Curso de RMarkdown

Estats Consultoria

Contents

Chapter 1

Introdução

A fim de aprimorar os membros, e passar a utilizar o software para fins internos, este curso tem como propósito ensinar R Markdown para gerar documentos.

1.1 O que é

Markdown é uma linguagem de marcação usada para formatar de maneira simples os textos redigidos e converte-los em HTML. John Gruber e Aaron Swartz, os criadores desse sistema, utilizaram marcadores (#, *, !, (), []) para inserir nos textos, elementos como: títulos, listas, formatação de fonte, imagens e tabelas.

Com base nisso, R markdown é uma extensão da linguagem markdown que pode ser usado em IDEs como o R Studio possibilitando assim empregar os recursos da linguagem markdown citados acima em conjunto com a linguagem R, permitindo a melhor organização de análises, relatórios e códigos em um só documento.

1.2 Possíveis tipos de Outputs

O R Markdown apresenta várias possibilidades de outputs (tipos de renderização), tais como nos formatos de documentos, apresentações, entre outros, sendo que em cada formato há várias opções de customização. A seguir estão os principais formatos:

Documentos:

- html_document – documento no formato HTML;
- pdf_document – documento no formato PDF (via o modelo LaTeX);
- word_document – documento no formato do editor de texto Microsoft Word (docx);

- odt_document – documento no formato dos editores de texto Libre Office e OpenDocument;
- rtf_document – documento no formato Rich Text Format (rtf).

Apresentações (slides):

- ioslides_presentation – apresentação no formato HTML com ioslides;
- beamer_presentation – apresentação no formato PDF com LaTeX Beamer;
- powerpoint_presentation – apresentação no formato power point.

Outros:

- flexdashboard::flex_dashboard – apresentação interativa com dashboards;
- htm_vignette – R package vignette no format HTML
- github_document – document no format GitHub

Pode-se escolher o output desejado quando for criar um documento conforme a figura 1. Para fazê-lo, deve-se clicar em `file > new file > R Markdown`. Será aberta uma aba com quatro opções de output previamente estabelecidas, sendo elas: 1) Document (HTML, PDF e Word); 2) Presentation (HTML (ioslides); 3)HTML (slidy); 4) PDF (Beamer); 5) PowerPoint; 6) Shiny (Shiny Document e Shy Presentation) 7) From Template (GitHub document e Package Vignette)

Além disso, pode-se alterar o formato utilizando a função abaixo, sendo que `render` refere-se ao local que está salvo seu documento e `output_format` ao tipo de documento desejado, conforme os exemplos apontados no início.

```
render("teste.Rmd", output_format = "pdf_document")
```

O mesmo pode ser feito para outros formatos. Abaixo está uma lista com todos os formatos suportados por padrão com o pacote `rmarkdown`.

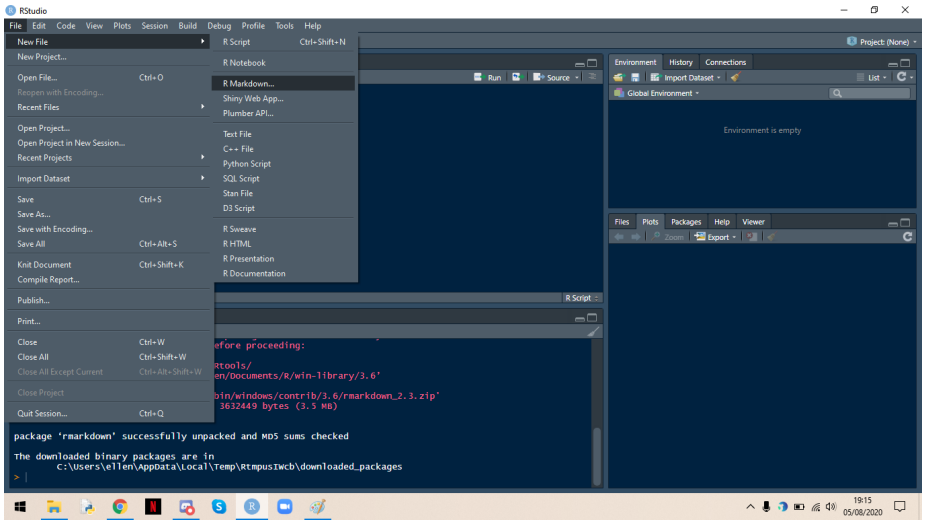
- `beamer_presentation`
- `context_document`
- `github_document`
- `html_document`
- `ioslides_presentation`
- `latex_document`
- `md_document`
- `odt_document`
- `pdf_document`
- `powerpoint_presentation`
- `rtf_document`
- `slidy_presentation`
- `word_document`

1.3 Criando o primeiro documento

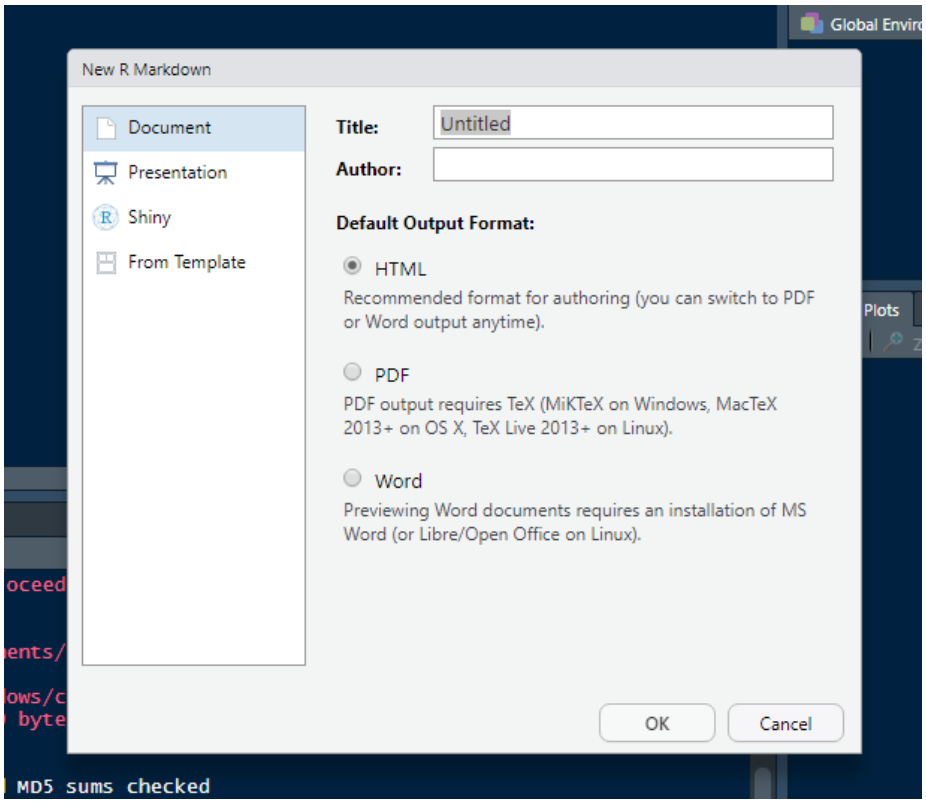
Para gerar um arquivo em R Markdown é necessário abrir o programa R Studio e instalar o pacote rmarkdown :

```
install.packages("rmarkdown")
```

Após a instalação do pacote no R Studio, siga os seguintes passos:



Em seguida, escolha o tipo de arquivo desejado:



Obs: Para gerar documentos em PDF, é necessário ter instalado em seu computador o programa Latex.

Seguindo os passos acima, você terá criado o seu primeiro documento em R Markdown. Vale ressaltar que é possível utilizar o R Markdown sem que tenha instalado o R Studio, porém, é necessário ter instalado o programa Pandoc.

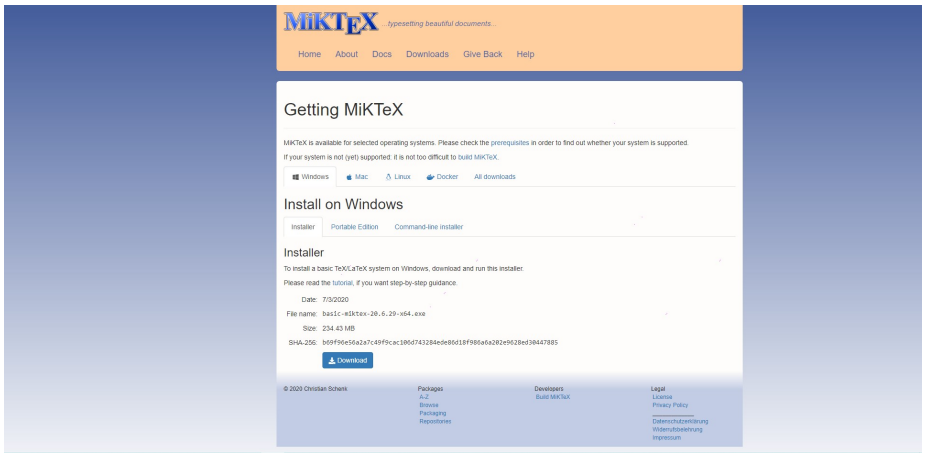
1.4 Download MiKTeX

Para exportar um arquivo PDF utilizando o R Markdown é necessário um motor LaTex pois é ele que irá converter o arquivo .tex em PDF.

É necessário que o programa MiKTeX esteja instalado no computador (download disponível em https://miktex.org/download).

1.4.1 Windows

Selecione a aba *Windows* e clique no botão de download.

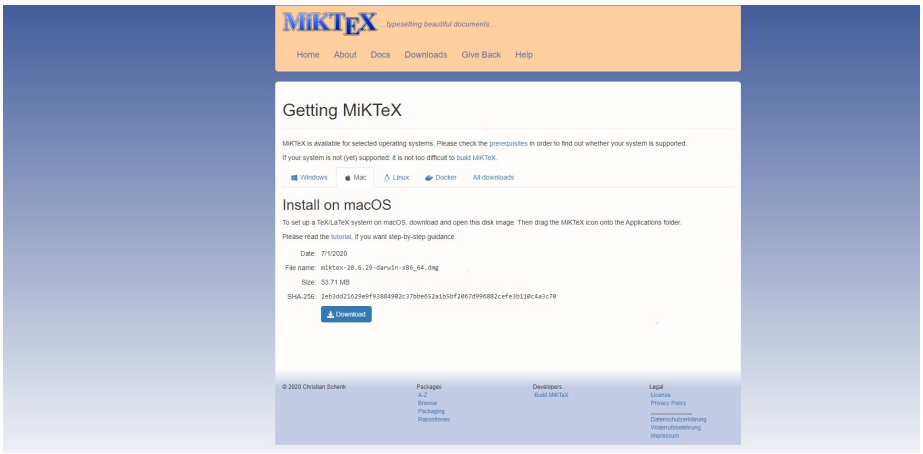


1.4.2 Linux

Selecione a aba *Linux* em seguida a aba de sua distribuição Linux para receber as instruções de instalação.

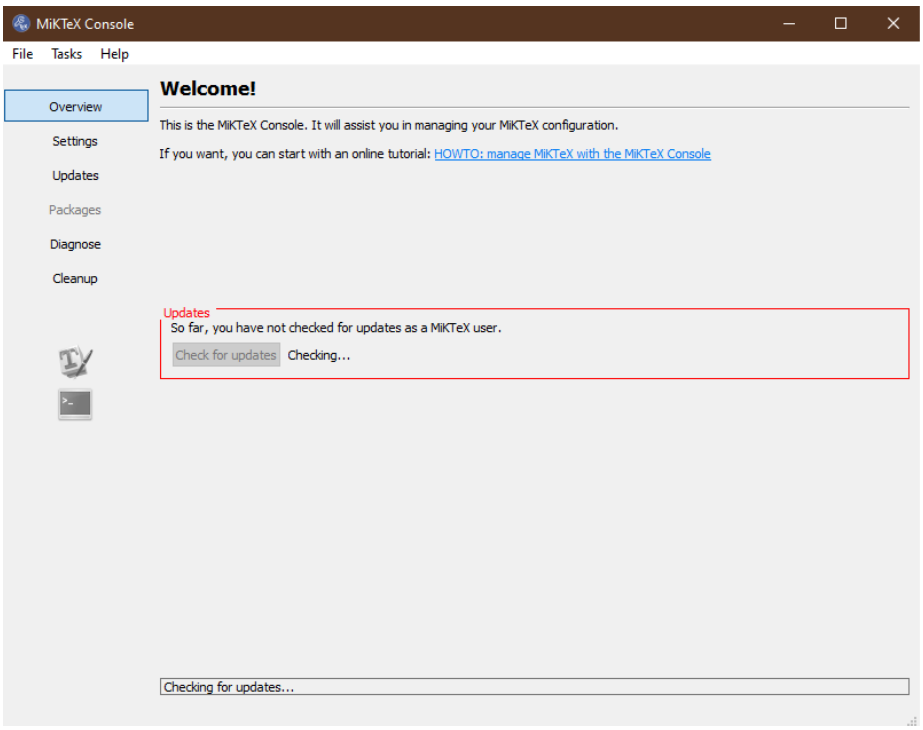
1.4.3 MacOS

Selecione a aba *MacOS* e clique no botão de download.

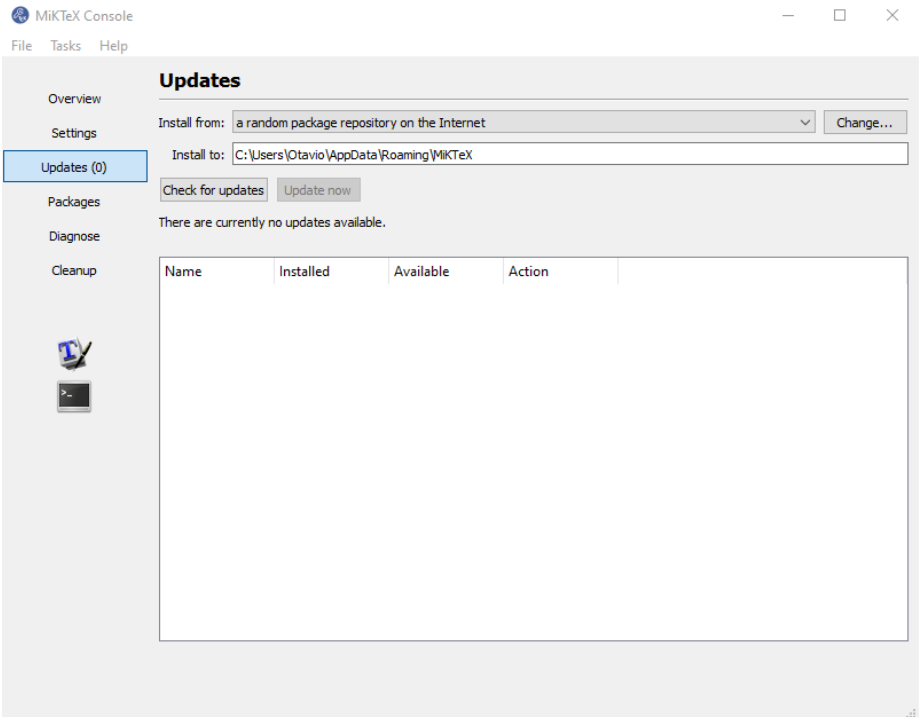


1.4.3.1 Configuração do MikTeX para Windows

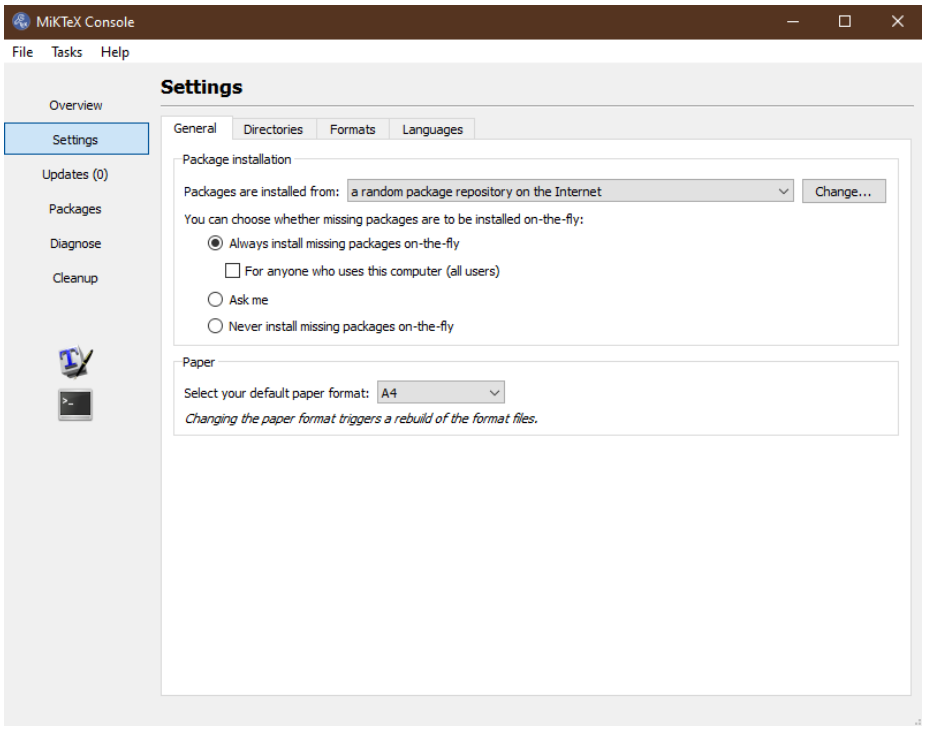
Após ter instalado o MikTeX, deve-se atualizá-lo. Para isso, ao abrir o MikTeX console seleciona-se a opção *chek for updates*.



Quando o MikTeX checar as atualizações, deve-se prosseguir para a aba *update* e selecionar *update now*.



Por fim, na aba *settings*, deve-se alterar em *You can chose whether missing packages are to be installed on-the-fly* a opção *Ask me para Always install missing packages on-the-fly*.


MiKTeX Console
File
Tasks
Help
Overview
Settings
Updates (0)
Packages
Diagnose
Cleanup
Settings
General
Directories
Formats
Languages
Package installation
Packages are installed from:
a random package repository on the Internet
Change...
You can choose whether missing packages are to be installed on-the-fly:
Always install missing packages on-the-fly
For anyone who uses this computer (all users)
Ask me
Never install missing packages on-the-fly
Paper
Select your default paper format:
A4
Changing the paper format triggers a rebuild of the format files.

Chapter 2

Sintaxe

2.1 Prêambulo

No início de um documento R Markdown, é utilizada a linguagem yaml para definir as configurações do seu arquivo. As configurações disponíveis no preâmbulo do documento R Markdown são variadas, podendo inclusive ter diferentes opções para diferentes tipos de arquivos. Uma das configurações mais importantes para se definir no seu preâmbulo é o tipo de documento a ser gerado.

2.1.1 Definindo tipo de output

A opçao de tipo de documento é definida da seguinte forma:

```
---
output: pdf_document
---
```

No código acima, `output` foi definido como `pdf_document`. Output é o formato final do seu documento, e `pdf_document` é, dentro do pacote `rmarkdown`, o formato pdf. Para gerar arquivos de formatos diferentes, é necessário somente que seja modificado a opção de `output` no preâmbulo.

Note que a opção definida no preâmbulo foi escrita entre duas linhas tracejadas. Não é possível definir as configurações fora dessas linhas tracejadas. Da mesma forma, não é possível escrever partes do documento ou rodar códigos de `R` dentro dessas linhas.

Por padrão no `yaml`, utilizado no preâmbulo de R Markdown, a opção a ser definida é escrita sem espaços, seguida de dois pontos, espaço e então a definição da opção.

Existem ainda opções dentro de outras, como por exemplo `pdf_document`, em que é possível definir configurações se o documento terá ou não sumário, quantos

níveis de cabeçalho serão utilizados no sumário, etc.

```
---
output: pdf_document:
  toc: true
  toc_depth: 3
  latex_engine: xelatex
---
```

No exemplo acima, foi definido `toc` (table of contents) como `true`, o que vai fazer com que seja gerado um sumário no documento final. Vale ressaltar que em `yaml` o valor lógico de verdadeiro é escrito com todas as letras minúsculas. A opção `toc_depth: 3`, dedfine que ao gerar o sumário, até três níveis de cabeçalho aparecerão no sumário.

2.1.2 Informações Gerais

Além do tipo de documento e das opções de cada tipo de documento, pode-se definir algumas opções gerais, como autor, título do documento, e data.

```
---
author: "Fulano"
output: pdf_document:
  toc: true
  toc_depth: 3
  latex_engine: xelatex
---
```

Seguindo com o preâmbulo já feito anteriormente, foi adicionada a opção `author`, que definirá o autor do documento. Por padrão, ele aparecerá na página inicial de diversos tipos de documento.

```
---
date: 1 de janeiro de 1970
output: pdf_document:
  toc: true
  toc_depth: 3
  latex_engine: xelatex
author: Fulano
---
```

Utilizando a opção `date` pode-se definir uma data para o documento. Do mesmo modo que o autor, a data aparece por padrão no início dos documentos. Vale ressaltar que, no exemplo acima, a escrita do nome do autor e data sem aspas não interfere no funcionamento do programa. Do mesmo modo, a troca da ordem dos elementos também não interfere, pois as opções do preâmbulo não necessitam de uma ordem específica. Contudo, as opções dentro de outras devem sempre estar abaixo da "opção mãe" com uma identação a mais.

```
---
author: Fulano
date: 1 de janeiro de 1970
title: Título
output: pdf_document:
  toc: true
  toc_depth: 3
  latex_engine: xelatex
---
```

Por fim definiu-se também a opção `title` que indica o título do documento final. Mais customizações no documento são abordadas no capítulo 4.

2.2 Textos

A linguagem R Markdown possui diversos tipos de customizações para textos. A seguir, são apresentadas as principais.

2.2.1 Títulos

É possível definir o título de uma seção, bem como seus respectivos subtítulos, ao colocar `#` antes do texto do título. A hierarquia dos títulos é definida pela quantidade de *hashtags*.

```
# Título Nível 1

## Título Nível 2

### Título Nível 3

#### Título Nível 4

##### Título Nível 5

###### Título Nível 6
```

2.2.2 Formatação de textos (negrito, itálico, sobrescrito, tachado e código)

Também é possível formatar o texto, podendo destacá-lo de diversas formas. A seguir, são exemplificadas as principais formatações.

```
**Negrito** __Negrito__
```

Negrito Negrito

```
*Italico* _italico_
```

Italico Italico

```
texto^sobrescrito^
```

texto^sobrescrito^

```
~~~tachado~~
```

~~tachado~~

2.2.3 Links

Para adicionar um link em um texto, deve-se utilizar a seguinte sintaxe:

```
[nome do link](url do link)
[Curso de RMarkdown](https://estatsej.github.io/curso_rmarkdown)
```

Exemplo: Link para a apostila o Curso de RMarkdown.

Nesse caso, o nome dado ao link aparecerá em destaque e ao clicar nele, o usuário será direcionado para o caminho especificado no link.

Caso não se deseje colocar um nome específico ou ocultar o link, pode-se simplesmente colocá-lo no meio do texto.

Exemplo: https://estatsej.github.io/curso_rmarkdown

2.2.4 Listas

Por fim, também pode-se criar listas em um documento R Markdown, podendo ser ela ordenada ou não.

2.2.4.1 Lista Ordenada

Para criar uma lista ordenada, basta numerar os itens colocando um ponto e um espaço após o número, como é mostrado a seguir.

```
1. Primeiro item
2. Segundo item
3. Terceiro item
```

1. Primeiro item
2. Segundo item
3. Terceiro item

2.2.4.2 Lista Não-Ordenada

Para listas não ordenadas, coloca-se apenas o caractere `-` ou `+`, seguido também de um espaço.

```
- Primeiro item
- Segundo item
- Terceiro item
```

- Primeiro item
- Segundo item
- Terceiro item

2.2.4.3 Subitens em uma lista

Para Adicionar subitens em uma lista, é preciso adcionar um espaço de tabulação (tab) antes do caractere que indica o item.

```
1. Item
    - Um sub-item
    - Outro sub-item
```

1. Item
    - Um sub-item
    - Outro sub-item

2.3 Inserindo Imagens

Antes de inserir a imagem escolhida pode-se definir a configuração global para todas as imagens. Lembrando que a configuração feita diretamente na imagem vai sobrepor a configuração global.

````
```{r setup include=FALSE}
library(knitr)
opts_chunk$set(echo = FALSE,
               out.width = "10%",
               fig.align = "center")
```
````

Se um gráfico ou imagem não for gerado a partir de um código feito por R, você poderá incluí-lo de duas maneiras:

- Usando a sintaxe Markdown `![texto](pasta/para/image)`. Nesse caso, você pode definir o tamanho da imagem usando os atributos `width` ou `height`, por exemplo:

```
![Logo da Estats](img/logoestats.jpeg){width=50%}
```

- Usando a função `knitr::include_graphics()` em qualquer parte do código. Você pode usar opções de configuração de tamanho como `out.width` ou `out.height` para esse exemplo:

````
```{r, echo=FALSE, out.width="50%", fig.cap="Logo da Estats"}
knitr::include_graphics("img/logoestats.jpeg")
```
````

Se você souber que deseja gerar a imagem apenas para um formato de saída específico, poderá usar uma unidade específica. Por exemplo, você pode usar

`out.width="300px"` se o formato de saída for HTML, mas no nosso exemplo o formato que usamos `out.width="50%"` é válido para qualquer saída.

E podemos aliar as imagens utilizando `fig.align`. Por exemplo, você pode centralizar imagens `fig.align="center"` ou alinhar à direita as imagens `fig.align="right'"`. Está opção funciona para saída HTML e LaTeX, mas pode não funcionar para outros formatos de saída (como o Word).

No código a seguir é utilizado quando o seu projeto tem múltiplas saídas (PDF,HTML,...) e ocorra problema na inclusão das imagens, então a sugestão para o melhor caminho foi fazer a validação para obter a saída utilizada, sabendo disso podemos fazer configuração e inserindo imagens especifica para cada saída.

````
```{r, fig.cap="Logo da Estats"}
if (knitr::is_html_output()) {
    knitr::include_graphics("img/logoestats.jpeg")
} else {
    knitr::include_graphics("img/logoestats.jpeg")
}
```
````

```r
if (knitr::is_html_output()) {
    knitr::include_graphics("img/logoestats.jpeg")
} else {
    knitr::include_graphics("img/logoestats.jpeg")
}
```

Figure 2.1: Logo da Estats

Os dois exemplos abaixo são casos de curiosidades:

- Nesse caso podemos utilizar vetor para colocar várias imagem juntas.

````
```{r image}
include_graphics(c("img1.jpeg", "img2.jpeg"))
```
````

- E nesse último caso podemos fazer repetição de uma imagem várias vezes.

````
```{r}
knitr::include_graphics(rep("img/logoestats.jpeg", 3))
```
````

```
knitr::include_graphics(rep("img/logoestats.jpeg", 3))
```

2.4 Expressões Matemáticas

As Equações inline em LaTeX podem ser escritas utilizando um par de cifrões, por exemplo, `$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$`(output atual: $f(k) = \binom{n}{k} p^k (1-p)^{n-k}$); expressões matemáticas do estilo de exibição podem ser escritas utlizando um par de cifrões duplos, por exemplo, `$$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$$`, ficando o output da seguinte maneira:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Você também pode usar ambientes matemáticos dentro `$ $` OU `$$ $$`, isto é

```
$$\begin{array}{ccc}
x_{11} & x_{12} & x_{13}\\
x_{21} & x_{22} & x_{23}
\end{array}$$
```

$$\begin{array}{ccc}
x_{11} & x_{12} & x_{13}\\
x_{21} & x_{22} & x_{23}
\end{array}$$

```
$$X = \begin{bmatrix}
1 & x_{1}\\
1 & x_{2}\\
1 & x_{3}
\end{bmatrix}$$
```

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}$$

```
$$\Theta = \begin{pmatrix}
\alpha & \beta\\
\gamma & \delta
\end{pmatrix}$$
```

$$\Theta = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

```
$$\begin{vmatrix}
a & b\\
c & d
\end{vmatrix}=ad-bc$$
```

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

2.5 Exemplos

Exemplo Sumário e Cabeçalhos

Exemplo formatação de textos

Chapter 3

Executando Código

3.1 Introdução

3.2 Flags

A lista com todas as opções de flags está disponível na documentação do **knitr**.

Abaixo estão listadas algumas das flags mais utilizadas.

- `eval`: Não executa o chunk quando recebe o parâmetro `FALSE`, porém o código do chunk será exibido no arquivo final.
- `echo`: Se recebe o parâmetro `FALSE`, o código chunk será executado, mas não exibido no relatório.
- `results`: Modifica a saída de código chunk. Essa flag possui os seguintes parâmetros: `markup`, `hold`, `asis` e `hide`. Os exemplos a seguir apresentam as modificações decorrentes de cada um.

`results='markup'`:

```
print("Curso de R Markdown")
```

```
## [1] "Curso de R Markdown"
```

`results='hold'`:

```
print("Curso de R Markdown")
```

```
## [1] "Curso de R Markdown"
```

`results='asis'`:

```
print("Curso de R Markdown")
```

```
[1] "Curso de R Markdown"
```

`results='hide'`:

```
print("Curso de R Markdown")
```

- `collapse`: Comprime o código do chunk e sua saída em apenas um bloco de código no relatório final. Exemplo:

`coollapse=TRUE`:

```
print("Curso de R markdown")
## [1] "Curso de R markdown"
```

- `warning` e `message`: Quando recebem o parâmetro `FALSE`, não serão exibidas mensagens ou avisos decorrentes de algum código do chunk no relatório final.

- `error`: Usando o parâmetro `TRUE`, se houver algum erro no código do chunk, será exibido no relatório final.

- `include`: Quando recebe o parâmetro `FALSE`, oculta o código e o resultado do chunk, mas o código será executado.

- `cache`: Passando o parâmeto `TRUE`, podemos compilar codigos e armazenar seus resultados em cache. Se o mesmo não for alterado, não será necessário compila-lo novamente.

- `fig.width and fig.height` ou `fig.dim=c(x,y)`: Dimensiona o tamanho da figura em polegadas. Exemplo: `fig.width=2 and fig.height=2` ou `fig.dim=c(2,2)`.

- `out.width and out.height`: Dimensiona a figura com a porcentagem da altura e largura desejada. Exemplo: `out.width='20%'`.

- `fig.aligh`: Define o alinhamento da figura na página. Isso ocorre de acordo com os parâmetros: `center`, `left` e `right`.

- `dev`: Altera o formato de arquivo das figuras do relatório. Podemos usar os parâmetros: `'svg'`, `'jpeg'`, `'pdf'`, `'png'`, `'bmp'`, entre outros.

- `fig.cap`: Adiciona legenda em figuras do relatório. A legenda será passada como parâmetro desta maneira: `fig.cap='legenda'`.

- `child`: Pode-se rodar outro arquivo de extensão *.Rmd* dentro de um novo código. Essa prática é recomendada para trabalhos longos que necessitem de maior organização. Para acessar o arquivo desejado é preciso passar o endereço com a seguinte sintaxe: `child='endereço\\arquivo.Rmd'`.

3.3 Linguagens Suportadas

Por meio do pacote **knitr**, que forneceu um grande número de motores de linguagem, o R Markdown suporta outras linguagens, tais como Python, Julia, C++, SQL. Esses motores de linguagem podem ser acessados dentro de blocos de código designando sua respectiva engine, as engines no knitr são:

```
names(knitr::knit_engines$get())
```

```
##  [1] "awk"         "bash"        "coffee"      "gawk"        "groovy"
##  [6] "haskell"     "lein"        "mysql"       "node"        "octave"
## [11] "perl"        "psql"        "Rscript"     "ruby"        "sas"
## [16] "scala"       "sed"         "sh"          "stata"       "zsh"
## [21] "highlight"   "Rcpp"        "tikz"        "dot"         "c"
## [26] "cc"          "fortran"     "fortran95"   "asy"         "cat"
## [31] "asis"        "stan"        "block"       "block2"      "js"
## [36] "css"         "sql"         "go"          "python"      "julia"
## [41] "sass"        "scss"        "theorem"     "lemma"       "corollary"
## [46] "proposition" "conjecture"  "definition"  "example"     "exercise"
## [51] "proof"       "remark"      "solution"
```

3.3.1 Linguagem Python

Para utilizar a linguagem python, é necessário instalar o pacote `reticulate` com o comando:

```
install.packages("reticulate")
```

Com esse pacote é possível executar todos os fragmentos de código Python em uma mesma seção. Para executar um trecho do código, usa-se o seguinte comando:

```
python.reticulate = FALSE
```

Vale ressaltar que se a versão do **kniter** utilizada for inferior a 1.18é necessário a atualizar os pacotes R. Por definição, o reticulate usa a versão padrão encontrada em seu path. Para encontrar onde esta instalado o Python no computador, usa-se o seguinte comando:

```
Sys.which("python")
```

Pode-se também escolher a versão de Python a ser utilizada como a função:

```
use_python()
```

Exemplo 1: Definindo a versão do Python.

```
library(reticulate)
use_python("/usr/local/bin/python")
```

Exemplo 2: Definindo e recuperando variáveis no Python.

```
py$name
```

Onde `name` é o nome da variável que você deseja usar na sessão. Para recuperar um valor Python também utiliza-se py$name.

Exemplo 3: Executando um bloco de código R e um bloco de código Python.

#código em R

```
x <- 42
print(x)
```

```
## [1] 42
```

#código em Python

```
x = 42 * 2
print(x)
```

```
## 84
```

Nesse caso o valor de `x` na sessão em Python é py$x, sendo diferente do valor atribuido a `x` da sessão em R.

3.3.2 Script Shell

Pode-se escrever scripts Shell em R Markdown, desde que o sistema possa executá-los, ou seja, o bash executável ou sh deve existir, o que normalmente não é um problema para usuários de Linux ou macOS. Para usuários windows, é necessário ainstalação de um software adicional como Cygwin ou o Subsistema Linux.

3.3.3 SQL

Como o mecanismo SQL usa o pacote DBI, deve-se estabelecer uma conexão DBI com um banco de dados, o que pode ser feito através do comando

```
DBI::dbConnect()
```

Pode-se usar essa conexão em um fragmento SQL através da opção de conexão, como demonstrado a seguir utilizando o SQLite.

Exemplo 4:

```
library(DBI)
db = dbConnect(RSQLite::SQLite(), dbname = "sql.sqlite")
```

Onde `sql.sqlite` é o nome do arquivo sqlite.

```
 ```{sql, connection = db}
SELECT * FROM trials
```
```

Para controlar o numero de registros exibidos, usa-se a opção `max.print` que é derivada da função sql.max.print.

O fragmento de código a seguir exemplifica os 15 primeiros registros.

```
 ```{sql, connection=db, max.print = 15}
SELECT * FROM trials
```
```

Caso deseja-se especificar que não haja nenhum limite no registro, usa-se `max.print = NA`.

Para atribuir os resultados da consulta SQL a um objeto R como um quadro de dados, basta utilizar o comando `output.var` como é mostrado a seguir:

```
 ```{sql, connection=db, output.var="trials"}
SELECT * FROM trials
```
```

Quando os resultados de uma consulta SQL são atribuídos a um quadro de dados, nenhum registro será impresso no documento. Porém é ossível imprimir manualmente o quadro de dados em um bloco R subsequente. Caso seja necesário ligar os valores das variáveis R em consultas SQL, pode-se fazê-lo prefaciando a variável R de referência com uma interrogação (?). Como mostra o exemplo a seguir:

```
subjects = 10
```

```
 ```{sql, connection=db, output.var="trials", eval = FALSE}
SELECT * FROM trials WHERE subjects >= ?subjects
```
```

Caso haja muitos fragmentos SQL, pode ser útil definir um padrão para a opção de fragmento de conexão no fragmento de configuração, de forma que não seja necessário especificar a conexão em cada fragmento individual. Conforme mostra a seguir:

```
library(DBI)
db = dbConnect(RSQLite::SQLite(), dbname = "sql.sqlite")
knitr::opts_chunk$set(connection = "db")
```

Observe que a opção de conexão deve ser uma string nomeando o objeto de conexão (não o próprio objeto). Depois de definido, pode-se executar blocos SQL sem especificar uma conexão explícita como é exemplificado a seguir:

```
 ```{sql}
SELECT * FROM trials
```
```

3.3.4 Rcpp

O mecanismo Rcpp permite a compilação de C ++ em funções R com o comando `rcpp::sourceCpp()`

O resultado é apresentado a seguir:

```
 ```{Rcpp}
#include <Rcpp.h>
using namespace Rcpp;
// [[Rcpp::export]]
NumericVector timesTwo(NumericVector x) {
return x * 2;
}
```
```

A execução desse trecho compilará o código e tornará a função C ++ com o comando `timesTwo ()` disponível para R.Pode-se armazenar em cache a compilação de blocos de código C ++ usando o cache de knitr padrão, ou seja, adicionar o comando `cache = TRUE`, da seguinte forma:

```
 ```{Rcpp, cache=TRUE}
#include <Rcpp.h>
using namespace Rcpp;
// [[Rcpp::export]]
NumericVector timesTwo(NumericVector x) {
return x * 2;
}
```
```

Em alguns casos, é desejável combinar todos os pedaços de código Rcpp em um documento em uma única unidade de compilação, pois economiza tempo. Para isso utiliza-se o comando `ref.label` junto com o comando `knitr::all_rcpp_labels()`.

Deste modo, é possível coletar todos os fragmentos do Rcpp, conforme mostra a seguir:

3.3.5 Todos os fragmentos de códigos C++ serão combinados em um único fragmento.

```
 ```{Rcpp, ref.label=knitr::all_rcpp_labels(), include=FALSE}
```
```

Primeiramente, é necessário incluir o leitor `Rcpp.h`

```
#include <Rcpp.h>
```

Depois, define-se a função:

```
// [[Rcpp::export]]
int timesTwo(int x) {
return x * 2;
}
```

Com isso, os dois fragmentos com código Rcpp serão compilados juntos no primeiro pedaço.

3.3.6 Stan

A linguagem Stan permite a incorporação da linguagem probabilística. Para tanto, utilize o comando `output.var`. A seguir é exemplificado um pouco de linguagem Stan:

```
parameters {
real y[2];
}
model {
y[1] ~ normal(0, 1);
y[2] ~ double_exponential(0, 2);
}
```

```
library(rstan)
fit = sampling(ex1)
print(fit)
```

3.3.7 Java Script

Se há a intenção de gerar um documento com saída HTML, é possível executar um Java Script dentro do R Markdown usando esta linguagem denominada js. O exemplo a seguir mostra commo mudar a cor do título para vermelho.

```
$('.title').css('color', 'red')
```

Da mesma forma, pode-se incorporar a regras CSS no documento de saída. O código a seguir exemplifica isso, mostrando como transformar o texto dentro do corpo do documento vermelho:

```
body {
color: red;
}
```

3.3.8 Julia

A linguagem Julia é suportada pelo pacote JuliaCall, de forma análoga a linguagem Python.

```
a = sqrt(2); # the semicolon inhibits printing
```

```
## 1.4142135623730951
```

3.3.9 C and Fortran

Para blocos de código que usam C ou Fortran, o **knitr** usa R CMD SHLIB para compilar o código e carregar o objeto (um arquivo `.so` no Unix ou `.dll` no Windows). Então você pode usar o comando `.C ()` ou `/ .Fortran ()` para as funções C / Fortran, como é mostrado a seguir, respectivamente:

```
void square(double *x) {
*x = *x * *x;
}
```

```
.C('square', 9)
```

```
## [[1]]
## [1] 81
```

```
.C('square', 123)
```

```
## [[1]]
## [1] 15129
```

3.4 Exemplo

Para que o código seja executado no meio de um texto, é necessário usar o caractere `, antes e depois do código descrito. Exemplo:

```
Este texto gera: `r format(Sys.Date(), "%d/%m/%Y")`.
```

Este texto gera: 14/10/2020.

O código também pode ser executado dentro de blocos. Blocos de código são inseridos da seguinte forma:

```
```{r}
format(Sys.Date(), "%d/%m/%Y")
```
```

O que gera o seguinte output ao ser compilado seu arquivo R Markdown:

```
format(Sys.Date(), "%d/%m/%Y")
```

```
## [1] "14/10/2020"
```

Table 3.1: Carros

speed	dist
4	2
4	10
7	4
7	22
8	16

Algumas das opções básicas de blocos de código já mencionadas:

````
```{r, echo = FALSE}
2 + 2
```
````

```
## [1] 4
```

````
```{r, eval = FALSE}
2 + 2
```
````

```
2 + 2
```

Exemplo de utilização de blocos de código para inserir imagens como descrito no capítulo 2

````
```{r}
knitr::include_graphics("img/logoestats.jpeg")
```
````

```
knitr::include_graphics("img/logoestats.jpeg")
```

Inserindo tabelas dentro de blocos de código:

````
```{r}
knitr::kable(cars[1:5,], caption = "Carros")
```
````

```
knitr::kable(cars[1:5,], caption = "Carros")
```

Exemplo utilizando outra linguagem suportada, neste caso python:

````
```{python}
print(5 ** 3)
```
````

```
print(5 ** 3)
```

```
## 125
```

Exemplos mais detalhados de R e de outras linguagens suportadas estão disponíveis neste link.

Chapter 4

Customização e Tipos de Arquivo

4.1 Opções avançadas no preâmbulo

4.2 Apresentação de slides

4.2.1 ioslides

Para criar uma apresentação ioslides de R Markdown, você especifica `ioslides_presentation` no output nos metadados YAML do seu documento. Você pode criar uma apresentação de slides dividida em seções usando as tags de cabeçalho `#` e `##` (você também pode criar um novo slide sem cabeçalho usando uma régua horizontal (`---`). Por exemplo, aqui está uma apresentação de slides simples:

```
---
title: "Curso de R Makrdown"
author: Departamento de Estatística
date: 12 de Agosto de 2020
output: ioslides_presentation
---

# Capítulo 1

## O que é?

- R
- RMarkdown
```

```
## Capítulo 2

- R
  - RMarkdown
```

Você também pode adicionar um subtítulo ou seção incluíndo o texto após o caractere da barra vertifical (`|`). Por exemplo:

```
## Isto é um texto | Exemplo de subtítulo ou seção
```

4.2.2 Modos de Exibição do Slide

Veja a seguir um conjunto de atalhos no teclado que permitem modos de exibição alternativos: - `f`: habilita o modo de tela cheia - `w`: habilita o modo janela - `o`: habilita o modo de visão geral - `h`: habilita o modo de highlight do código - `p`: exibe as anotações presentes

Pressione a tecla `Esc` para sair dos modos de exibição escolhidos.

4.2.3 Aparência Visual

Tamanho da Apresentação

Você pode exibir a apresentação usando um wider form ao utilizar a opção `widescreen`. Você pode especificar que um texto menor seja usado com a opção `smaller`. Por exemplo:

```
---
output:
  ioslides_presentation:
    widescreen: true
    smaller: true
---
```

Você também pode habilitar a opção `smaller` como uma opção slide-by-slide adicionando o atributo `.smaller` no cabeçario do slide:

```
## Este é um exemplo {.smaller}
```

Velocidade de Transição

Você pode customizar a velocidade de transição de um slide usando a opção `transition`. Isto pode ser `"default"`, `"slower"`, `"faster"`, ou um valor número com os números em segundos (por exemplo, `0.10`). Um exemplo:

```
---
output:
  ioslides_presentatiom:
    transition: slower
---
```

4.2.3.0.1 Destacando o código É possível selecionar subconjuntos de código para ênfase adicional, adicionando um comentário especial de "destaque" ao redor do código. Por exemplo:

```
### <b>
x <- 10
y <- x * 2
### </b>
```

A região destacada será exibida com uma fonte em negrito. Quando você quiser ajudar o público a se concentrar exclusivamente na região destacada, pressione a tecla `h` o resto do código desaparecerá.

4.2.3.0.2 Adicionando um Logotipo Você pode adicionar um logotipo à apresentação usando a opção `logo`(por padrão, o logotipo será exibido em um quadrado de 85x85 pixels). Por exemplo:

```
---
output:
  ioslides_presentation:
    logo: logo.png
---
```

O gráfico do logotipo será redimensionado para 85x85 (se necessário) e adicionado ao slide de título. Uma versão de ícone menor do logotipo será incluída no rodapé esquerdo de cada slide. O logotipo na página de título e o elemento retangular que o contém podem ser personalizados com CSS. Por exemplo:

```
.gdbar img {
  width: 300px !important;
  height: 150px !important;
  margin: 8px 8px;
}

.gdbar {
  width: 400px !important;
  height: 170px !important;
}
```

Esses seletores devem ser colocados no arquivo de texto CSS. Da mesma forma, o logotipo no rodapé de cada slide pode ser redimensionado para qualquer tamanho desejado. Por exemplo:

```
slides > slide:not(.nobackground):before {
  width: 150px;
  height: 75px;
  background-size: 150px 75px;
}
```

Isso fará com que o logotipo do rodapé tenha 150 por 75 pixels de tamanho.

4.2.3.0.3 Tabelas O modelo ioslides tem um estilo padrão atraente para tabelas, então você não deve hesitar em adicionar tabelas para apresentar conjuntos de informações mais complexos. Pandoc Markdown suporta várias sintaxes para definir tabelas, que são descritas no Manual do Pandoc.

4.2.3.0.4 Layout Avançado Você pode centralizar o conteúdo em um slide adicionando o `.flexbox` e `.vcenter` atributos para o título do slide. Por exemplo:

```
## Código {.flexbox .vcenter}
```

Você pode centralizar horizontalmente o conteúdo, encerrando-o em uma tag `div` com a classe `centered`. Por exemplo:

```
<div class="centered">
Este texto está centralizado.
</div>
```

Você pode fazer um layout de duas colunas usando a classe `columns-2`. Por exemplo:

```
<div class="columns-2">
  ![](image.png)

  - Bullet 1
  - Bullet 2
  - Bullet 3
</div>
```

Observe que o conteúdo fluirá pelas colunas, portanto, se você quiser ter uma imagem de um lado e o texto do outro, certifique-se de que a imagem tenha altura suficiente para forçar o texto para o outro lado do slide.

4.2.3.0.5 Cor do Texto Você pode colorir o conteúdo usando classes de cores básicas `red`, `blue`, `green`, `yellow`, e `gray` (ou variações deles, por exemplo, `red2`, `red3`, `blue2`, `blue3`, etc.). Por exemplo:

```
<div class="red2">
Este texto é vermelho
```

4.2.3.0.6 Modo de Apresentação Uma janela separada do apresentador também pode ser aberta (ideal para quando você está apresentando em uma tela, mas tem outra tela que é particular para você). A janela permanece sincronizada com a janela principal da apresentação e também mostra as notas do apresentador e uma miniatura do próximo slide. Para ativar o modo de apresentador, adicione `?presentme=true` ao URL da apresentação. Por exemplo:

```
minha-apresentacao.html?presentme=true
```

A janela do modo de apresentador será aberta e sempre reabrirá com a apresentação até que seja desativada com:

```
minha-apresentacao.html?presentme=false
```

Para adicionar notas do apresentador a um slide, inclua-as dentro de uma seção "notes" `div`. Por exemplo:

```
<div class="notes">
Este é a minha *nota*.

- Pode conter markdown
</div>
```

4.2.3.0.7 Impressão e Output em PDF Você pode imprimir uma apresentação ioslides a partir de navegadores que tenham um bom suporte para CSS de impressão (até o momento, o Google Chrome tem o melhor suporte). A impressão mantém a maioria dos estilos visuais da versão HTML da apresentação. Para criar uma versão PDF de uma apresentação, você pode usar o menu `Print to PDF` do Google Chrome.

4.2.3.0.8 Templates Customizados Você pode substituir o modelo Pandoc subjacente usando a opção `template`:

```
---
title: "Curso de RMarkdown"
output:
  ioslides_presentation:
    template: quarterly-report.html
---
```

No entanto, observe que o nível de personalização que pode ser alcançado é limitado em comparação com os modelos de outros formatos de saída, porque os slides são gerados por formatação personalizada escrita em Lua e, como tal, o modelo usado deve incluir a string `RENDERED_SLIDES` como pode ser encontrado no arquivo de modelo padrão com o caminho `rmarkdown:::rmarkdown_system_file("rmd/ioslides/default.html")`

4.2.4 Slidy

Para criar uma apresentação Slidy de RMarkdown, você precisa especificar o `slidy_presentation` no output nos metadados YAML do seu documento. Você pode criar uma apresentação de slides dividida em seções usando as tags de cabeçalho `# #` você também pode criar um novo slide sem cabeçalho usando uma régua horizontal (`---`). Por exemplo, aqui está uma apresentação de slides simples:

```
---
```

```
title: "Curso de R Makrdown"
author: Departamento de Estatística
date: 12 de Agosto de 2020
output: slidy_presentation
---

# Capítulo 1

## O que é?

- R
- RMarkdown

## Capítulo 2

- R
- RMakrdown
```

Modos de Exibição

Veja a seguir um conjunto de atalhos no teclado que permitem modos de exibição alternativos:

- `'C'`: Exibe o indíce
- `'F'`: Alterna a exibição do rodapé.
- `'A'`: Alterna a exibição de slides atuais para todos os slides (útil para imprimir folhetos).
- `'S'`: Dimunue o tamanho da Fonte.
- `'B'`: Aumentya o tamanho da Fonte.

Tamanho do Texto

Você pode usar a opção `font_adjustment` para aumentar ou diminuir o tamanho da fonte padrão (por exemplo, `-1` ou `+1`) para toda a apresentação. Por exemplo:

```
---
output:
  slidy_presentation:
    font_adjustment: -1
---
```

Se quiser diminuir o tamanho do texto em um slide especifíco, você pode usar o atributo de slide `.smaller`. Vejamos um exemplo:

```
## Isto é um exemplo{.smaller}
```

Se quiser aumentar o tamanho do texto em um slide especifíco, você pode usar o atributo de slide `.bigger`. Vejamos um exemplo:

```
## Isto é um exemplo{.bigger}
```

Também é possível ajustar manualmente o tamanho da fonte padrão durante a sua apresentação usando o `'S'` (smaller) e `B`(bigger).

Elementos de Rodapé

Você pode adicionar uma contagem regressiva ao rodapé de seus slides usando a opção `duration` (a duração é especificada em minutos). Por exemplo:

```
---
output:
  slidy_presentation:
    duration: 45
---
```

Você também pode adicionar um texto personalizado ao rodapé (por exemplo, o nome da sua organização, universidade ou Copyright) usando a opção `footer`. Por exemplo:

```
---
output:
  slidy_presentation:
    footer: "Copyright (c) 2020, Curso de RMarkdown"
---
```

4.2.5 Beamer

Temas

Você pode especificar os temas do Beamer usando as opções `theme`, `colortheme`, e `fonttheme`. Por exemplo:

```
---
output:
  beamer_presentation:
    theme: "AnnArbor"
    colortheme: "dolphin"
    fonttheme: "structurebold"
---
```

Nível do Slide

A opção `slide_level` define o nível de título que define slides individuais.. Por padrão, este é o nível de cabeçalho mais alto na hierarquia, seguido imediatamente pelo conteúdo, e não por outro cabeçalho, em algum lugar do documento. Este padrão pode ser sobrescrito especificando um explícito `slide_level`:

```
---
output:
```

```
  beamer_presentation:
    slide_level: 2
---
```

4.2.6 PowerPoint

Para criar uma apresentação do PowerPoint a partir do R Markdown, você especifica a opção `powerpoint_presentation` no output nos metadados YAML do seu documento. Note por favor que este formato de output só está disponóivel na versão v1.9 do **rmakrdown** e requer a última versão do Pandoc 2.10.1. (O Pandoc é um conversor de documentos, como por exemplo de Markdown para HTML). Você pode verificar as versões dos seus pacotes de **rmakrdown** e Pandoc com o comando `packageVersion('rmarkdown')` e `rmarkdown::pandoc_version()` respectivamente, no R. Abaixo está um exemplo rápido de um output PowerPoint:

```
---
title: "Curso de R Makrdown"
author: Departamento de Estatística
date: 12 de Agosto de 2020
output: powerpoint_presentation
---

# Capítulo 1

## O que é?

- R
- RMarkdown

## Capítulo 2

- R
- RMakrdown
```

O nível de slide (isto é, o nível de título que define slides individuais) é determinado da mesma forma que uma apresentação Beamer, e você pode especificar um nível explícito por meio de `slide_level` da opção sob `powerpoint_presentation`. Você também pode iniciar um novo slide sem cabeçalho usando uma régua horizontal `---`

Você pode gerar a maioria dos elementos suportados pelo Markdown do Pandoc no output do PowerPoint, como texto em negrito / itálico, notas de rodapé, marcadores, expressões matemáticas LaTeX, imagens e tabelas, etc.

Observe que imagens e tabelas sempre serão colocadas em novos slides.Os únicos elementos que podem coexistir com uma imagem ou tabela em um slide são o cabeçalho do slide e a legenda da imagem / tabela. Quando você tem um parágrafo de texto e uma imagem no mesmo slide, a imagem será movida

para um novo slide automaticamente. As imagens serão dimensionadas automaticamente para caber no slide e, se o tamanho automático não funcionar bem, você pode controlar manualmente os tamanhos das imagens: para imagens estáticas incluídas por meio da sintaxe Markdown `![]()` você pode usar o `width` e/ou `height` atributos em um par de chaves após a imagem, por exemplo, `![caption](foo.png){width=40%}` para imagens geradas dinamicamente a partir de blocos de código R, você pode usar o bloco `fig.width` e `fig.height` para controlar os tamanhos.

```
:::::: {.columns}
::: {.column width="40%"}
Content of the left column.
:::

::: {.column width="60%"}
Content of the right column.
:::
::::::
```

Templates Customizados

Como documentos do Word você pode personalizar a aparência das apresentações do PowerPoint passando um documento de referência personalizado por meio da opção `reference_doc`, por exemplo:

```
---
title: "Curso de RMarkdown"
output:
  powerpoint_presentation:
    reference_doc: curso-rmarkdown.pptx
---
```

Observe que a opção `reference_doc` requer a versão 1.9 ou superior do **rmarkdown**:

```
if (packageVersion('rmarkdown') <= '1.9') {
    install.packages('rmarkdown')  # atualiza o pacote rmarkdown do CRAN
}
```

Basicamente, qualquer modelo incluído em uma versão recente do Microsoft PowerPoint deve funcionar. Você pode criar um novo arquivo `*.pptx` do menu do PowerPoint `Arquivos -> Novo` com o template desejado, salve o novo arquivo e use-o como documento de referência (template) através da opção `reference_doc`. O Pandoc lerá os estilos no modelo e os aplicará à apresentação do PowerPoint a ser criada a partir do R Markdown.

4.2.7 Reveal.js

O pacote **revealjs** fornece um tipo de output no formato `revealjs::revealjs_presentation` que pode ser usado para criar outro estilo de slides HTML5 com base na biblioteca JavaScript **reveal.js.** Você pode instalar o pacote R do CRAN:

```
install.packages("revealjs")
```

Para criar uma apresentação reveal.js de R Markdown, você especifica `revealjs_presentation` no output nos metadados YAML do seu documento. Você pode criar uma apresentação de slides dividida em seções usando as tags de cabeçalho `#` e `##` (você também pode criar um novo slide sem cabeçalho usando uma régua horizontal (`---`). Por exemplo, aqui está uma apresentação de slides simples:

```
---
title: "Curso de R Makrdown"
author: Departamento de Estatística
date: 12 de Agosto de 2020
output: revealjs::revealjs_presentation
---

# Capítulo 1

## O que é?

- R
- RMarkdown

## Capítulo 2

- R
- RMakrdown
```

4.2.8 Xaringan

O pacote **xaringan** é uma extensão R Markdown baseada no JavaScript da biblioteca remark.js para gerar apresentações HTML5 de um estilo diferente.

O nome “xaringan” veio do Sharingan no mangá e anime japoneses “Naruto”. A palavra foi deliberadamente escolhida para ser difícil de pronunciar para a maioria das pessoas (a menos que você tenha assistido ao anime), porque seu autor (eu) amava muito o estilo e estava preocupado que se tornasse muito popular. A preocupação era um tanto ingênua , porque o estilo é realmente muito personalizável e os usuários começaram a contribuir com mais temas para o pacote posteriormente. O pacote **xaringan** é uma extensão R Markdown baseada no JavaScript da biblioteca remark.js; a biblioteca remark.js suporta apenas Markdown, e o xaringan adicionou o suporte para R Markdown, bem como outros

utilitários para tornar mais fácil construir e visualizar slides. Possibilitando que você crie apresentações ninjas!.

4.3 Documentos

4.3.1 Documento de texto OpenDocument

Para criar um documento OpenDocument Text (ODT) a partir do R Markdown, você especifica o `odt_document` no formato de saída nos metadados YAML do seu documento:

```
---
title: "Curso"
author: Estats
date: Agosto 01, 2020
output: odt_document
---
```

Semelhante a `word_document`,você, também pode fornecer um documento de referência de estilo para `odt_document` percorrer a `reference_odt` na configuração. Para obter os melhores resultados, o documento ODT de referência deve ser uma versão modificada de um arquivo ODT produzido usando rmarkdown. Por exemplo:

```
---
title: "Curso"
output:
  odt_document:
    reference_odt: meu-estilo.odt
---
```

4.3.2 Documento do Word

Para criar um documento do Word a partir do R Markdown, você especifica o `word_document` no formato de saída nos metadados YAML do seu documento:

```
---
title: "Curso"
author: Estats
date: Agosto 01, 2020
output: word_document
---
```

A característica mais notável dos documentos do Word é o modelo do Word, também conhecido como "documento de referência de estilo". Você pode especificar um documento a ser usado como referência de estilo na produção de um `*.docx` arquivo (um documento do Word). Isso permitirá que você personalize itens como margens e outras características de formatação. Para obter os melhores resultados, o documento de referência deve ser uma versão modificada de um

`.docx` arquivo produzido usando rmarkdown. O caminho de tal documento pode ser passado para o `reference_docx` argumento do `word_document` no formato de saída. Passe `"default"` para usar os estilos padrão. Por exemplo:

```
---
title: "Curso"
output:
  word_document:
    reference_docx: meu-estilo.docx
---
```

4.3.3 Documento de PDF

Para criar um documento PDF a partir do R Markdown, você especifica o `pdf_document` no formato de saída nos metadados YAML:

```
---
title: "Curso"
author: Estats
date: Agosto 01, 2020
output: pdf_document
---
```

Dentro dos documentos R Markdown que geram saída em PDF, você pode usar LaTeX bruto e até mesmo definir macros LaTeX.

Observe que a saída de PDF (incluindo slides Beamer) requer uma instalação do LaTeX:

```
install.packages('tinytex')
tinytex::install_tinytex()  # install TinyTeX
```

4.3.3.1 Índice

Você pode adicionar um sumário usando a `toc` na cofiguração e especificar a profundidade dos cabeçalhos aos quais se aplica usando a `toc_depth` na cofiguração. Por exemplo:

```
---
title: "Curso"
output:
  pdf_document:
    toc: true
    toc_depth: 2
---
```

Se a profundidade do TOC não for especificada explicitamente, o padrão é 2 (significando que todos os cabeçalhos de nível 1 e 2 serão incluídos no TOC), enquanto o padrão é 3 pol `html_document`.

Você pode adicionar numeração de seção aos cabeçalhos usando a `number_sections` na cofiguração:

```
---
title: "Curso"
output:
  pdf_document:
    toc: true
    number_sections: true
---
```

Se você estiver familiarizado com LaTeX, `number_sections: true` significa `\section{}` e `number_sections: false`significa `\section*{}` para seções em LaTeX (também se aplica a outros níveis de "seções" como `\chapter{}`, e `\subsection{}`).

4.3.3.2 Opções de imagem

Existem várias opções que afetam a saída de figuras em documentos PDF:

- `fig_width` e `fig_height` pode ser usado para controlar a largura e altura da figura padrão.
- `fig_crop` controla se o `pdfcrop` utilitário, se disponível em seu sistema, é aplicado automaticamente a figuras em PDF.
- `fig_caption` controla se as figuras são renderizadas com legendas.
- `dev` controla o dispositivo gráfico usado para renderizar figuras.

```
---
title: "Curso"
output:
  pdf_document:
    fig_width: 7
    fig_height: 6
    fig_caption: true
---
```

4.3.3.3 Impressão de Tabela

Você pode aprimorar a exibição padrão da tabela por meio da `df_print` na configuração. Os valores válidos são apresentados na Tabela abaixo:

Opção	Descrição
default	Chame o `print.data.frame` método genérico
kable	Use a `knitr::kable()` função
tibble	Use a `tibble::print.tbl_df()` função

Por Exemplo:

```
---
title: "Curso"
output:
  pdf_document:
    df_print: kable
---
```

4.3.3.4 Destaque de sintaxe

A `highlight` na configuração especifica o estilo de realce da sintaxe. Seu uso em `pdf_document` é o mesmo que `html_document`. Por exemplo:

```
---
title: "Curso"
output:
  pdf_document:
    highlight: tango
---
```

4.3.3.5 Opções LaTeX

Muitos aspectos do modelo de látex usado para criar documentos PDF podem ser personalizados usando top-level YAML metadados (note que estas opções não aparecem por baixo da `output` secção, mas sim aparecer no nível superior, juntamente com `title`, `author` e assim por diante). Por exemplo:

```
---
title: "Curso"
output: pdf_document
fontsize: 11pt
geometry: margin=1in
---
```

Algumas variáveis de metadados disponíveis são exibidas na Tabela abaixo:

Variável	Descrição
lang	Código de idioma do documento |
fontsize	Tamanho da fonte (por exemplo, `10pt`, `11pt`, ou `12pt`)
documentclass	Classe de documento LaTeX (por exemplo, `article`)
classoption	Opções para documentclass (por exemplo, `oneside`)
geometry	Opções para geometria e class(por exemplo, `margin=1in`)
mainfont, sansfont, monofont, mathfont	Fontes de documentos (funciona apenas com `xelatexe lualatex`)

Variável	Descrição
linkcolor, urlcolor, citecolor	Cor para links internos, externos e de citação

4.3.3.6 Pacotes LaTeX para citações

Por padrão, as citações são processadas por meio do `pandoc-citeproc`, o que funciona para todos os formatos de saída. Para saída em PDF, às vezes é melhor usar pacotes LaTeX para processar citações, como `natbib` ou `biblatex`. Para usar um desses pacotes, basta definir a opção `citation_package` como `natbib` ou `biblatex`, por exemplo:

```
---
output:
  pdf_document:
    citation_package: natbib
---
```

4.4 HTML

Originalmente a linguagem markdown foi feita para facilitar a geração de arquivos html, então por padrão, R Markdown te suporte para a criação de arquivos html.

Todo arquivo R Markdown sem tipo de output especificado, é transformado em html.

Para gerar arquivos html a partir de um arquivo R Markdown, é utilizado `html_document` como output:

```
---
output: html_document
---
```

4.4.1 Súmario

Como demonstrado no capítulo 2 é possível gerar um sumário com todas as seções do arquivo criado utilizando `toc_depth`, é possível também definir os níveis de cabeçalho a serem apresentados no sumário com a opção `toc_depth`.

Definir essas configurações para um arquivo html funciona da mesma forma que para um arquivo pdf, basta adicionar a opção e o nível que será definido dentro das opções de output html no seu preâmbulo, como demonstrado abaixo:

```
---
output:
  html_document:
    toc: true
```

```
    toc_depth: 3
---
```

Outra opção disponível para o cabeçalho é `toc_float`, quando definida como verdadeira, o sumário não ficara fixo no início do arquivo hmtl, ele ficará sempre visivel, independente da parte do arquivo que esteja sendo visualizada. A opção `toc_float` pode ser definida da seguinte forma:

```
---
output:
  html_document:
    toc: true
    toc_float: true
---
```

Dentro de `toc_float` existem duas outras opções a serem definidas, `collapsed`, que por padrão é definida como verdadeira, e `smooth_control`, que também é definida como verdadeira po padrão.

A opção `collapsed` define o que aparecerá no sumário o tempo todo, quando definido como verdadeiro, ele esconde as subseções de sessões que não estejam sendo visualizadas, as expandindo quando se estiver olhando essa determinada sessão, e quando definido como falso, o sumário mostra todas as seções e subseções o tempo todo.

A opção `smooth_control` controla a animação da transição para uma seção quando clicada no sumário, se estiver definida como verdadeira, a transição é animada de forma suave, se estiver definida como falsa a transição é imediata.

Segue exemplo de definição das opções `collapsed` e `smooth_control`:

```
---
output:
  html_document:
    toc: true
    toc_float:
      collapsed: false
      smooth_control: false
---
```

4.4.2 Numerando seções

É possível numerar ou não seções utilizando R Markdown, para definir essa configuração é utilizada a opção `number_sections` dentro de `html_document`. Se definida como verdadeira as seções são numeradas, se definidas como falsas as seções não são numeradas.

Exemplo:

```
---
```

```
output:
  html_document:
    toc: true
    number_sections: true
---
```

4.4.3 Abas

Gerando arquivos html, podemos adicionar abas no meio do arquivo, para isso é utilizada a classe de cabeçalho `.tabset`, isso fará com que todo sub cabeçalho feito a partir desse cabeçalho seja uma aba diferente.

Exemplo:

```
---
## Seção com abas {.tabset}

### Aba 1

Conteúdo da aba 1

### Aba 2

Conteúdo da aba 2
---
```

4.4.4 Aparência

Existem varias formas de modificar a aparência dos documentos html, desde opções no preâmbulo até a utilização de arquivos css.

No preâmbulo, utilizando a opção `theme`, é possível definir um tema bootstrap. Os temas suportados por R Markdown são: `default`, `cerulean`, `journal`, `flatly`, `darkly`, `readable`, `spacelab`, `united`, `cosmo`, `lumen`, `paper`, `sandstone`, `simplex`, e `yeti`. Por padrão a opção `theme` é definida como `null`, isso faz com que o arquivo html seja gerado sem temas bootstrap.

Outra opção de preâmbulo é a opção `highlight`, que definirá como será exibido o realce da sintaxe dos códigos do documento. As opções disponíveis são: `default`, `tango`, `pygments`, `kate`, `monochrome`, `espresso`, `zenburn`, `haddock`, `breezedark`, e `textmate`.

Exemplo:

```
---
output:
  html_document:
    theme: readable
    highlight: pygments
```

```
---
```

Utilizando arquivos CSS

Para incluir um arquivo css a seu output html, basta usar a opção `css` dentro de `html_document`:

```
---
output:
  html_document:
    css: arquivo.css
---
```

Vale ressaltar que o arquivo css deve estar na mesma pasta que o arquivo R Markdown para ser utilizado como foi no exemplo, caso o arquivo esteja em uma pasta diferente é preciso informar o caminho para o arquivo.

É possível dentro do seu arquivo R Markdown definir classes e ids para suas seções da mesma forma que se define abas. Por exemplo:

```
## Cabeçalho {#cabecalho-2 .enfase}
```

Se fosse criado um cabeçalho da forma acima, poderiamos modificar esta seção no arquivo css usado da seguinte forma:

```
#cabecalho-2 {
  color: blue;
}

.enfase {
  font-size: 1.2em;
}
```

Opções de figuras

É possível definir opções no preâmbulo configurações para as imagens geradas em blocos de código. Nessas opções podemos definir atributos como altura da imagem, largura da imagem, etc.

Utilizando a opção `fig_width` é possível definir a largura das imagens geradas, e utilizando `fig_height` é possivel controlar altura da imagem.

`fig_caption` comando para colocar legenda nas figuras.

```
---
title: "Estats"
output:
  html_document:
    fig_width: 7
    fig_height: 6
    fig_caption: true
---
```

4.4.5 Impressão de Tabela

Quando a `df_print` opção é definida como `paged`, as tabelas são impressas como tabelas HTML com suporte para paginação em linhas e colunas.

TABELA

```
---
title: "Estats"
output:
  html_document:
    df_print: paged
---
```

4.4.6 Personalização avançada

Mantendo Markdown

Quando o `knitr` processa um arquivo de entrada R Markdown, ele cria um `*.md` arquivo Markdown ( ) que é subsequentemente transformado em HTML. Se quiser manter uma cópia do arquivo Markdown após a renderização, você pode fazer isso usando a `keep_md` as opções:

```
---
title: "Estats"
output:
  html_document:
    keep_md: true
---
```

Includes

Você pode fazer uma personalização mais avançada da saída, incluindo conteúdo HTML adicional. Para incluir conteúdo no cabeçalho do documento ou antes / depois do corpo do documento, você usa a `includes` opção a seguir:

```
---
title: "Estats"
output:
  html_document:
    includes:
      in_header: header.html
      before_body: consultoria_prefix.html
      after_body: consultoria_suffix.html
---
```

Modelos personalizados

Você também pode ultilizar modelos usando o comando `template` nas opções:

```
---
```

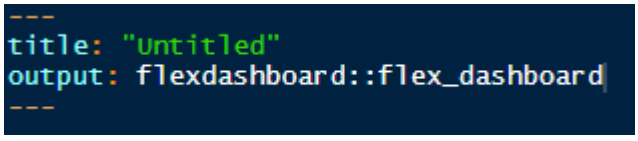
```
title: "Estats"
output:
  html_document:
    template: consultoria.html
---
```

4.5 Pacotes úteis

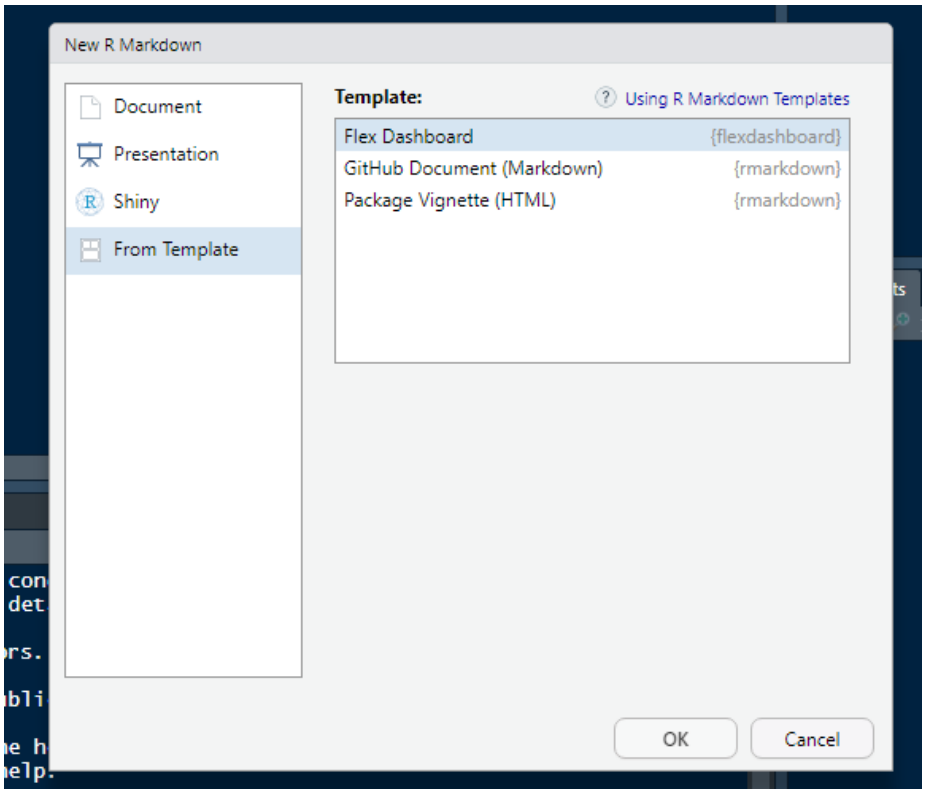
4.5.1 Flexdashboard

Um dos pacotes úteis do RMarkdown é o flexdashboard. Esse pacote possui várias funções como: - Publicar um grupo de visualizações de dados como um painel; - Incorporar uma ampla variedade de componentes, incluindo widgets HTML, gráficos R, dados tabulares, medidores, caixas de valor e anotações de texto. -Especificar layouts baseados em linha ou coluna (os componentes são redimensionados de forma inteligente para preencher o navegador e adaptados para exibição em dispositivos móveis). - Crie storyboards para apresentar sequências de visualizações e comentários relacionados; -Opcionalmente, use o Shiny para gerar visualizações dinamicamente.

Para criar um painel no RMarkdown, existem duas opções: A primeira é criar um documento em RMarkdown com flexdashboard::flex_dashboard como formato de saída.



Para a segunda opção é necessário primeiro instalar o pacote “flexdashboard” no RStudio, depois siga os passos abaixo: File -> New File -> RMarkdown Depois:



Para maior conhecimento acesse, https://rmarkdown.rstudio.com/flexdashboard/

4.5.2 Blogdown

Outro pacote que também pode ser utilizado é o "blogdown". Ele é utilizado para construção de sites. Com esse pacote, pode-se escrever uma postagem de blog ou uma página geral em um documento de RMarkdown. Para mais informações acesse, https://bookdown.org/yihui/blogdown/

4.5.3 Pkgdown

O pacote "pkgdown" facilita a construção de um site de documentação para um pacote R. Ele ajuda a organizar diferentes partes da documentação. Por exemplo, páginas de ajuda, vinhetas e notícias, de uma forma mais visual, com um estilo agradável. Para maior aprofundamento do pacote, acesse http://pkgdown.r-lib.org

4.5.4 Bookdown

O “bookdown” é um pacote que facilita a construção de livros. Ele também oferece melhorias importantes: - Recursos de formatação, como referência cruzada e numeração de figuras, equações e tabelas; - Os documentos podem ser facilmente exportados em muitos formatos adequados como PDF, e-books e sites HTML. Para maior entendimento do bookdown, acesse https://bookdown.org/yihui/bookdown/