

R

Sergio Pedro R Oliveira

03 março 2023

Contents

1	Objetivo	2
2	Livro de referência	2
3	Cap 1 - Instalação do R e Rstudio	2
4	Cap 2 - Pacote base e funções estatísticas básicas	2
4.1	Operações matemáticas básicas	2
4.2	Vetor	3
4.3	Tabela de dados (data.frame) e matrizes	4
4.4	Acessando valores em posições especificadas dos objetos - vetor , matriz e data.frame . . .	5
4.5	Visualizando dados	6
4.6	Funções estatísticas básicas	8
5	Cap 3 - Principais pacotes	10
5.1	Instalação de pacotes	10
5.2	Pacotes	10
5.3	Carregamento de pacotes	11
5.4	Obter ajuda (informações) sobre pacotes	11
6	Sites para uso Remote do R	12
7	Cap 4 - R Markdown	13
7.1	Preâmbulo	13
7.2	<i>Chunks</i> (códigos embutidos)	16
7.3	Titulos e subtítulos	19
7.4	Pular linha	19
7.5	Listas	20
7.6	Notas de rodapé (cliqueáveis)	21
7.7	Inserir tabelas	22
7.8	Hiperlinks e imagens	32
7.9	Fórmulas LaTeX	33
7.10	Letras gregas	38
7.11	Formatação (Fontes)	39
7.12	Abas	39

8	Cap 5 - Pacotes do Tidyverse e identificando/mudando tipos de variáveis	40
9	Cap 6 - Pacote data.table	43
10	Cap 7 - Gráficos básicos e pacote ggplot2	44
11	Andamento dos Estudos	47
11.1	Assunto em andamento:	47
	Referências	48

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 matemáticas básicas

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]

Nome da operação	Operação	Resultado
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 princípio *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 informar a posição. A contagem da posição começa a partir do 1.
Ex.:
`vetor <- c(5,18,89)`
`vetor[1]`
- Para as **matrizes**, é necessário 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 vetor, 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 argumento pertence.
- Basicamente diz se o objeto é numerico, string, vetor, lista, data.frame, matriz, ...
- Sintaxe:
class(*argumento*)

4.6 Funções estatísticas básicas

Função	Descrição
<code>apply(D,i,f)</code>	Retorna os valores resultantes da aplicação da função <code>f</code> ao objeto <code>D</code> , linhas <code>i=1</code> , ou colunas <code>i=2</code> .
<code>c(valor1, valor2, valor3)</code>	Concatena uma sequência de valores seja numérico ou de caracteres. Neste último caso os valores devem estar entre aspas.
<code>cbind(x1, x2, ..., xn)</code>	Cria uma matriz com <code>n</code> colunas formada pelos vetores <code>x1, x2, ..., xn</code> .
<code>ceiling(x)</code>	Retorna o menor inteiro maior ou igual ao valor <code>x</code> .
<code>cor(x,y)</code>	Calcula o coeficiente de correlação.
<code>cumsum(x)</code>	Retorna um vetor com valores acumulados em soma sobre os elementos de <code>x</code> .
<code>cumprod(x)</code>	Retorna um vetor com valores acumulados em produto sobre os elementos de <code>x</code> .
<code>cummin(x)</code>	Retorna um vetor com valores acumulados em mínimo sobre os elementos de <code>x</code> .
<code>cummax(x)</code>	Retorna um vetor com valores acumulados em máximo sobre os elementos de <code>x</code> .
<code>data.frame(x1, x2, ..., xn)</code>	Cria um dataframe com os valores <code>x1, x2, ..., xn</code> .
<code>det(M)</code>	Calcula o determinante da matriz quadrada <code>M</code> .
<code>dim(M)</code>	Retorna as dimensões do objeto <code>M</code> .
<code>diff(x)</code>	Retorna um vetor com a diferença entre os valores de <code>x</code> .
<code>eigen(M)</code>	Retorna os autovalores e os autovetores da matriz quadrada <code>M</code> .
<code>floor(x)</code>	Retorna o maior inteiro menor ou igual a <code>x</code> .
<code>identical(x,y)</code>	Verifica se os vetores são idênticos.
<code>intersect(x,y)</code>	Realiza a interseção de dois conjuntos.
<code>head(D)</code>	Mostra o cabeçalho do objeto <code>D</code> .
<code>length(x)</code>	Calcula o comprimento do vetor <code>x</code> .
<code>mean(x)</code>	Calcula a média do vetor <code>x</code> .
<code>median(x)</code>	Calcula a mediana do vetor <code>x</code> .
<code>min(x)</code>	Calcula o mínimo de <code>x</code> .
<code>max(x)</code>	Calcula o máximo de <code>x</code> .
<code>ncol(M)</code>	Retorna o número de colunas da matriz <code>M</code> .
<code>nrow(M)</code>	Retorna o número de linhas da matriz <code>M</code> .
<code>polyroot(x)</code>	Encontra as raízes do polinômio de ordem <code>n</code> cujos coeficientes são representados no vetor <code>x</code> em ordem decrescente.
<code>prod(x)</code>	Multiplica os valores de <code>x</code> .
<code>quantile(x,k)</code>	Calcula o percentil de ordem $0 \leq x \leq 1$ dos valores de <code>x</code> .
<code>Re(x)</code>	Retorna a parte real de um vetor <code>x</code> .
<code>rep(x,k)</code>	Cria um vetor repetindo a sequência <code>x</code> <code>k</code> vezes.
<code>round(x,k)</code>	Arredonda o valor <code>x</code> com <code>k</code> casas decimais.
<code>sd(x)</code>	Calcula o desvio-padrão do vetor <code>x</code> .
<code>seq(i,j,k)</code>	Cria uma sequência de <code>i</code> até <code>j</code> com tamanho de passo <code>k</code> .
<code>setdiff(x,y)</code>	Retorna um vetor contendo os elementos do conjunto diferença entre <code>x</code> e <code>y</code> .
<code>setequal(x,y)</code>	Verifica se os elementos dos vetores <code>x</code> e <code>y</code> são iguais, independentemente da frequência em que aparecem no vetor.
<code>solve(A,b)</code>	Resolve $Ax=b$, retornando <code>x</code> .
<code>sort(x)</code>	Ordena os valores de vetor <code>x</code> em ordem crescente.
<code>sort(x, decreasing = T)</code>	Ordena os valores de <code>x</code> em ordem decrescente.

Função	Descrição
<code>str(D)</code>	Retorna a estrutura do objeto D.
<code>sum(x)</code>	Soma os valores de x.
<code>union(x,y)</code>	Retorna os elementos da união entre x e y.
<code>var(x)</code>	Calcula a variância do vetor x.
<code>var(x,y)</code>	Calcula a covariância entre x e y.
<code>View(D)</code>	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 variáveis instalações simultâneas:
`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 relatórios (html, pdf, doc, md).
- **knitr**
Interpretação e compilação do documento rmd.
- **data.table**
Exploração de data.frames.
- **janitor**
Limpeza de dados.
- **DescTools**
Análise 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.
 - **purrr**
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**.
2. 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 carregá-lo no script.
- As principais formas de carregar um pacote no script é através 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 instalá-lo no computador.
- Uma ótima saída quando necessário 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://rdr.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 Título

title: “Título 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 (modelo inglês).
date: “r Sys.Date()”
- Outra opção é usar o a função dentro de um *chunk* “r format(Sys.time(), ‘%d %B %Y’)”. A data será gerada no modelo: 02 agosto 2004.
date: “r format(Sys.time(), ‘%d %B %Y’)”
Obs.: *chunk* deve ser colocado entre acentos graves.

7.1.4 Tipo do Documento (*output*)

- *output*: o tipo de saída, 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.
- O comando **number_sections: true** adiciona numeração aos capítulos do sumário.

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**, através do comando:
bibliography: 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 necessário criar um capítulo no final para as referências. A bibliografia vai ser alocada no final do documento, logo neste último capítulo. A bibliografia é sempre inserida ao final do documento.
- Por fim, para aparecer as referências 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 inseridos 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 linguagem definida.
- Três sinais de acento grave (crases) para fechar o *chunk*.
- Outras formas de inserir *chunks* é através do botão *Insert*, na área superior da tela do script, do **RStudio**.
- Observação:
mensagem 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 sequencia um texto, 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 necessário indentar os tópicos.

7.5.2 Listas não numeradas

- Os principais símbolos (na ordem de utilização) da lista não numerada:
 - Asterisco(*)
 - Mais(+)
 - Traço(-)
- Para inserir uma lista dentro de uma outra lista, é necessário 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.

¹R Markdown: The Definitive Guide. Yihui Xie, J. J. Allaire, Garrett G. Grolemund. Disponível em: <https://bookdown.org/yihui/rmarkdown/>

7.7 Inserir tabelas

7.7.1 Formato de tabela padrão

- A tabela mais simples é através do padrão:
 - Primeira linha:
Cabecalho 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 início do tracejado para representar alinhamento do texto a esquerda.
 - Dois pontos no início e no fim do tracejado para representar alinhamento centralizado do texto.
 - Dois pontos 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.
- Argumentos do **kable**:

- **format**

Tipos de formatos que a tabela pode ser representada.

```
knitr::kable(head(mtcars[, 1:4]), "pipe")
```

- * pipe

- * simple

- * latex

- * html

- * rst

- **col.names**

O nome das colunas.

Podemos usar o argumento **col.names** para substituir os nomes das colunas por um vetor de novos nomes.

```
knitr::kable(iris, col.names = c('We', 'Need', 'Five', 'Names', 'Here'))
```

- **row.names**

Adiciona nome as linhas.

- **align**

Para alterar o alinhamento das colunas da tabela.

Podemos usar um vetor contendo os valores consistindo dos caracteres **l** (esquerda), **c** (centro) e **r** (direita).

```
kable(..., align = c("l","c",...))
```

ou

```
knitr::kable(iris2, align = "lccrr")
```

- **caption**

Adiciona uma legenda a tabela.

```
knitr::kable(iris2, caption = "An example table caption.")
```

- **digits**

Define o número máximo de casas decimais.

```
knitr::kable(d, digits = 4)
```

```
knitr::kable(d, digits = c(5, 0, 2))
```

- **format.args**

Define o formato me que os números serão apresentados.

* scientific

Se é no formato científico (**true** ou **false**).

```
knitr::kable(d, digits = 3, format.args = list(scientific = FALSE))
```

* big.mark

Como será a separação para números grandes.

```
knitr::kable(d, digits = 3, format.args = list(big.mark = ",", scientific = FALSE))
```

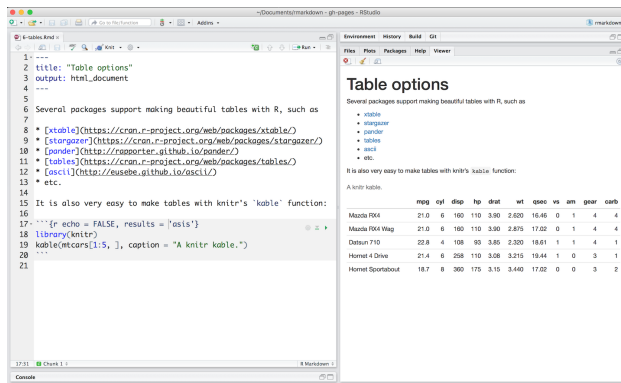
— escape

Ativa (**TRUE**) e desativa (**FALSE**) os caracteres especiais.

```
knitr::kable(d, format = "latex", escape = TRUE)
```

- Exemplo:

```
library(knitr)
kable(head(var_dataframe,10))
```



7.7.3.3.2 kableExtra

- Para mais opções de formatação do `knitr::kable`, temos o pacote `kableExtra`.
- `kableExtra` é um pacote complementar ao `knitr::kable`, por conta disto, é necessário chamar a função `kable` (primeiramente), e concatenar as funções do pacote `kableExtra` pelo operador pipe `%>%`.

```
library(knitr)
library(kableExtra)
kable(iris) %>%
  kable_styling(latex_options = "striped")
```

- Definir o tamanho da fonte:

```
kable(head(iris, 5), booktabs = TRUE) %>%
  kable_styling(font_size = 8)
```

- Estilizar linhas e colunas específicas:

– Funções:

- * **row_spec**
Especifica a linha que vai ser estilizada.
- * **column_spec**
Especifica a coluna que vai ser estilizada.

– Estilizações:

- * negrito (**bold**)
- * italico (**italic**)
- * fundo preto (**background**)
- * fonte branca (**color**)
- * sublinhado (**underline**)
- * espaçamento (**monospace**)
- * ângulo (**angle**)

```
kable(head(iris, 5), align = 'c', booktabs = TRUE) %>%
  row_spec(1, bold = TRUE, italic = TRUE) %>%
  row_spec(2:3, color = 'white', background = 'black') %>%
  row_spec(4, underline = TRUE, monospace = TRUE) %>%
  row_spec(5, angle = 45) %>%
  column_spec(5, strikeout = TRUE)
```

- Alterar o tamanho da tabela, preenche todo espaço disponível (**full_width**).

```
kable(head(dados, 10), col.names = c("Gênero", "Álcool", "Memória", "Latência")) %>%
  kable_styling(full_width = FALSE)
```

- **bootstrap_options**

- Cores alternadas entre linhas (**bootstrap_options** = c("striped")).

```
kable(head(dados, 10), col.names = c("Gênero", "Álcool", "Memória", "Latência")) %>%
  kable_styling(full_width = F, bootstrap_options = c("striped"))
```

- Deixando a tabela mais condensada/junta (**bootstrap_options** = c("striped", "condensed")).

```
kable(head(dados, 10), col.names = c("Gênero", "Álcool", "Memória", "Latência")) %>%
  kable_styling(full_width = F, bootstrap_options = c("striped", "condensed"))
```

- Agrupar linhas e colunas.

Podemos agrupar conjunto de linhas, ou colunas, e dar um cabeçalho para elas.

- Agrupar colunas:

Através da função **add_header_above** podemos dar nome aos agrupamentos e definir o número de colunas agrupadas.

```
iris2 <- iris[1:5, c(1, 3, 2, 4, 5)]
names(iris2) <- gsub('[.] +', '', names(iris2))
kable(iris2, booktabs = TRUE) %>%
  add_header_above(c("Length" = 2, "Width" = 2, " " = 1)) %>%
  add_header_above(c("Measurements" = 4, "More attributes" = 1))
```

- Agrupar linhas:

Através da função **pack_rows** e do argumento **index** podemos dar nome aos agrupamentos e definir o número de linhas agrupadas.

```
iris3 <- iris[c(1:2, 51:54, 101:103), ]
kable(iris3[, 1:4], booktabs = TRUE) %>%
  pack_rows(index = c("setosa" = 2, "versicolor" = 4, "virginica" = 3))
```

7.7.3.3.3 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")
```

7.7.3.3.4 **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 células.

As células podem estar em negrito e itálico 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 célula**)

* Opções de alinhamento de célula:

```
· "right"

· "left"
```

- “center”

* Formas de alinhamento de célula:

- 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 célula**) O tamanho máximo da célula são 30 caracteres, caso ultrapasse esse tamanho, o texto será quebrado e adicionado a baixo, ainda na célula.

Para ajustar o tamanho da célula (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)

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	►
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	
1-5 of 32 rows 1-10 of 12 columns										
Previous				1	2	3	4	5	6	7 Next

7.8 Hiperlinks e imagens

7.8.1 Hiperlinks

- Sintaxe:
[Nome do Link](Endereço do Link)
- Exemplo:
Canal do YouTube

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*sCJzUnDilAuvGr11lJeXKw.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 Fórmulas LaTeX

7.9.1 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: `$$equação$$`

7.9.2 Superescrito e subscritos

- Superescrito `a^2` = a^2
- Subscrito `a_2` = a_2
- Agrupado `a^{2+2}` = a^{2+2}
- Subscrito dois índices `$a_{i,j}$` = $a_{i,j}$
- Combinando super e subscrito `a_2^3` = a_2^3
- Derivadas `x'` = x'

7.9.3 Sublinhados, sobrelinhas e vetores

Fórmula	Símbolo
<code>\$\hat{a}\$</code>	\hat{a}
<code>\$\bar{b}\$</code>	\bar{b}
<code>\$\overrightarrow{a\ b}\$</code>	\overrightarrow{ab}
<code>\$\overleftarrow{c\ d}\$</code>	\overleftarrow{cd}
<code>\$\widehat{d\ e\ f}\$</code>	\widehat{def}
<code>\$\overline{g\ h\ i}\$</code>	\overline{ghi}
<code>\$\underline{j\ k\ l}\$</code>	\underline{jkl}

7.9.4 Frações, matrizes e chavetas

- Fração:
 $\frac{1}{2}$

- pmatrix:

```
 $\begin{pmatrix} x & y \\ z & v \end{pmatrix}$ 
```

$$\begin{pmatrix} x & y \\ z & v \end{pmatrix}$$

- bmatrix:

```
 $\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$ 
```

$$\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$$

- Bmatrix:

```
 $\begin{Bmatrix} x & y \\ z & v \end{Bmatrix}$ 
```

$$\begin{Bmatrix} x & y \\ z & v \end{Bmatrix}$$

- vmatrix:

```
 $\begin{vmatrix} x & y \\ z & v \end{vmatrix}$ 
```

$$\begin{vmatrix} x & y \\ z & v \end{vmatrix}$$

- Vmatrix:

```
 $\begin{Vmatrix} x & y \\ z & v \end{Vmatrix}$ 
```

$$\begin{Vmatrix} x & y \\ z & v \end{Vmatrix}$$

$$\begin{vmatrix} x & y \\ z & v \end{vmatrix}$$

- matrix:

```

 $\begin{matrix} x & y \\ z & v \end{matrix}$ 

```

```

 $x \quad y$ 
 $z \quad v$ 

```

7.9.5 Expressões

- Combinação

$$\${n \choose k} = \binom{n}{k}$$

- Função piso

$$\${\lfloor x \rfloor} = \lfloor x \rfloor$$

- Função teto

$$\${\lceil x \rceil} = \lceil x \rceil$$

- Sobrechaves

$$\${\overbrace{1+2+\cdots+100}}^{5050}$$

- Sobchaves

$$\${\underbrace{1+2+\cdots+100}_{5050}}$$

- Função por partes

$$f(n) = \begin{cases} n/2, & \text{se } n \text{ é par} \\ 3n+1, & \text{se } n \text{ é ímpar} \end{cases}$$

$$f(n) = \begin{cases} n/2, & \text{se } n \text{ é par} \\ 3n+1, & \text{se } n \text{ é ímpar} \end{cases}$$

- Limites

$$\${\lim_{n \rightarrow \infty} x_n} = \lim_{n \rightarrow \infty} x_n$$

- Integral

$$\int_{-N}^N e^x dx$$

- Integral Linear

$$\oint_C x^3 dx + 4y^2 dy$$

- Integral Múltipla

$$\iiint_V \mu(u,v,w) du dv dw$$

- Somatório

$$\sum_{k=1}^N k^2$$

- Somatório com dois índices

$$\sum_{\substack{0 \leq i < m \\ 0 \leq j < n}} k_{i,j}$$

- Produtório

$$\prod_{i=1}^N x_i$$

- Raiz n-ésima

$$f(x) \approx \sqrt[n]{x}$$

7.9.6 Sinais e setas

- Principais sinais e setas:

Fórmula	Símbolo
<code>\sim</code>	\sim
<code>\simeq</code>	\simeq
<code>\cong</code>	\cong
<code>\leq</code>	\leq
<code>\geq</code>	\geq
<code>\equiv</code>	\equiv
<code>\approx</code>	\approx
<code>\neq</code>	\neq
<code>\leftarrow</code>	\leftarrow
<code>\rightarrow</code>	\rightarrow
<code>\leftrightarrow</code>	\leftrightarrow
<code>\longleftarrow</code>	\longleftarrow
<code>\longrightarrow</code>	\longrightarrow
<code>\mapsto</code>	\mapsto
<code>\longmapsto</code>	\longmapsto
<code>\nearrow</code>	\nearrow
<code>\searrow</code>	\searrow
<code>\swarrow</code>	\swarrow
<code>\nwarrow</code>	\nwarrow
<code>\uparrow</code>	\uparrow
<code>\downarrow</code>	\downarrow
<code>\updownarrow</code>	\updownarrow

- Guia de fórmulas:
http://pt.wikipedia.org/wiki/Ajuda:Guia_de_edic~ao/F~ormulas_TeX

7.10 Letras gregas

- Expressões matemáticas, ou letras gregas, devem vir entre símbolos de \$.

Fórmula	Símbolo
<code>\$\alpha\$</code>	α
<code>\$\beta\$</code>	β
<code>\$\gamma\$</code>	γ
<code>\$\delta\$</code>	δ
<code>\$\epsilon\$</code>	ϵ
<code>\$\varepsilon\$</code>	ε
<code>\$\zeta\$</code>	ζ
<code>\$\eta\$</code>	η
<code>\$\theta\$</code>	θ
<code>\$\vartheta\$</code>	ϑ
<code>\$\iota\$</code>	ι
<code>\$\kappa\$</code>	κ
<code>\$\lambda\$</code>	λ
<code>\$\mu\$</code>	μ
<code>\$\nu\$</code>	ν
<code>\$\xi\$</code>	ξ
<code>\$\pi\$</code>	π
<code>\$\varpi\$</code>	ϖ
<code>\$\rho\$</code>	ρ
<code>\$\varrho\$</code>	ϱ
<code>\$\sigma\$</code>	σ
<code>\$\varsigma\$</code>	ς
<code>\$\tau\$</code>	τ
<code>\$\upsilon\$</code>	υ
<code>\$\phi\$</code>	ϕ
<code>\$\varphi\$</code>	φ
<code>\$\chi\$</code>	χ
<code>\$\psi\$</code>	ψ
<code>\$\omega\$</code>	ω

- Para letra maiúscula, inicie a letra na fórmula com letra maiúscula.
 $\delta = \text{\texttt{$\delta$}}$
 $\Delta = \text{\texttt{$\Delta$}}$

7.11 Formatação (Fontes)

- 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 subscritos₂ (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`.

7.12 Abas

- Aplica a um `#titulo` um comando (`{.tabset}`) que transforma em abas os `##subtitulo` com os gráficos e tabelas contidos neles.
- Muito útil para relatórios dinâmicos (**html**).
- Exemplo:
`# titulo {.tabset}`

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

1. identificando/mudando tipos de variáveis

- i. Identificando o tipo de variável:
Uso do **is**.
- ii. Mudando o tipo de variável:
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.
 - i. manipulação 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.

ii. 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)

2. 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.
- i) 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.

ii) Quais formatos podemos utilizar no ggplot2 - *geom_forma*?

Forma	Tipo de grafico
<i>geom_area</i> ou <i>geom_ribbon</i>	Produce um grafico para visualizar área sob a curva ou entre curvas.
<i>geom_bar</i> ou <i>geom_col</i>	Produce um grafico de colunas do vetor x.
<i>geom_bar</i> + <i>coord_polar</i>	Produce um grafico circular (Pizza).
<i>geom_boxplot</i>	Produce o boxplot de x.
<i>geom_curve</i>	Produce um grafico em curva.
<i>geom_density</i>	Produce um grafico da densidade de x.
<i>geom_dotplot</i>	Produce um grafico de pontos.
<i>geom_histogram</i>	Produce um histograma do vetor x.
<i>geom_line</i> , <i>geom_abline</i> , <i>geom_hline</i> , <i>geom_vline</i>	Produce um grafico de linhas
<i>geom_point</i>	Produce um grafico de dispersão entre x e y.
<i>geom_qq</i> ou <i>geom_qq_line</i>	plota os quantis de x usando como base a curva normal.
<i>geom_tile</i> , <i>geom_rect</i> ou <i>geom_raster</i>	Produce uma grade de retangulos.
<i>geom_violin</i>	Produce um grafico em forma de violino.

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

Função	Efeito no grafico
<i>autoplot</i>	Produce um grafico apropriado para o tipo de variavel.
<i>coord_cartesian</i>	Coordenada cartesiana.
<i>coord_fixed</i>	Coordenada cartesiana com razão entre eixo x e y fixada.
<i>coord_flip</i>	Inverte a posição dos eixos x e y.
<i>coord_polar</i>	Coordenada polar.
<i>geom_blank</i>	Janela em branco.
<i>geom_jitter</i>	Produce um efeito jitter.
<i>geom_smooth</i>	Produce uma curva suavizada.
<i>geom_text</i>	Aplica texto a janela grafica.
<i>scale_fill</i> (=brewer ou grey ou gradient)	Define a escala de cores.
<i>scale_*_contínuos</i>	Define parametros para o eixo x ou y contínuos.
<i>scale_*_discrete</i>	Define parametros para o eixo x ou y discreto.
<i>scale_*_manual</i>	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 pretas de 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**. Rio de Janeiro: Departamento de estatística da UFF; Alta Books Editora, 2021.