



TREINAMENTO FERRAMENTA R

Curso EAD FERRAMENTA R
Apostila com os comandos das vídeoaulas e
exercícios de fixação



Informações

CURSO: R – Teoria e Prática	DIREITOS: Uso da apostila deve ser autorizado, caso o estudante venha a utilizá-la fora do ambiente virtual do curso. Enviar e-mail para solicitação.
DATA CRIAÇÃO: 14/07/2014	REVISÃO: 3.0
DATA ÚLTIMA MODIFICAÇÃO: 05/09/2014	VERSÃO: 1.8
AUTOR Grimaldo Lopes de Oliveira	EMAIL: grimaldo_lopes@hotmail.com
SOBRE O PROFESSOR Grimaldo é baiano e soteropolitano. Fez graduação em Estatística pela Universidade Federal da Bahia e logo em seguida uma especialização na Área de Mineração de Dados/BI na Faculdade Visconde de Cairu. Após esta formação, "mergulhou" na área. Trabalha com Ferramentas Estatísticas a muito tempo SPSS, SAS, EPI-INFO e R, quando começou a extrair dados dos bancos de dados e passou a gerar análises estatísticas nestas ferramentas. Atualmente trabalha com banco de dados na PRODEB. Também é mestre em Gestão e Tecnologia Aplicadas à Educação (Gestec), pela Universidade do Estado da Bahia e editor do blog BI com Vatapá. Conheça Grimaldo: Blog: www.bicomvatapa.blogspot.com Site: www.aprendavirtual.com - (Aulas EAD) Facebook: www.facebook.com/groups/bicomvatapa/ Perfil: br.linkedin.com/in/grimaldo	

Sumário

INTRODUÇÃO	4
APRESENTAÇÃO	3
Sobre este documento.....	3
Recursos necessários	3
Atualizações deste documento	4
1. OBJETIVO	5
2. ENTENDENDO O R.....	6
3. INSTALAÇÃO DO R-STUDIO E R	7
4. PRIMEIROS PASSOS COM O R	9
5. USO DO HELP.....	10
6. OBJETOS NO R	11
VETORES.....	12
MATRIZES.....	16
DATA FRAMES.....	19
LISTAS.....	19
FUNÇÕES.....	20
7. WORKSPACE DO R(ÁREA DE TRABALHO)	20
8. PACOTE DO R	21
9. TRABALHANDO COM LEITURA DE ARQUIVOS EXTERNOS	24
10. SUMARIZANDO DADOS	28
SELECIONANDO DADOS	31
11. GRÁFICOS	33
HISTOGRAMAS.....	33
RAMO E FOLHA	34
BOX-PLOT.....	35
GRÁFICO DE DISPERSÃO	35

GRÁFICO DE BARRAS.....	37
GRÁFICO DE PIZZA OU SETORES.....	38
MELHORIAS NOS GRÁFICOS	38
12. PROGRAMAÇÃO	40
13. USO DA ESTATÍSTICA	44
14. EXERCÍCIOS FINAIS PARA EMISSÃO DO CERTIFICADO DE PARTICIPAÇÃO	60
14.1 Primeira Bateria de Exercícios	60
14.2 Segunda Bateria de Exercícios	61
14.3 Terceira Bateria de Exercícios	62
14.4 Quarta Bateria de Exercícios.....	62
CONSIDERAÇÕES IMPORTANTES	64
16. CONSIDERAÇÕES IMPORTANTES.....	64
16.1 Futuras atualizações.....	64
APÊNDICE.....	65
I. GLOSSÁRIO DE SIGLAS E TERMOS	65

Conheça o Professor Grimaldo Oliveira



Sou professor das pós-graduações das universidades **UNIFACS, CATÓLICA DO SALVADOR e ISL Wyden**. Mestre pela **Universidade do Estado da Bahia (UNEB)** no Curso de Mestrado Profissional Gestão e Tecnologias Aplicadas à Educação (GESTEC). Possuo Especialização em Análise de Sistemas pela Faculdade Visconde de Cairu e **Bacharelado em Estatística** pela Universidade Federal da Bahia. Atuo profissionalmente como consultor há mais de **15 anos nas áreas de Data Warehouse, Mineração de Dados, Ferramentas de Tomada de Decisão e Estatística**.

Idealizador do treinamento online **BI PRO** com + de 10 módulos contendo todas as disciplinas para formação completa na área de dados. Quem participa do **BI PRO** tem acesso gratuito: todos os meus cursos de dados da Udemy, + ebook **BI COMO DEVE SER - O Guia Definitivo**, espaço de mentoria para retirada de dúvidas, respostas das atividades. Acesse www.bipro.com.br

Autor do eBook **BI COMO DEVE SER - O Guia Definitivo**, com ele você poderá entender os conceitos e técnicas utilizados para o desenvolvimento de uma solução BI, tudo isso de forma objetiva e prática, com linguagem acessível tanto para técnicos quanto gestores e analista de negócio. Acesse www.bicomodeveser.com.br

Site de **cupons** do prof. Grimaldo, com desconto de todos os seus cursos de dados da Udemy, atualizado diariamente com diversas promoções, incluindo cursos gratuitos. Acesse <https://is.gd/CUPOMCURSOSPROFGRIMALDO>



Idealizador do Blog **BI COM VATAPÁ**, reúne informações diversas sobre a área de dados com detalhes sobre o mundo de Business Intelligence, Big Data, Ciência de dados, Mineração de dados e muitos outros. Acesse <http://bicomvatapa.blogspot.com/>

Introdução

"A persistência é o caminho do êxito."

Charlie Chaplin (1895-1976)

Apresentação

O curso de R, foi idealizado para permitir um rápido aprendizado prático ao aluno, através de uma interação com o professor a partir de vídeoaulas, que facilitam o entendimento dos diversos comandos do R. Durante o curso, o aluno terá que assistir as vídeoaulas e praticar diretamente no seu computador. Será necessário que o aluno tenha em sua máquina uma versão da ferramenta R instalada. O aluno terá um prazo máximo de curso de 3 meses para sua conclusão, mas poderá solicitar uma única prorrogação pelo tempo que desejar, entretanto o aluno será avisado por e-mail sobre o término do curso. Durante todo o curso, o aluno poderá retirar dúvidas com o professor e colegas, através do fórum de dúvidas ou por e-mail.

Sobre este documento

O objetivo deste documento é fornecer ao aluno, os passos necessários para que este aprenda a trabalhar com a ferramenta R na sua plenitude, onde detalhes mais específicos sobre cada comandos serão explicados, através de uma linguagem direta, facilitando a construção de programas.

Recursos necessários

Para acesso ao curso é necessário que o aluno tenha uma internet de rápido acesso para assistir aos vídeos, além de um leitor pdf da apostila do curso*, devem ser utilizados os navegadores de internet Firefox(Mozilla) ou Google Chrome com às versões mais atuais para acesso ao site de aulas EAD, além do software R* instalado na máquina do aluno.

****Estes softwares não são fornecidos pelo treinamento, o aluno é responsável pela aquisição destes, caso os mesmos sejam pagos.***

Atualizações deste documento

Futuras modificações poderão ocorrer no conteúdo deste documento em decorrência de possíveis ajustes na documentação do curso, sejam elas oriundas do professor ou devido a atualizações pertinentes que possam ser demandadas pelos alunos, dentro de critérios lógicos que não afetarão os objetivos para o qual este documento foi criado.

Aulas

“Foi um grande conselho o que ouvi certa vez, dado a um jovem: «Faça sempre o que tiver medo de fazer».”

Ralph Emerson (1803-1882)

As informações abaixo estão nas vídeoaulas do curso, acompanhe os vídeos e re-execute os procedimentos para fixação

1. Objetivo

O objetivo principal do curso é permitir que aspectos básicos com ênfase no entendimento de aspectos básicos da linguagem R, sua estrutura e a forma de operação sejam compreendidos. É importante frisar que nenhum método e/ou modelo estatístico em particular é discutido em detalhes seja em seus fundamentos ou alternativas para análises. O curso em questão não pretende ensinar estatística, será necessário que o aluno tenha conhecimento em estatística para entender as saídas de dados dos métodos que serão apresentados. Os métodos estatísticos são usados ao longo do texto simplesmente para ilustrar aspectos do uso da linguagem.

Embora, na maior parte do texto assume-se familiaridade com conceitos e métodos básicos de estatística, alguns tópicos especializados são usados em algumas sessões e, não sendo de interesse de leitor, podem ser ignorados, sem prejuízo ao acompanhamento das demais partes do texto. De todo modo, não será assumido nenhum conhecimento prévio do R. O curso foi preparado e estruturado para que desde o iniciante até o mais avançado em técnicas estatísticas aprenda a trabalhar com a ferramenta R. O material pode ser acompanhado utilizando o R instalado em outros sistemas operacionais, tal como Linux.

Portanto, pretendemos que você retire o melhor proveito do curso e aprenda de verdade a trabalhar com o R, então mãos a obra.

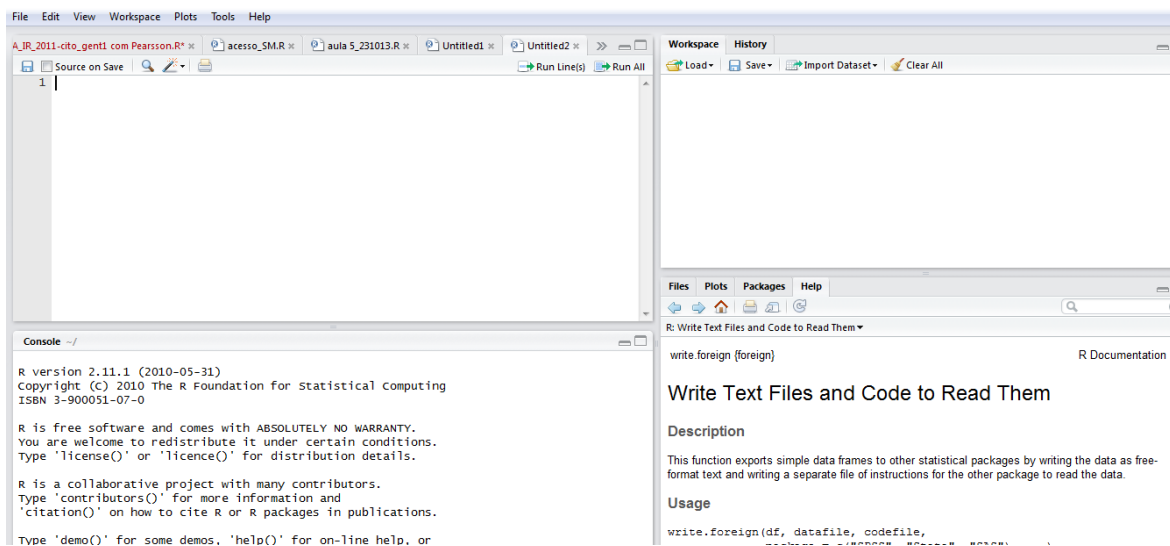
2. Entendendo o R

O R possui uma janela com algumas poucas opções para você trabalhar, note abaixo a tela de entrada do programa. As análises feitas no R são digitadas diretamente na linha de comandos. Na "**linha de comandos**" você irá digitar os comandos e funções que deseja utilizar. Para que você identifique aonde deve digitar um comando, basta que você encontre o sinal **>** (**sinal de maior**) indica o **prompt** e quer dizer que o R está pronto para receber comandos. Em alguns casos um **sinal de +** aparecerá no lugar do prompt, isso indica que ficou faltando algo na linha de comandos anterior (**isso acontece quando houve um erro, ou quando a finalização do comando só ocorrerá nas próximas linhas**). Caso esteja errado pressione **Esc** para retornar ao prompt normal **>** e sumir com o sinal de **+**.

Note o começo de cada linha com os comandos do R há um sinal do prompt, **>**, e em alguns casos um sinal de **+**, **não digite estes sinais**. Os comandos que você digita aparecem em **azul** e o output (saída dos métodos, comandos) do R aparece entre **[]**.

Sempre após digitar os comandos tecle **Enter** para que eles sejam executados.

Para facilitar nossos trabalhos, vamos trabalhar com o **R Studio**, que é um conjunto de ferramentas integradas para facilitar a produtividade do uso do R, pois facilita o trabalho com a ferramenta, permitindo um entendimento mais facilitado da execução dos métodos e comandos em 4 janelas múltiplas contendo **console**, **help de comandos e métodos**, **histórico de comandos**, **plot dos dados** e área contendo informações sobre a área de armazenamento de dados (**workspace**).



Tela de Entrada do R Studio

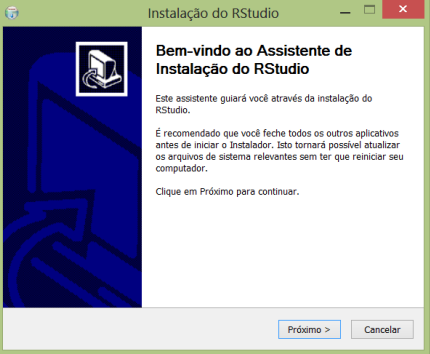
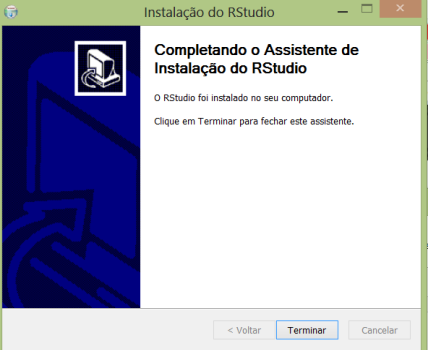
Download da ferramenta, você encontra no endereço:

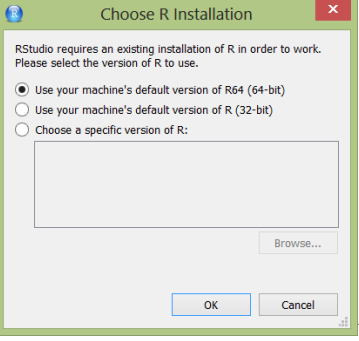
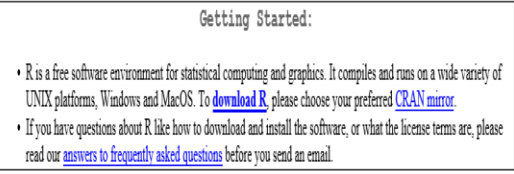
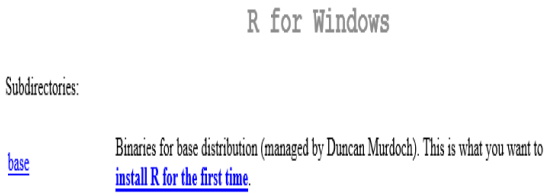
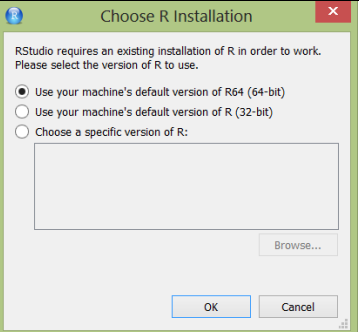
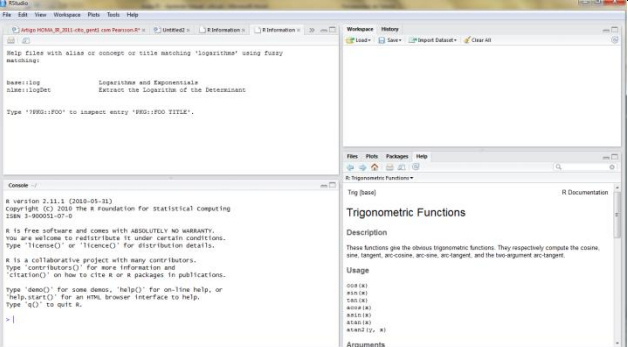
<http://www.rstudio.com/products/rstudio/>

3. Instalação do R-studio e R

A instalação serão duas etapas, iniciaremos através do R-studio e logo em seguida a ferramenta R. Esta separação existe, pois o R-studio é um programa receptor da ferramenta para facilitar o manuseio com o R, pois caso contrário teríamos que digitar todos os comandos, durante o curso você entenderá estas facilidades.

Passos:

1- Download do R-studio no site.	http://www.rstudio.com/products/rstudio/
2- Inicie o R-studio na sua máquina, primeira tela será. Instale como um programa qualquer do Windows, siga as instruções padrão.	
3- Ao final você deve chegar a esta tela.	

<p>4- O próximo passo é instalar a ferramenta R, você deve ir ao site do projeto R e baixar a última versão, mas existe uma forma direta de associar o R-studio ao R. Execute o R-studio e a tela ao lado aparecerá, solicitando que seja baixado a última versão do R.</p>	
<p>5- Você será direcionado ao endereço: www.r-project.org. Ao clicar em download R, você será direcionado para alguns dos servidores espalhados no mundo.</p>	
<p>6- Escolha sua primeira instalação e pronto, você irá instalar, igual a um programa qualquer do Windows, escolha sempre as opções default.</p>	
<p>7- Você deve retornar ao passo 4 e escolher o diretório onde está instalada a ferramenta R.</p>	
<p>8- Ao final esta deve ser a aparência do R-studio com a ferramenta R associada.</p>	

4. Primeiros passos com o R

Vamos iniciar nosso curso, trabalhando com os comandos básicos do R e no decorrer do curso, vamos nos aprofundar nos assuntos com um grau de dificuldade maior.

Vamos utilizar o R para avaliar algumas expressões aritméticas bem simples:

```
> 4+5+6
[1] 15
> 5+7*8+5
[1] 66
> 16/2-3
[1] 5
> 7*5**6
[1] 109375
```

Note: A execução dos comandos em azul e a saída vêm entre cochetes [].

A operação de potência é indicada por **. Alternativamente pode-se usar o símbolo ^, por exemplo 5*7^2 produziria o mesmo resultado que 5*7**2.

Outras funções matemáticas:

```
> sqrt (64)
[1] 8
> sin(4.556765)
[1] -0.987915
> log(10)
[1] 2.302585
```

Aqui está uma lista resumida de algumas funções aritméticas no R:

sqrt()	raiz quadrada
abs()	valor absoluto (positivo)
sin() cos() tan()	funções trigonométricas
asin() acos() atan()	funções trigonométricas inversas
sinh() cosh() tanh()	funções hiperbólicas
asinh() acosh() atanh()	funções hiperbólicas inversas
exp() log()	exponencial e logaritmo natural
log10() log2()	logaritmo base-10 e base-2
gamma()	função Gamma de Euler
factorial	fatorial (n!)
choose()	número de combinações ($\frac{n!}{x!(n-x)!}$)
combn()	todas conjuntos gerados pela combinações de certo número de elementos

As expressões acima podem ser combinadas de diversas formas, como no exemplo:

```
> sqrt(64)/sin(3.5676)
[1] -19.35929
```

5. Uso do Help

Em diversas situações você desejará fazer uma determinada análise cujo nome da função você ainda não conhece. Nestes casos existem três maneiras para descobrir uma função que faça aquilo que você deseja.

A primeira é pesquisar dentro do R usando palavras chave usando a função **help.search()**.

Por exemplo, vamos tentar descobrir como se calcula o seno no R.

```
> help.search("seno")
```

Veja que o R não encontrou nenhuma função, pois o R é desenvolvido em língua inglesa, portanto, precisamos buscar ajuda usando palavras em inglês.

```
> help.search("sin")
```

Com esse comando o R irá procurar, dentro dos arquivos de help, possíveis funções para calcular o seno. Uma janela irá se abrir contendo possíveis funções. Você também pode substituir por apenas ?? (duas interrogações)

```
> ??sin
```

Também é possível buscar ajuda na internet, dentro do próprio R, buscando diretamente no site do R, com a função `RSiteSearch()`

```
> RSiteSearch("sin")
```

Esta execução apenas funcionará se o seu computador estiver conectado à internet.

Para ver os arquivos de ajuda do R use o comando `help(nome.da.função)` ou `? nome.da.função`. Por exemplo, vamos ver o help da função `log`:

```
> help(log)
```

Existem alguns Arquivos na web com resumo dos comandos da ferramenta R, são os chamados cartões de referência, abaixo alguns link's para que você faça download.

- <http://www.leg.ufpr.br/~paulojus/misc/refcard.pdf> Criado por Jonathan Baron

-

6. Objetos no R

O R é uma linguagem orientada à objetos, ou seja, todos os dados que trabalharemos serão armazenados : variáveis, dados, matrizes, funções, etc. tudo será armazenado na memória do computador na forma de objetos. Veja abaixo como distinguir cada um deles, claro que você irá entendendo aos poucos, iremos trabalhando com os objetos e você naturalmente irá compreender:

Vetores:

É quando um objeto armazena uma sequência de valores podendo ser numéricos ou de caracteres (letras, palavras).

Matrizes:

É caracterizado como uma coleção de vetores em linhas e colunas, como na matemática, onde no segundo grau se estuda o assunto matrizes. Todos os vetores devem ser do mesmo tipo (numérico ou de caracteres).

Data Frame:

O exatamente o mesmo que uma matriz, podendo, entretanto aceitar vetores de tipos diferentes (numérico e caracteres). Geralmente quando trabalhamos na ferramenta R nós guardamos os dados em objetos do tipo data frame, pois sempre temos variáveis numéricas e variáveis categóricas (por exemplo, peso de uma pessoa e seu nome).

Listas:

É composto por um conjunto de vetores, data frames ou de matrizes. É a forma que a maioria das funções retorna os resultados.

Funções:

As funções é o tipo de objeto dos mais utilizados, pois seu uso é característico das chamadas análises estatísticas, podendo fazer diversos cálculos.

Características dos Objetos:

Para que você compreenda de forma fácil, como um determinado valor é armazenado na ferramenta R é importante entender as atribuições dos objetos.

No exemplo abaixo, note que atribuiremos um valor em um objeto chamado de dados usando o símbolo <- o valor do seno de (9.7979).

```
> dados<- sin(9.7979)
> dados
[1] -0.3645244
```

A saída de dados **[1] -0.3645244** apresenta apenas 1 resultado, o que informa que este objeto só tem armazenado 1 informação, caracterizando um vetor de apenas 1 informação.

Você pode substituir o símbolo de <- pelo = sem nenhum problema.

```
> dados=sin(9.7979)
> dados
[1] -0.3645244
```

A operação acima dizemos que o objeto dados “**recebe**” o valor do seno(9,7979), isso é retirado de uma expressão muito conhecida no meio de computação o *get* (recebe).

Os objetos apresentam seus valores apenas quando digitados na linha de comando, enquanto estão “recebendo” dados nada é exibido.

```
> var1=656*767
> var2= sqrt(7687)
> var3=var2-var1
> var3
[1] -503064.3
```

Importante Saber!

- Todo objeto deve começar por um caracter alfabético;
- O objeto não pode utilizar palavras reservadas da ferramenta R como: **break, else, for, function, IF, in, next, repeat, while, FALSE, Inf, NA, NaN, NULL, TRUE, etc.**

Valores faltantes e especiais

Toda vez que trabalharmos com objetos e por algum motivo algum dado esteja faltando ou incorreto, é possível que você se depare com os resultados abaixo:

- **NA: Not Available**, constitui dados faltantes.
- **NaN: Not a Number**, constitui um valor que não é representável por um número.
- **Inf** e **-Inf**: mais ou menos infinito.

```
> c(99,0,-3)/0
[1] Inf NaN -Inf
```

Vetores

Para trabalhar com vetores, vamos utilizar a função **C** que significa **concatenar**, facilitando a construção de vetores.

Vamos trabalhar com algumas funções utilizando vetores.

Crie o objeto **cidade**, contendo cidades do nordeste do Brasil.

```
> cidade=c("salvador","recife","pernambuco","maceio","natal")
> length(cidade)
[1] 5
```

- A função **length** informa a quantidade de dados contida no vetor **cidade**.

```
> populacao=c(15000,20000,30000,17000,23000)
> sum(populacao)
[1] 105000
> max(populacao)
[1] 30000
> min(populacao)
[1] 15000
> numero_elementos.cidade=length(cidade)
> numero_elementos.cidade
[1] 5
> media.cidade=sum(populacao)/numero_elementos.cidade
> media.cidade
[1] 21000
```

- Acima, geramos a média da população das cinco cidades, para isso criamos um vetor população hipotético, armazenando a quantidade de habitantes. Em seguida utilizamos a função **sum** para somar o total de habitantes, a função **max** para saber a cidade que tem a maior população, a função **min** para saber quem tem a menor população, criamos um novo objeto chamado **numero_elementos** (note que o vetores cidade deve ser colocado para informar de qual vetor este se referencia) que armazena o total de elementos que a função **length** informa, criamos o objeto **media** que guarda a média da população das cinco cidades.

Para facilitar tudo poderíamos utilizar a função média da ferramenta R.

```
> mean(populacao)
[1] 21000
```

Outros comandos

- Repete o número 1 quarenta vezes.

```
> num1=(1:40)
> num1
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27
[28] 28 29 30 31 32 33 34 35 36 37 38 39 40
```

- Executa a sequência de 10 até 100, saltando de 5 em 5.

```
> num2=seq(10,100,by=5)
> num2
[1] 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90
95 100
```

- Comando que exibe a informação de TRUE ou FALSE caso o valor contido no vetor **num3** seja maior que 40.

```
> num3=(num2>40)
> num3
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE
TRUE TRUE
[14] TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

- Comando que exibe a informação dos valores maiores que quarenta, caso o valor contido no vetor **num2** seja maior que 40.

```
> num2[num2>40]
[1] 45 50 55 60 65 70 75 80 85 90 95 100
```

- Comando repete o número 10 cinquenta vezes.

```
> num4=rep(10, 50)
> num4
[1] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10
[28] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10
10
```

- Comando que repete o número 10 quatro vezes e o número 12 sete vezes.

```
> num5=rep(c(10,12),c(4,7))
> num5
[1] 10 10 10 10 12 12 12 12 12 12 12
```

- Podemos acrescentar um determinado dado ao vetor no início dos dados.

```
num6=c(7,num5)
> num6
[1] 7 10 10 10 10 12 12 12 12 12 12 12
```

- Podemos acrescentar um determinado dado ao vetor no final dos dados.

```
> num7=c(num5, 78)
> num7
[1] 10 10 10 10 12 12 12 12 12 12 12 78
```

- Podemos mesclar um determinado dado caracter ao vetor, mesmo que este só tenha dados numéricos, mas este vetor passará de numérico para um vetor caracter.

```
> num_palavra=c("bahia", num5)
> num_palavra
[1] "bahia" "10" "10" "10" "10" "12" "12" "12"
"12" "12"
[11] "12" "12"
```

- Podemos realizar somatórios, produtórios ou qualquer outro cálculo sequencialmente com o vetor.

```
> num8=num6 + 2:13
> num8
[1] 9 13 14 15 16 19 20 21 22 23 24 25
```

- Podemos repetir determinados elementos uma quantidade especifica de vezes.

```
> rep(5:9, each = 3)
[1] 5 5 5 6 6 6 7 7 7 8 8 8 9 9 9
```

- Podemos repetir uma sequência de dados uma quantidade especifica de vezes.

```
> rep(5:9,3)
[1] 5 6 7 8 9 5 6 7 8 9 5 6 7 8 9
```

- Exibe apenas os quatro primeiros elementos de um vetor.

```
> num6[1:4]
[1] 7 10 10 10
```

- Exibe uma amostra extraída de uma distribuição normal de média 400 e desvio padrão 30.

```
> amostra <- round(rnorm(10, m = 400, sd = 30))
> amostra
[1] 400 433 340 416 397 383 389 379 399 365
```

- Exibe os dados do vetor amostra ao contrário.

```
> rev(amostra)
[1] 365 399 379 389 383 397 416 340 433 400
```

- Exibe os dados do vetor amostra em order ascendente.

```
> sort(amostra)
[1] 340 365 379 383 389 397 399 400 416 433
```

- Exibe os dados do vetor amostra com os seus respectivos rank's.

```
> order(amostra)
[1] 3 10 8 6 7 5 9 1 4 2
```

- Retorna as posições específicas do vetor que obedecem a um determinado critério.

```
> which(amostra<400)
[1] 3 5 6 7 8 9 10
```

- Para a eliminação de um determinado dado dentro de um vetor, basta colocar a posição deste entre parênteses.

```
> amostra[-2]  
[1] 400 340 416 397 383 389 379 399 365
```



OBS: SEMPRE QUE SE DESEJAR ACESSAR UM DETERMINADO VALOR EM UM VETOR UTILIZE COCHETES[]. PARA ACESSAR MAIS DE UM VALOR USE "C" PARA CONCATENAR DENTRO DOS COLCHETES [C(4,7,...)]

Matrizes

Matrizes são objetos criados a partir da utilização de elementos de um vetor em linhas e colunas. A seleção de elementos, as chamadas **submatrizes**, são realizadas usando o símbolo de `[,]`. Perceba uma diferença entre vetores e matrizes que consiste na separação entre seus elementos em linhas e colunas utilizando a vírgula, antes da vírgula indica-se a(s) linha(s) e depois a(s) coluna(s) a serem selecionadas. Cada um destes componentes da matriz é um vetor de nomes.

Vamos trabalhar com algumas funções utilizando matrizes.

- Para criar uma matriz, que inicialmente carrega dados *default*, basta utilizar o comando **matrix**. O primeiro argumento da função `matrix` diz respeito à quantidade de dados, a segunda opção informará a quantidade de colunas em uma determinada matriz, note que automaticamente é estabelecida a quantidade de linhas, sempre deve ser múltiplo a **quantidade de dados/número de colunas**.

```
> amostra2=matrix(1:10, ncol = 5)  
> amostra2  
      [,1] [,2] [,3] [,4] [,5]  
[1,]     1     2     3     4     5  
[2,]     6     7     8     9    10
```

- Caso deseje o contrário, que a disposição dos elementos seja registrada nas colunas, utilize o argumento **byrow**.

```
> amostra3=matrix(1:10, ncol = 5, byrow = T)  
> amostra3  
      [,1] [,2] [,3] [,4] [,5]  
[1,]     1     2     3     4     5  
[2,]     6     7     8     9    10
```

- Caso deseje saber a quantidade de elementos em uma determinada matriz.

```
> length(amostra2)  
[1] 10
```

- Quantas linhas e colunas há em uma determinada matriz.

```
> dim(amostra2)
[1] 2 5
```

- Quantas linhas há em uma determinada matriz.

```
> nrow(amostra2)
[1] 2
```

- Quantas colunas há em uma determinada matriz.

```
> ncol(amostra2)
[1] 5
```

- Caso você deseje acessar um determinado elemento, basta colocar entre [] o número da linha e da coluna.

```
> amostra2[2,3]
[1] 6
```

- Quais são os elementos de uma determinada coluna.

```
> amostra2[,3]
[1] 5 6
```

- Quais são os elementos de uma determinada linha.

```
> amostra2[1,]
[1] 1 3 5 7 9
```

- Caso você deseje acessar ao mesmo tempo linhas e colunas. Note que os valores são reorganizados dinamicamente, mas a matriz **amostra2** permanece de forma intacta.

```
> amostra2[1:2,2:4]
      [,1] [,2] [,3]
[1,]    3    5    7
[2,]    4    6    8
```

- Caso você deseje alterar as nomenclaturas padrão da matriz, isso pode ser realizado através do comando **dimnames**.

```
> dimnames(amostra2) <- list(c("Linha1", "Linha2"), c("Coluna1",
"Coluna2", "Coluna3", "Coluna4", "coluna5"))
> amostra2
      Coluna1 Coluna2 Coluna3 Coluna4 coluna5
Linha1      1      3      5      7      9
Linha2      2      4      6      8     10
```

- Caso você deseje incluir determinados valores em uma matriz com duas colunas, utilize o comando **cbind**. O número maior de elementos na entrada de argumentos **cbind(1 argumento, 2 argumento)** definirá a quantidade de elementos.

```
> amostra3 <- cbind(1:3, 10:15)
> amostra3
      [,1] [,2]
[1,]     1    10
[2,]     2    11
[3,]     3    12
[4,]     1    13
[5,]     2    14
[6,]     3    15
```

- Soma dos elementos das linhas da matriz, funções **apply** ou **rowSums**.

```
> apply(amostra2, 1, sum)
Linha1 Linha2
     25     30
> rowSums(amostra2)
Linha1 Linha2
     25     30
```

- Soma dos elementos das colunas da matriz, funções **apply** ou **colSums**.

```
> apply(amostra2, 2, sum)
Coluna1 Coluna2 Coluna3 Coluna4 Coluna5
       3       7      11      15      19
> colSums(amostra2)
Coluna1 Coluna2 Coluna3 Coluna4 Coluna5
       3       7      11      15      19
```

- Média dos elementos da coluna de uma matriz.

```
> colMeans(amostra2)
[1] 1.5 3.5 5.5 7.5 9.5
```

- Média dos elementos da linha de uma matriz.

```
> rowMeans(amostra2)
[1] 5 6
```

- Soma dos elementos de uma matriz mais outra matriz.

```
> amostra2+amostra3
      Coluna1 Coluna2 Coluna3 Coluna4 Coluna5
Linha1       2       5       8      11      14
Linha2       8      11      14      17      20
```

- Produto entre os elementos de uma matriz por outra matriz.

```
> amostra2*amostra3
      Coluna1 Coluna2 Coluna3 Coluna4 Coluna5
Linha1       1       6      15      28      45
Linha2      12      28      48      72     100
```

- Subtração entre os elementos de uma matriz pela outra matriz.

```
> amostra3-amostra2
      Coluna1 Coluna2 Coluna3 Coluna4 Coluna5
```

Linha1	0	-1	-2	-3	-4
Linha2	4	3	2	1	0

Data Frames

A diferença básica entre data frames e matrizes é que utilizando o objeto matriz, você só pode ter colunas do mesmo tipo (numéricas, caracteres), no caso de data frames, você pode ter um tipo diferente para cada coluna.

Vamos ver.

- Criamos dois vetores, um numérico e outro caractere. A única forma de juntar é através de data frame.

```
> d1=c(4,5,6,7,8,9)
> d2=c("a","b","c","d","e","f")
> data.frame(d1,d2)
  d1 d2
1  4  a
2  5  b
3  6  c
4  7  d
5  8  e
6  9  f
```

Listas

A lista é caracterizada por ser genérica e pode armazenar diversos formatos diferentes de dados em um único objeto, é um conjunto de objetos.

Vejamos:

- Criamos uma lista com três diferentes tipos.

```
> lista1 <- list(A = 1:5, B = "SALVADOR CAPITAL DA FELICIDADE", C =
matrix(1:4,ncol = 4))
> lista1
$A
[1] 1 2 3 4 5

$B
[1] "SALVADOR CAPITAL DA FELICIDADE"
```

```
$C  
[1,] [,1] [,2] [,3] [,4]  
      1  2  3  4
```

- Para listar um objeto que pertence a uma lista, basta colocar entre colchetes.

```
> lista1[2]
```

```
$B  
[1] "SALVADOR CAPITAL DA FELICIDADE"
```

- Para listar os elementos que estão dentro de um determinado objeto que pertence a uma lista, basta colocar entre duplos colchetes.

```
> lista1[[2]]  
[1] "SALVADOR CAPITAL DA FELICIDADE"
```

Funções

É o que é mais comum utilizado na ferramenta R, o uso de funções é através de seu nome, basta digitar para saber seu conteúdo. É um programa criado para resolver uma determinada tarefa.

Ex: plot

```
> plot  
function (x, y, ...)  
{  
  if (is.function(x) && is.null(attr(x, "class"))) {  
    if (missing(y))  
      y <- NULL  
    hasylab <- function(...) !all(is.na(pmatch(names(list(...)),  
      "ylab")))  
    if (hasylab(...))  
      plot.function(x, y, ...)  
    else plot.function(x, y, ylab = paste(deparse(substitute(x)),  
      "(x)"), ...)  
  }  
  else useMethod("plot")  
}  
<environment: namespace:graphics>
```

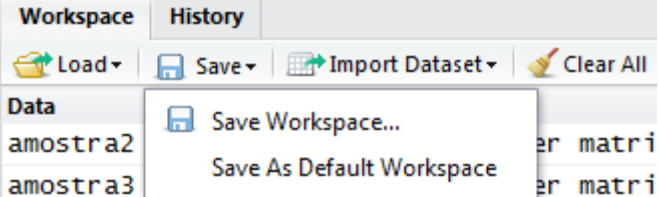
7. Workspace do R(Área de Trabalho)

Sempre que você abrir a ferramenta R para trabalhar, esta armazena temporariamente todas as saídas de objetos e seus conteúdos, é a chamada **Workspace ou Área de Trabalho**. Para que você mantenha tudo que foi feita na sessão que está trabalhando, você deve salvá-la. Você pode digitar o comando **save.image** ou salvar diretamente no R

Studio, que é a forma mais fácil. Ao salvar a workspace, o arquivo terá um ícone do R na pasta que você gravou, a partir do qual você irá abrir o R. Vamos praticar as duas formas:

- 1) Digitando o comando:

```
> save.image("E:/Grimaldo/Aprenda Virtual/Curso EAD/Aula R/Curso R.RData")
```
- 2) Acesse o menu do R studio.

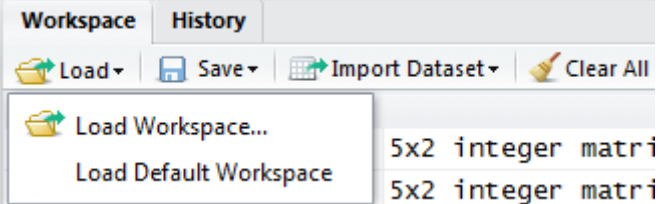
1) Basta clicar em Save-> Save workspace . Escolha o diretório e dê um nome ao arquivo.	
---	--

Pronto, tudo que você fez nesta sessão não será mais perdido.

Quando quiser utilizar todos os objetos que foram salvos nesta sessão, basta utilizar o comando **Load** ou através do menu, escolha a opção **Load**.

- 1) Utilizando a digitação do comando **Load**.

```
load("E:/Grimaldo/Aprenda Virtual/Curso EAD/Aula R/Curso R.RData")
```
- 2) Acesse o menu do R studio.

1) Basta clicar em Load -> Load Workspace . Escolha o diretório e clique no nome do arquivo.	
--	--

8. Pacote do R

O programa R é composto por uma quantidade mínima de estruturas, pois o entendimento é que você quando desejasse uma estrutura mais avançada, poderá ir até o site do Projeto R e baixar o programa, os chamados **pacotes**.

O R é dividido em 3 partes:

- 1) O R-base, que contém as funções principais disponíveis quando iniciamos o programa.
- 2) Possui os **pacotes recomendados** (recommended packages) que são instalados junto com o R-base mas não são carregados quando iniciamos o programa. Por exemplo o pacote MASS – e há vários outros. Para usar as funções que estão dentro dos pacotes deve-se carregá-los antes com o comando **library()**. Por exemplo o comando **library(MASS)** carrega o pacote MASS.
- 3) E por último os **pacotes contribuídos** (contributed packages) não são instalados junto com o R-base. Estes pacotes disponíveis na página do R são pacotes oficiais. Estes pacotes adicionais fornecem funcionalidades específicas e para serem utilizados devem ser copiados, instalados e carregados. Para ver a lista deste pacotes com uma descrição de cada um deles acesse a página do R e siga os links para CRAN e Package Sources.

Antes que você instale um pacote você pode ver se ele já está instalado/disponível no seu computador. Para isto digite o comando:

```
> require(NOME_DO_PACOTE)
```

Após a execução do comando, caso o pacote retorne a informação **T** é porque o pacote **já está instalado/disponível**. Se retornar a informação **F**, você deverá instalar o pacote, vejamos como fazer esta tarefa:

A instalação e uso dos pacotes vai depender:

- De qual sistema operacional você está utilizando.
- Você deve estar conectado a internet.
- Devesse digita o comando **install.packages(<nome_pacote>)** e instalá-lo.
- Para confirmar cheque através dos comandos **require** ou **library**, caso prefira verifique no menu package se o mesmo foi instalado.

O site da ferramenta R que contém a lista de pacotes é: <http://cran.rstudio.com/>

Para instalar o pacote no **sistema windows**, utilize:

```
> install.packages(<nome_pacote>)
```

Vamos instalar o pacote do **Algoritmo Bethel**: [bethel](#)

```
> install.packages('bethel')
```

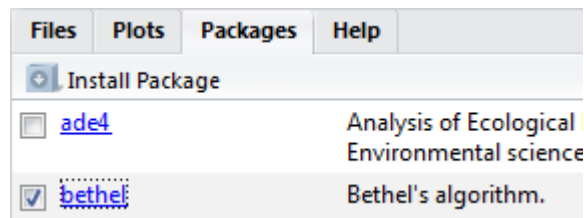
Você deve ter uma mensagem parecida com essa:

```
> install.packages('bethel')
Warning in install.packages :
  argument 'lib' is missing: using 'C:/Users/lucivalda/Documents/R/win-
library/2.11'
Warning in install.packages :
  cannot open: HTTP status was '404 Not Found'
Warning in install.packages :
  cannot open: HTTP status was '404 Not Found'
Warning in install.packages :
  unable to access index for repository
http://www.stats.ox.ac.uk/pub/Rwin/bin/windows/contrib/2.11
trying URL
'http://cran.cnr.Berkeley.edu/bin/windows/contrib/2.11/bethel_0.2.zip'
Content type 'application/zip' length 133530 bytes (130 kb)
opened URL
downloaded 130 kb
```

package 'bethel' successfully unpacked and MD5 sums checked

Note que a ferramenta tentou de várias formas baixar o pacote bethel, caso apareça a mensagem **successfully unpacked and MD5 sums checked** é que tudo ocorreu bem.

No R-studio verifique na aba Package se o pacote foi instalado, veja abaixo:



Todo o pacote tem as chamadas citações, para que você possa colocar em seus projetos de pesquisa, basta executar o comando **citation()**.

```
> citation("bethel")
To cite package 'bethel' in publications use:
```

```
Michele De Meo (2009). bethel: Bethel's algorithm.. R package version
0.2.
http://CRAN.R-project.org/package=bethel
```

A BibTeX entry for LaTeX users is

```
@Manual{,
  title = {bethel: Bethel's algorithm.},
  author = {Michele De Meo},
  year = {2009},
  note = {R package version 0.2},
  url = {http://CRAN.R-project.org/package=bethel},
}
```

ATTENTION: This citation information has been auto-generated from the package DESCRIPTION file and may need manual editing, see `'help("citation")'`

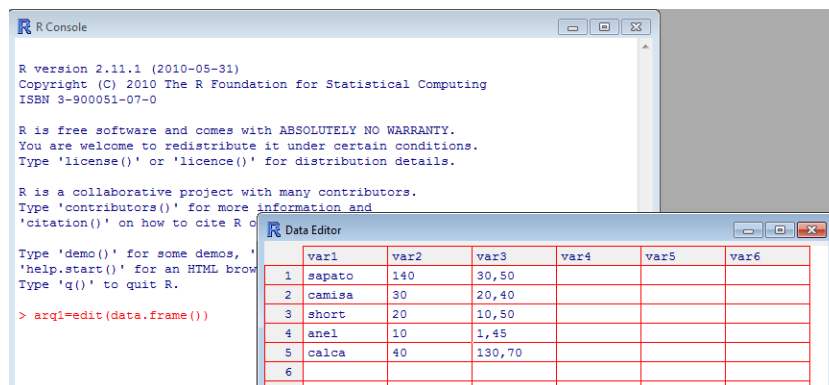
9. Trabalhando com leitura de arquivos externos

Trabalharemos com uma das tarefas mais importante que existem na ferramenta R, a **importação, exportação, alteração** de dados oriundos de arquivos externos. O mais comum é a utilização e leitura de arquivos no formato texto (**.txt**), mas é possível ler dados de quaisquer formatos, basta que se utilize e que existam funções dentro do R para realizar esta tarefa. Os mais comuns são o uso das funções:

- **edit(data.frame())** – Abre uma planilha de dados que permite que você digite informações. Ao terminar de digitar, basta sair da planilha através do botão <QUIT>. Caso deseje fazer correções ou alterar algum dado, utilize o comando **fix()**. Tudo será gravado em um data frame, ou seja, tipos diferentes de dados.



OBS: ESTA FUNÇÃO SÓ É PERMITIDA DENTRO DA FERRAMENTA R, NO R-STUDIO NÃO É POSSÍVEL EXECUTÁ-LA.

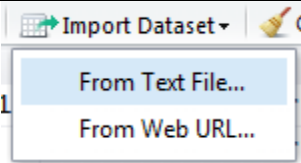
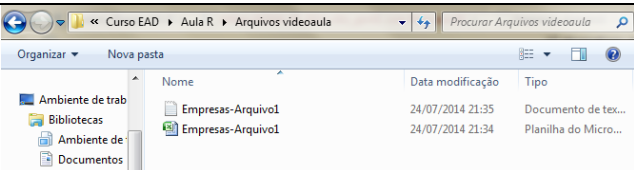
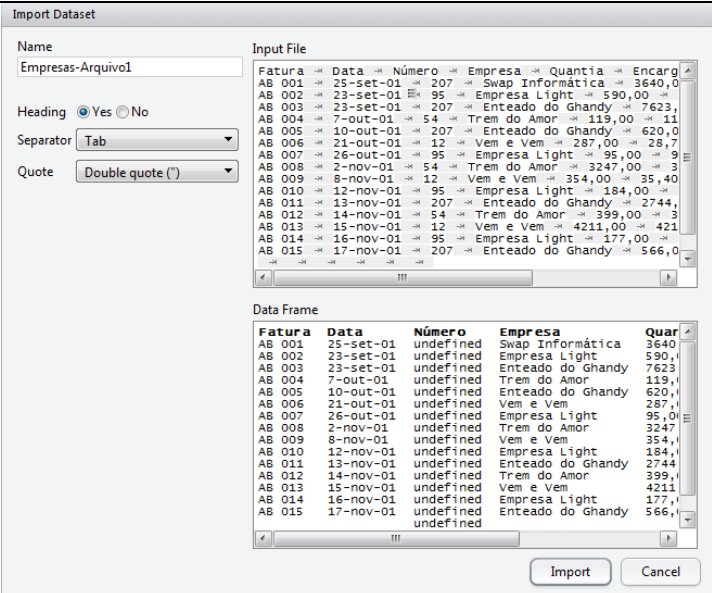


> arq1

```
      var1 var2   var3
1 sapato  140  30,50
2 camisa   30  20,40
3 short    20  10,50
4 anel     10   1,45
5 calça    40 130,70
```

- **read.table("nome_do_arquivo.txt")** – através deste comando você pode ler arquivos no formato texto. Caso você tenha o arquivo no formato Excel, pode solicitar que seja salvo no formato texto. O comando **scan()** também pode ser utilizado para esta tarefa, mas é bastante primitivo e está em desuso.

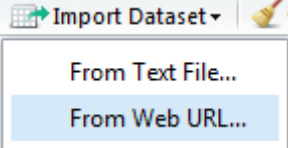
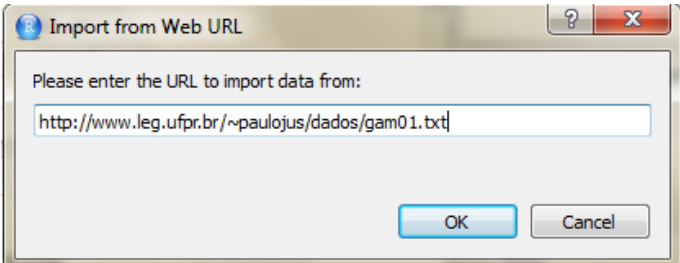
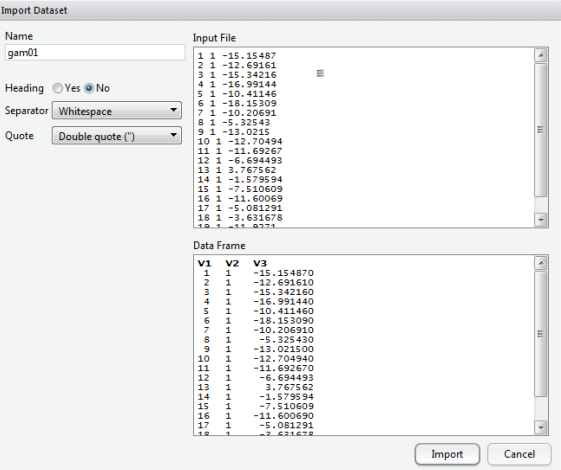
- 1) Vamos abrir o arquivo **Empresas-Arquivo1.xlsx** e convertê-lo para o formato texto separado por tabulações, gravando o arquivo **Empresas-Arquivo1.txt**.
- 2) Depois copie o seu arquivo para sua área de trabalho do R (Working directory of R), no meu computador é : **E:/Grimaldo/Aprenda Virtual/Curso EAD/Aula R/Empresas-Arquivo1.txt**
- 3) A maneira mais fácil de ler qualquer arquivo no R-Studio neste formato, é utilizando o menu **import dataset**, veja:

1) Basta clicar em Import Dataset -> From Text File Escolha o diretório e clique no nome do arquivo.	
2) Abrirá outra janela, solicitando que você escolha o nome do arquivo, no nosso caso Empresas-Arquivo1.txt	
3) Automaticamente o R-Studio informará a composição do arquivo, confirme que é tabulado e com cabeçalho .	
4) É então criado o data frame Empresa-Arquivo1 . A importação pode ser realizada através do comando read.delim()	<pre>`Empresa-Arquivo1` <- read.delim("E:/Grimaldo/Aprenda Virtual/Curso EAD/Aula R/Empresas-Arquivo1.txt")</pre>

- 4) você pode ler um arquivo direto da internet, basta utilizar no menu import dataset a opção **From Web URL**. Vamos utilizar a URL abaixo como exemplo:

<http://www.leg.ufpr.br/~paulojus/dados/gam01.txt>

A maneira mais fácil de ler qualquer arquivo no R-Studio neste formato, é utilizando o menu **import dataset**, veja:

<p>1) Basta clicar em Import Dataset -> From Web URL Digite a URL onde está o arquivo que será carregado.</p>	
<p>2) Abrirá outra janela, solicitando que você escolha o nome do arquivo, no nosso exemplo o arquivo gam01.txt</p>	
<p>3) Automaticamente o arquivo é reconhecido e Solicitamos que seja importado.</p>	

Você pode ler diversos formatos, o pacote **foreign** tem todos os objetos necessários a leitura de dados, veja abaixo:

Digite: **help(foreign)**

read.arff	Read Data from ARFF Files
read.dbf	Read a DBF File
read.dta	Read Stata Binary Files
read.epiinfo	Read Epi Info Data Files
read.mtp	Read a Minitab Portable Worksheet
read.octave	Read Octave Text Data Files
read.S	Read an S3 Binary or data.dump File
read.spss	Read an SPSS Data File
read.ssd	Obtain a Data Frame from a SAS Permanent Dataset, via read.xport
read.systat	Obtain a Data Frame from a Systat File
read.xport	Read a SAS XPORT Format Library

Dentro da ferramenta R, existem uma série de dados já pré-formatados, para conhecer basta digitar o comando **data()**, aparecerá uma lista de datasets disponíveis que você pode treinar da forma que desejar. Abaixo alguns exemplos:

AirPassengers	Monthly Airline Passenger Numbers 1949-1960
BJsales	Sales Data with Leading Indicator
BJsales.lead (BJsales)	Sales Data with Leading Indicator
BOD	Biochemical Oxygen Demand
CO2	Carbon Dioxide Uptake in Grass Plants
ChickWeight	Weight versus age of chicks on different diets
DNase	Elisa assay of DNase
EuStockMarkets	Daily Closing Prices of Major European Stock Indices, 1991-1998
Formaldehyde	Determination of Formaldehyde
HairEyeColor	Hair and Eye Color of Statistics Students
Harman23.cor	Harman Example 2.3
Harman74.cor	Harman Example 7.4
Indometh	Pharmacokinetics of Indomethicin
InsectSprays	Efectiveness of Insect Sprays
JohnsonJohnso	Quarterly Earnings per Johnson & Johnson Share
LakeHuron	Level of Lake Huron 1875-1972
LifeCycleSavings	Intercountry Life-Cycle Savings Data

Para a leitura de dados em banco de dados, é necessário a instalação do pacote **RODBC**, basta digitar o comando **install.packages('RODBC')**, que será instalada as bibliotecas de leituras ODBC, igual a qualquer sistema de banco de dados.

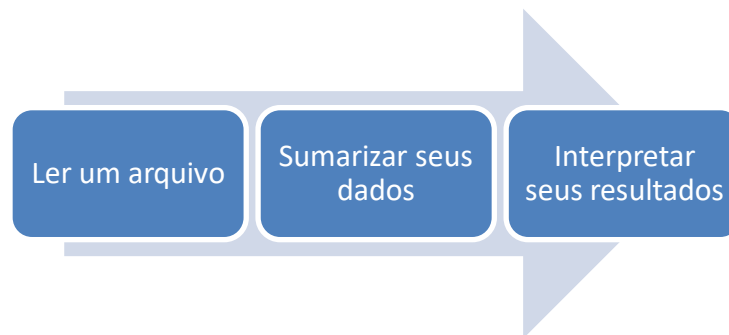
Vejamos um exemplo de leitura de dados de uma base de dados qualquer:

```
myconn <-odbcConnect("fornecedor", uid="fornec", pwd="34rt#yt")
vendedorbd <- sqlFetch(myconn, Vendedor)
produtosbd <- sqlQuery(myconn, "select * from Produto where Produto.totestoque>20")
close(myconn)
```

Duas tabelas estão sendo carregadas **Vendedor** e **Produtos**, sendo que a segunda tabela há uma seleção de dados.

10. Sumarizando Dados

Iniciaremos uma das atividades mais comuns na ferramenta R, a sumarização de um conjunto de dados, corriqueiramente faremos a operação básica de :



Portanto , vamos iniciar a leitura de um arquivo e em seguida utilizaremos os comandos mais comuns da ferramenta R. A forma mais simples de descrever quantitativamente observações é agrupá-las em categorias e contar quantas observações pertence a cada categoria.

No R a forma mais direta de obter **contagens (frequências)** é através da função **table**. Vamos ler o arquivo **Vendas-Arquivo2.csv** que será armazenado em um data frame, por ter variáveis de diversos tipos. Podemos nos perguntar quais são as filiais que vendem produtos de uma determinada empresa (**variável filial**):

```
> vendas = read.csv(file="E:/Grimaldo/Aprenda Virtual/Curso EAD/Aula  
R/Arquivos videoaula/Vendas-Arquivo2.csv", sep=";")
```

```
> table(vendas$FILIAL)
```

```
    Natal    Recife    Salvador  
      74         47         77
```


- Para listar as primeiras linhas de um data frame, que acabou de ser carregado de um arquivo.

> head(vendas)

	FILIAL	PARCELAS	MESCOMPRA	PESOGGRAMAS	VALOR	PRODUTO
1	Salvador	1	1	210	80	Boneca
2	Salvador	1	1	170	80	Boneca
3	Salvador	1	1	720	70	patins
4	Salvador	1	1	200	80	carrinho
5	Salvador	1	1	750	170	carrinho
6	Salvador	1	1	320	80	Boneca

- Você pode definir novas variáveis dentro do data frame que você acabou de criar. Vamos criar a variável desconto, que guardará 10% de todas as vendas realizadas.

> vendas\$DESCONTO=vendas\$VALOR*0.1

> head(vendas)

	FILIAL	PARCELAS	MESCOMPRA	PESOGGRAMAS	VALOR	PRODUTO	DESCONTO
1	Salvador	1	1	210	80	Boneca	8
2	Salvador	1	1	170	80	Boneca	8
3	Salvador	1	1	720	70	patins	7

- Podemos cruzar as variáveis, buscando uma contagem cruzada.

> table (vendas\$FILIAL,vendas\$PRODUTO)

	Boneca	carrinho	damas	patins	pipa	tremzinho
Natal	23	51	0	0	0	0
Recife	15	25	0	0	7	0
Salvador	13	54	3	1	3	3

- Podemos apresentar a distribuição de frequência relativa. Vejamos qual é o produto e filial que mais possuem vendas.

> tabela= table (vendas\$FILIAL,vendas\$PRODUTO)

> tabela.relativa = tabela /nrow(vendas)

> tabela.relativa

	Boneca	carrinho	damas	patins	pipa	tremzinho
Natal	0.116161616	0.257575758	0.000000000	0.000000000	0.000000000	0.000000000
Recife	0.075757576	0.126262626	0.000000000	0.000000000	0.035353535	0.000000000
Salvador	0.065656566	0.272727273	0.015151515	0.005050505	0.015151515	0.015151515

- Agora apresentando em percentual multiplicado por 100

> tabela.relativa = tabela /nrow(vendas) * 100

> tabela.relativa

	Boneca	carrinho	damas	patins	pipa	tremzinho
Natal	11.6161616	25.7575758	0.0000000	0.0000000	0.0000000	0.0000000
Recife	7.5757576	12.6262626	0.0000000	0.0000000	3.5353535	0.0000000
Salvador	6.5656566	27.2727273	1.5151515	0.5050505	1.5151515	1.5151515

- Existe uma função chamada de **tappy** que facilita a aplicação de uma determinada função aplicando em todos os dados, de acordo com um grupo específico, será

tapply(dados, grupos, função). Calcularemos a média e o somatório de todos os pesos em gramas por cada filial.

```
> tapply(vendas$PESOGAMAS, vendas$FILIAL, mean)
      Natal Recife Salvador
205.1351 413.1915 294.9351
```

```
> tapply(vendas$PESOGAMAS, vendas$FILIAL, sum)
      Natal Recife Salvador
15180    19420    22710
```

- Existe uma função chamada de **summary** retorna um conjunto de estatísticas descritivas (**sumário**) de todas as variáveis de um determinado data frame conforme o seu tipo.

```
> summary(vendas$PESOGAMAS)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 30.0   190.0   250.0   289.4   340.0   840.0
```

Existe uma série de estatísticas descritivas na ferramenta R, as mais conhecidas são:

Estatística Descritiva	Nome da Função
Média	mean
Mediana	median
Mínimo	min
Máximo	max
Amplitude de variação	range
Quartis e quantis	quantile
Distância Interquartil (Inter Quarter Range)	IQR
Variância	var
Desvio padrão (Standard Deviation)	sd
Desvio absoluto mediano (Mean Absolut Deviation)	mad

Vamos a alguns exemplos:

```
> mean(vendas$PESOGAMAS)      # Média
[1] 289.4444
> median(vendas$PESOGAMAS)    # Mediana
[1] 250
> min(vendas$PESOGAMAS)      # Mínimo
[1] 30
> max(vendas$PESOGAMAS)      # Máximo
[1] 840
> range(vendas$PESOGAMAS)     # Máximo - Mínimo
```

```

[1] 30 840
> quantile(vendas$PESOGAMAS) # Quartis
 0% 25% 50% 75% 100%
 30 190 250 340 840
> IQR(vendas$PESOGAMAS) # Índice Interquartil.(Q75%-Q25%)
[1] 150
> var(vendas$PESOGAMAS) # Variância
[1] 23459.08
> sd(vendas$PESOGAMAS) # Desvio-Padrão
[1] 153.1636
> mad(vendas$PESOGAMAS) # Desvio Absoluto Mediano
[1] 103.782

```

Selecionando Dados

Agora na parte de manipulação de dados, vamos especificamente trabalhar com conectores lógicos que nos ajudarão na **seleção de determinados dados**, muito importante na manipulação de qualquer dado.

Vamos primeiro aprender quais são os conectores lógicos que facilitam a seleção de dados:

CONECTOR	SIGNIFICADO
>	Maior
<	Menor
>=	Maior ou igual
<=	Menor ou igual
==	Igualdade
!=	Diferente
&	E
	Ou

Agora vamos utilizar um comando que já conhecemos o **which**, lembre-se? Ele irá selecionar as posições de acordo com a condição que criarmos, depois utilizaremos o **[]** que é a forma de traduzir posição em dados, vejamos:

Inicialmente vamos mostrar apenas as posições de uma determinada seleção de dados.

```

> which(vendas$FILIAL=="Salvador")
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25 26 27
[28] 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49
50 51 52 53 54
[55] 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76
77

```

Agora para exibir o conteúdo em cada uma destas posições, utilize o `[]`, lembre-se estamos em um arquivo de dados então o comando `[]` necessita que você determine se a seleção será em linha ou coluna, no nosso caso são as linhas, então deveremos acrescentar a `","` para esta informação.

```
> vendas[which(vendas$FILIAL=="Salvador"),]
  FILIAL PARCELAS MESCOMPRA PESOGRAMAS VALOR PRODUTO
1  Salvador         1         1        210    80   Boneca
2  Salvador         1         1        170    80   Boneca
3  Salvador         1         1        720    70   patins
4  Salvador         1         1        200    80 carrinho
5  Salvador         1         1        750   170 carrinho
6  Salvador         1         1        320    80   Boneca
....
```

Execute os comandos abaixo e veja o que acontece:

```
> vendas[which(vendas$FILIAL!="Salvador"),]
> vendas[which(vendas$FILIAL=="Salvador" & vendas$PRODUTO=="Boneca"),]
> vendas[which(vendas$FILIAL=="Salvador" | vendas$PRODUTO=="Boneca"),]
> vendas[which(vendas$PESOGRAMAS>=210 & vendas$PESOGRAMAS<=310),]
```

Vamos selecionar apenas uma coluna, com um determinado critério e listaremos na tela, mas precisaremos alterar a quantidade de linhas impressas, utilize o comando abaixo:

```
> options(max.print=5.5E5)
```

Agora execute:

```
> sort(head(vendas[which(vendas$PESOGRAMAS>=210 &
vendas$PESOGRAMAS<=250),]$PESOGRAMAS,n=1000L))
```

Para colocar todo o data frame em ordem, como exemplo pelo peso em gramas, faça o seguinte:

```
> ordem.vendas=order(vendas$PESOGRAMAS)
> vendas[ordem.vendas,]
  FILIAL PARCELAS MESCOMPRA PESOGRAMAS VALOR PRODUTO
129  Natal         4         5         30    50 carrinho
127  Natal         4         3         40    50 carrinho
128  Natal         4         4         60    50 carrinho
134  Natal         4        10         60    50 carrinho
...
```

Para colocar em **ordem decrescente** o data frame para o atributo peso em gramas, faça o seguinte:

```
> vendas[order(vendas$PESOGRAMAS,decreasing=TRUE),]
  FILIAL PARCELAS MESCOMPRA PESOGRAMAS VALOR PRODUTO
99  Recife         2         1        840   120   Boneca
82  Recife         2         1        810   120 carrinho
```

Outra forma de se trabalhar com dados é através do comando **ifelse** que permite tomar uma ação de acordo com uma condição específica, **IF** significa **SE** e **ELSE** significa **SENÃO**.

Vamos a um exemplo:

```
> resultado=ifelse(vendas$VALOR>90,"acima das expectativas","abaixo das expectativas")
> table(resultado)
resultado
abaixo das expectativas  acima das expectativas
                118                80
```

Podemos combinar mais de uma condição para selecionar os dados:

```
> resultado=ifelse(vendas$VALOR>90&vendas$PESOGAMAS>200,"acima","abaixo")
> table(resultado)
resultado
abaixo  acima
    127    71
```

11. Gráficos

O R possui um ambiente de trabalho rico para geração de gráficos de diversos tipos, podem gerar gráficos de duas formas:

- **Gráficos para análise de dados:** são gráficos que permitam visualizar o mais claro possível os padrões presentes nos dados. Esses gráficos disponibilizados de forma rápida no R e as formas de construí-los permitem inúmeras interações com os elementos de informação nos gráficos.
- **Gráficos prontos para apresentação:** Gráficos de apresentação são mais elaborados. Sua construção no R exige mais tempo e conhecimento, pois o R não oferece recursos interativos para manipular os elementos picturais dos gráficos.

Vamos iniciar com gráficos simples de análise de dados:

Histogramas

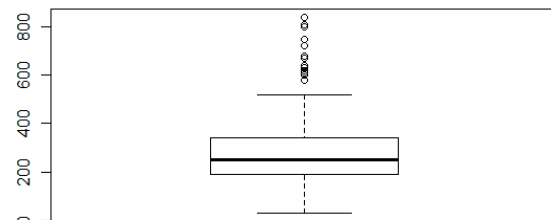
Um dos gráficos mais utilizados em estatística, facilita a exibição de qual função de densidade estamos analisando.


```
3 | 555556788888999
4 | 011223
4 | 55588
5 | 1112
5 | 8
6 | 001234
6 | 778
7 | 2
7 | 5
8 | 0114
```

Box-Plot

O gráfico de box-plot oferece como saída estatísticas descritivas como média, máximo, mínimo, quartis de forma a facilitar o estudo do comportamento das variáveis, vejamos:

```
> boxplot( vendas$PESOGRAMAS )
```



```
> boxplot( PESOGRAMAS ~  
PRODUTO, data=vendas )
```

Obs: Caso deseje aumentar a janela do gráfico, basta setar as informações na função par.

```
Ex.: > par( mar=c(5,4,4,2) )
```

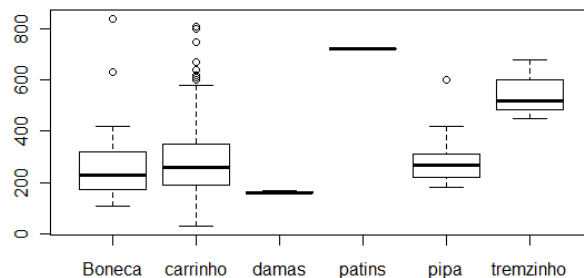
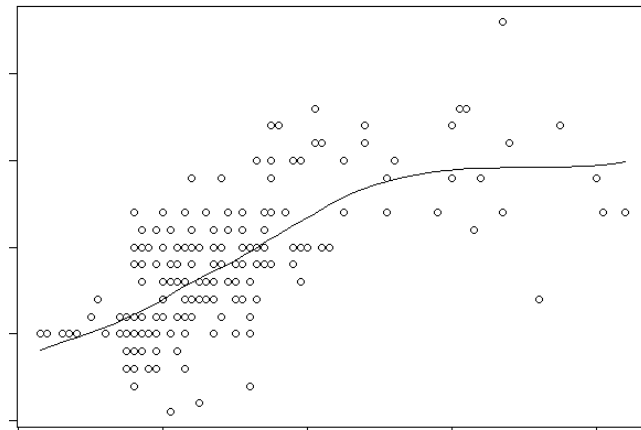


Gráfico de Dispersão

Sempre que desejar criar um gráfico para entender a relação entre duas variáveis, utilize primeiro o gráfico de dispersão, veja como é simples:

```
> par( mar=c(5,4,4,2) )
> plot( vendas$PESOGAMAS,
vendas$VALOR )
> scatter.smooth(
vendas$PESOGAMAS,
vendas$VALOR)
```

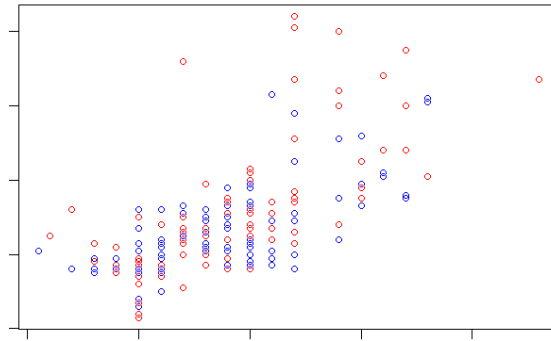


VARIAÇÕES DO GRÁFICO

```
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "h", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "l", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "p", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "b", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "c", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "s", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "s", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "n", col =
"red")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "n", col =
"red",main="babab")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "n", col =
"red",main="PLOTE DOS DADOS")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "p", col =
"red",main="PLOTE DOS DADOS")
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "p", col =
"red",main="PLOTE DOS DADOS",lwd=10)
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "p", col =
"red",main="PLOTE DOS DADOS",lwd=50)
> scatter.smooth( vendas$PESOGAMAS, vendas$VALOR,type = "p", col =
"red",main="PLOTE DOS DADOS",lwd=5)
```

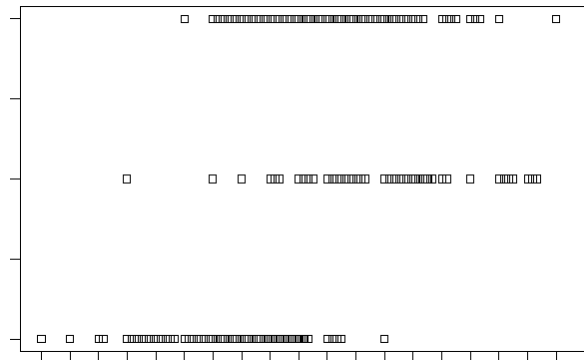


```
> par( mar=c(5,4,4,2) )  
> plot(vendas$VALOR, vendas$PESOGRA  
MAS, col=c("red", "blue"))
```



Quando você deseja realizar o gráfico de dispersão com uma variável categórica, precisamos utilizar a função **stripchart**.

```
> par( mar=c(5,4,4,2) )  
> stripchart(vendas$FILIAL~vendas  
$VALOR, vertical=T)  
  
Obs: Desconcentrando os pontos que  
estão sobrepostos  
  
>  
stripchart(vendas$FILIAL~vendas$  
VALOR, vertical=T, method="stack")
```

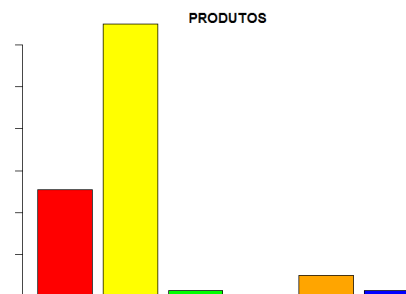


OBS: PLOT(XAXIS,YAXIS) CRIA UM GRÁFICO DE PONTOS SE XAXIS É UMA VARIÁVEL CONTÍNUA E UM BOXPLOT SE XAXIS É UMA VARIÁVEL CATEGÓRICA (SE É UM FATOR)

Gráfico de Barras

Gráfico de barras é um dos mais comuns utilizados nas apresentações de universidades, empresas e por profissionais. Todos os programas de planilhas eletrônicas possuem este tipo de gráfico, vejamos como criá-lo:

```
> par( mar=c(5,4,4,2) )  
> vendas.freq=table(vendas$PRODUTO)  
> barplot(vendas.freq,col=colors)  
> barplot(vendas.freq,col=colors,main="PRODUTOS")
```

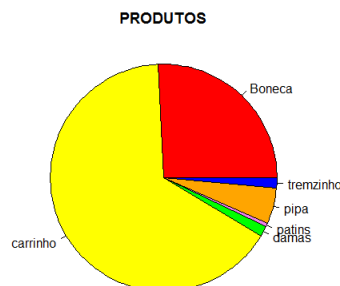


Para que você conheça as formas de qualquer tipo de gráfico, basta digitar:
example(<nome_função_gráfica>
> **example(barplot)**

Gráfico de Pizza ou Setores

É um tipo de gráfico que exibe a distribuição dos dados em frequências (absolutas, percentuais), vejamos:

```
> par( mar=c(5,4,4,2) )  
> pie(vendas.freq,main="PRODUTOS",col=colors)
```



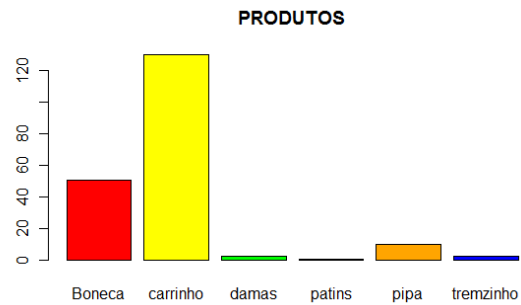
Melhorias nos Gráficos

Caso você deseje colocar um texto qualquer dentro da área do gráfico, utilize o comando **locator**, você terá a possibilidade de colocar informações além dos tradicionais eixos X e Y.

Como fazemos isto:

1) Carregue o gráfico que você deseja. Ex:

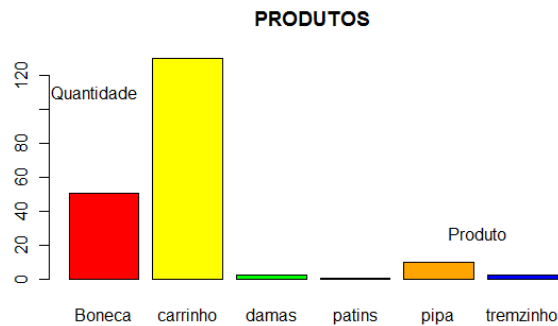
```
> barplot(vendas.freq, col=colors, main="PRODUTOS")
```



2) Vamos colocar duas informações : PRODUTO e Quantidade.

```
> text(locator(2), c("Produto", "Quantidade"))
```

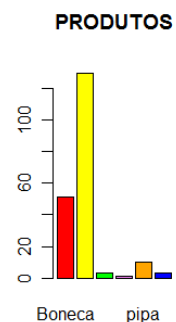
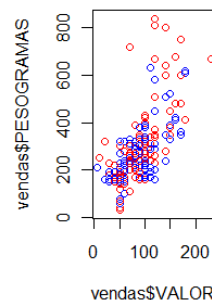
3) Agora Clique nas posições que desejar, deve ficar da seguinte forma:



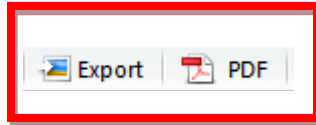
Se você deseja colocar dois gráficos em uma única janela, de forma que possam ser comparados, utilize o comando **par(mfrow=c(nl,nc))** # nl indica o número de linhas e nc o número de colunas que a janela deverá ter. Primeiro vamos dividir a janela em duas colunas.

Como exemplo, vamos exibir dois gráficos em uma única janela:

```
> par(mfrow=c(1,2))
> plot(vendas$VALOR, vendas$PESOGRAMAS, col=c("red", "blue"))
> barplot(vendas.freq, col=colors, main="PRODUTOS")
```



Para retornar a exibição para apenas 1 gráfico, basta repetir a função **par(mflow=c(1,1))**.
Para imprimir um gráfico, na janela plot há uma opção para exportação em PNG e PDF.



Caso você esteja fora do R-studio, você poderá dentro do R clicar com o botão direito do Mouse sobre o gráfico e depois em "**save as metafile**", os formatos são PDF, JPEG, BMP, PNG.

12. Programação

Todos nós sabemos que a ferramenta R é extremamente útil para aqueles que desejam fazer análise de dados através dos pacotes disponíveis na ferramenta. Entretanto ela pode ser vista como uma linguagem de programação, onde o usuário pode criar as suas próprias funções. Esta é uma das maiores vantagens da ferramenta. Além de ser um programa para análises estatísticas, o R é acima de tudo uma linguagem de programação. Você tem duas opções para aprender a usar o R ou aprender a fazer programas básicos em R. Primeiro devemos entender como construímos uma **função** ou **function**.

A sintaxe é:

```
function(lista de argumentos)  
{  
corpo da função  
}
```


- A **lista de argumentos** é a passagem de parâmetros para o seu programa, podem ser passados mais de um parâmetro, tudo separado por vírgula.

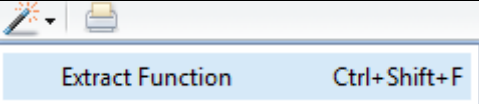
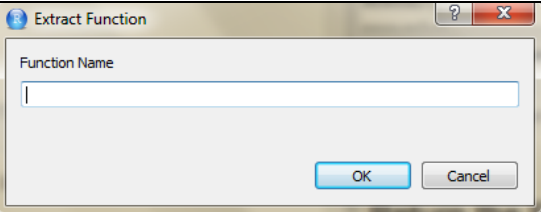
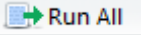
- o **corpo da função** é onde você irá programar o seu algoritmo, ou seja, irá definir a lógica de programação da sua função.

Para dar um nome a sua função basta colocar:

```
minha.função<-function(lista de argumentos){corpo da função}
```

No R-studio você tem um editor de texto para criar a sua função:

1) Você deve escolher File -> New -> R script	
---	--

	<p>Curso EAD Ferramenta R</p> <p>Prof. Grimaldo Oliveira</p> <p>grimaldo_lopes@hotmail.com</p>
2) Pronto o editor está livre, digite a função ao lado.	<pre>{ sample(x,n,replace=T) }</pre>
3) Agora que você digitou a função, no nosso exemplo será criado um vetor com algumas amostras de dados através da função sample .	
4) Digite o nome da função: ganho	
5) O próprio R-studio organizará e ficará da seguinte forma:	<pre>ganho <- function (x, n) { { sample(x,n,replace=T) } }</pre>
6) Marque toda a função e clique em RUN ALL .	
7) Verifique na Aba workspace se a função apareceu.	<pre>ganho(x, n)</pre>
8) Pronto agora em linha de comando digite:	<pre>> informe=c("Brasil","Alemanha","inglaterra") > ganho(informe,10) Resultado pode ser: [1] "Alemanha" "Brasil" "Brasil" "inglaterra" "Brasil" "Brasil" [7] "Brasil" "inglaterra" "Alemanha" "Alemanha"</pre>

Agora você percebe que foi criada uma função onde são passados 2 parâmetros, o vetor contendo os dados que serão amostrados e a quantidade de amostras com reposição. Ao contrário de chamar a função **sample**, você utiliza a função **ganho** criada por você.

Comando FOR

O comando **FOR** é utilizado para criar loopings de execução.

A sintaxe do comando é: **for (variável in 1:n) {comandos}**

Por exemplo, vamos utilizar o comando FOR para gerar a sequência de **Fibonacci**.

```
> fibonacci=numeric(0)
> fibonacci[c(1,2)]=1
> for (x in 3:12) {fibonacci[x]=fibonacci[x-2]+fibonacci[x-1]}
> fibonacci
[1] 1 1 2 3 5 8 13 21 34 55 89 144
```

Note que o vetor fibonacci guardou a soma dos valores que foram passados inicialmente `fibonacci[c(1,2)]=1`, o comando FOR foi responsável de circular a variável **x** iniciando em 3 e terminando em 12 um a um.

Outro Exemplo

```
> megasena<-function(numjogos) { # cria a função com nome de megasena
+ numeros<-matrix(NA,6,numjogos) # cria o arquivo que recebe os jogos
+ for(x in 1:numjogos){
+   numeros[,x]<-sample(1:60,6)
+ }
+ return(numeros)
+ }
> megasena(20)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]	[,13]	[,14]	[,15]
[1,]	10	50	41	30	44	35	49	57	14	21	17	49	32	45	19
[2,]	30	27	18	31	9	55	3	58	19	46	44	1	15	26	31
[3,]	49	32	54	24	51	3	51	28	1	43	42	21	21	10	23
[4,]	20	18	30	40	21	53	2	49	27	36	28	43	28	39	27
[5,]	25	12	53	17	15	56	44	12	59	58	40	28	48	4	40
[6,]	58	51	3	32	55	19	14	33	31	53	41	48	8	30	60

	[,16]	[,17]	[,18]	[,19]	[,20]
[1,]	54	36	57	34	21
[2,]	26	31	39	35	5
[3,]	4	34	32	41	39
[4,]	21	11	19	40	29
[5,]	18	44	17	56	58
[6,]	3	6	1	7	35

13. Uso da Estatística

Caro Aluno, conforme havíamos detalhado desde o início do curso, o objetivo principal dos nossos estudos é na descoberta das potencialidades da ferramenta R, de forma a permitir uma compreensão do seu funcionamento, mas de todo modo vimos aqui algumas facilidades estatísticas que a ferramenta pode oferecer, entretanto neste tópico irei apresentar algumas estatísticas e manipulações de dados que facilitam a aplicação de teste estatísticos e de análise de dados em geral.

Vamos importar o arquivo **Cidadao-Arquivo3.csv**

Com o arquivo importando, vamos entender dois conceitos importantes na estatística, variáveis quantitativas e qualitativas:

- qualitativas
 - nominais
 - ordinais
- quantitativas
 - discretas
 - contínuas

Estas podem ser resumidas por tabelas, gráficos e/ou medidas.

As variáveis pertencentes ao arquivo **Cidadao-Arquivo3.csv**, **NÃO** são numéricas e sim categóricas. No R variáveis categóricas são definidas usando o comando **factor()**, que vamos usar para redefinir nossas variáveis. Redefinimos a variável **Est.civil** com os rótulos (labels) **solteiro** e **casado** associados aos níveis (levels) **1** e **2**. Para variável **Grau.instrucao** usamos o argumento adicional **ordered = TRUE** para indicar que é uma **variável ordinal**. Na variável **regiao** codificamos assim: 2=capital, 1=interior, 3=outro. Ao final inspecionamos as primeiras linhas do conjunto de dados digitando usando **head()**.

```
> cidadao=`Cidadao-Arquivo3` # reduzindo o nome do arquivo
> head(cidadao) #parte do arquivo
```

Cidadao	Est.civil	Grau.Instrucao	Idade	Salario	Irmão	regiao
1	1	1	1	45	4.00	1
2	2	2	1	46	4.56	3
3	3	1	1	34	5.25	4
4	4	1	2	35	5.73	3
5	5	1	1	36	6.26	3
6	6	2	1	56	6.66	2

```
> cidadao$Est.civil=factor(cidadao$Est.civil, label = c("solteiro",
"casado"),levels = 1:2)
```

```
[1] solteiro casado solteiro solteiro solteiro casado casado casado solteiro
[10] casado solteiro casado casado solteiro solteiro solteiro solteiro casado
[19] casado solteiro casado
Levels: solteiro casado
```



```
> cidadeao$Grau.Instrucao=factor(cidadeao$Grau.Instrucao, label =  
c("1Grau","2Grau"),level = 1:2, ordered = T)  
> cidadeao$Grau.Instrucao  
  
[1] 1Grau 1Grau 1Grau 2Grau 1Grau 1Grau 1Grau 1Grau 2Grau 2Grau 2Grau 1Grau 2Grau  
[14] 2Grau 2Grau 2Grau 1Grau 1Grau 1Grau 2Grau 1Grau  
Levels: 1Grau < 2Grau  
  
> cidadeao$regiao <- factor(cidadeao$regiao, label = c("capital",  
"interior","outro"), lev = c(2, 1,3))  
> cidadeao$regiao  
  
[1] interior capital interior outro outro interior capital interior capital  
[10] interior capital interior capital capital capital outro capital outro  
[19] outro capital interior  
Levels: capital interior outro  
  
> cidadeao  
  Cidadeao Est.civil Grau.Instrucao Idade Salario Irmao regiao  
1         1 solteiro          1Grau   45    4.00      1 interior  
2         2 casado          1Grau   46    4.56      3 capital  
3         3 solteiro          1Grau   34    5.25      4 interior  
4         4 solteiro          2Grau   35    5.73      3 outro  
5         5 solteiro          1Grau   36    6.26      3 outro  
6         6 casado          1Grau   56    6.66      2 interior
```

Para criar uma nova variável utilize o comando **transform()**, vamos criar uma variável que guarde o aumento salarial de 20%.

```
> cidadeao=transform(cidadeao,aumento.sal=Salario*1.20)
```

Análise univariada

A análise univariada consiste basicamente em, para cada uma das variáveis individualmente:

- **Classificar a variável quanto a seu tipo:**
 - 1) **qualitativa** (nominal ou ordinal) ou
 - 2) **quantitativa** (discreta ou contínua).
- Gerar uma tabela, gráfico e/ou medidas que resumam a variável.

Vamos trabalhar com a **variável qualitativa nominal** Est.civil, podemos gerar uma tabela de frequência, um gráfico de setores que é o mais indicado para o tipo de variável.

✓ Frequência Absoluta

```
> civil=table(cidadeao$Est.civil)  
> civil
```

```
solteiro  casado
      11      10
```

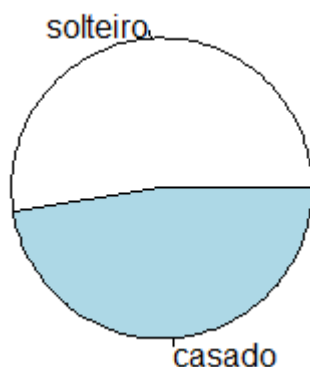
✓ **Frequência Relativa**

```
> prop.table(civil)
```

```
solteiro  casado
0.5238095 0.4761905
```

✓ **Gráfico de Setores**

```
> pie(prop.table(civil))
```



✓ **Moda**

```
> names(civil)[which.max(civil)]
[1] "solteiro"
```

Existem alguns específicos gráficos de acordo com o tipo da variável:

Tipo de Variável	Gráfico
Variável qualitativa nominal	Setores - <code>pie(prop.table(civil))</code>
Variável qualitativa ordinal	Barras - <code>barplot(instrucao)</code>
Variável quantitativa discreta	Frequência absoluta – <pre>> irmao.fa=table(cidadao\$Irmao)</pre> <pre>> plot(irmao.fa)</pre> Frequência relativa – <pre>> irmao.fr=prop.table(irmao.fa)</pre> <pre>> Plot(irmão.fr)</pre>
Variável quantitativa continua	Histograma <pre>> hist(cidadao\$Salario, main = "")</pre> Box-plot <pre>> boxplot(cidadao\$Salario)</pre>

Por curiosidade, para se obter uma frequência relativa de uma variável contínua, teremos que saber a amplitude dos dados (máximo e mínimo), a quantidade de classes (utilizaremos o critério de sturges).

```
> range(cidadao$salario)
[1] 4.00 11.06
> nclass.Sturges(cidadao$salario)
[1] 6
> salario.fr <- table(cut(cidadao$salario, seq(4.00, 11.06, l = 7)))
> salario.fr

      (4,5.18] (5.18,6.35] (6.35,7.53] (7.53,8.71] (8.71,9.88] (9.88,11.1]
              1              3              4              3              6              3
> prop.table(salario.fr)

      (4,5.18] (5.18,6.35] (6.35,7.53] (7.53,8.71] (8.71,9.88] (9.88,11.1]
      0.05      0.15      0.20      0.15      0.30      0.15
```

Análise Bivariada

Na análise bivariada basicamente procuramos identificar relações entre duas variáveis. Assim como na análise univariada estas relações podem ser resumidas também por gráficos, tabelas e/ou medidas estatística. Os estudos realizados nestas variáveis vai depender dos tipos das variáveis envolvidas. Vamos considerar três possibilidades:

- qualitativa vs qualitativa
- qualitativa vs quantitativa
- quantitativa vs quantitativa



OBS: AS RELAÇÕES ENTRE DUAS VARIÁVEIS DEVEM SER EXAMINADAS COM CUIDADO, POIS Podem ser OCULTADAS/MASCARADAS POR UMA OU MAIS VARIÁVEIS EXISTENTES NÃO CONSIDERADAS NA ANÁLISE. ESTAS SÃO CHAMADAS VARIÁVEIS DE CONFUNDIMENTO.

1) **Análise Qualitativa x Qualitativa:** Este tipo de análise envolve duas variáveis criando a chamada tabela de cruzamento ou tabela de contingência e pode ser apresentada de várias formas. A forma mais adequada de apresentação vai depender dos objetivos da análise e da interpretação desejada para os dados.

Vamos iniciar cruzando duas variáveis Estado Civil e Grau de Instrução.

> `tab.cr=(cidadeao$Est.civil,cidadeao$Grau.Instrucao)`

	1Grau	2Grau
solteiro	4	7
casado	8	2

Podemos saber o somatório de cada linha ou coluna e suas frequências relativas através do comando **addmargins()**

> `addmargins(tab.cr,margin=)`

	1Grau	2Grau	Sum
solteiro	4	7	11
casado	8	2	10
Sum	12	9	21

> `addmargins(tab.cr,margin=1)`

	1Grau	2Grau
solteiro	4	7
casado	8	2
Sum	12	9

> `addmargins(tab.cr,margin=2)`

	1Grau	2Grau	Sum
solteiro	4	7	11
casado	8	2	10

A frequência relativa faremos com o comando conhecido **prop.table()**

> `prop.table(tab.cr,margin=1)`

	1Grau	2Grau	
solteiro	0.3636364	0.6363636	→ 100%
casado	0.8000000	0.2000000	

> `prop.table(tab.cr,margin=2)`

	1Grau	2Grau
solteiro	0.3333333	0.7777778
casado	0.6666667	0.2222222
	→ 100%	

Neste tipo de análise o ideal é gerar gráficos de barras, vejamos:

```
> barplot(tab.cr, legend = T) # gráficos para Grau de instrução x Estado Civil
> barplot(t(tab.cr), legend = T) # gráficos do inverso
> barplot(prop.table(tab.cr), beside = T, legend = T) # gráfico de barras
separado por Grau de instrução
```

Uma medida estatística para análise de duas variáveis qualitativas é o teste Chi-quadrado, através do comando **summary()** conhecido como medida de associação entre duas

variáveis, podemos executar o teste e verificar se há associação entre estado civil e grau de instrução.

```
> summary(tab.cr)
Number of cases in table: 21
Number of factors: 2
Test for independence of all factors:
  chisq = 4.073, df = 1, p-value = 0.04358
Chi-squared approximation may be incorrect
```

O resultado apresenta dados insuficientes para a análise, portanto é necessário que um novo agrupamento ou novos dados sejam coletados para que a estatística Chi-quadrado seja exibida corretamente.

2) Análise Qualitativa x Quantitativa: Para este tipo de análise vamos utilizar agrupamento de **tabelas em classe** e a análise de gráfico deve ser realizada através de **box-plot**. Vamos utilizar o comando **cut()** para separar a tabela em quartis, a instrução **include.lowest=TRUE** para incluir os extremos dos dados, os valores de mínimo e máximo. As variáveis que trabalharemos serão salário (quantitativa) e Grau de instrução (qualitativa)

Para exibir os quartis , para se ter uma ideia da separação das classes, utilize o comando **quantile()**

```
> quantile(cidadao$Salario)
 0%   25%   50%   75%  100%
4.00  6.66  8.12  9.35 11.06

> salario.qt=cut(cidadao$Salario, quantile(cidadao$Salario),
include.lowest = T)
> tab.sl=table(cidadao$Grau.Instrucao,salario.qt)
> tab.sl
```

	salario.qt [4,6.66]	(6.66,8.12]	(8.12,9.35]	(9.35,11.1]
1Grau	5	2	1	4
2Grau	1	3	4	1

Para análise de frequência relativa, lembre-se do comando **prop.table()**

```
> prop.table(tab.sl,margin=1)
salario.qt
  [4,6.66] (6.66,8.12] (8.12,9.35] (9.35,11.1]
1Grau 0.41666667 0.16666667 0.08333333 0.33333333 → 100%
2Grau 0.11111111 0.33333333 0.44444444 0.11111111

> prop.table(tab.sl,margin=2)
salario.qt
  [4,6.66] (6.66,8.12] (8.12,9.35] (9.35,11.1]
1Grau 0.8333333 0.4000000 0.2000000 0.8000000
2Grau 0.1666667 0.6000000 0.8000000 0.2000000
→100%
```

Vamos então obter um gráfico dos salários para cada nível de instrução, ou seja, os salários(variável resposta) serão explicados pelos graus de instrução(variável explicativa):

```
> boxplot(cidadao$Salario~cidade$Grau.Instrucao)
```

Podemos obter, outras estatísticas descritivas: Média e desvio padrão. Tudo através do comando **tapply()**, no nosso exemplo em relação a salário e grau de instrução.

```
> tapply(cidadao$Salario,cidade$Grau.Instrucao, mean)
      1Grau      2Grau
7.550000 8.423333
```

```
> tapply(cidadao$Salario,cidade$Grau.Instrucao, sd)
      1Grau      2Grau
2.370462 1.423165
```

3) Análise Quantitativa vs Quantitativa: Para este tipo de análise podemos gerar tabelas de classes para ambas as variáveis, no exemplo possuímos salário e idade, que são variáveis quantitativas. O gráfico explorado será o gráfico de dispersão, onde perceberemos que uma variável é explicada por outra.

Vamos gerar as classes da variável idade:

```
> idade.qt=cut(cidade$Idade,quantile(cidade$Idade),include.lowest = T)
```

Gerando as tabelas de classes com **frequência absoluta e relativa**:

```
> table(idade.qt,salario.qt)
      idade.qt      salario.qt
      [21,34]      [4,6.66] (6.66,8.12] (8.12,9.35] (9.35,11.1]
      (34,45]      0          3          4          0
      (45,56]      3          0          0          1
      (56,78]      2          1          1          1
      (56,78]      1          1          0          3
```

```
> prop.table(table(idade.qt,salario.qt),margin=1)
      idade.qt      salario.qt
      [21,34]      [4,6.66] (6.66,8.12] (8.12,9.35] (9.35,11.1]
      (34,45] 0.0000000 0.4285714 0.5714286 0.0000000
      (45,56] 0.7500000 0.0000000 0.0000000 0.2500000
      (56,78] 0.4000000 0.2000000 0.2000000 0.2000000
      (56,78] 0.2000000 0.2000000 0.0000000 0.6000000
```

```
> prop.table(table(idade.qt,salario.qt),margin=2)
      idade.qt      salario.qt
      [21,34]      [4,6.66] (6.66,8.12] (8.12,9.35] (9.35,11.1]
      (34,45] 0.0000000 0.6000000 0.8000000 0.0000000
      (45,56] 0.5000000 0.0000000 0.0000000 0.2000000
      (56,78] 0.3333333 0.2000000 0.2000000 0.2000000
      (56,78] 0.1666667 0.2000000 0.0000000 0.6000000
```

Para geração dos gráficos utilizaremos a função **plot()**

```
> plot(cidadao$Idade,cidadao$Salario)
```

No gráfico você irá notar que à medida que a idade(variável explicativa,preditora,independente) avança os salários(variável resposta, dependente – está sempre no eixo y) são maiores, confirmando uma relação entre as variáveis.

Podemos fechar o uso de estatísticas para variáveis quantitativas, através do chamado coeficiente de correlação que analisa através de uma medida variando de **-1 até +1** se há uma forte, nenhuma ou fraca correlação entre as variáveis:

Correlação linear:

```
> cor(cidadao$Idade,cidadao$Salario)
[1] 0.06533893
```

Correlação Spearman:

```
> cor(cidadao$Idade,cidadao$Salario,method = "spearman")
[1] 0.07289313
```

Ambas apresentam valores próximos de zero, não confirmando as correlações no caso.

Teste de Hipóteses

Serão apresentados alguns testes de hipóteses para ilustração da ferramenta R, importante que você obtenha conhecimento teórico antes de utilizar estas funções da ferramenta R, o nosso curso não se propõe ao aprendizado teórico das análises estatísticas geradas aqui, cabe ao aluno o aprendizado prévio.

1) Média de uma distribuição normal com variância desconhecida

Para realizar inferência sobre a média de uma amostra considerando com uma distribuição Normal, devemos verificar qual é o seu intervalo de confiança.

Sabemos que o intervalo de confiança para média de uma distribuição normal com média desconhecida é dado por:

$$\left(\bar{x} - t_{\alpha/2} \sqrt{\frac{S^2}{n}}, \bar{x} + t_{1-\alpha/2} \sqrt{\frac{S^2}{n}} \right)$$

No nosso caso vamos partir de um conjunto de dados para utilizar o teste t para calcular o intervalo para esta média desconhecida e com um tamanho de amostra pequeno.

No nosso exemplo consideremos:

Sejam os dados abaixo representando o **tempo** (segundos) de um chute de penalidade máxima, dos gols marcados pela seleção brasileira nas últimas copas do mundo. Considerado como tendo distribuição Normal e deseja-se fazer inferência sobre a média que é desconhecida obtendo um intervalo de confiança. Trinta chutes foram sorteados dentro 70 últimos gols de pênalti marcados. Os dados foram os seguintes (em minutos):

```
3.7 1.6 4.2 3.3 3.2 4.1 6.1 2.5 3.1 4.3
3.2 4.2 3.1 4.2 5.2 3.2 4.7 2.1 2.3 2.1
3.1 4.6 2.1 1.5 1.8 2.7 3.1 4.7 2.5 3.4
```

```
> chute=c(3.7,1.6,4.2,3.3,3.2,4.1,6.1,2.5,3.1,4.3,3.2,4.2,3.1,4.2,5.2,3.2,
4.7,2.1,2.3,2.1,3.1,4.6,2.1,1.5,1.8,2.7,3.1,4.7,2.5,3.4)
```

```
> t.test(chute)
```

One Sample t-test

```
data: chute
t = 16.1906, df = 29, p-value = 4.610e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 2.909347 3.750653
sample estimates:
mean of x
 3.33
```

O intervalo de confiança default é de 95% testa à igualdade de média a zero (p-value = 6.610e-16), em um teste bilateral.

Para obter um intervalo de confiança diferente de 95%, para forçar o teste em uma média específica, vamos dizer que você saiba a média da população e queira testar a hipótese:

H_0 : Média igual a 2
 H_a : Média é maior que 2

```
> t.test(chute, alt = "greater", mu = 2, conf.level = 0.99)
```

One Sample t-test

```
data: chute
t = 6.4665, df = 29, p-value = 2.234e-07
alternative hypothesis: true mean is greater than 2
```


99 percent confidence interval:
2.823623 Inf
sample estimates:
mean of x
3.33

2) Teste χ^2 para aderência à uma certa distribuição

Muitas vezes desejamos saber se uma determinada amostra tem as mesmas características (distribuição) de uma determinada população, desta maneira podemos utilizar o teste Chi-quadrado para determinar qual hipótese aceitar.

H_0 : Segue a distribuição esperada
 H_a : Não segue a distribuição esperada

Considere a quantidade de indivíduos de acordo com os seus pesos, é retirada uma determinada amostra para cinco grupos de indivíduos, separados por classes sociais:

Classe A	Classe B	Classe C	Classe D	Classe E
180	30	50	60	80

Seja o padrão da população:

Classe A	Classe B	Classe C	Classe D	Classe E
16	4	6	7	5

Vamos calcular a um nível de significância de $\alpha = 5\%$.

A estatística de teste $\chi_c^2 = \sum_i \frac{(o_i - e_i)^2}{e_i}$

Vejamos:

```
> o <- c(180, 30, 50, 60, 80)
> e <- c(16, 4, 6, 7, 5)/38
> chisq.test(o, p = e)
```

Chi-squared test for given probabilities

```
data: o
X-squared = 23.7905, df = 4,
p-value = 8.798e-05
```

Portanto como o p-valor é menor que 0.05, a conclusão é que **rejeita-se H_0** ao nível de 5%, ou seja, a hipótese que a população pertence a amostra não é confirmada, esta não segue o padrão de pesos da população estudada.

3) Comparação de duas médias

Sempre quando temos uma variável qualitativa com dois níveis e outra quantitativa o interesse em geral está em comparar as médias da quantitativa para cada grupo da qualitativa. Para isto podemos utilizar o **teste T**. Há dois tipos de teste T: para amostras independentes com variâncias iguais ou desiguais, ou para amostras pareadas.

No nosso exemplo, vamos testar se um determinado tipo de chocolate que é produzido por duas empresas diferentes, realmente difere em relação a sua produção pelo seu teor de cacau encontrado nos chocolates. Vamos comparar as médias dos teores de cacau a um nível de 5% de significância e considerando as variâncias iguais, tiramos uma amostra de tamanho=8:

Só Cacau 0.21.23.42.14.52.35.24.2
Puro Sabor 2.12.14.23.52.34.21.30.7

As hipóteses são:

$$H_0 : \mu_1 = \mu_2$$
$$H_a : \mu_1 \neq \mu_2$$

```
> sc=c(0.2,1.2,3.4,2.1,4.5,2.3,5.2,4.2)
> pc=c(2.1,2.1,4.2,3.5,2.3,4.2,1.3,0.7)
> t.test(sc, pc, var.eq = TRUE, conf = 0.95)
```

Two Sample t-test

```
data: sc and pc
t = 0.4412, df = 14, p-value = 0.6658
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.303165  1.978165
sample estimates:
mean of x mean of y
  2.8875    2.5500
```

Pelo teste verificasse que as médias são iguais, ou seja não rejeitamos a hipótese nula (H_0), note o p-valor(**0.6658**) é maior que 0.05.

Se a amostra for pareada, ou seja, são duas coletas para os mesmos indivíduos, então basta acrescentar a informação: **paired=TRUE**

```
> t.test(sc, pc, var.eq = TRUE, conf = 0.95, paired=TRUE)
```

4) Comparação de médias múltiplas pelo teste de Tukey

Muitas vezes temos mais de uma média em nossos experimentos, portanto desejamos comparar as médias entre fatores e grupos. Vamos carregar a base de dados **Escola-Arquivo4.csv**, que contém as taxas de aprovação dos alunos por determinadas escolas para os mesmos professores, ou seja, os professores lecionam em escolas diferentes e queremos saber se há diferenças nas taxas de aprovação entre escolas. Para facilitar utilize a importação do dataset no R-Studio, ou digite o comando a seguir:

```
> escola=`Escola-Arquivo4` <- read.csv("E:/Grimaldo/Aprenda  
Virtual/Curso EAD/Aula R/Arquivos videoaula/Escola-Arquivo4.csv",  
sep=";")  
View(`Escola-Arquivo4`)
```

Consideramos taxa de aprovação a variável que determinará as médias por escola e professores, criamos uma interação, note pelo símbolo de ":".

```
> escola.media= aov(escola$taxa_aprovacao ~ escola$escolas +  
escola$professores + escola$escolas:escola$professores, data = escola)
```

```
> escola.tk=TukeyHSD(escola.media)  
> escola.tk  
Tukey multiple comparisons of means  
95% family-wise confidence level
```

```
Fit: aov(formula = escola$taxa_aprovacao ~ escola$escolas +  
escola$professores + escola$escolas:escola$professores, data = escola)
```

```
$`escola$escolas`  
      diff      lwr      upr      p adj  
es2-es1 -5.325 -13.716052  3.066052 0.2632241  
es3-es1 -4.025 -12.416052  4.366052 0.4546932  
es3-es2  1.300  -7.091052  9.691052 0.9178202  
  
$`escola$professores`  
      diff      lwr      upr      p adj  
p2-p1 -0.09166667 -5.731578  5.548244 0.973136  
  
$`escola$escolas:escola$professores`  
      diff      lwr      upr      p adj  
es2:p1-es1:p1 -4.475 -19.251842 10.301842 0.9239345  
es3:p1-es1:p1 -3.650 -18.426842 11.126842 0.9666007  
es1:p2-es1:p1  0.725 -14.051842 15.501842 0.9999846  
es2:p2-es1:p1 -5.450 -20.226842  9.326842 0.8440854  
es3:p2-es1:p1 -3.675 -18.451842 11.101842 0.9656296  
es3:p1-es2:p1  0.825 -13.951842 15.601842 0.9999708  
es1:p2-es2:p1  5.200  -9.576842 19.976842 0.8675048  
es2:p2-es2:p1 -0.975 -15.751842 13.801842 0.9999332  
es3:p2-es2:p1  0.800 -13.976842 15.576842 0.9999749  
es1:p2-es3:p1  4.375 -10.401842 19.151842 0.9303175
```

```
es2:p2-es3:p1 -1.800 -16.576842 12.976842 0.9986690
es3:p2-es3:p1 -0.025 -14.801842 14.751842 1.0000000
es2:p2-es1:p2 -6.175 -20.951842 8.601842 0.7663948
es3:p2-es1:p2 -4.400 -19.176842 10.376842 0.9287538
es3:p2-es2:p2 1.775 -13.001842 16.551842 0.9987553
```

pelos p-valores encontrados não encontramos diferenças de médias das taxas de aprovação entre os professores que ensinam em escolas diferentes.

Regressão Linear Simples

A regressão linear simples é utilizada para analisar uma relação que ocorre entre variáveis quantitativas (contínuas). É chamado de “linear”, pois se considera que o resultado esperado é verificado em uma função linear, do tipo $y = ax + b$. Para exemplificar o uso da regressão linear simples, carregaremos o arquivo **Bovinos-Arquivo5.csv**, onde procuraremos estudar se o percentual de proteína de bovinos aumenta conforme a quantidade ingerida de uma determinada ração.

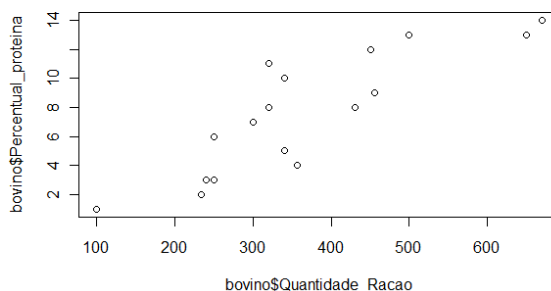
```
> `Bovinos-Arquivo5` <- read.csv("E:/Grimaldo/Aprenda Virtual/Curso
EAD/Aula R/Arquivos videoaula/Bovinos-Arquivo5.csv", sep=";")
```

```
> bovino=`Bovinos-Arquivo5` <- read.csv("E:/Grimaldo/Aprenda
Virtual/Curso EAD/Aula R/Arquivos videoaula/Bovinos-Arquivo5.csv",
sep=";")
```

Para utilizar a função de regressão linear simples, vamos chamar a função **lm()** e a função **summary()** com suas variações para linear models.

Primeiro vamos plotar os dados para verificar se existe uma relação linear entre as variáveis.

```
> plot(bovino$Quantidade_Racao,bovino$Percentual_proteina)
```



Pelo gráfico aparentemente parece que os pontos estão crescentes e formando uma reta. Vamos aplicar a função `lm()` para confirmar ou não.

```
> rls=lm(bovino$Percentual_proteina ~ bovino$Quantidade_Racao)
> summary.aov(rls)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
bovino\$Quantidade_Racao	1	3371.6	3371.6	62.509	9.953e-07 ***
Residuals	15	809.1	53.9		

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(rls)
```

Call:
`lm(formula = bovino$Percentual_proteina ~ bovino$Quantidade_Racao)`

Residuals:

Min	1Q	Median	3Q	Max
-13.0125	-3.7277	-0.1322	6.7723	9.6783

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	13.73520	4.86168	2.825	0.0128 *
bovino\$Quantidade_Racao	0.09797	0.01239	7.906	9.95e-07 ***

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.344 on 15 degrees of freedom
Multiple R-squared: 0.8065,    Adjusted R-squared: 0.7936
F-statistic: 62.51 on 1 and 15 DF,  p-value: 9.953e-07
```

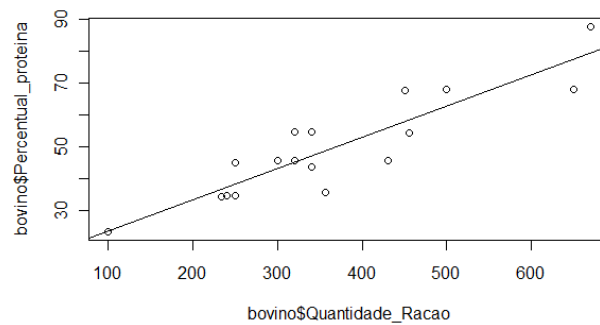
Note que o modelo de regressão linear simples pela análise do p-valor informa que existe uma relação linear entre as variáveis, com a determinação do modelo pela função:

$$\text{Percentual_proteina} = 13.73520 + 0.09797 * \text{Quantidade_ração}$$

Por exemplo, se desejarmos saber qual seria o valor estimado do Percentual de proteína de um boi, se este se alimentasse de 350 quilos de ração:

$$\text{Percentual_Proteina} = 13.73520 + 0.09797 * 350 \rightarrow \mathbf{48,0247}$$

Para verificarmos a reta estimada façamos:



Regressão Múltipla

A regressão múltipla se diferencia da regressão linear simples, por possuir mais de uma preditora (X), ou seja, para uma variável resposta(Y) podemos ter mais de uma variável explicativa(X).

Vamos utilizar o mesmo arquivo **Bovinos-Arquivo5.csv** e acrescentar a variável Percentual de carne magra no modelo.

Da mesma forma que trabalhamos na regressão linear simples, vamos chamar a função **lm()** e a função **summary()** com suas variações, para linear models.

```
> rlm=lm(bovino$Percentual_proteina ~ bovino$Quantidade_Racao +
bovino$Percentual_Carne_Magra)
> summary.aov(rlm)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
bovino\$Quantidade_Racao	1	3371.6	3371.6	68.0330	9.579e-07 ***
bovino\$Percentual_Carne_Magra	1	115.3	115.3	2.3256	0.1495
Residuals	14	693.8	49.6		

 Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
 > summary(rlm)

```
Call:
lm(formula = bovino$Percentual_proteina ~ bovino$Quantidade_Racao +
    bovino$Percentual_Carne_Magra)
```

Residuals:

Min	1Q	Median	3Q	Max
-14.0887	-2.2565	-0.2885	4.6368	10.7687

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.49957	9.85102	0.051	0.960271
bovino\$Quantidade_Racao	0.07966	0.01689	4.716	0.000331 ***

	Curso EAD Ferramenta R Prof. Grimaldo Oliveira grimaldo_lopes@hotmail.com
--	---

```
bovino$Percentual_Carne_Magra 0.25317    0.16601    1.525 0.149526
```

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 7.04 on 14 degrees of freedom
```

```
Multiple R-squared:  0.834,    Adjusted R-squared:  0.8103
```

```
F-statistic: 35.18 on 2 and 14 DF,  p-value: 3.467e-06
```

Note que o modelo de regressão múltipla informa um p-valor abaixo de 0.05, foi significativa apenas para a quantidade de ração, não apresentando relevância para os demais atributos, portanto não poderemos utilizar todos os atributos na definição do modelo múltiplo:

14. Exercícios Finais para Emissão do Certificado de Participação

Agora caro aluno, caso esteja apto, com todas as dúvidas retiradas e todas as vídeoaulas assistidas, inicie sua bateria de exercícios para que tenha direito ao certificado de participação.

O aluno deve realizar os exercícios e caso tenha dúvidas deve consultar o professor, mas o aluno deve tentar realizar **TODOS OS EXERCÍCIOS**, estejam corretos ou não.

O professor irá avaliar o seu grau de entendimento. Caso o professor não esteja satisfeito com o seu rendimento, este solicitará que você refaça os exercícios ou revise algumas aulas.

O professor tem total autonomia para ajudar e gerenciar as atividades dos alunos e decidir quando o aluno terá direito ao certificado de participação.

14.1 Primeira Bateria de Exercícios

Para os resultados abaixo mostre os comandos:

1) Carregue o vetor

[1] 29 67 78 46 56 67

2) Selecionar o quinto e último elemento do vetor que você criou acima

[1] 56 67

3) Crie o vetor abaixo na seguinte ordem de valores

[1] 32 45 56 98 103 154

4) Crie um vetor com repetição de sequência

[1] 1 1 1 2 2 2 3 3 3 4 4 4

5) Crie um vetor que guarde dados alfanuméricos

[1] "Bahia" "Fluminens" "Palmeiras" "Santos"

6) Crie um conjunto de dados que responda as informações abaixo. Para saber se as informações estarão corretas, considere a tabela de frequências a seguir. Entre com os dados usando o tipo de objeto adequado.

	Casado		Solteiro	
Idade	Homem	Mulher	Homem	Mulher
Menor que 50	30	16	34	13
50 a 70	14	32	23	17
Maior que 70	27	23	44	32

- 6.1) Encontre o número total de pessoas.
- 6.2) Informe quantos são homens casados.
- 6.3) Informe quantos são homens.
- 6.49 Informe a proporção da distribuição, qual o maior percentual?

14.2 Segunda Bateria de Exercícios

Para os resultados abaixo mostre os comandos:

Há vários conjuntos de dados que são embutidos na ferramenta R como por exemplo, o conjunto `mtcars`. Para ver uma lista completa com o conjunto de dados disponíveis digite `data()`.

Carregue a base `cars` e execute os comandos a seguir:

> `Data(cars)`

- 1) Calcule a `mean()`, `var()`, `sd()`, `median()`, `quantile()` do campo **speed**.
- 2) Encontre o número máximo e mínimo do campo **dist**.
- 3) Exiba o gráfico de ramo-e-folhas para o campo **speed**.
- 4) Encontre os coeficientes de correlação linear e de Spearman para os campos **dist** e **speed**.
- 5) Crie um gráfico de dispersão para os campos **dist** e **speed**.
- 6) Exiba as primeiras linhas do conjunto de dados **cars**.
- 7) Coloque em ordem decrescente os dados do campo **speed**.
- 8) Utilizando o comando `which`, filtre os dados cujo o campo **speed** seja maior que 80.
- 9) Crie o campo tempo pela fórmula **dist/speed**.
- 10) Calcule o tempo médio total.

14.3 Terceira Bateria de Exercícios

Para os resultados abaixo mostre os resultados e comandos:

1) Para os dados abaixo, utilize o teste **t-student** para encontrar se há diferença entre médias das duas amostras. As amostras foram retiradas de dois grupos de portadores de uma doença rara, eles tomaram um remédio específico para a baixa de colesterol, devemos verificar se há diferença entre as médias dos valores de colesterol, após a ingestão do remédio criado pelos laboratórios A e B, considere uma confiança de 99% e que existem igualdade das variâncias:

Lab. A	141	138	142	135	137	129
Lab. B	148	132	136	141	139	128

2) Encontre intervalos de confiança de 95% para a média de uma distribuição Normal com variância desconhecida dos pesos em gramas de rãs, dada a amostra abaixo:

12.6 11.9 18.5 9.8 12.6 11.7 12.1 13.2 13.2 9.3 12.3 12.2 9.3 11.7 18.4 12.5 9.1 8.3 12.7 8.2

14.4 Quarta Bateria de Exercícios

Para os resultados abaixo mostre os resultados e comandos:

1) Construa uma função chamada "calculator" que calculará automaticamente o valor da renda de um cidadão por 12 meses, esta renda será acrescida de 20% a cada mês:

Por exemplo:

Salário atual= 1000 reais

Aumento em janeiro =20%

Salário de fevereiro= 1200 reais (salário de janeiro + 20%)

Aumento em fevereiro =20%

Salário de março= 1440 reais (salário de fevereiro + 20%)

.....

A função deve receber o salário e o valor de percentual de aumento.

	Curso EAD Ferramenta R Prof. Grimaldo Oliveira grimaldo_lopes@hotmail.com
--	---

Ex: > `calculator(5000,30)` # quer dizer que o salário é de 5000 mil e o aumento é de 30%.

Considerações Importantes

“A alegria que se tem em pensar e aprender faz-nos pensar e aprender ainda mais”

Aristóteles (Início dos tempos)

16. Considerações Importantes

- Importante que você saiba que você pode retirar dúvidas com o professor no momento que desejar, para isso entre em contato via e-mail, o mesmo está na folha de rosto da apostila ou através do fórum dos alunos;
- Lembre-se que uma internet de banda larga ajudará na visualização dos vídeos, quando temos lentidão no acesso da internet isso influenciará na aprendizagem rápida do curso;
- Importante que você trabalhe com a ferramenta R diariamente, ou um espaço de tempo de uma aula para outra pequeno, pois isso facilitará seu entendimento;
- Lembre-se o curso tem um custo baixo, para permitir que mais colegas possam realizar o curso e retire dúvidas com os professores e colegas, não compartilhe seu usuário e senha, pois prejudica uma cadeia de profissionais que trabalharam no curso.
- Qualquer dificuldade não hesite e entre em contato com o professor do curso, passe um e-mail.

16.1 Futuras atualizações

Toda necessidade de inclusão de novos comandos e exercícios, devem ser solicitados ao professor, lembre-se você pode melhorar e muito o curso informando problemas no acesso e sobre algum erro encontrado e identificado na apostila.

Apêndice

Glossário de Siglas e Termos

“Mesmo desacreditado e ignorado por todos, não posso desistir, pois para mim, vencer é nunca desistir..”

Albert Einstein (1879-1955)

I. Glossário de Siglas e Termos

A seguir estão disponíveis em ordem alfabética, a relação de siglas e termos frequentemente utilizados durante a criação do curso.

- A -

ALUNO – É o indivíduo que recebe formação e instrução de um ou vários professores ou mestres para adquirir ou ampliar seus conhecimentos.

- E -

EAD – É uma modalidade de educação mediada por tecnologias em que alunos e professores estão separados espacial e/ou temporalmente, ou seja, não estão fisicamente presentes em um ambiente presencial de ensino-aprendizagem.

- P -

PROFESSOR - É uma pessoa que ensina uma ciência, arte, técnica ou outro conhecimento.

PROGRAMAÇÃO - É um método padronizado para comunicar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador.

- R -

FERRAMENTA R – É uma linguagem criada para a criação de cálculos estatísticos e gráficos. Foi criada originalmente por Ross Ihaka e por Robert Gentleman