Readme.rmd

Sergio Pedro R Oliveira

2023-02-23

Table of Contents

# 1 Objetivo

Estudo dirigido de linguagem R.

# 2 Livro de referência

Utilizando a Linguagem R.  
Editora: ALTA BOOKS EDITORA

# 3 Cap 1 - Instalação do R e Rstudio

* Download da linguagem R:  
  <https://www.r-project.org/>
* Download Rstudio IDE:  
  <https://posit.co/downloads/>

# 4 Cap 2 - Pacote base e funções estatísticas básicas

## 4.1 Operações matematicas basicas

| Nome da operação | Operação | Resultado |
| --- | --- | --- |
| Adição | 5+4 | [9] |
| Subtração | 6-2 | [4] |
| Multiplicação | 7\*3 | [21] |
| Divisão | 45/9 | [5] |
| Potência | 2^2 | [4] |
| Raiz | sqrt(121) | [11] |
| Exponencial | exp(0) | [1] |
| Log na base e | log(1) | [0] |
| Log na base 10 | log10(1) | [0] |
| Log na base 2 | log2(4) | [2] |
| Log na base 3 ou qualquer outra | log(9,3) | [2] |

## 4.2 Vetor

* Para criar um vetor usamos a função *c*().
* Os argumentos são separados por virgula dentro do parênteses.
* strings devem estar entre aspas duplas.  
  Ex.: *c*(“um”,“sete”,“nove”)
* Vetores são compostos de elementos todos do mesmo tipo.
* Armazenando vetores em um objeto:  
  Ex.: *obj\_qualquer* <- *c*(1,2,3)

## 4.3 Tabela de dados (**data.frame**) e **matrizes**

### 4.3.1 **data.frame**

* Uma tabela onde cada coluna é um vetor.
* Como cada coluna é um vetor, cada coluna pode ser de um tipo diferente.  
  Ex.: nome\_data.frame <- **data.frame**(vetor\_1, vetor\_2)
* Acrescentando uma nova coluna ao data.frame.  
  Ex.: nome\_data.frame <- **data.frame**(nome\_data.frame, vetor\_3)
* Para visualizar um **data.frame** podemos usar a função **View**().  
  Ex.: **View**(nome\_data.frame)

### 4.3.2 **Matrizes**

* A diferença entre **matrizes** e **data.frames**, é que no caso das matrizes todas as colunas e linhas devem ser do mesmo tipo. Enquanto nos **data.frames** as colunas podem ser de tipos diferentes.
* Para adicionar uma coluna numa matriz, usamos a função cbind().  
  Ex.: nome\_matriz <- **cbind**(vetor\_1, vetor\_2, …)
* Para adicionar uma linha numa matriz, usamos a função rbind().  
  Ex.: nome\_matriz <- **rbind**(vetor\_3, vetor\_4, …)
* Quando inserimos dados (vetor) de naturezas diferentes (tipos) numa matriz, ela converte todos os dados para um único tipo. A principio *string* (*chr*).

## 4.4 Acessando valores em posições especificadas dos objetos - **vetor**, **matriz** e **data.frame**

### 4.4.1 Caso **vetor** e **matriz**

* Podemos acessar os valores do objeto tipo **vetor** e **matriz**, informando a posição entre colchetes [].
* Para os **vetores** precisamos apenas informa a posição. A contagem da posição começa a partir do 1.  
  Ex.:  
  vetor <- c(5,18,89)  
  vetor[1]
* Para as **matrizes**, é necessario informar a posição [*linha*, *coluna*]. A contagem da posição começa a partir do 1.  
  Ex.:  
  Mc[1,2]
* Para acessar todos os valores de uma *linha* da **matriz**, podemos determinar a *linha* e deixar a *coluna* em branco.  
  Ex.: Mc[1,]
* Para acessar todos os valores de uma *coluna* da **matriz**, podemos determinar a *coluna* e deixar a *linha* em branco.  
  Ex.: Mc[,2]

### 4.4.2 Caso **data.frame**

* No caso do **data.frame** podemos acessar os valores das colunas informando, “nome do **data.frame**” “$” “nome da coluna”.  
  Sintaxe:  
  **nome\_dataframe$nome\_coluna**
* O **data.frame** também aceita as mesmas formas de acessar posições que as **matrizes**.

## 4.5 Visualizando dados

### 4.5.1 **View**() - visualização de dados

* Podemos visualizar dados de duas formas:  
  + Escrevendo o nome da variável  
    O valor dela será impressa na tela.
  + Atraves da função **View**()  
    Ao chamar a função View() e colocar dentro a variavel que queremos ver, será exibido uma nova janela com o valor da variável numa tabela.

### 4.5.2 **str**() - estrutura de objetos

* A função “**str**()” retorna a estrutura do objeto do argumento.
* Retorna diversos dados, entre eles:  
  + A classe do objeto.
  + Tamanho do objeto.
  + A lista, ou vertor, dos campos com o tipo e tamanho.
* Sintaxe:  
  **str**(*argumento*)

### 4.5.3 **summary**() - resumo de variáveis

* A função **summary**() retorna o resumo de variaveis.
* O retorno depende do argumento (se for um vetor, uma lista, um data.frame).
* O retorno para uma matriz ou **data**.**frame**, vai ser os metodos aplicados a cada campo/coluna.
* O retorno da função, no geral, retorna diversos metodos aplicados aos dados, tais como:  
  + valor mínimo
  + 1º quantil
  + valor da mediana
  + valor da media
  + 3º quantil
  + valor máximo
* Sintaxe:  
  **summary**(*nome\_variavel*)

### 4.5.4 **class**() - classe de objetos

* A função “**class**()” retorna a que classe do objeto do argemunto pertence.
* Basicamente diz se o objeto é numerico, string, vetor, lista, data.frame, matriz, …
* Sintaxe:  
  **class**(*argumento*)

## 4.6 Funções estatísticas básicas

| Função | Descrição |
| --- | --- |
| apply(D,i,f) | Retorna os valores resultantes da aplicação da função f ao objeto D, linhas i=1, ou colunas i=2. |
| c(valor1, valor2, valor3) | Concatena uma sequência de valores seja númerico ou de caracteres. Neste último caso os valores devem estar entre aspas. |
| cbind(x1, x2, …, xn) | Cria uma matriz com n colunas formada pelos vetores x1, x2, …, xn. |
| ceiling(x) | Retorna o menor inteiro maior ou igual ao valor x. |
| cor(x,y) | Calcula o coeficiente de correlação. |
| cumsum(x) | Retorna um vetor com valores acumulados em soma sobre os elementos de x. |
| cumprod(x) | Retorna um vetor com valores acumulados em produto sobre os elementos de x. |
| cummin(x) | Retorna um vetor com valores acumulados em mínimo sobre os elementos de x. |
| cummax(x) | Retorna um vetor com valores acumulados em máximo sobre os elementos de x. |
| data.frame(x1, x2, …, xn) | Cria um dataframe com os vatores x1, x2, …, xn. |
| det(M) | Calcula o determinante da matriz quadrada M. |
| dim(M) | Retorna as dimensões do objeto M. |
| diff(x) | Retorna um vetor com a diferença entre os valores de x. |
| eigen(M) | Retorna os autovalores e os autovetores da matriz quadrada M. |
| floor(x) | Retorna o maior inteiro menor ou igual a x. |
| identical(x,y) | Verifica se os vetores são idênticos. |
| intersect(x,y) | Realiza a interseção de dois conjuntos. |
| head(D) | Mostra o cabeçalho do objeto D. |
| length(x) | Calcula o comprimento do vetor x. |
| mean(x) | Calcula a média do vetor x. |
| median(x) | Calcula a mediana do vetor x. |
| min(x) | Calcula o mínimo de x. |
| max(x) | Calcula o máximo de x. |
| ncol(M) | Retorna o número de colunas da matriz M. |
| nrow(M) | Retorna o número de linhas da matriz M. |
| polyroot(x) | Encontra as raízes do polinômio de ordem n cujos coeficientes são representados no vetor x em ordem decrescente. |
| prod(x) | Multiplica os valores de x. |
| quantile(x,k) | Calcula o percentil de ordem dos valores de x. |
| Re(x) | Retorna a parte real de um vetor x. |
| rep(x,k) | Cria um vetor repetindo a sequência x k vezes. |
| round(x,k) | Arredonda o valor x com k casas decimais. |
| sd(x) | Calcula o desvio-padrão do vetor x. |
| seq(i,j,k) | Cria uma sequência de i ate j com tamanho de passo k. |
| setdiff(x,y) | Retorna um vetor contendo os elementos do conjunto diferença entre x e y. |
| setequal(x,y) | Verifica se os elementos dos vetores x e y são iguais, idenpendentemente da frequência em que aparecem no vetor. |
| solve(A,b) | Resolve Ax=b, retornando x. |
| sort(x) | Ordena os valores de vetor x em ordem crescente. |
| sort(x, decreasing = T) | Ordena os valores de x em ordem decrescente. |
| str(D) | Retorna a estrutura do objeto D. |
| sum(x) | Soma os valores de x. |
| union(x,y) | Retorna os elementos da união entre x e y. |
| var(x) | Calcula a variância do vetor x. |
| var(x,y) | Calcula a covariância entre x e y. |
| View(D) | Mostra o dataframe em janela separada. |

# 5 Cap 3 - Principais pacotes

## 5.1 Instalação de pacotes

* sintaxe de instalação:  
  **install.packages**(“*nome do pacote*”)
* sintaxe de variais instalações simultaneas:  
  **install.packages**(c(“*nome do pacote*”,“*nome do pacote*”,…), dependencies = **TRUE**)

## 5.2 Pacotes

1. Principais pacotes:

* **stringr**  
  Pacote para trabalhar com strings (texto).
* **Rmarkdown**  
  Produção de relatorios (html, pdf, doc, md).
* **knitr**  
  Interpretação e compilação do documento rmd.
* **data.table**  
  Exploração de data.frames.
* **janitor**  
  Limpeza de dados.
* **DescTools**  
  Analise descritiva de dados.
* **tidyverse**  
  conjunto de pacotes.  
  + **readr**  
    Importação e leitura de arquivos de dados.
  + **tibble**  
    estruturação de data.frame.
  + **dplyr**  
    Manipulação de data.frame.
  + **tidyr**  
    Organização de data.frame.
  + **ggplot2**  
    Visualização de dados, produção de gráficos.
  + **purr**  
    Manipulação de vetores e listas.
* **foreign**  
  Leitura e gravação de dados armazenados por algumas versões de “Epi Info”, “Octave”, “Minitab”, “S”, “SAS”, “SPSS”, “Stata”, “Systat”, “Weka” e para leitura e gravação de alguns “dBase” arquivos.
* **devtools**  
  Para instalar pacotes que não estejam no **CRAN**.

1. Pacotes auxiliares ao pacote **ggplot2**:

* **ggthemes**
* **grid**

## 5.3 Carregamento de pacotes

* Para poder utilizar o conjunto de funções de um determinado pacote, não basta apenas instalar o pacote, é preciso carrega-lo no script.
* As principais formas de carregar um pacote no script é través dos comandos *library*() e *require*().  
  **library**(*nome\_pacote*)  
  **require**(*nome\_pacote*)
* Outra possibilidade, é ao usar um função especificar a qual pacote ela pertence.  
  *nome\_pacote***::***função*.

## 5.4 Obter ajuda (informações) sobre pacotes

Duas formas de se conseguir informações sobre determinado pacote é através dos comandos:  
1. **package?***nome\_pacote*  
2. **help**(**package** = “*nome\_pacote*”)

# 6 Sites para uso Remote do R

* Alguns sites que possibilitam utilizar o R básico, sem que seja necessário instala-lo no computador.
* Uma otima saída quando necessario utilizar em algum computador público (lan houses, hotéis, laboratórios, …)

1. <http://rstudio.cloud/>
2. <http://jupyter.org/try>
3. <http://www.tutorialspoint.com/execute_r_online.php>
4. <http://github.com/datacamp/datacamp_light>
5. <http://rdrr.io/snippets>
6. <http://www.jdoodle.com/execute-r-online>
7. <http://rextester.com/l/r_online_compiler>
8. <http://rnotebook.io>

# 7 Cap 4 - R Markdown

## 7.1 Preâmbulo

### 7.1.1 **Titulo**

*title*: “Titulo desejado”

### 7.1.2 **Autor**

* Para inserir um autor:  
  *author*: “Nome do autor”
* Para inserir varios autores:  
  *author*:  
  + autor\_1^[instituto]
  + autor\_2^[instituto]

### 7.1.3 **Data**

* O comando “*date*:”, adiciona uma data ao documento.
* Podemos adicionar uma data qualquer para o documento no formato “dd/mm/aaaa”.  
  *date*: “dd/mm/aaaa”
* Outra possibilidade é usar uma função dentro de um *chunk* “r Sys.Date()”, para adicionar a data atual do sistema.  
  *date*: “r Sys.Date()”  
  Obs.: *chunk* deve ser colocado entre acentos graves.

### 7.1.4 **Tipo do Documento** (*output*)

* *output*: o tipo de saida, podem ser:  
  + Documentos:  
    - *pdf\_document*
    - *md\_document*
    - *html\_document*
    - *word\_document*
    - *odt\_document*
    - *rtf\_document*
  + Apresentação:  
    - *powerpoint\_presentation*
    - *ioslides\_presentation*
    - *beamer\_presentation*
  + mais:  
    - *flexdashboard::flex\_dashboard*
    - *github\_document*

### 7.1.5 **Sumário**

Para inserir o sumário no documento, basta colocar o comando “*doc*: *yes*” indentado dentro do tipo de saída.

### 7.1.6 Formatação desejada

Para determinar a formatação desejada, basta salvar um arquivo com o nome *estilo*.*docx*, que contenha a formatação e referenciar o arquivo, indentado dentro do tipo de arquivo, através do comando “*reference\_docx*: caminho/…/estilo.docx”.

### 7.1.7 **Abstract**

*Abstract*: “Texto de abstract”.

### 7.1.8 **Bibliografia**

* Ter um arquivo \*.bib com as referencias.
* Adicionar o arquivo \*.bib no preâmbulo do **R Markdown**, atravês do comando:  
  *bibliograpy*: caminho/arquivo.bib
* Um arquivo \*.csl com o estilo da citação.  
  Este arquivo pode ser obtido no site:  
  <https://www.zotero.org/styles>  
  Pesquisar por: “abnt”  
  Opção: “Instituto de Pesquisa Econômica Aplicada - ABNT (Português - Brasil)”
* Adicionar o arquivo \*.csl no preâmbulo do R Markdown, através do comando:  
  *csl*: caminho/arquivo.csl
* É necessario criar um capítulo no final para as referências. A bibliografia vai ser alocada no final do documento, logo neste ultímo capítulo. A bibliografio é sempre inserida ao final do documento.
* Por fim, para aparecer as referencias elas precisam ser citadas no texto.  
  As principais formas de citar uma referência num texto de **R Markdown** é:  
  + Uma citação:  
    Exemplo do comando: [@ chave\_da\_referencia]  
    Exemplo de como fica no arquivo final: (Alcoforado, 2021).
  + Mais de uma citação ao mesmo tempo:  
    Exemplo do comando: [@ chave\_da\_referencia\_1, @ chave\_da\_referencia\_2]

## 7.2 *Chunks* (códigos embutidos)

### 7.2.1 Códigos embutidos no texto

* Podemos embutir códigos ao longo do texto.
* Para inserir um código que será rodado no meio do texto, usamos um sinais de crase para abrir, definimos a linguagem (normalmente r), o comando que desejamos e um sinal de crase para fechar o código.  
  Este é um código embutido
* Para rodar pequenos comandos no meio do texto códigos embutidos é uma ótima opção.
* Exemplo:  
  O resultado do comando 1:3 é criar uma sequencia com os valores 1:3. A soma destes valores é sum(1:3).  
  O resultado do comando 1:3 é criar uma sequencia com os valores 1, 2, 3. A soma destes valores é 6.

### 7.2.2 Chunk

* Códigos em R, ou em outras linguagens, podem ser inseriodos nos documentos através de *chunks*.
* *Chunks* são blocos de programação.
* A principal forma de inserir *chunks* é:
* Três sinais de acento grave (crases) para abrir o *chunk*.
* Na primeira linha, definir a linguagem do bloco de programação:  
  + **R**
  + **Python**
  + **Julia**
  + **C++**
  + **SQL**  
    …
* Para dar um nome ao *chunk*, após definir a linguagem de programação basta colocar o nome do *chunk*. Nomear o *chunk* facilita determinar sua função dentro do relatório/documento.
* Ainda na primeira linha, considerações sobre o bloco de programação (*chunk options*):  
  + *include*  
    Mostra (*true*), ou não (*false*), o código e os resultados no arquivo finalizado. O R Markdown ainda executa o código e o resultado dele ainda pode ser usado em outro bloco de programação.  
    *include* = *false* | *true*
  + *echo*  
    Impede (*false*), ou não (*true*), que o código apareça, não afeta o resultado.  
    *echo* = *false* | *true*
  + *results*  
    “*hide*” mostra o código e omite o resultado.  
    *results* = “*hide*”
  + *message*  
    Imprede (*false*), ou não (*true*),que mensagens geradas por código apareçam no arquivo finalizado.  
    *message* = *false* | *true*
  + *warning*  
    Imprede (*false*), ou não (*true*), que avisos gerados pelo código apareçam no final.  
    *warning* = *false* | *true*
  + *fig.cap*  
    Adiciona uma legenda aos resultados gráficos.  
    *fig*.*cap* = “…”
* Bloco de programação, escrito na linguagme definida.
* Três sinais de acento grave (crases) para fechar o *chunk*.
* Outras formas de inserir *chunks* é atraves do botão *Insert*, na área superior da tela do script, do **RStudio**.
* Observação:  
  *messagem* e *warning* igual a *false* é muito utilizado quando se carrega bibliotecas (**library**) no *chunk*, evita que as mensagens do carregamento apareçam.

### 7.2.3 Configurando imagens e tabelas dentro do *chunk*

* Os comandos de configuração de imagem no *chunk* são inseridos no cabeçalho do *chunk*.
* Principais comando de configuração de imagens com *chunk*:  
  + **fig.width** =  
    Largura da figura em cm na janela gráfica.
  + **fig.height** =  
    Altura da figura em cm na janela gráfica.
  + **fig.align** =  
    Alinha a figura no arquivo final (“left”, “right” ou “center”).
  + **fig.cap** = ” “  
    Texto para legenda.
  + **dpi** =  
    Valor referente a qualidade da imagem, padrão é 72.
  + **out.width ou out.height** =  
    Porcentagem do tamanho original da imagem.

### 7.2.4 Global *Chunk*

* Para definir as opções globais que se aplicam a cada parte do seu arquivo, chame knitr::opts\_chunk$set em uma parte do código.
* O **knitr** tratará cada opção que você passar para knitr::opts\_chunk$set como um padrão global que pode ser substituído em cabeçalhos de blocos individuais.

## 7.3 Titulos e subtitulos

* Ao utilizar o comando # e em seguencia um texo, geramos um titulo.  
  # Titulo
* A cada # que adicionamos, diminuimos uma camada de subtitulos.  
  ## Subtitulo

## 7.4 Pular linha

* Para que duas frases fiquem em linhas separadas, dê dois espaços entre elas.
* Os dois espaços funcionam também para deixar uma linha em branco.
* Outra forma é adicinal “\”, tem o mesmo efeito.

## 7.5 Listas

### 7.5.1 Listas numeradas

* Basta inserir o número seguido de ponto e espaço.  
  1. Tópico da lista numerada
* A ordem das principais camadas de lista numeradas são:  
  + Número  
    1. Primeira camada
  + Algarismos romanos  
    i) Segunda camada
  + Letra  
    A. Terceira camada
* Para inserir uma lista dentro de uma outra lista, é necessario indentar os tópicos.

### 7.5.2 Listas não numeradas

* Os principais simbolos (na ordem de utilização) da lista não númerada:  
  + Asterisco(\*)
  + Mais(+)
  + Traço(-)
* Para inserir uma lista dentro de uma outra lista, é necessario indentar os tópicos.

## 7.6 Notas de rodapé (clicáveis)

* Há duas opções para criar uma nota de rodapé:

1. Escrever ao final do texto [^1] e então (pode ser logo abaixo, ou depois) escrever a nota de rodapé:  
   “Essa informação não é um consenso [^1]”  
   [^1]: Esta é uma nota de rodapé.
2. Colocar a informação da nota de rodapé no meio do texto, e o R numerará automaticamente:  
   “Essa informação não é um consenso ^[Esta é uma nota de rodapé]”

* Observação:  
  A informação da nota de rodapé deve estar separado do texto por uma linha, no primeiro caso, ou contida na nota no link clicável, como no segundo caso.
* Exemplo:  
  O RMarkdown é uma ferramenta excelente para documentar seus códigos e apresentar os resultados. As muitas funcionalidades dele são descritas detalhadamente no livro R Markdown: The Definitive Guide [[1]](#footnote-74).

## 7.7 Inserir tabelas

### 7.7.1 Formato de tabela padrão

* A tabela mais simples é atraves do padrão:  
  + Primeira linha:  
    Cabeçalho das colunas, separado por barra vertical(|).
  + Segunda linha:  
    - Tracejados (pelo menos 3), para representar cada coluna, com dois pontos onde se espera que o texto esteja alinhado:
      * Dois pontos no inicio do tracejado para representar alinhamento do texo a esquerda.
      * Dois pontos no inicio e no fim do tracejado para representar alinhamento centralizado do texto.
      * Dois ponstos no final do tracejado para representar alinhamento do texto a direita.
    - Cada coluna separada por barra vertical.
  + Terceira linha em diante:  
    Cada dado de linha em uma linha, com os dados de cada coluna separado por barras verticais.

### 7.7.2 Criador de tabelas online para R Markdown

Site que ajuda a construir tabelas para **R Markdown**:  
<https://tablesgenerator.com/markdown_tables>

### 7.7.3 Tabelas provenientes de banco de dados

#### 7.7.3.1 Mostrar todos os dados

Dentro do *chunk* chamar a variável que contém um **dataframe**, para imprimir ela na tela.

#### 7.7.3.2 Mostrar apenas os primeiros dados

* Dentro do *chunk* chamar a variável que contém um **dataframe**, e usar a função **head**() que mostra as 5 primeiras linhas. Podemos adicionar o parâmetro de quantidade de linhas desejamos apresentar.
* Exemplo:  
  **head**(*var\_dataframe*, *n\_linha*)

#### 7.7.3.3 Bibliotecas para criação de Tabelas

##### 7.7.3.3.1 **kable**

* Dentro do *chunk*, podemos chamar a biblioteca **knitr**, e usar a função **kable**(), onde podemos chamar como argumento a variável **dataframe** (e outras funções).
* A função **kable**(), apresenta uma tabela em formato mais profissional.
* Exemplo:  
  **library**(**knitr**)  
  **kable**(**head**(*var\_dataframe*,10))



##### 7.7.3.3.2 **xtable**

* A biblioteca **xtable** converte um objeto R em um objeto **xtable**, que pode ser expresso como uma tabela **LaTeX** ou **HTML**.
* Dentro do *chunk*, podemos chamar a biblioteca **xtable**, e usar a função **xtable**(), que recebe como argumentos a variável **dataframe** (e outras funções) e o *tipo* da saída para a tabela (**LaTeX** ou **HTML**).  
  library(xtable)  
  xtable(dataframe, type = "latex")

library(xtable)  
  
coluna1 <- c(1,2,3,4,5,6)  
coluna2<- c("a","b","c","d","e","f")  
tab <- data.frame(coluna1,coluna2)  
  
xtable(tab,type = "latex")  
xtable(tab,type = "html")

##### 7.7.3.3.3 **pander**

* O principal objetivo do pacote **pander** R é oferecer uma ferramenta de fácil renderização de objetos R no markdown do Pandoc.
* Um dos recursos mais populares do **pander** é pandoc.table, renderizando a maioria dos objetos R tabulares em tabelas de remarcação com várias opções de configuração:  
  + *Style* (**Estilo**)  
    - “*simple*”  
      style = "simple"
    - “*grid*”  
      style = "grid"
    - “*markdown*”  
      style = "markdown"
  + *Caption* (**Legenda**)  
    caption = "Legenda"
  + *Highlighting cells* (**Celulas destacadas**)  
    Comandos para destacar linhas, colunas e celulas.  
    As celulas pode estar em negrito e italico ao mesmo tempo.  
    - Italics (*italico*):  
      emphasize.italics.rows(1)  
      emphasize.italics.cols(2)  
      emphasize.italics.cells(which(t > 20, arr.ind = TRUE))
    - strong (**negrito**):  
      emphasize.strong.rows(1)  
      emphasize.strong.cols(1)  
      emphasize.strong.cells(which(t > 20, arr.ind = TRUE))
    - verbatim (estilo literal):  
      emphasize.verbatim.rows(1)  
      emphasize.verbatim.cols(2)  
      emphasize.verbatim.cells(which(t > 20, arr.ind = TRUE))  
      Exemplo:  
      emphasize.italics.cols(1)  
      emphasize.italics.rows(1)  
      emphasize.strong.cells(which(t > 20, arr.ind = TRUE))  
      pandoc.table(t)
  + *Justify* (**Alinhamento da celula**)  
    - Opções de alinhamento de celula:  
      * “*right*”
      * “*left*”
      * “*center*”
    - Formas de alinhamento de celula:  
      * Alinhando tudo de uma vez:  
        justify = "right"
      * Alinhando cada coluna separadamente:  
        justify = c("right","center","left")
  + *Table and Cell width* (**Largura**)  
    - split.table (**Largura tabela**) A largura máxima da tabela são 80 caracteres, caso ultrapasse esse tamanho, a tabela será quebrada e a parte excendente será inserida abaixo, como uma continuação. Para desligar essa opção e aumentar o tamanho da tabela, basta adicionar a opção *Inf*.  
      split.table = Inf
    - split.cell (**Largura celula**) O tamanho máximo da celula são 30 caracteres, caso ultrapasse esse tamanho, o texto será quebrado e adicionado a baixo, ainda na celula.  
      Para ajustar o tamanho da celula (definir o número de caracteres) existem três opções:  
      * Todas de uma vez.  
        split.cell = 40
      * Coluna por coluna.  
        split.cell = c(40,20,5)
      * Em termos de porcentagem.  
        split.cell = "40%"  
        split.cell = c("80%","20%","40%")
* Exemplo:  
  library(pander)  
  pandoc.table(dataframe, justify = "center", caption = "Exemplo de tabela")

#### 7.7.3.4 Tabela para paginas web

* Dentro do *chunk*, podemos chamar a biblioteca **rmarkdown**, e usar a função **paged\_table**(), onde podemos chamar como argumento a variável **dataframe**.
* Esse tipo de tabela é ideal para aplicações *web*.
* Separa os dados por páginas, de maneira dinâmica e com interação do usuário.
* Mostra dez linhas por página.
* Exemplo:  
  **library**(**rmarkdown**)  
  **paged\_table**(*var\_dataframe*)



## 7.8 Hiperlinks e imagens

### 7.8.1 Hiperlinks

* Sintaxe:  
  [Nome do Link](Endereço do Link)
* Exemplo:  
  [Canal do YouTube](https://www.youtube.com/)

### 7.8.2 Imagens

* Existem duas formas de pegar uma imagem são elas:  
  + Pegar a imagem de um endereço da web (igual a hiperlink).  
    ![Legenda](https://miro.medium.com/max/600/1\*sCJzUnDilAuvGrlllJeXKw.jpeg)
  + Pegar a imagem de uma pasta no computador (adicionar caminho ate a imagem).  
    ![Legenda](Cap4-R\_markdown/RMarkdown.png)
* Sintaxe:  
  ![Legenda](Endereço da Imagem)
* Exemplo:  
  

## 7.9 Equações

* As equações no **R Markdown** são escritas com a linguagem **LaTeX**.
* Para que a equação apareça no meio do texto, devemos escrevê-la entre dois cifrões: $equação$
* Para que a equação apareça no formato destacado (display), deve ser colocada entre quatro cifrões:  
  $$equação$$

## 7.10 Letras gregas

## 7.11 Formatação

* Para deixar uma palavra em **negrito**, coloque-a entre quatro asteriscos: \*\*negrito\*\*.
* Para deixar uma palavra em *itálico*, coloque-a entre dois asteriscos: \*itálico\*.
* Para deixar uma palavra em ~~tachado~~, coloque-a entre dois til: ~~tachado~~.
* Para deixar caracteres sobrescritos, coloque-os entre acentos circunflexos: ^1^.
* Para deixar caracteres subscritos, coloque-os entre til: ~1~.
* Outra forma de escrever subscritos (forma *LaTeX*), colocar no formato subscrito equação do *LaTeX*: subscrito$\_{2}$.
* Para destacar um termo como código, coloque-o entre crases (backticks): `código`.
* Para criar uma citação (quote), escreva o texto após um sinal de maior: > Citação.
* Vetores
* Frações, matrizes e chavetas
* Expressões
* Sinais e setas

# 8 Cap 5 - Pacotes do Tidyverse e identificando/mudando tipos de variaveis

1. identificando/mudando tipos de variaveis  
   1. identificando  
      uso do **is**.
   2. mudando o tipo de variavel:   
      uso do **as**.
2. pacotes do Tidyverse:

* **readr**  
  Leitura de dados.
* **tibble**  
  Tipo de data.frame.
* **magrittr**  
  Operador pipe ‘%>%’, concatena linhas de comando.
* **dplyr**  
  Manipulação de dados.  
  1. munipulação de dados:  
     + *select*  
       seleciona e retorna as colunas selecionadas da tabela.
     + *pull*  
       extrai uma coluna de uma tabela de dados e retorna ela como vetor.
     + *filter*  
       filtra linhas.
     + *distinct*  
       remove linhas com valores repetidos.
     + *arrange*  
       reordena ou combina linhas.
     + *mutate*  
       cria novas colunas.
     + *transmute*  
       cria novas colunas, mas não adiciona na base de dados.
     + *summarise*  
       sumariza valores.
     + *group\_by*  
       permite operações por grupo.
     + *add\_column*  
       adiciona novas colunas.
     + *add\_row*  
       adiciona novas linhas.
     + *rename*  
       renomeia uma coluna.
  2. combinando tabelas de dados:  
     + *bind\_cols*  
       Une duas tabelas lado a lado. acrescenta numeração as colunas repetidas.  
       É necessario que tenha o mesmo numero de linhas nas duas tabelas para fazer essa combinação.
     + *bind\_rows*  
       Une duas tabelas sobrepostas.  
       Quando não há correspondencia o comando retorna **NA**.
     + *inner\_join*  
       A tabela final será o resultado da intersecção das duas colunas de x e y, que possuem pelo menos uma coluna em comum, a coluna chave.  
       Junta duas colunas pela interseção.
     + *left\_join*  
       Une duas tabelas, definindo qual será a tabela principal e a unida a esquerda da outra. Esse fator muda a interpretação das linhas/registros correspondentes uma na outra, no caso, a tabela principal e tabela que será colocada a esquerda.  
       É necessario que tenha pelo menos uma coluna em comum, uma coluna chave.
     + *right\_join*  
       Une duas tabelas, definindo qual será a tabela principal e a unida a direita da outra. Esse fator muda a interpretação das linhas/registros correspondentes uma na outra, no caso, a tabela principal e tabela que será colocada a direita.  
       É necessario que tenha pelo menos uma coluna em comum, uma coluna chave.
     + *full\_join*  
       Une duas tabelas. Prestar atenção na junção das linhas/registros que formam novas informações, atraves da junção de correspondentes.  
       É necessario que tenha pelo menos uma coluna em comum, uma coluna chave.
     + *intersect*  
       Retorna a interseção entre tabelas.
     + *union*  
       Retorna a união de tabelas.
     + *setdiff*  
       Retorna a diferença entre tabelas.
     + *setequal*  
       Esse comando verifica se duas tabelas de dados possuem linhas com os mesmos valores, independentemente da ordem em que tais valores se apresentem. retorna **TRUE**, se os registros forem iguais, ou **FALSE**, se os registros forem diferentes.
* **tidyr**  
  Organização de dados.  
  + *pivot\_longer* ou *gather*  
    Converte a tabela de dados para o formato longo. (larga -> longo)
  + *pivot\_wider* ou *spread*  
    Converte a tabela de dados para o formato larga. (longo -> larga)
  + *separate*  
    Separa as respostas que estão em uma unica coluna para diversas colunas.
  + *unite*  
    O comando unite é utilizado para unir duas ou mais colunas em uma unica coluna.
  + *complete*  
    Completa as combinações de duas colunas, se não houver valor completa com *NA*.
  + *drop\_na*  
    Elimina as linhas, especificadas ou não, com valor NA.
  + *replace\_na*  
    Substitui o valor NA por outro valor especificado.

# 9 Cap 6 - Pacote data.table

1. **data.table**  
   * Manipulando linhas
   * Manipulando colunas
   * Sumarizando dados
   * Operando um subconjunto de dados
     + *lapply*
   * modificando dados com set:
     + *set*  
       modificando um valor.
     + *setnames*  
       modificando nome da coluna.
     + *setorder*  
       modificando ordem das linhas.
     + *setcolorder*  
       modificando ordem das colunas.

# 10 Cap 7 - Gráficos basicos e pacote ggplot2

1. Gráficos basicos:

* Gráfico de barras  
  **barplot**
* Gráfico circular (pizza)  
  **pie**
* Gráfico de linhas  
  **plot**
  + Para adicionar mais linhas no grafico.  
    **lines**
* Gráfico de dispersão  
  + Para obter a correlação.  
    **cor**(x,y)
  + Para obter o coeficiente da reta de regressão.  
    **lm**(y ~x)$coef
  + Adiciona a reta tracejada.  
    **abline**
* Histograma  
  **hist**
* **Boxplot** (diagrama de caixa)

1. Pacote **ggplot2**

* Constroi diversos tipos de graficos a partir da mesma estrutura de componentes:  
  + *data*: referente ao banco de dados.
  + *geom\_forma*: um rol de tipos possiveis de representação dos dados.
  + *coord\_system*: referente ao sistema de coordenadas, que podem ser cartesianas, polares e projeção de mapas.

1. O que precisa para fazer o grafico?  
   A. Um nome de objeto para guardar o grafico (uma variavel).  
   B. A base de dados que será utilizada para a plotagem.  
   **ggplot**(*data***=***nome\_da\_base*)  
   C. Descrever como as variaveis serão utilizadas na plotagem:  
   **aes**(*x***=**…, *y***=**…, …)  
   D. Especificar o tipo de grafico:  
   *geom\_forma*(…)  
   E. Utilizar o operador “**+**” para adicionar camadas ao objeto **ggplot** criado.  
   F. Pacotes auxiliares como *ggthemes* e *grid*, dentre outros.
2. Quais formatos podemos utilizar no ggplot2 - geom\_forma?

| Forma | Tipo de grafico |
| --- | --- |
| geom\_area ou geom\_ribbon | Produz um grafico para visualizar área sob a curva ou entre curvas. |
| geom\_bar ou geom\_col | Produz um grafico de colunas do vetor x. |
| geom\_bar+coord\_polar | Produz um grafico circular (Pizza). |
| geom\_boxplot | Produz o boxplot de x. |
| geom\_curve | Produz um grafico em curva. |
| geom\_density | Produz um grafico da densidade de x. |
| geom\_dotplot | Produz um grafico de pontos. |
| geom\_histogram | Produz um histograma do vetor x. |
| geom\_line, geom\_abline, geom\_hline, geom\_vline | Produz um grafico de linhas |
| geom\_point | Produz um grafico de dispersão entre x e y. |
| geom\_qq ou geom\_qq\_line | plota os quantis de x usando como base a curva normal. |
| geom\_tile, geom\_rect ou geom\_raster | Produz uma grade de retangulos. |
| geom\_violin | Produz um grafico em forma de violino. |

1. Nome dos argumentos para adicionar efeito em graficos do pacote ggplot2.

| Função | Efeito no grafico |
| --- | --- |
| autoplot | Produz um grafico apropriado para o tipo de variavel. |
| coord\_cartesian | Coordenada cartesiana. |
| coord\_fixed | Coordenada cartesiana com razão entre eixo x e y fixada. |
| coord\_flip | Inverte a posição dos eixos x e y. |
| coord\_polar | Coordenada polar. |
| geom\_blank | Janela em branco. |
| geom\_jitter | Produz um efeito jitter. |
| geom\_smooth | Produz uma curva suavizada. |
| geom\_text | Aplica texto a janela grafica. |
| scale\_fill\_(=brewer ou grey ou gradient) | Define a escala de cores. |
| scale\_\*\_continuos | Define parametros para o eixo x ou y continuos. |
| scale\_\*\_discrete | Define parametros para o eixo x ou y discreto. |
| scale\_\*\_manual | Define parametros para os eixos manualmente. |

* Definindo um tema para o grafico **ggplot**.  
  + *theme\_gray*  
    Fundo cinza e linhas grandes brancas.
  + *theme\_bw*  
    O classico preto e branco. Otimo para projetor.
  + *theme\_linedraw*  
    Linhas pretasde varias larguras num fundo branco. semelhante ao theme\_bw.
  + *theme\_light*  
    Semelhante ao theme\_linedraw, porem com as linhas mais cinza claro, para dar atenção aos dados.
  + *theme\_dark*  
    Versão escura do theme\_light, com o fundo escuro, util para criar linhas finas coloridas.
  + *theme\_minimal*  
    Um tema minimalista sem anotações de fundo.
  + *theme\_classic*  
    Tema classico, com linhas do eixo x e y, sem linhas de grade.
  + *theme\_void*  
    Um tema completamente vazio.

# 11 Andamento dos Estudos

## 11.1 Assunto em andamento:

Atualmente estou estudando Cap.7, pacote ggplot2.  
E revisando Cap.4 - R Markdown.

# Referências

ALCOFORADO, L. F. [**UTILIZANDO A LINGUAGEM R: conceitos, manipulação, visualização, modelagem e elaboração de relatórios**](https://altabooks.com.br/produto/utilizando-a-linguagem-r/). Rio de Janeiro: Departamento de estatística da UFF; Alta Books Editora, 2021.

1. R Markdown: The Definitive Guide. Yihui Xie, J. J. Allaire, Garrett Grolemund. Disponível em:  
   <https://bookdown.org/yihui/rmarkdown/>  
    [↑](#footnote-ref-74)