

O R está com tudo!

Viviana G R Lobo
vivianalobo@id.uff.br

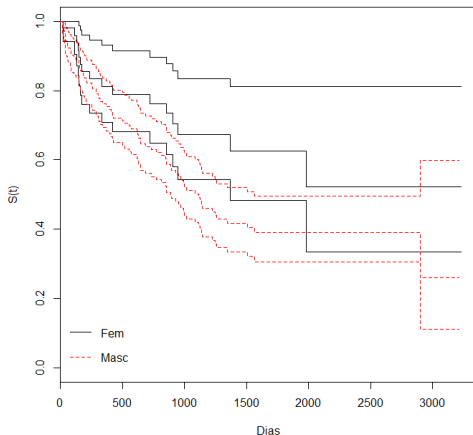
Departamento de Contabilidade - UFF

9^o Semana da Estatística

23 de Outubro de 2017.

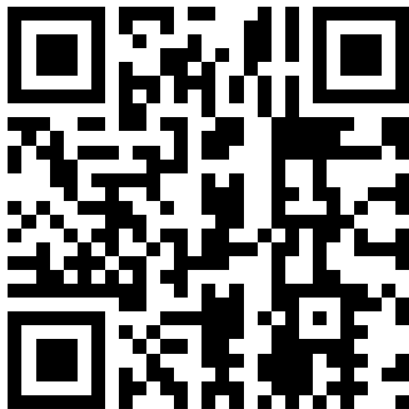
Motivação (Análise de Sobrevivência)

Considere estudar o tempo de sobrevivência de indivíduos com a determinada doença de acordo com o sexo.



Motivação (QR Code)

QR code é um código de barras, que foi criado em 1994, e possui esse nome pois dá a capacidade de ser interpretado rapidamente pelas pessoas.



R!

- O R é uma linguagem de programação voltado para análise de dados.
- O R é altamente expansível com o uso dos pacotes, que são bibliotecas para funções específicas ou áreas de estudo específicas.
- A popularidade se baseia por ser um software de uso livre e de fácil manuseio.
- A linguagem é muito similar com outras linguagens de programação como S-plus, C++ e outros.

Vantagens

- facilidade no manuseio e armazenamento de dados;
- operadores vetoriais e matriciais;
- uma ampla e coerente variedade de ferramentas para análise de dados
- que atua de forma integrada;
- facilidades gráficas como a confecção imediata de figuras que podem ser salvas em diversos formatos;
- linguagem de programação simples e eficiente;
- integração com outros softwares, por exemplo, C++ e WinBUGS;
- calculadora!

Desvantagens

- pode ser lento, por exemplo, quando se trata de simulações intensivas

R!

- O download do software **R** é gratuito de acordo com a licença GNU e oferece um sistema composto por software livre, isto é, que respeita a liberdade dos usuários.
 - Está disponível para as plataformas UNIX, Windows e MacOS.
 - Para baixar, basta acessar o site <https://www.r-project.org/> e
- ① Clique em CRAN (*Comprehensive R Archive Network*)
 - ② Escolha o espelho de sua preferência (CRAN mirrors)
 - ③ Opções de download: Download R for Linux / Download R for (Mac) OS X / Download R for Windows

Interface

- A interface se baseia em uma janela com poucas opções na barra de ferramentas.
- Não há planilhas para digitação de banco de dados ou ícones na barra de ferramentas para cálculos.
- As análises são realizadas por linhas de comandos, ou seja, usuário **manda** e o programa **executa**.
- Todas as informações ficam armazenadas na memória do computador. Podemos salvá-las, descartá-las e/ou manipulá-las.

Interface

- Cada vez que é inicializado o **R** é criado um novo *workspace*. Dentro deste *workspace* o usuário poderá criar qualquer tipo de análise. Você tem a opção de salvar este *workspace*. Sempre que o usuário utilizar o **R** e desejar carregar algum trabalho já realizado, basta carregar o *workspace* salvo em alguma pasta de trabalho inserida no computador.
- Para salvar qualquer *workspace* vá em arquivo (*File*), salvar área de trabalho (*workspace*), escolher a pasta desejada (chamaremos de diretório). Para carregar, basta ir em arquivo, carregar área de trabalho e escolher o arquivo.
- O usuário também tem a opção de salvar um editor, no formato **.R**, ao invés de salvar uma área de trabalho. Basta estar em **Editor R > Arquivo > Salvar > salvar como > meuarquivo.R**.

Pacotes do R (*package*)

- Quando instalamos o programa a partir da configuração padrão, alguns pacotes disponibilizados de maneira automática.
- Para tarefas mais complicadas pode ser necessário o uso de pacotes adicionais.
- Para instalar qualquer pacote basta ir em **Pacotes > Instalar Pacotes (escolher a CRAN utilizada) > nome do pacote**. Podemos utilizar o comando `> install.packages()` com o nome do pacote desejado entre aspas.

Por exemplo para instalar o pacote MASS, no prompt:

```
> install.packages("MASS") # deve estar conectado a internet  
> # carregar o pacote quando necessário:  
> library(MASS)  
> require(MASS) # carregar o pacote quando necessário
```

O Editor R Studio

- **RStudio** é um software livre de ambiente de desenvolvimento integrado para R.
- Possui 4 janelas de operação:
 - ① canto superior esquerdo: editor (*script*) ;
 - ② canto superior direito: *workspace* e *history*;
 - ③ canto inferior direito: disponibiliza *help!*, gráficos, pacotes e arquivos;
 - ④ canto inferior esquerdo: console do R.
- Disponível no site `www.rstudio.org` e com procedimento de instalação igual ao R.

Vamos conhecer o programa!

Primeiro Acesso

- ① o sinal `>` indica o *prompt* e significa que o programa está pronto para receber comandos impostos pelo usuário.
- ② o sinal `+` indica que “faltou” ou deve acrescentar algo na linha de comando anterior.
- ③ para fazer um comentário na linha de comando basta digitar `> # meu comentario.`
- ④ Esc retorna ao *prompt* normal.
- ⑤ é possível criar um arquivo editor do *R* para ser salvo posteriormente.
- ⑥ para rodar um comando no Console basta digitar **ENTER**, para rodar um comando no Editor bastar utilizar as teclas **CTRL + enter**.
- ⑦ para limpar o Console basta digitar **CTRL + L**.
- ⑧ para remover **todos** os objetos do console basta digitar `> rm(list=ls(all=TRUE))`

Primeiro Acesso

- Verificar em qual diretório de trabalho de R está usando o comando

```
> getwd()
```

- Para mudar o diretório de trabalho use o comando `setwd()`

```
> setwd("C:/Documents and Settings/Administrador/Meus documentos/nomedapasta")
```

- Verifique se mudança funcionou a partir do comando `getwd()`
- Verifique se o seu diretório fixado está vazio a partir do comando `dir()`
- Salvando objetos do seu workspace a partir da extensão `.RData`

```
> save.image(file="C:/Documents and Settings/Administrador/Meus documentos/  
+ nomedapasta/nomedoarquivo.RData")
```

Help!

Geralmente, o comando `help()` ou `?help` do possui 10 tópicos básicos:

- ① *Description* - faz um resumo geral sobre o uso da função.
- ② *Usage* - mostra como a função deve ser utilizada e quais argumentos podem ser especificados.
- ③ *Arguments* - explica o que é cada um dos argumentos.
- ④ *Details* - explica alguns detalhes sobre o uso e aplicação da função.
- ⑤ *Value* - mostra o que sai no output após usar a função.
- ⑥ *Note* - notas sobre a função.
- ⑦ *Authors* - lista os autores que criaram a função.
- ⑧ *References* - referências para os métodos usados.
- ⑨ *See also* - mostra outras funções relacionadas que podem ser consultadas.
- ⑩ *Examples* - exemplos do uso da função (copie e cole os exemplos no *R* para ver como funciona).

Importando Dados

- Dados que já estão disponíveis em outro programa, pode ser importado para o R sem necessidade digitá-los novamente.
- Há dois modos de fazer isso:
 - ① especificando o caminho do arquivo;
 - ② especificando apenas o nome do arquivo se o diretório estiver fixo.

Importando Dados

Carregando arquivos do tipo .txt

Considere importar para o [R Studio](#) os dados do arquivo XX. Para importar o arquivo, basta digitar o comando `read.table()`

- Escolhendo o diretório de trabalho

```
> dados <- read.table("C:/users/Viviana/Dropbox/Minicurso/R/dado.txt", header=TRUE)
```

- No diretório de trabalho

```
> dados <- read.table("dado.txt",header=T)
> # Usar o argumento HEADER=T para diferenciar os dados do nome da variavel.
```


Importando Dados

Carregando arquivos do tipo .csv

Considere importar para o [R Studio](#) dados do arquivo XX. Para importar o arquivo, basta digitar o comando `read.csv()`

- Escolhendo o diretório de trabalho

```
> dados <- read.csv("C:/users/Viviana/Dropbox/Minicurso/R/dado.csv", header=TRUE, sep=";")
```

- No diretório de trabalho

```
> dados <- read.csv("dado.csv",header=T, sep=";")  
> # necessario dizer que os dados sao separados sep;
```

Importando Dados

Exemplo

(1) Carregar conjunto de dados `ipec.csv`

```
> ipec.data<- read.table("ipec.csv", header=T, sep=";")
```

(2) Salvando e Carregando um conjunto de dados

```
> A <- matrix(5:20,4,4) # criando matriz
> write.table(A,"meudado.txt",row.names=FALSE,col.names=FALSE)
# salvando dados em .txt
> write.csv(A,"meudado.csv",row.names=FALSE) # salvando da-
dos em .csv
> read.table(A,"meudado.txt")
> read.csv(A,"meudado.csv")
```

Operações Básicas

A utilização mais básica do R é usá-lo como calculadora. As operações matemáticas básicas são:

- + soma;
- - subtração;
- * multiplicação;
- / divisão;
- ^ exponenciação.

Operações Básicas

Algumas funções aritméticas implementadas no R:

<code>sum()</code>	soma
<code>sqrt()</code>	raiz quadrada
<code>abs()</code>	valor absoluto (positivo)
<code>prod()</code>	multiplicação
<code>factorial()</code>	fatorial de um número
<code>sin()</code> <code>cos()</code> <code>tan()</code>	funções trigonométricas
<code>asin()</code> <code>acos()</code> <code>atan()</code>	funções trigonométricas inversas
<code>sinh()</code> <code>cosh()</code> <code>tanh()</code>	funções hiperbólicas
<code>asinh()</code> <code>acosh()</code> <code>atanh()</code>	funções hiperbólicas inversas
<code>exp()</code> <code>log()</code>	exponencial e logaritmo natural
<code>log10()</code>	logaritmo base-10

Comandos de lógica

Vamos entender o significado dos comandos abaixo:

```
> Maior que >= maior que ou igual a
< Menor que <= menor que ou igual a
== igualdade
!= diferença
> x<-c(1,2,9,4,5)
> y<-c(1,2,6,7,8)
> 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
```

Vamos analisar alguns exemplos!

O que são objetos?

- O **R** é uma linguagem orientada à objetos: **variáveis**, **vetores**, **matrizes**, **funções** que são armazenados na memória ativa do computador na forma de objetos.
- Se chamamos um objeto de **x** e esse objeto possui algum valor (numérico ou não numérico), por exemplo 4, ao digitarmos seu nome ele retorna a este valor.

Veja no exemplo

O que são objetos?

```
> x <-4  
> x  
[1] 4
```

Podemos armazenar um valor em um objeto com certo nome (o usuário tem livre arbítrio para escolher o nome do objeto).

```
> (x<- factorial(4)) # o resultado aparece imediatamente  
[1] 24
```

Alternativamente podemos utilizar o símbolos <- ou =.

```
> x <- sin(pi) # este é o formato tradicional  
> x = sin(pi)
```

Ver conjunto de dados cars do R.

O que são objetos?

Vetores: representam um conjunto de valores numéricos ou de caracteres (letras e/ou palavras). Para criar utilizamos a função `c()`, **concatenação**.

```
> v1<- c(1,4,5,6,8,10) # vetor numerico
> v1[1] # representa o primeiro elemento do vetor
> v1[6] # representa o último elemento do vetor
> v1[c(1,4)] # acessa os elementos 1 e 4 do vetor
> v1[-2] # exclui o elemento de posicao 2 do vetor
> v1[1]<- 100 # fixa o valor 100 para o primeiro elemento
do vetor
> vogal<- c("a","e","i","o","u") # vetor de caracteres
> v2<- NULL # cria um vetor nulo
```


Comando rep()

```
> vog1<- rep("a",10) # vetor que repete a vogal a, 10 vezes  
> binaria<- c(rep(c("sim","nao"),c(4,5)), rep("sim",2))  
# repete "sim"e "nao",  
# 4 e 5 vezes respectivamente e repete "sim"duas vezes
```

Comando seq()

```
> seq(from=1,to=10,by=2) # 1,3,5,7,9  
> seq(1,10, length=10)  
> 1:10 # tambem e sequencia
```

[Abrir arquivo exemploPag25.R](#)

Suponha um vetor genérico x . Podemos fazer algumas operações a partir dele:

```
> sum(x) # soma de todos os elementos
> min(x) # valor mínimo
> max(x) # valor máximo
> summary(x) # resumo
> length(x) # comprimento do vetor
> sort(x) # ordena em ordem crescente
> sort(x, TRUE) # ordena em ordem decrescente
> round(x, 2) # arredonda os valores em 2 casas decimais
> is.vector(x) verifica se é um vetor
> is.matrix(x) verifica se é uma matriz
> is.numeric(x) verifica se é um vetor numérico
> is.character(x) verifica se é um vetor de caracteres
```

Exemplo

(1) Altura dos alunos do curso de Cálculo I, supondo que há 20 alunos inscritos na disciplina.

```
> x<- seq(1.53, 1.85, by = 0.001)
> set.seed(123)
> altura<- sample(x,20)
```

O que são objetos?

Matrizes representam uma coleção de vetores de mesma natureza organizado em linhas e colunas. Para criar, utilizamos a função `matrix()`.

```
matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,  
        dimnames = NULL)
```

- Criar uma matriz de 3 linhas e 3 colunas a partir de uma sequência de 1:9

```
> m1<- matrix(1:9, nrow=3, byrow=FALSE) # preenchimento por  
coluna
```

A partir da matriz:

```
> length(m1) # comprimento da matriz
> dim(m1) # dimensao da matriz
> nrow(m1) # numero de linhas
> ncol(m1) # numero de colunas
> m1[1,2] # primeiro elemento da segunda coluna
> m1[2,2] # segundo elemento da segunda coluna
> m1[,2] # todos os elementos da segunda coluna
> m1[3,] # todos os elementos da terceira linha
> m1[-2,] # todas as linhas exceto a segunda
```

O que são objetos?

Arrays são generalizações das matrizes e podem ter várias dimensões. Para criar, utilizamos a função `array()`.

```
array(data = NA, dim = length(data), dimnames = NULL)
```

```
> # Exemplo 1  
array(1:3, c(2,4))  
> # Exemplo 2  
> A <- array(1:24,dim=c(2,3,4))  
> # 4 matrizes de dimensão 2 x 3  
> A[1,2,4] # linha 1, coluna 2 do bloco 4
```

[Abrir arquivo exemploPag30.R](#)

O que são objetos?

Data frames o mesmo que uma matriz, mas aceita vetores de natureza diferente. Para criar, utilizamos a função `data.frame()`.

- A partir de vetores numéricos de mesmo comprimento

```
> df1 <- data.frame(X = 1:10, Y = c(51, 54, 61, 67, 68, 75,
77, 75, 80, 82))
> names(df1) # nomes atribuídos aos vetores
> df1$X # visualiza o primeiro vetor X
> df1$Y # visualiza o segundo vetor Y
> attach(df1)
```

Ver `help(attach)`

- Mistura de vetores numéricos e caracteres

```
> d2<- data.frame(X= seq(1,5,by=1), Y= vogal)
```

O que são objetos?

Listas representam um conjunto de **vetores**, **dataframes** ou de **matrizes** e não precisam ter o mesmo comprimento. Para criar, utilizamos a função `list()`.

```
> X<- 1:10  
> Y<- ``probabilidade``  
> Z<- matrix(1:9, ncol=3)  
  
> lista1<- list(X,Y,Z)  
> lista1
```


Classe de objetos

O usuário pode verificar a qual classe um objeto `x` pertence a partir dos seguintes comandos:

```
> is.numeric(): verifica se o objeto é numérico
> is.vector(): verifica se o objeto é vetor
> is.matrix(): verifica se o objeto é matriz
> is.data.frame(): verifica se o objeto é data frame
> is.list(): verifica se o objeto é lista
> is.character(): verifica se o objeto é caracter.
```

É possível, em alguns casos, transformar um objeto de uma classe específica por outra classe, substituindo o `is` pelo comando `as`.

```
> setwd(C:/Users/viviana/Dropbox/Minicurso R)
> inflacao<- read.table("dadoInflacao.csv", sep=";")
> is.matrix(inflacao)
> as.matrix(inflacao)
```

[Abrir arquivo exemploPag33.R](#)

Exercício

- Considere um modelo de regressão linear simples da forma

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2)$$

- Reescrevendo o modelo na forma matricial, temos

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \quad \boldsymbol{\varepsilon} \sim N(\mathbf{0}, \sigma^2 \mathbf{I})$$

em que $\mathbf{y} = (y_1, \dots, y_n)'$, \mathbf{X} é uma matriz de desenho com primeira coluna de 1's e segunda coluna igual a $(x_1, \dots, x_n)'$, $\boldsymbol{\beta} = (\beta_0, \beta_1)'$, $\boldsymbol{\varepsilon} = (\varepsilon_1, \dots, \varepsilon_n)'$.

- O estimador de mínimos quadrados (e de máxima verossimilhança) de $\boldsymbol{\beta}$ é

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

```
> beta.hat<- solve(t(X)%*%(X))%*%t(X)%*%y
```

[Abrir arquivo exemploPag34.R](#)

Análise Descritiva dos Dados

Apresentar funções que buscam sumarizar as informações disponíveis sobre o comportamento de uma variável.

```
> mean(x):  média da variável x
> median(x):  mediana da variável x
> quantile(x,p):  quantil da variável x
> range(x):  mínimo e máximo da variável x
> summary(x):  esquema de 5 números da variável x
> table(x):  retorna a frequências de valores de x
> var(x):  variância da variável x
> sd(x):  desvio padrão da variável x
> cov(x,y):  covariância entre duas variáveis x e y
> cor(x,y):  correlação entre duas variáveis x e y
```

[Abrir arquivo exemploPag35.R](#)

Análise Gráfica

Produzir gráficos é uma ótima forma para apresentar e explorar dados em um primeiro momento.

Alguns comandos interessantes que estão dentro da função plot

```
> main="Título do gráfico": título no gráfico
> pch= número: muda o tipo do ponto no gráfico
> col= "nome da cor": cor no gráfico
> xlab("nome"): nome da variável que representa o eixo x
> ylab("nome"): nome da variável que representa o eixo y
> xlim(c(min,max)): limite da variável que representa o eixo x
> ylim(c(min,max)): limite da variável que representa o eixo y
```

Análise Gráfica

```
> barplot(x)
> pie(x)
> hist(x)
> boxplot(x)
> plot(density(x))
```

Janela de gráficos

Podemos dividir uma janela gráfica para analisar mais de um gráfico pela função `par()`.

```
> par(mfrow=c(nl,nc)) # nl indica o número de linhas
e nc o número de colunas que a janela deverá ter.
```

Analisando um conjunto de dados (1)

Índice de Placa Bacteriana

Analisaremos um conjunto de dados disponível no livro de *Estatística Básica, dos autores Bussab, W. e Morettin, P.* que representa medidas de um Índice de placa bacteriana obtidas de 26 crianças em idade pré escolar **antes** e **depois** do uso de uma escova experimental e uma escova convencional.

[Abrir arquivo exemploPag38.R](#)

Analizando um conjunto de dados (2)

Um estudo sobre os níveis de ozônio na costa sul da Califórnia para os anos de 1980 a 1991 calculou o número de dias que os níveis de ozônio passaram de $0.20ppm$. O pesquisador acredita que o número de dias depende de um índice meteorológico que é calculado fornecido a cada ano.

ano	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991
dias	88	91	58	82	81	65	61	48	61	43	33	36
índice	17.2	18.2	16.0	17.2	18.0	17.2	16.9	17.1	18.2	17.3	17.5	16.6

[Abrir arquivo exemploPag39.R](#)

Distribuições de Probabilidade

Seja X uma variável de interesse no estudo. Os valores associados a X podem ser do tipo:

- Discreto: se número de valores possíveis de X for enumerável
- Contínuo: pode assumir qualquer valor numérico em um determinado intervalo ou série de intervalos.

Comandos Gerais

d: calcula a densidade de probabilidade $f(x)$ no ponto, ou seja, $P(X = x)$

p: calcula a função de probabilidade acumulada $F(x)$ no ponto, ou seja, $P(X \leq x)$

q: calcula o quantil correspondente a uma dada probabilidade

r: retira uma amostra da distribuição

Pretendemos explorar essas quatro operações utilizando **exemplos** de algumas distribuições de probabilidade.

Distribuições de Probabilidade

Algumas distribuições

Distribuição	Comando	Argumento
Binomial	binom	size, prob
Poisson	pois	lambda
Geométrica	geom	prob
Hipergeométrica	hyper	m, n
Beta	beta	shape1, shape2
Exponencial	exp	rate=1
Normal	norm	mean=0, sd=1
Gama	gamma	shape, rate=1
T-Student	t	df
F-Snedecor	f	df1, df2
Chi-Quadrado	chisq	df

Distribuições de Probabilidade

Gerando amostras aleatórias

- Gerar dados aleatórios com distribuição Uniforme a partir da função `runif()`
- Gerar dados aleatórios com distribuição Normal a partir da função `rnorm()`
- Gerar dados aleatórios com distribuição Poisson a partir da função `rpois()`
- Gerar dados aleatórios com distribuição Binomial a partir da função `rbinom()`

A partir dos itens acima vamos fazer algumas análises.

Exemplos

- Gere uma amostra de tamanho 200 da Distribuição Normal. Faça um histograma e compare com a curva teórica.
- Faça um gráfico da função de distribuição acumulada da distribuição Chi-Quadrado com 2 graus de liberdade.
- Faça um gráfico da função de densidade

$$f(x) = 0,4(2\pi)^{-1/2}\exp\left\{-\frac{(x-2)^2}{2}\right\} + 0,6(2\pi)^{-1/2}\exp\left\{-\frac{(x+2)^2}{2}\right\}$$

[Abrir arquivo exemploPag43.R](#)

Exercício (Distribuição Binomial)

Seja X a v.a. que representa o número total de sucessos e, n ensaios de Bernoulli, com probabilidade p de sucesso, $0 < p < 1$.

$$P(X = x) = C_x^n p^x (1 - p)^{n-x}, \quad x = 0, 1, \dots$$

(1) A probabilidade de uma peça artesanal ser feita com perfeição por um artesão é de 50%. Considerando que o artesão produz, de maneira independente, 6 peças por dia, pede-se:

- Obter a distribuição de probabilidades, ou seja, as probabilidades associadas aos possíveis valores da variável aleatórias discreta x , em que X é número de peças perfeitas produzidas pelo artesão num único dia.
- Plotar o gráfico com os valores da probabilidade calculada.
- Plotar o gráfico da distribuição acumulada.

[Abrir arquivo exemploPag44.R](#)

Exercício (Distribuição Poisson)

A variável aleatória X , que denota o número de contagens no intervalo, possui uma distribuição de Poisson com parâmetro λ e é representada através da função:

$$P(X = x) = \frac{e^{-\lambda} \lambda^x}{x!}, \quad x = 0, 1, 2, \dots, n,$$

(2) Em um fio delgado de cobre, o número de falhas no fio segue a distribuição de Poisson, com uma média de 2,3 falhas por milímetro.

- Determine a probabilidade de existir exatamente 2 falhas em um milímetro de fio.
- Sabendo que o número máximo de erros no teste de qualidade é de 10 erros/mm, verifique as probabilidades de que ocorram de 0 a 10 falhas no fio. Plote o gráfico da distribuição.

[Abrir arquivo exemploPag45.R](#)

Introdução a Funções

O programa **R** além de ser muito útil como ferramenta de análise de dados ele permite que o usuário crie suas próprias funções. A sintaxe básica para escrever funções é dada pelos seguintes comandos

```
> minha.funcao <- function(){  
  comandos da minha função  
}
```

Uma vez definida, a função pode ser chamada como qualquer outra função ou comando interno do **R**. As funções podem ou não ter argumentos,

```
> minha.funcao <- function(argumentos){  
  comandos da minha função  
}
```

Exercícios

- Criar uma função que calcula a média de um conjunto de dados. Sabemos que no R o comando que calcula a média é `mean(x)`.
- Criar uma função que calcule a variância de um conjunto de dados. No R, sabemos que a função se chama `var(x)`.
- Criar uma função que retorne o montante M após um tempo t de um investimento X a uma taxa de juros i considerando:
 - ▶ **juros simples** $M = X(1 + it)$;
 - ▶ **juros compostos** $M = X(1 + i)^t$.

[Abrir arquivo exemploPag47.R](#)

Loopings e Condicionamentos

O comando `if` é utilizado para validar uma condição e a partir disto executar o código de acordo com o resultado.

```
> if(condição verdadeira){  
comandos  
}
```

O comando `else` é utilizado de maneira conjunta ao `if`, representando a condição caso o contrário.

```
> if(condição verdadeira){  
comandos A  
}else{ comandos B }
```

O comando `print()`, é usado para descrever o conteúdo de qualquer objeto.

Loopings e Condicionamentos

Exemplos

- Vamos avaliar a nota final do aluno no curso de Probabilidade I. Sabemos que se o aluno possuir uma média final igual ou acima de 6 ele é aprovado, caso contrário é reprovado.
- Criar uma função considerando a nota de todos os alunos no curso de Probabilidade I.

[Abrir arquivo exemploPag49.R](#)

Loopings e Condicionamentos

Os operadores lógicos `||` (“ou”) e `&&` (“e”) são úteis quando precisamos verificar mais de uma condição ao mesmo tempo.

Exemplo

- Avaliar se o aluno está reprovado, na VS ou Aprovado.

[Abrir arquivo exemploPag49.R](#)

Loopings e Condicionamentos

- O comando **for** é utilizado para fazer loopings, ou seja, cria um ciclo a partir de um contador. O ciclo é interrompido quando uma condição que dependa do contador não é verdadeira.

Ele funciona da seguinte maneira:

```
> for(contador in sequência){  
  comandos  
}
```

- O que isso quer dizer? Para cada valor do **contador** o programa irá calcular os **comandos** até o último valor da sequência.

Se chamarmos *i* de contador e *1:n* de sequência, então:

```
> for(i in 1:n){  
  comandos  
}
```

Loopings e Condicionamentos

Para salvar resultados de *loopings*, podemos criar objetos vazios que receberá os valores calculados.

```
> nome.objeto <- NULL # no caso em que o objeto é um vetor.  
  
> nome.objeto<- matrix(NA, ncol , nrow) # no caso em que o  
objeto é uma matriz
```

A função `cat()` é muito parecida com a função `print()` e também pode ser usada para escrever objetos ou constantes.

Funciona com um número qualquer de argumentos, sendo que sua função é transformar os seus argumentos em *strings* para depois escrevê-los no Console.

Exemplos

- Crie uma função que encontre o máximo de um vetor. Compare o resultado de sua implementação com a função `max()` do [R](#).
- Crie uma função que calcule o fatorial de n . Compare o resultado de sua implementação com a função `factorial()` do [R](#).
- Crie uma função que calcule a soma de um vetor. Compare o resultado de sua implementação com a função `sum()` do [R](#).
- Crie uma função que calcule a soma acumulada de um vetor. Compare o resultado de sua implementação com a função `cumsum()` do [R](#).

[Abrir arquivo exemploPag53.R](#)

Loopings e Condicionamentos

Já o comando `while` cria um ciclo a partir de uma condição e interrompe este ciclo quando a condição se torna falsa.

```
> while(condição é verdadeira){  
  comandos  
}
```

Podemos interpretar o comando `while` da seguinte maneira: enquanto a condição for verdadeira vamos repetir o bloco de instruções do ciclo.

Exemplo

Utilizando `while` e `cat`

```
> x <- rnorm(1) ##gerando um numero aleatorio da distribui-  
cao normal  
> while(x < 1){ #enquanto x for menor que 1, faça os se-  
guintes comandos abaixo  
#escreve que x vale um número aleatório menor que 1  
>cat("x=", x )  
>x <- rnorm(1)  
>if(x>=1){ #condição para nova linha  
>cat()  
> }  
> }
```

[Abrir arquivo exemploPag55.R](#)

APLICAÇÕES

Aplicação 1 (Séries Temporais)

(1) Gerar dados de um modelo Auto Regressivo dado pela forma

$$y_t = 0.5y_{t-1} + \epsilon_t, \quad \epsilon_t \sim N(0, 1)$$

- assume $t = 100$
- modelo AR(1)
- criar um gráfico da série temporal

[Abrir arquivo aplicacao1.R](#)

Aplicação 1 (Séries Temporais)

(2) Faça o gráfico das séries `USAccDeaths` e `AirPassengers` do banco de dados do [R](#). Identifique características como tendência, padrão sazonal, alteração de amplitude e mudança estrutural.

[Abrir arquivo aplicacao1.R](#)

Aplicação 2 (Modelos Lineares)

(1) Uma seguradora do ramo de saúde deseja estudar os valores gastos com reembolsos de despesas com exames laboratoriais para homens que tem o seu plano contratado, pois verificou um aumento significativo em sua sinistralidade no último ano. Para isso, a seguradora observou uma amostra de 60 homens com idade entre 40 e 80 anos e ajustou um modelo de regressão linear simples.

Considere um modelo de regressão linear simples para entender esta relação, da forma

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2)$$

Aplicação 2 (Modelos Lineares)

Neste exercício iremos simular os dados e ajustar o modelo:

- Simular a variável X que representa a idade do segurado (variável explicativa)
- Simular os erros, que por suposição, são normalmente distribuídos com média zero e variância constante para cada segurado.
- Simular a variável Y que representa os gastos em despesas (R\$) associado a cada segurado (variável resposta)
- Fazer o gráfico de dispersão para avaliar essa relação
- Calcular o coeficiente de correlação
- Ajustar a reta pelo comando `lm()` ou `lsfit()`

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, \quad i = 1, \dots, 60.$$

O ajuste é feito a partir do estimador de mínimos quadrados

- Verique que a reta passa pelo ponto (\bar{x}, \bar{y})

[Abrir arquivo aplicacao2.R](#)

Aplicação 2 (Modelos Lineares)

(2) Analisar os resíduos do modelo

$$z_i = \frac{y_i - \hat{y}_i}{\hat{\sigma}} \sim N(0, 1)$$

(3) Verificar normalidade dos resíduos, a partir da função `qqplot()`

Os pontos devem estar dispostos próximo a reta identidade $x = y$ para que haja normalidade nos resíduos. Por consequência, podemos considerar a população normalmente distribuída.

[Abrir arquivo aplicacao2.R](#)

Aplicação 3 (Monte Carlo)

- Se a integração analítica não é possível ou é complicada, podemos aproximá-la usando métodos numéricos.
- O método originalmente foi desenvolvido por físicos para resolver integrais.

Queremos integrar a função $h(x)$ no intervalo (a,b) . Podemos escrever a integral de interesse:

$$I = \int_a^b h(x)dx = \int_a^b \frac{h(x)}{f(x)} f(x)dx$$

Dessa forma, o integrando pode ser visto como o valor esperado de $h(x)/f(x)$ sob a densidade $f(x)$ com suporte no intervalo (a, b) . A integração de Monte Carlo é dada por:

$$I = \int_a^b h(x)dx = \int_a^b \frac{h(x)}{f(x)} f(x)dx \approx \frac{1}{n} \sum_{i=1}^n \frac{h(x_i)}{f(x_i)},$$

com (x_1, \dots, x_n) uma amostra de $f(x)$.

Aplicação 3 (Monte Carlo)

(1) Suponha que queremos integrar a função

$$h(x) = [\cos(x) + \sin(20x)]^2,$$

no intervalo $(0,1)$. Nesse caso, poderíamos resolver analiticamente. O valor exato dessa integral é 0,965.

Usando Monte Carlo:

1. Gerar U_1, \dots, U_n da $U(0,1)$;
2. Calcular $\frac{1}{n} \sum_{i=1}^n h(u_i)$

[Abrir arquivo aplicacao3.R](#)

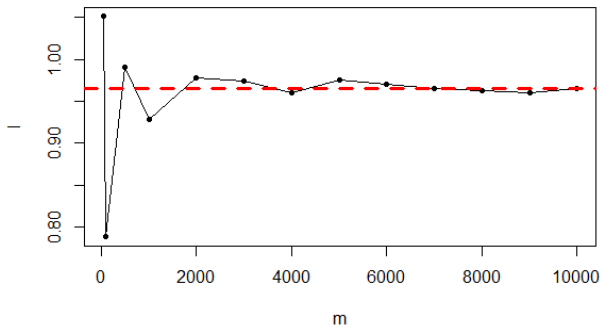


Figura: Analisando a convergência da integral.

Aplicação 3 (Monte Carlo)

(2) Queremos calcular $P(Z \leq t)$ para $Z \sim N(0, 1)$, quando $t = 1,96$:

$$P(Z \leq t) = \int_{-\infty}^t \frac{1}{\sqrt{2\pi}} \exp \left\{ -\frac{1}{2}z^2 \right\} dz = E [I(Z \leq t)] .$$

Sabemos que essa integral não pode ser resolvida analiticamente.

Usando Monte Carlo:

1. Gerar z_1, z_2, \dots, z_n da $N(0, 1)$;
2. Calcular $I = \frac{1}{n} \sum_{i=1}^n I(z_i \leq t)$.

[Abrir arquivo aplicacao3.R](#)

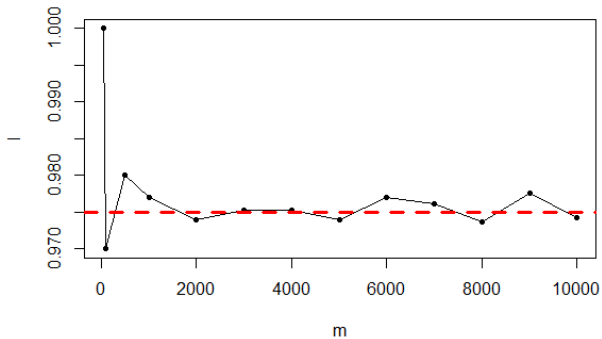


Figura: Analisando a convergência da integral.

Aplicação 4 (Estatística Espacial)

O **R** tem muitos pacotes para plotar mapas. Alguns pacotes utilizados, são: `sp`, `maptools`, `ggplot2`, `ggmap`, `maps` dentre outros.

```
> library(maps) #mapas simples, eixos, escala, cidades  
> library(maptools) #Ler ESRI shapefiles  
> library(ggplot)  
> library(ggmap) # acessa coordenadas do google maps  
> library(geoR)
```

Aplicação 4 (Estatística Espacial)

(1) Considerar a lista das unidades federativas do Brasil por produto interno bruto (PIB) nominal per capita referente ao ano de 2014, com valores em reais brasileiros, disponível em [Wikipedia](#)

- Plotar o mapa do Brazil por estados
- Considerar o PIB per capita em 2014 em reais
- Classificar os estados por classes de acordo com o PIB

[Abrir arquivo aplicacao4.R](#)

Aplicação 5 (Google Maps no R)

(1) Visualizar as localizações dos campus da Universidade Federal Fluminense em Niterói por meio gráfico.

- Construir um banco de dados com os endereços dos campus.
- Instalar o pacote ggmap.
- Descobrir as coordenadas geográficas a partir do endereço de cada campus.
- Utilizar o google maps.

Aplicação 5 (Google Maps no R)

- Banco de dados (nome, endereço, latitude, longitude)

```
> data.uff<- data.frame(UFF =NA, Endereco = NA, lat = NA,  
long = NA)
```

- Utilizar pacote ggmap

```
> require(ggmap)  
> ?geocode # retorna as coordenadas de cada ponto  
> ?qmap # plota o gráfico pelo google maps
```

[Abrir arquivo aplicacao5.R](#)

Aplicação 5 (Google Maps no R)

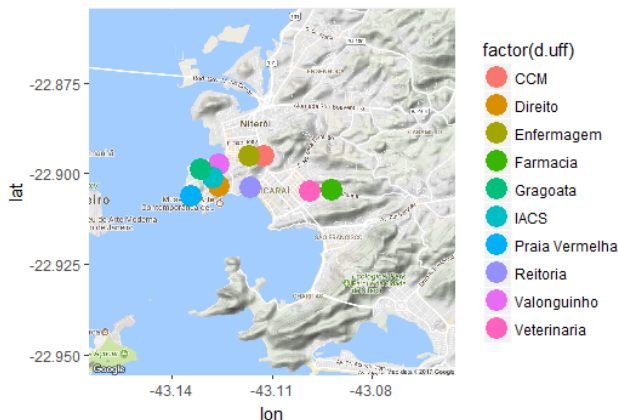


Figura: Localizações espaciais referente aos Campus da UFF em Niterói.

Aplicação 6 (Análise de Sobrevivência)

- Seja a variável T como o tempo de sobrevivência, isto é, o tempo até a ocorrência de um evento.
- T é considerada uma variável aleatória contínua e positiva

(1) Tempo de Sobrevivência (em meses) de 10 pacientes em diálise

Paciente (i)	1	2	3	4	5	6	7	8	9	10
Tempo (T_i)	22	6	12	43	23	10	35	18	36	29

Representar a trajetória do tempo de sobrevida dos pacientes graficamente.

[Abrir arquivo aplicacao6.R](#)

Aplicação 6 (Análise de Sobrevivência)

(2) Tempo de Sobrevivência (em meses) de 10 pacientes em diálise e a ocorrência de censura à direita.

Paciente (i)	1	2	3	4	5	6	7	8	9	10
Tempo (T_i)	22	6	12	43	23	10	35	18	36	29
Status (δ_i)	1	0	1	0	1	1	1	0	1	1

Representar a trajetória do tempo de sobrevida dos pacientes graficamente considerando o status do paciente.

- Censura refere-se à perda de informação decorrente de não se ter observado a **data de ocorrência** do desfecho.
- Na censura à direita sabe-se que o tempo entre o início da observação e o evento é maior do que o tempo de fato observado.

[Abrir arquivo aplicacao6.R](#)

Aplicação 6 (Análise de Sobrevivência)

(3) Tempo de Sobrevivência (em meses) de pacientes participantes da coorte aberta

Paciente (i)	1	2	3	4	5	6	7	8	9	10
Tempo Inicial (I)	0	15	0	5	10	0	0	12	3	15
Tempo Final (F)	22	21	12	48	33	10	35	30	39	44
Tempo Decorrido (T)	22	6	12	43	23	10	35	18	36	29
Status (δ_i)	1	0	1	0	1	1	1	0	1	1

Representar a trajetória do tempo de sobrevida dos pacientes graficamente considerando o status do paciente e a coorte aberta.

- Na coorte aberta o momento de entrada dos pacientes na coorte varia.

[Abrir arquivo aplicacao6.R](#)

Aplicação 6 (Análise de Sobrevivência)

- Seja a variável T como o tempo de sobrevivência, isto é, o tempo até a ocorrência de um evento.
- T é considerada uma variável aleatória contínua e positiva
- A função de sobrevivência $S(t)$, pode ser definida como a probabilidade de um indivíduo sobreviver por mais do que um determinado tempo t , ou por no mínimo um tempo igual a t :

$$S(t) = P(T > t)$$

$S(t)$ sempre será contínua a direita, não crescente.

Aplicação 6 (Análise de Sobrevivência)

(4) Dados de Aids (ipecc.csv)

Dados referente a uma amostra de 193 indivíduos diagnosticados com Aids (critério CDC 1993) provenientes de uma população de 1591 pacientes HIV positivos atendidos entre 1986 e 2000 no Ipec - Instituto de Pesquisa Clínica Evandro Chagas/Fiocruz.

- O tempo de sobrevivência é definido como o tempo entre o diagnóstico de Aids e o óbito ou censura.
- Coorte aberta com dados registrados no formato (I, F, δ) .
- O primeiro dia do estudo (20 de Outubro de 1987) foi chamado de dia 1, e o tempo de entrada e saída de cada indivíduo foi contado como o número de dias a partir dessa data.





[Abrir arquivo aplicacao6.R](#)

Aplicação 6 (Análise de Sobrevivência)

- Representar graficamente o tempo até o evento
- Estimar e representar graficamente as funções de sobrevivência a partir de um conjunto de dados (Estimador Kaplan-Meier)
- Estimar a função de sobrevivência de Kaplan-Meier estratificando por covariáveis.

Abrir arquivo `aplicacao6.R`

Referências Bibliográficas

-  CARVALHO, Marilia et al *Análise de Sobrevivência: Teoria e aplicações em saúde*. Editora FioCruz, 2^o Edição.
-  LOBO, Viviana G. R. *Notas de Aula*.
-  PEREIRA, João. B. *Minicurso de Introdução ao R*. Primeira Semana da Atuária, 2016, UFF.
-  ROBERT, C. e CASELLA, G. *Introducing Monte Carlo Methods with R*. Springer.