundo critérios, combinando os resultados em um data frame.

Loopings e Condicionamento

- **if else** ou **ifelse**

ifelse(condição,tarefa1,tarefa2).

Ex: ifelse(aux>100,0,aux)

```
if(condição1 & ... & condiçãoN){
      conjunto de tarefas
   }else{
      conjunto de tarefas}.
```

Ex: aux=100;b=10

```
if(aux>100 & b!= 10){
      d=10
   }else{
      d=20
}
```

Loopings e Condicionamento

- **if else** ou **ifelse**

```
if(condição1 | ... | condiçãoN ){
      conjunto de tarefas
   }else{
      conjunto de tarefas
}
```

- **for**

```
for(i in 1:N){
      conjunto de tarefas
}
```

- **while**

```
while(condição satisfeita){
      conjunto de tarefas
}
```

Exemplos de *Loopings* e Condicionamento

- Criar uma variável chamada ano.2000 que assume 1 se ano for igual a 2000 e 0 caso contrário.
- **if else** ou **ifelse**

```
ano.2000=ifelse(dad2$ano,1,0).
```

- O que os comandos abaixo fazem??

```
if(a>0 & b>10 ){
      d=a+b ; k=a-b
}else{
      k=a+b
}.
```

Exemplos de *Loopings* e Condicionamento

- **for**

```
a=0
for(i in 1:10) {
      a=a+1
}
aux=NULL for(i in 1:5){
      print(i)
      aux[i]=i+sqrt(i)
}
```

- **while**

```
a=0;d=20
while(a<3){
      a=a+1
      d=d+20
}
```

Criando Funções

- Para criar uma função é importante definir suas **entradas** (argumentos) e **saídas**.
- Os argumentos devem estar entre parênteses e os comandos da função devem estar entre chaves.

  nome.da.funcao=function(argumento1,...,argumentoN)
  {tarefa1 ; tarefa2 ; tarefa3}

- Para chamar a função basta digitar o nome da função seguida de seus argumentos entre parênteses.
  nome.da.funcao(argumento1,...,argumentoN)

Departamento de Estatística UFF

Exemplos de Criando Funções

1. funcao.teste=function(x,y,z){
   $x + y + z^2$
   }.
2. Criem uma função que calcula a raiz quadrada de um número.
3. Criem uma função que calcula o fatorial de um número.
4. Criem uma função que calcula o valor da função $f(x) = e^{-(x-10)^2}$, $x \in \Re$.
5. Criem uma função que calcula o valor da função $f(y) = 3e^{-3y}$, $y > 0$.

Exemplos de Criando Funções

1. Crie uma função que recebe um número como argumento e retorna o inverso desse número. De modo que, se o elemento de entrada for 0 a função deve retornar a seguinte mensagem: "função não definida para o valor 0".

2. Crie uma função que recebe um número como entrada e, se o mesmo não for NA, ela retorna o número ao quadrado e se o mesmo for NA ela retorna a seguinte mensagem: "A função não pode ser aplicada em um NA".

3. Crie uma função que recebe como argumento um vetor e retorna a soma dos quadrados deste vetor e caso receba outro tipo de objeto retorne a seguinte mensagem: "A função só pode ser aplicada em um vetor"

Departamento de Estatística UFF

Exemplos de Criando Funções

```
fun.teste=function(x){
if(is.vector(x)){
val1=sum(x)
val2=prod(x)
cat("soma =", val1, "\n", "produtório =", val2, "\n")
}else{
print("A função só pode ser aplicada em um vetor")
}
}
```

Funções Gráficas

- Principal função gráfica: **plot**.

Argumentos	Descrição
type	tipo de plot que deve ser feito (linha, pontos)
main	título do gráfico
sub	subtítulo para o gráfico
xlab, ylab	título para o eixo x e y
xlim, ylim	dimensiona os eixos x e y
col	cor do gráfico
lwd	largura da linha
lty	tipo de linha (pontilhada, contínua)
cex	dimensiona o tamanho do símbolo
pch	define o tipo de símbolo a ser utilizado no gráfico

Tipos de Gráficos

- **plot(x,y)** - plot das coordenadas em x e em y.
- **barplot** - gráfico de barras.
- **hist** - histograma.
- **pie** - gráfico de setores (pizza).
- **stem** - ramos e folhas.
- **ts.plot** - usado para séries temporais.
- **qqnorm** - Q-Q plot.

Departamento de Estatística UFF

Exemplos de Gráficos

- **plot(x,y)**
  Exemplo 1:

```
y=c(10,8,4,0,4,8,10)
x=-3:3
plot(x,y,pch=20,col=1:7,cex=4)
palette()
```

  Exemplo 2:

```
palette(rainbow(10))
palette()
plot(x,y,pch=20,col=1:7,cex=4)
```

  Exemplo 3:

```
plot(3:4,col=c(rgb(69,139,0,maxColorValue=255),
     rgb(139,62,47,maxColorValue=255)),pch=20,cex=2)
```

Exemplos de Gráficos

- **hist**

```
x=c(seq(10),rep(2:5,3),rep(1:4,by=2))
hist(x,main="Histograma de x")
hist(x, breaks=seq(0,10,by=5), main="Histograma de x")
hist(x, breaks=c(0,1,5,8,10), main="Histograma de x")
```

- **barplot**

```
a=c(rep(1,5),rep(2,11))
s=rep(1:2, each = 2, times = 4)
barplot(table(a))
barplot(prop.table(table(a))*100)
barplot(prop.table(table(a))*100,ylab="Porcentagem",
names=c("Homem","Mulher"),xlab="Gênero",col="red")
```

Exemplos de Gráficos

- **barplot**

  barplot(table(a,s))

  barplot(table(a,s),legend.text=c("Trabalha","Não Trabalha"),names=c("Homem","Mulher"))

  barplot(table(a,s),legend.text=c("Trabalha","Não Trabalha"), names=c("Homem","Mulher"),ylim=c(0,11),col=c("blue","red"))

- **pie**

  a=c(rep(1,5),rep(2,11))

  pie(a)

  pie(table(a),labels=c("A","B"),col=c("dark blue","magenta"))

Exemplos de Gráficos

- **boxplot**

```
a=c(1,2,3,4,3,3,3,2,2,1,2,1,2,2,10,2,3,5,3,1)
b=c(rep("A",10),rep("B",10))
dad=as.data.frame(cbind(a,b))
boxplot(a~b,col=2,main="Boxplot")
boxplot(a~b,col=2,horizontal=TRUE,ylab="Idade")
```

- **ts.plot**

```
ts.plot(a,xlab="Idade")
plot(a,type="l",xlab="Idade")
```

Exemplos de Gráficos

- **stem**

  a=c(0,0.6,0,1,1.7,1,2,3.2,3,3.7,3,3.2,3,3,3,3,3,4.3,4,5,5,5.5,5.6,5,5)
  stem(a)
  stem(a,scale=2)

- **Gráfico de uma função criada**

  Crie a função: $f(x) = \frac{e^{-(x-3)^2}}{2}$ com o nome funcao.teste

  a=seq(2,8,by=0.1)
  plot(a,funcao.teste(a),type="l")

  curve(funcao.teste,2,8) # Mesmo efeito que o comando anterior

Exemplos de Gráficos

- **Criando uma janela para múltiplos gráficos**
  par(mfrow=c(i,j)) - *i*: número de linhas e *j*: número de colunas

  Exemplo:

  ```
  a=c(2,4,6,7,8,2,3,4,5)
  b=c(3,5,4,1,0,0,0,0,2)
  par(mfrow=c(1,2))
  boxplot(a,b)
  ```

- **Como salvar um gráfico???**

Departamento de Estatística UFF

Acrescentando elementos em um gráfico

- **lines** e **points** - acrescenta linhas e pontos em um gráfico.

  plot(-4:4, -4:4, type = "n")

  points(3, 4, col = "red")

  points(3, 2, col = "red",pch=2)

  points(1, 2, col = "red",pch=3)

  points(0, 0, col = "red",pch=4)

  points(0, 0, col = "red",pch=5)

- **abline** - acrescenta linhas verticais e horizontais.

  a=c(1,2,1,2,3,4,4,5,5,5,5,6,6,7,1,2,3,2) hist(a)

  abline(v=3.5,col="red",lwd=2)

Acrescentando elementos em um gráfico

- **text** - acrescenta texto na janela do gráfico e **locator** - localiza a posição do texto no gráfico através do cursor.

```
sexo=c(rep(1,20),rep(2,40))

pie(table(sexo),col=c("red","blue"),labels=c("Masculino",
"Feminino"))

text(locator(n=1),paste(round(prop.table(table(sexo))[1],
digits=2)*100,"%"))

text(locator(n=1),paste(round(prop.table(table(sexo))[2],
digits=2)*100,"%"))
```

Integração numérica e Derivadas

- **integrate** - usada para integração numérica em uma dimensão.

  f.x = function(x){x^2}

  curve(f.x,-5,5)

  integrate(f.x,-4,4)

- **quote** e **D** - utilizados para obter resultados de derivadas de expressões

  f = quote(sin(x) + (1/2)*x^2)

  f

  df.dx = D(f, "x")

  df.dx

Distribuições de probabilidade

O R inclui a funcionalidade para operações com distribuições de probabilidades. Para cada distribuição há 4 operações básicas indicadas pelas letras:

- **d** - calcula a densidade de probabilidade de $f(x)$ no ponto.
- **p** - calcula a função de probabilidade acumulada $F(x)$ no ponto.
- **q** - calcula o quantil correspondente a uma dada probabilidade.
- **r** - retira uma amostra da distribuição.

Distribuição normal

- dnorm(-2)
- dnorm(2)
- pnorm(2)
- qnorm(0,975)
- qnorm(0,5)
- rnorm(10)
- args(rnorm)
- rnorm(20,mean=25,sd=5)

Distribuição normal

- Seja $X \sim N(25, 32)$. Calcule:

  $P(15 < X < 21)$

  $P(X < 21)$

  $P(X > 24)$

- Gere uma amostra de tamanho 200 de uma normal com $\mu = 10$ e $\sigma^2 = 15$.

- Faça um histograma da amostra gerada.

- Faça um gráfico da distribuição $N(10, 15)$ e plote no mesmo gráfico que o histograma.

- Como fazer o gráfico da distribuição acumulada da N(0,1)?

Distribuição binomial

- args(dbinom)
- dbinom(2,10,.22)
- Seja $X \sim Binomial(20; 0,8)$. Calcule: $P(X = 7)$

  $P(X < 8)$

  $P(X \geq 8)$

  $P(7 < X \leq 15)$
- Gere uma amostra de tamanho 50 de uma Binomial com $n = 50$ e $p = 0,5$.
- Faça um gráfico apropriado para a amostra obtida no item anterior.
- Como fazer o gráfico da distribuição acumulada de uma binomial?

Outras distribuições

- dexp - densidade da exponencial.
- rchisq - retira amostra de uma qui-quadrado.
- qt - calcula o quantil de uma t-student.
- pf - calcula $F(x)$ de uma F.
- rbeta - retira uma amostra de uma beta.
- dgamma - densidade de uma gama.
- rpois - retira amostra de uma poisson.
- dunif - densidade de uma uniforme (contínua).
- E a uniforme (discreta)?!

Departamento de Estatística UFF

QQplot

O qq-plot é um gráfico dos dados ordenados contra os quantis esperados de uma certa distribuição.

Quanto mais próximo os pontos estiverem da bissetriz do primeiro quadrante mais próximos os dados observados estão da distribuição considerada.

Portanto para fazer o qqplot seguimos os passos:

1. obter os dados,
2. obter os quantis da distribuição teórica,
3. fazer um gráfico dos dados ordenados contra os quantis da distribuição.

QQplot

- **qqnorm**
  1. Gere uma amostra de tamanho 100 de uma distribuição normal com $\mu =$ **sua idade** e $\sigma^2 =$ **seu peso** e armazene em um objeto chamado amostra.
  2. Faça:
     qqnorm(amostra)
     qqline(amostra)

- **qqplot**
  1. Gere uma amostra de tamanho 100 de uma distribuição qui-quadrado com 5 graus de liberdade e armazene em um objeto chamado amostra.
  2. Faça:
     quantis = qchisq(ppoints(length(amostra)), df = 5)
     qqplot(quantis, amostra)
     abline(0,1)

Maximização de uma função

Como faço para maximizar uma função?

Suponha que você esteja interessado em maximizar a função

$$f(x) = 2 - (x-3)^2, \qquad -3 < x < 5.$$

Siga os seguintes passos:

1. usando o *function* escreva a função a ser maximizada,
2. plote o gráfico da função,
3. utilize o comando **optimise(função,c(-3,5),maximum=T)**

Maximização de uma função

Qual o nosso principal interesse em encontrar o máximo de uma função?

Suponha que você esteja interessado em encontrar a **estimativa** de máxima verossimilhança do parâmetro λ da distribuição de poisson, se foi observada uma amostra com os seguintes valores

$$x = c(2, 3, 1, 2, 3, 4, 5, 3, 3, 1).$$

Siga os seguintes passos:

1. usando o *function* escreva a função a ser maximizada,
2. plote o gráfico da função,
3. utilize o comando **optimise(função,c(0,10),maximum=T)**
4. acrescente no gráfico da função de verossimilhança uma linha vertical vermelha no valor da estimativa obtida para λ.

Maximização de uma função

```
fx2=function(x,lambda){
exp(-length(x)*lambda)*lambda^{sum(x)}*prod(1/factorial(x))
}

dados=c(2,3,1,2,3,4,5,3,3,1)

curve(fx2(c(2,3,1,2,3,4,5,3,3,1),x),0,10) # lembrar que no curve x é o que varia!!!

optimise(fx2,c(0,10),maximum=T,x=c(2,3,1,2,3,4,5,3,3,1))
```

Maximização de uma função

Encontre a **estimativa** de máxima verossimilhança do parâmetro λ da distribuição exponencial, maximizando a **função de log-verossimilhança**, se foi observada uma amostra com os seguintes valores

$$x = c(2.5, 3.6, 7.1, 2.1, 3, 4.8, 9.5, 2.3, 3.3, 1.9).$$

Funções *apply* e *sapply*

Função *apply* retorna um vetor de valores obtidos pela aplicação de uma função as marginais de uma matriz.

apply(X, **marginal**, **função**),

em que X é sua matriz de valores, **marginal** assume os valores 1 ou 2 (linha e coluna, respectivamente) e **função** é a função que você deseja aplicar as marginais.

```
a=matrix(c(1,10,2,3,4,5,6,1,3,4,6,7),ncol=4)
a
apply(a,1,mean)
apply(a,2,mean)
apply(a,2,sum)
apply(a,2,sd)
```

Funções *apply* e *sapply*

```
teste=function(x){
sum(x2)
}
```

```
apply(a,2,teste)
```

E se X for uma lista. A função apply funciona?

```
a=list(x=seq(1,10),y=rep(3,25),z=sample(c(1,2,3,4,8,10),21,rep=T))
```

```
a
```

```
apply(a,2,sum)
```

Experimente fazer:

```
sapply(a,sum)
```

```
sapply(a,quantile)
```

Verificando resultados teóricos por meio de simulação

```
n=10
normal=rnorm(n)
qui=rchisq(n,10)

t=normal/sqrt(qui/10)

hist(t,freq=FALSE,main="Amostra de uma t")
curve(dt(x,10),add=TRUE)
```

Verificando resultados teóricos por meio de simulação

```
par(mfrow=c(2,2))
for(n in c(5,10,50,1000)){
normal=rnorm(n)
qui=rchisq(n,10)

t=normal/sqrt(qui/10)

titulo=paste("n =",n)
hist(t,freq=FALSE,main=titulo)
curve(dt(x,10),add=TRUE)
}
par(mfrow=c(1,1))
```

Distribuição amostral da média

TLC: Para uma a.a.s. $(X_1, X_2, \ldots, X_n)$ retiradas de uma população com média μ e variância σ^2 finita, a distribuição amostral da média $\bar{X}$ aproxima-se para n grande, de uma distribuição normal, com média μ e variância σ^2/n.

Considere a população $\{1,3,5,5,7\}$ e faça o que se pede:

1. Retire 1000 amostras de tamanho n desta população.
2. Armazene as amostras de modo que em cada coluna fique uma amostra de tamanho n.
3. Obtenha as médias de cada amostra.
4. Faça uma representação gráfica para ajudar a visualizar o resultado do TLC.

Distribuição amostral da média

```
num.amostra=1000
n=2
a=matrix(NA,ncol=num.amostra,nrow=n)
for(i in 1:num.amostra){
a[,i]=sample(c(1,3,5,5,7),n,rep=T)
}
resultado=apply(a,2,mean)
hist(resultado,freq=F)
curve(dnorm(x,mean=4.2,sd=2.28/sqrt(n)),add=T)
abline(v=mean(resultado),col="red")
abline(v=4.2,col="blue")
```

Repita o que foi feito variando o valor de n, use 3, 4, 20, 200 e 400.

Distribuição amostral da média

Refazendo o mesmo exercício para uma população Normal com $\mu = 30$ e $\sigma^2 = 40$. num.amostra=1000

```
n=2
a=matrix(NA,ncol=num.amostra,nrow=n)
for(i in 1:num.amostra){
a[,i]=rnorm(n,mean=30,sd=sqrt(40))
}
resultado=apply(a,2,mean)
hist(resultado,freq=F)
curve(dnorm(x,mean=30,sd=sqrt(40/n)),add=T)
abline(v=mean(resultado),col="red")
abline(v=30,col="blue")
```