

# Dominando *RMarkdown*

Felipe Sodré Mendes Barros

2021-04-05



# Contents

<b>1</b>	<b>Prerequisites</b>	<b>5</b>
<b>2</b>	<b>Domine <i>RMarkdown</i></b>	<b>7</b>
2.1	Sobre o <i>Markdown</i> . . . . .	7
2.2	Sobre o <i>RMarkdown</i> . . . . .	7
2.3	Vantagens . . . . .	8
2.4	Instalação . . . . .	8
2.5	Antes de começar . . . . .	8
<b>3</b>	<b>Criação do documento</b>	<b>11</b>
<b>4</b>	<b>Sintaxe Markdown</b>	<b>13</b>
<b>5</b>	<b>Trabalhando com códigos R</b>	<b>15</b>
5.1	<i>labels</i> . . . . .	15
<b>6</b>	<b><i>configurações do code chunk</i></b>	<b>17</b>
<b>7</b>	<b>Inserir <i>plot</i></b>	<b>19</b>
7.1	Inserir legenda ao <i>plot</i> . . . . .	19
<b>8</b>	<b><i>inline code</i></b>	<b>23</b>
<b>9</b>	<b>Inserindo tabela</b>	<b>25</b>
9.1	Knitir . . . . .	25

<b>10 Inserindo imagem</b>	<b>27</b>
10.1 Forçando ordem de aparição das imagens . . . . .	27
<b>11 Referência cruzada</b>	<b>29</b>
11.1 De gráfico ( <i>plot</i> ) . . . . .	29
11.2 De imagem . . . . .	31
11.3 De seção . . . . .	32
11.4 De tabela . . . . .	32
<b>12 Nota de roda-pé</b>	<b>35</b>
12.1 Primeira possibilidade . . . . .	35
12.2 Segunda possibilidade . . . . .	36
12.3 Pontos importantes: . . . . .	36
<b>13 referências bibliográficas</b>	<b>37</b>
13.1 Adicionando autor, abstract e afiliações . . . . .	39
<b>14 Notações matemáticas</b>	<b>41</b>
14.1 <i>Inline</i> . . . . .	41
14.2 <i>Em destaque</i> . . . . .	41
<b>15 Configuração Markdown</b>	<b>43</b>
15.1 Numerar as seções/capítulos . . . . .	43
15.2 Table of cotent . . . . .	43
15.3 Mais infos . . . . .	43
<b>16 References</b>	<b>45</b>

# Chapter 1

## Prerequisites

Este livro está baseado em *RMarkdown* e talvez você tenha que instalar alguma coisa de *LaTeX*.

```
install.packages("rmarkdown")  
# 'knitr'
```



## Chapter 2

# Domine *RMarkdown*

A ideia deste material é sistematizar algumas experiências que tive e buscar uma abordagem pedagógica frente a distintos materiais disponíveis na internet relacionado à elaboração de relatórios com R.

É um livro aberto, que pretendo melhorar ao longo do tempo. Se voê encontrar algum erro ou se tem algum dúvida ou que sugerir alguma melhoria, basta entrar em contato.

### 2.1 Sobre o *Markdown*

*Markdown* é uma linguagem simples para formatação de textos. A ideia é que a partir dessa linguagem você possa transformar seu texto em um documento de *Word*, *HTML* ou até mesmo *PDF*, tornando a formatação do texto mais simples e **objetiva**.

### 2.2 Sobre o *RMarkdown*

O *RMarkdown*, é um pacote que nos possibilita integrar a linguagem *R* com o *Markdown*. Dessa simples junção, podemos criar documentos ao mesmo tempo em que desenvolvemos nossas análises. Ou seja, ao mesmo tempo em que estamos analisando dados com *R* já podemos ir criando um relatório, por exemplo.

Com o *RMarkdown*, pode-se também elaborar outros tipos de documentos, como apresentações. Vamos ver primeiro como elaborar documentos de texto e, mais à frente, vemos como trabalhar com apresentações.

## 2.3 Vantagens

Usar o *RMarkdown* traz várias vantagens, das quais cito: - Fomenta a reprodutibilidade;

- Permite que mais de uma pessoa colabore em relatórios (usando *Git* e *GitHub*);
- Permite apresentar o código usado, caso isso seja pertinente;
- Facilita a criação e incorporação de gráficos/tabelas e dados do projeto ao relatório;

## 2.4 Instalação

Para trabalhar com o *RMarkdown*, basta instalar a biblioteca de mesmo nome:

```
# instalando
# install.packages("rmarkdown", dependencies = TRUE)
library(rmarkdown)
```

Você perceberá que outras bibliotecas tbm serão instaladas.

## 2.5 Antes de começar

Alguns pontos básicos:

- A extensão do arquivo é *.Rmd* (*R* + *md* [*Markdown*]), independente do tipo de documento que se pretende criar (*HTML*, *PDF* ou *Doc*);
- Ao criar um documento *Rmd* novo, o mesmo terá sempre um exemplo (conteúdo) básico;
- Todo documento *Rmd* deverá ter três elementos:
  - Cabeçalho de configuração *Markdown* (começa e termina com `---`) (Figure 2.1);

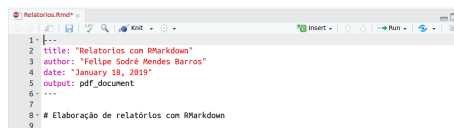


Figure 2.1: Cabeçalho de configuração *Rmd*.

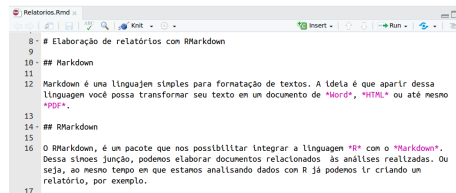




```
31 Para trabalhar com o "markdown", basta instalar a biblioteca de mesmo nome:
32 ```{r, eval=FALSE}
33 # instalando
34 # install.packages("markdown")
35 library(markdown)
36 ```
37 Você perceberá que outras bibliotecas tbm serão instaladas.
38
```

Figure 2.2: Exemplo de trecho de código *R* em um documento *Rmd*.

- Pedacos de código (*code chunk*) (começa e terminam com “`````”) ( Figure 2.2 );
- O texto em sí, usando a linguagem *Markdown* para definição de título, negrito, etc ( Figure 2.3 );



```
8 # Elaboração de relatórios com RMarkdown
9
10 ## Markdown
11
12 Markdown é uma linguagem simples para formatação de textos. A ideia é que apartir dessa
13 linguagem você possa transformar seu texto em um documento de "Word", "HTML" ou até mesmo
14 "ppt".
15
16 ## RMarkdown
17
18 O RMarkdown, é um pacote que nos possibilita integrar a linguagem "R" com o "Markdown".
19 Dessa sines junção, podemos elaborar documentos relacionados às análises realizadas. Ou
20 seja, ao mesmo tempo em que estamos analisando dados com R já podemos tr criando um
21 relatório, por exemplo.
22
```

Figure 2.3: Texto com sintaxe *Markdown*. Falaremos sobre as sintaxes mais pra frente.

Uma vez criado o documento com o texto e as análises feitas, o mesmo será **renderizado** e transformado em um documento no formato solicitado.

**Infelizmente alguns recursos mostrados nesse documento funcionam apenas para a renderização em PDF;**



## Chapter 3

# Criação do documento

Ao criar um novo documento *Rmd*, você poderá escolher entre *HTML*, *PDF* ou *Word* ( Figure 3.1 ). Escolha o que for de sua preferencia e **não se preocupe pois é possível mudar o tipo de documento**, sempre que se fizer pertinente (também veremos isso mais à frente na seção 3.0.1 ).

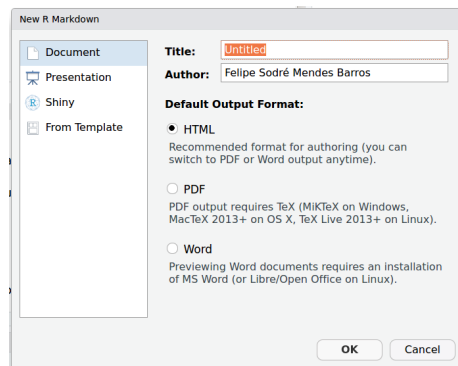


Figure 3.1: Criando um novo documento *Rmarkdown* (*\*.Rmd*)

Cada trecho de código, sempre que estiverem dentro da área delimitada para código `r` ( Figure 2.2 ), pode ser executado para se ter conhecimento de seu resultado. O resultado apresentado fará parte do documento final, uma vez que o mesmo seja **renderizado**.

A qualquer momento você pode **renderizar** o documento o transformado em *HTML*, *PDF* ou *Doc*, usando o botão *Knit*. E nesse menu que podemos mudar o formato final do documento, caso seja necessário.

### 3.0.1 Exercício I (alterando formato *output*)

1. Criar um documento *Rmarkdown* e executar-lo sem modificar nada;
  - Perceba como está a área de configuração do seu *.Rmd*;
2. Execute-o novamente, mas dessa vez alterando o formato final pelo menu do *Knit* ( Figure 3.2 );
  - Veja novamente a área de configuração do *Rmd*. O que mudou?

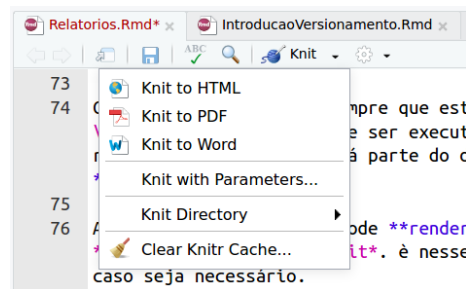


Figure 3.2: Menu do Knit

## Chapter 4

# Sintaxe Markdown

Como mencionado anteriormente, ao usar um arquivo *\*.Rmd* temos a possibilidade de mesclar a linguagem de formatação de texto *Markdown* com códigos *R*. Vamos começar entendendo a sintaxe *Markdown*.

A sintaxe *Markdown* é toda relacionada à **formatação de texto**. Como vocês puderam observar ao *renderizar* o documento padrão (exercício anterior), o símbolo `##` define que naquela linha estará o texto com um título de segunda ordem no nível hierarquico, já que o título de primeira ordem seria definido com apenas um `#`. A hierarquia de títulos é, portanto, definida pela sequência de `#`. Por exemplo, na figura Figure 2.3:

- O texto “Elaboração de relatórios com RMarkdown” será o título do documento, por possuir apenas um `#`.
- Logo em seguida, será incluído um título de segunda hierarquia (subtítulo) com texto “Markdown” (já que o mesmo é precedido por dois `#`).
- A sequência segue até satisfazer o nível hierarquico de títulos do seu documento;

Uma coisa bem legal desse sistema de formatação de texto é que, **por ser declarativo por texto (ou comando em sintaxe *Markdown*)**, fica fácil colaborar com outras pessoas sem perder as configurações, além de deixar claro a estrutura do texto. Isso desde que sempre se edite o arquivo *Rmd*. Caso contrário isso dependerá dos conhecimentos dos usuários de softwares de edição de texto, como o *MS Word*.

Outros elementos importantes para formatação de texto são:

- **Negrito:** Para colocar uma palavra ou frase em negrito, basta usar dois asterístico (`**`) para iniciar o trecho e dois asterístico para fechar o texto

a ser enfatizado (ex.: **\*\*trecho em negrito\*\***).

- *Itálico*: Similar ao **negrito**, mas usa-se apenas um asterístico no início e outro no fim (ex.: *\*trecho em italico\**).
- ~~riscado~~: Basta inserir dois tís (~~) ao início e outros dois ao fim do trecho a ser riscado (ex.: ~~~~trecho riscado~~~~);

#### Pontos importantes:

Os elementos de formatação de texto, como **negrito**, *italico*, ~~riscado~~, dentre outros, deverão ser seguidos por texto, sem espaço entre o elemento de formatação e o texto a ser formatado. Contudo, já com relação aos títulos, é necessário dar espaço entre o # e o texto a receber a formatação de título.

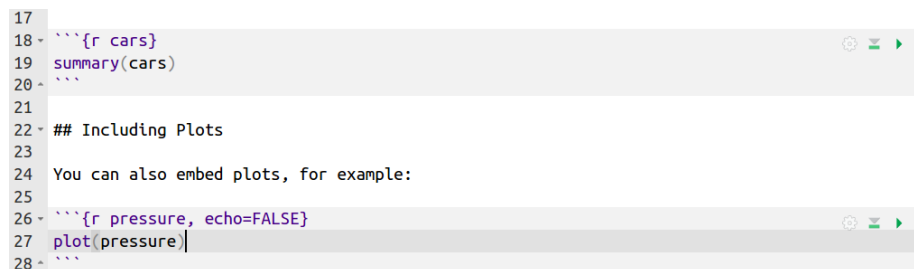
#### 4.0.1 Exercício II (Sintaxe markdown)

1. Colocar todo o primeiro parágrafo como riscado;
2. Colocar algumas paravras em **negrito**;
3. Colocar outras palavras em *intálico*;
4. Alterar a hierarquia dos títulos;
5. Cria um novo título de primeira hierarquia;
6. Renderize em formato *PDF*. Analise o indice do documento. Obedeceu a hierarquia definida?
7. Renderize em formato *.doc*. Analise o indice do documento. Obedeceu a hierarquia definida?

## Chapter 5

# Trabalhando com códigos R

Já vimos como identificar e inserir código *R* em nosso documento de texto. Vimos que existe um padrão para isso. Mas vamos a alguns detalhes ( Figure 5.1 ):



```
17
18 - ```{r cars}
19   summary(cars)
20 - ```
21
22 - ## Including Plots
23
24   You can also embed plots, for example:
25
26 - ```{r pressure, echo=FALSE}
27   plot(pressure)
28 - ```
```

Figure 5.1:

Após inserir o padrão que delimita um trecho de código *R* em nosso documento, vemos que dentro das chaves temos vários elementos... Vamos explorá-los!

### 5.1 *labels*

Na Figure 5.1 vemos, logo após a linguagem que estamos trabalhando (*r*, neste caso), um texto: **cars**, no primeiro trecho de código; e **pressure**, no segundo trecho de código. Esse texto é uma *label* que **podemos** criar (**opcional**) para cada trecho de código. Ele serve para facilitar na **compilação** do documento final. Caso algum trecho de código apresente erro, o erro e a *label* serão informados, o que facilitará nossa vida quando estivermos trabalhando com um documento com vários trechos de código diferentes.

Alguns detalhes com relação a essa *label*:

1. Não podem conter espaço;
2. Quando não informada, será criada uma label padrão *unnamed-chunk-1*. O número final é sequencial em referencia à sequencias aos quais os trechos de código aparecem sem *label*.



## Chapter 6

# *configurações do code chunk*

Ainda tendo como referência a Figure 5.1, vemos no segundo trecho de código a opção *echo=FALSE* logo após o texto da *label* e separado por vírgula. Se trata de uma das **várias** opções de configuração desse trecho de código.

Como são muitas opções, vamos trabalhar com apenas quatro delas:

1. *echo*: Dever conter valor *TRUE/FALSE* e define se o texto do código inserido deverá constar no texto do documento. Por padrão está definida como *TRUE*, já que se usa muito o *RMarkdown* para fins didáticos, onde se tem interesse de mostrar o código usado e o resultado do código.
2. *eval*: Também definido como *TRUE/FALSE*, define se o trecho de código deverá ser executado. Como padrão está definido como *TRUE*. Pode parecer estranho a existência/necessidade dessa opção. Afinal, porque necessitaríamos um trecho de código se não for para executá-lo? Pense num exemplo: você está criando um documento mostrando como funciona o comando de instalação de bibliotecas (“install.packages(ggplot)”, por exemplo). Se você não alterar o *eval* para *FALSE*, sempre que você *renderizar* seu documento ele fará a instalação da biblioteca. É o que queremos? Imagino que não...
3. *include*: Definido por padrão como *TRUE*, define se o trecho de código, **que será executado** apresentará no texto do documento seus resultados, mensagens e avisos (*warnings*); Também parece sem sentido? Vamos a um exemplo: Queremos, em algum momento salvar um determinado arquivo em *.csv*. Mas não precisamos ter isso no exposto em nosso texto final, então podemos usar essa opção e, sem precisar mexer em várias

opções (*hecho*, *warning*, etc) já desabilitar qualquer tipo de mensagem relacionado a essa tarefa ao mesmo tempo que garantimos que a mesma seja executada.

4. *fig.width*: Vimos na Figure 5.1 que podemos inserir um gráfico direto ao nosso documento sem a necessidade de salva-lo como uma figura e depois inseri-lo ao texto. Contudo, como faríamos para ajustar o tamanho do gráfico? Para isso serve as opções *fig.width* e *fig.height*. Essas opções devem ser seguidas de um valor numérico (exemplo, *fig.width = 7*) representando o tamanho de largura/altura da figura em *inches*;

## Chapter 7

# Inserir *plot*

Perceba que o segundo trecho de código do documento padrão é um simples comando `plot(pressure)`. Isso, por si só, faz com que o *plot* realizado seja incluído no documento final ao *renderizá-lo*.

### 7.1 Inserir legenda ao *plot*

Um parâmetro de configuração, não mencionado na seção anterior ( 6 ), mas que será fundamental, caso queira inserir uma selegenda ao seu gráfico é a opção `fig.cap`. Essa opção de configuração deverá receber a string (portanto, deverá estar entre aspas) a ser exibida como legenda.

**Exemplo:**

```
\\`{r, fig.cap="Gráfico de dispersão entre variável x e y."}  
plot(c(1,2,3), c(3,2,1))  
\\`
```

**Resultado:**

```
plot(c(1,2,3), c(3,2,1))
```

Veremos mais opções de configuração dos gráficos na seção 11

#### 7.1.1 Exercício (opções de código R)

1. No *Rmd* criando anteriormente, altere os parâmetros dos trechos de código. Quais são os resultados das alterações realizadas?

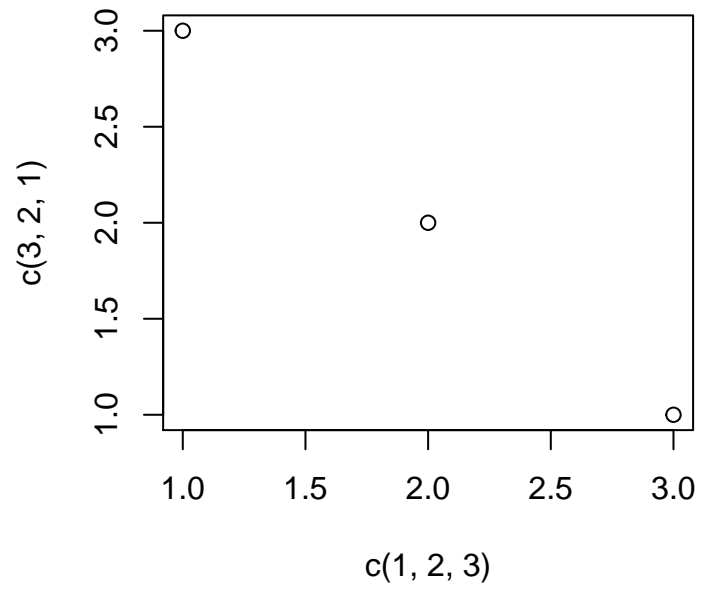


Figure 7.1: Gráfico de dispersão entre variável x e y.

2. Altere o gráfico usando *ggplot*;
  - Perceba que há que carregar a biblioteca (incluí-lo no trecho de código).



## Chapter 8

### *inline code*

Já vimos como inserir trechos de código no nosso texto. Uma coisa que não vimos é o chamado *inline code*, que é a possibilidade de escrever um código curto (de uma linha) e que terá seu resultado inserido na mesma posição e que se encontra o código.

Para usá-lo basta usar uma crase (`) seguida pela definição da linguagem de programação a ser usada (neste caso, *R*) e posteriormente, o comando a ser executado. Exemplo: `` r length( c(1, 2, 3, 4)) ``

Para que serve? Vamos imaginar que estamos escrevendo um artigo e precisamos dizer a quantidade de dados que usamos. Uma possibilidade (vou chamá-la de tradicional) seria consultar a quantidade de linhas ou o *length* de nossos dados no *R* e escrevê-lo posteriormente no nosso texto. Mas e se nossos dados mudarem? Teremos que fazer isso de novo. Aí é que entra o *inline code*.

#### **Exemplo:**

Em nossa análise foi realizada a média de `` r length( c(1, 2, 3, 4)) `` valores numérico sorteados ao azar, cujo resultado foi de `` r mean( c(1, 2, 3, 4)) ``;

#### **Resultado:**

Em nossa análise foi realizada a média de 4 valores numérico sorteados ao azar, cujo resultado foi de 2.5;





## Chapter 9

# Inserindo tabela

Outro elemento muito importante a ser inserido nos relatórios são as tabelas. Há algumas formas de adicionar os *dataframes* ou *tibles* ao nosso relatório de forma automática e configurada. Vamos usar o pacote **knitr**.

### 9.1 Knitir

Para adicionar nosso *dataframe* ao relatório, vamos usar a função **kable()**, do pacote **knitr**, da seguinte forma:

**Exemplo:**

```
\````{r InserirTabela}
knitr::kable(head(mtcars))
\````
```

**Resultado:**

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

Incentivo a vocês a darem uma olhada no *help* da função **kable**, para identificarem os parâmetros de configuração que afetarão diretamente o *layout* final da tabela.

Table 9.1: Analise dos carros.

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

- pander
- xtable
- comparativo

### 9.1.1 Adicionando titulo à tabela

**Exemplo:**

```

\```{r}
knitr::kable(head(mtcars), caption="Analise dos carros.")
\```

```

**Resultado:**

## Chapter 10

# Inserindo imagem

Já vimos como inserir um gráfico gerado direto de um código *R*. Lembra? Isso foi na seção 7. Mas sabemos bem que nem todas as imagens e figuras de nossos documentos e/ou artigos são provenientes do *R*. Então precisamos saber como inseri-las.

Para fazer isso basta incluir o seguinte código ! [] (endereco/para/imagem.png). A exclamação tem que se mantida. E dentro dos colchetes, podemos escrever a legenda da imagem, quando acharmos pertinente. **Veja a seção 15 para saber o que é preciso ser modificado na configuração do *Markdown* para que a legenda de imagens sejam apresentadas.** Não se preocupem... é uma linha que muda tudo!

Das maiores fontes de erro ao inserir imagens, antecipamos:

- Caminho para imagem errado<sup>1</sup>;
- Nome errado do arquivo;
- Falta de parágrafo antes e depois de onde deverá ir a imagem <sup>2</sup>;

### 10.1 Forçando ordem de aparição das imagens

Se você chegou a inserir várias imagens no seu relatório, já deve ter se deparado com um comportamento estranho do *Rmarkdown* em colocar as imagens em

---

<sup>1</sup>Só esse tema vale um capítulo... Mas em resumo, podemos usar um caminho relativo, que considera sempre a partir da pasta de trabalho atual (*Working Directory*); Ou caminho absoluto, que informará todas as pastas desde o diretório “raiz” (não recomendado!);

<sup>2</sup>Isso faz parte de algumas coisas inexplicáveis do *Markdown*... As vezes precisamos dar espaço entre parágrafos para que alguns recursos funcionem. As vezes não. Nisso a experiencia ajuda bastante.

posições diferentes da ordem inserida no relatório. Isso é normal e, na verdade tem a ver com LaTeX. O *LaTeX* busca otimizar espaço e por isso é comum que algumas imagens estejam onde deveriam estar. A notícia boa é que podemos resolver isso.

Basta adicionar ao cabeçalho o seguinte:

- `header-includes:` - Permite adicionar comando latex
- `\usepackage{float}` - Informa que usaremos o pacote ‘float’, necessário para o que queremos
- `\floatplacement{figure}{H}` - Informa que queremos o posicionamento das imagens no local ao qual ela é “chamada”. `h` faz referência a “here”.

```
header-includes:  
- \usepackage{float}  
- \floatplacement{figure}{H}
```

## Chapter 11

# Referência cruzada

Por referência cruzadas entendemos a possibilidade de referir-nos a imagens e seções de nosso documento no corpo de nosso texto, sem termos que nos preocupar com seus respectivos números, facilitando **muito** a edição e a manutenção de nossos textos. Isso pelo fato de ambos serem sempre atualizados segundo a ordem das figuras/seções a cada *renderização*.

### 11.1 De gráfico (*plot*)

Vimos em 7 como inserir um gráfico (*plot*) em nosso documento, lembra?

```
\```\r}
plot(c(1,2,3), c(3,2,1))
\```\r
```

E vimos como inserir legenda a ele:

```
\```\r, fig.cap="Gráfico de dispersão entre variável x e y.""}
plot(c(1,2,3), c(3,2,1))
\```\r
```

Mas como usá-lo em referência cruzada?

Para criar referência cruzada com os gráficos criados através do *R*, precisaremos criar uma *label*. A *label*, assim como no caso dos *trechos de código* ( 5.1 ) deverão se um identificador único e exclusivo para cada gráfico. Mas a *label* usada, não é a mesma que a do *code-chunk*. Por isso precisamos inserir dentro da *string* da configuração da legenda (*fig.cap*) o comando `\\label{LabelDoGrafico}` onde o texto dentro de chaves serão a *label* do gráfico.

Por agora só criamos a *label* do gráfico. Para fazer a referência à esse gráfico em nosso texto, usaremos o comando `\ref{LabelDoGrafico}` onde quisermos referir-nos à figura.

*Exemplo:*

```
\```\r s, fig.cap = "\\label{fig:plotTest}Gráfico de dispersão entre as variáveis x e y
plot(c(1,2,3,4), c(1,2,3,4))
\```\r
```

Aqui farei referência à `\ref{fig:plotTest}` onde vemos o gráfico de dispersão entre as variáveis x e y.

*Resultado:*

```
plot(c(1,2,3,4), c(1,2,3,4))
```

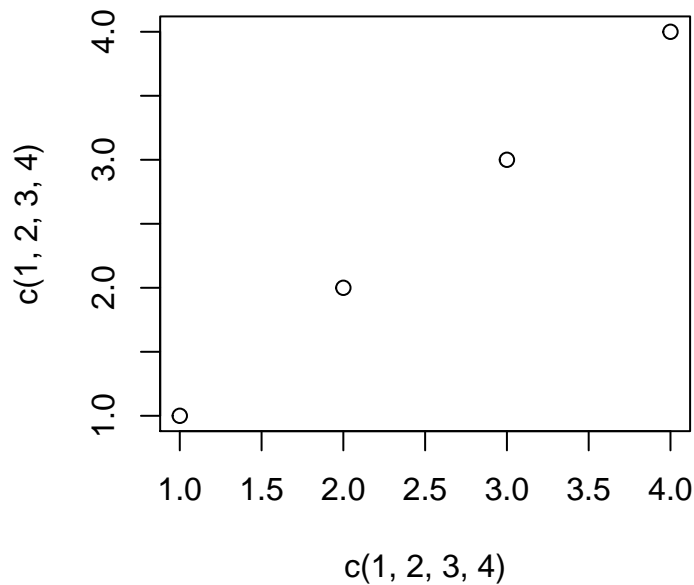


Figure 11.1: Gráfico de dispersão entre as variáveis x e y.

Aqui farei referência à figura 11.1 onde vemos o gráfico de dispersão entre as variáveis x e y.

## 11.2 De imagem

Vimos na seção 10 como inserir uma imagem. Lembra?

```

```

Vimos ainda que podemos inserir uma legenda a essa figura. Lembra?

```
![legenda da imagem vai aqui](./caminho/para/imagem.png)
```

Pois bem, se queremos fazer uma referencia a essa imagem usando a *ferramenta* de referência cruzada, basta, entre os colchetes, inserir uma *label* a essa mesma imagem: (`\label{LabelDessaImagem}`). Assim como as *labels* dos trechos de código (conforme mencionado na seção section 5.1), elas deverão ser como um ID e não poderão se repetir em diferentes imagens:

```
![legenda da imagem vai aqui \label{LabelDessaImagem}](./caminho/para/imagem.png)
```

O que fizemos até agora foi criar uma *label* para a imagem que queremos referenciar em alguma parte do texto. Agora, para fazer a referencia a essa imagem, basta usar `\autoref{LabelDessaImagem}`.

**Exemplo:**

```
![O mistério da restauração foi desvendado por esta
imagem. \label{SolucaoRestauracao}](./img/RestorationAnswers.jpg)
veja a \autoref{LabelDessaImagem} ela é muito legal a responde
todas as questões científicas sobre a restauração!
```

**Resultado:**

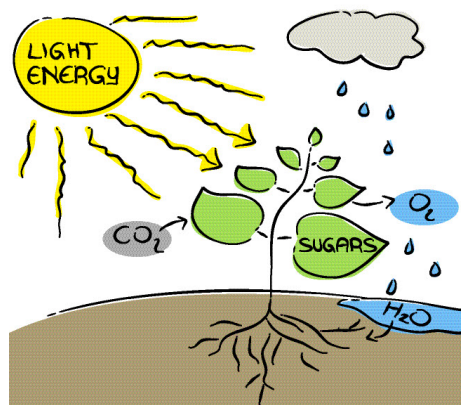


Figure 11.2: O mistério da restauração foi desvendado por esta imagem.

veja a Figure 11.2 ela é muito legal a responde todas as questões científicas sobre a restauração!

Table 11.1: Analise dos carros.

	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108

### 11.3 De seção

Quando quisermos fazer referencia a alguma seção (Título, subtítulo, ...), precisaremos criar uma *label* nessa seção, usando o *comando* `{#labelsecao}`, para depois referenciá-la no texto usando `\ref{labelsecao}`.

Repare que, para criar a seção, devemos inserir `#` antes do texto da *label*. E no comando `\ref{}` a label vai sem o `#`. Outra observação importante, não use espaço nas *labels*!

**Exemplo:**

Ver a seção `\ref{InserindoImagem}` para saber mais informações

**Resultado:**

Ver a seção 10 para saber mais informações

### 11.4 De tabela

Assim como nos casos anteriores, precisaremos adicionar uma *label* à nossa tabela. Faremos isso adicionando `\\label{labelDaTabela}` **dentro da tring do título**. Estamos usando `\` duas vezes para que a mesma não seja confundida com o título. **Fique atento pois essa é uma fonte de erro bem comum.**

**Exemplo:**

```
\\`{r}
knitr::kable(mtcars[1:3,1:3], align='c', caption="Analise dos carros. \\label{tabelamt
\\`
```

**Resultado:**

```
knitr::kable(mtcars[1:3,1:3], align='c', caption="Analise dos carros. \\label{tabelamt
```



E, como você já deve imagina, para fazer referencia à tablea, basta usar `\autoref{labelDaTable}`

**Exemplo:**

Na `\autoref{tabelamtcars}` apresentamos todos os dados usados.

**Resultado:**

Na Table 11.1 apresentamos todos os dados usados.



## Chapter 12

# Nota de roda-pé

Para inserir nota de roda-pé é muito fácil, e há duas possibilidades para você escolher.

### 12.1 Primeira possibilidade

A primeira, que acredito ser a mais simples (mas não necessariamente, mais organizada, no sentido textual) é inserindo `^[ ]` ao fim do trecho que deverá conter a nota, com o texto dentro dos colchetes. Exemplo: `^[aqui dentro vai o texto da nota]`

**Exemplo:**

Quando lhe disserem que você não consegue, lembre-se que os grandes heróis já ouviram isso e nunca desistiram `^[Frase retirada de: https://www.mundodasmensagens.com/frases-efeito/]`.

**Resultado:**

Quando lhe disserem que você não consegue, lembre-se que os grandes heróis já ouviram isso e nunca desistiram <sup>1</sup>.

Porque eu acho que talvez não seja, necessariamente, a forma mais organizada? Pois imagine que você queira colocar uma nota de roda-pé ao fim de uma frase que está no meio de um parágrafo. Você terá no seu *Rmd* a nota de roda-pé no meio do parágrafo. E isso pode confundir a algumas pessoas ou dificultar a leitura do paragrafo inteiro (quando não renderizado, claro).

Não deixe de conferir alguns pontos importantes na seção 12.3.

---

<sup>1</sup>Frase retirada de: <https://www.mundodasmensagens.com/frases-efeito/>

## 12.2 Segunda possibilidade

Outra possibilidade que nos obriga a sermos mais cuidadosos é inserindo, após o trecho que deva conter a nota: [^1]. Com isso vc está criando a “ancoragem” à nota de roda-pé. **Ou seja, está informando que ali deverá entrar um valor superescrito que estará linkado com a nota de roda-pé.** E para criar a nota, basta colocar em qualquer momento do texto [^1]: seguindo do texto de nota.

### Exemplo:

A vida não se acaba quando deixamos de viver e sim quando deixamos de buscar algo nela! [^1].

[^1]: Frase retirada de: <https://www.mundodasmensagens.com/frases-efeito/>

### Resultado:

A vida não se acaba quando deixamos de viver e sim quando deixamos de buscar algo nela! <sup>2</sup>.

## 12.3 Pontos importantes:

- Para a primeira opção ( 12.1 ) :
  - Perceba que não é necessário inserir um valor numérico antes, dentro ou depois dos colchetes, como se faz necessário com a segunda opção;
- Para a segunda opção (12.2) :
  - O valor numérico dentro do colchete não pode ser repetido (ou usado para outra nota diferente);
  - O valor numérico dentro do colchete não precisa representar a ordem de aparição das notas! Isso facilita em muito por não exigir reescrever todos os valores de nota de roda-pé subsequentes a uma nota removida;

---

<sup>2</sup>Frase retirada de: <https://www.mundodasmensagens.com/frases-efeito/>

## Chapter 13

# referências bibliográficas

Aqui começamos a fazer a transição de um simples relatório a um potencial artigo... Vamos incluir bibliografia!

Qualquer gerenciador de bibliografias (**Zotero**, **Mendeley**, etc) possibilita extrair os artigos no formato *.bib*. Se vocês selecionarem os artigos/livros que vão usar e exportá-los como *.bib* e abri-los em um editor do texto<sup>1</sup>, poderá ver algo assim:

```
@article{Akcakaya2009,
abstract = {The World Conservation Union (IUCN) defined a set of categories for conservation status},
author = {Akcakaya, H Resit and Ferson, Scott and Burgman, Mark a and Keith, David a and Georgina M. M.},
doi = {10.1046/j.1523-1739.2000.99125.x},
file = {:home/felipe/Documentos/ConsistentIUCN.pdf:pdf},
issn = {0888-8892},
journal = {Conservation biology},
keywords = {Artigo pos,Revisao{\_}Intro},
mendeley-tags = {Artigo pos,Revisao{\_}Intro},
number = {4},
pages = {1001--1013},
title = {{Making Consistent IUCN Classifications under Uncertainty}},
url = {http://onlinelibrary.wiley.com/doi/10.1046/j.1523-1739.2000.99125.x/full},
volume = {14},
year = {2009}
}
```

Uma vez que tenhamos exportado os artigos a serem usados para o formato *.bib* e com o arquivo na mesma pasta onde estamos trabalhando nosso texto <sup>2</sup>,

---

<sup>1</sup>editor de texto não é a mesma coisa que Word! Abra com um **notepad**

<sup>2</sup>Caso o arquivo esteja em uma pasta diferentes, será necessário informar o caminho à pasta

precisaremos informar ao *Markdown* que vamos usar a opção de bibliografia e qual será o arquivo *.bib* a ser usado. Para isso, adicionaremos à configuração do *RMarkdown* a seguinte linha: `bibliography: Referencias.bib`.

Repare como está a configuração deste documento que vc está lendo:

```
---
title: "Relatorios com RMarkdown"
date: 12/02/2019
output:
  pdf_document: default
  word_document: default
bibliography: Referencias.bib
---
```

Neste caso estou dizendo que o arquivo com as referências bibliográficas estão no arquivo “Referencias.bib”.

Tendo feito isso, agora já posso fazer as citações, da seguinte forma:

`[@Akcakaya2009] = (Akcakaya et al., 2009)`

ou

`@Akcakaya2009 = Akcakaya et al. (2009)`

Caso queiramos adicionar mais referencias:

`[@Akcakaya2009, @Akcakaya2009] = (Akcakaya et al., 2009, Akcakaya et al. (2009))`

ou

`@Akcakaya2009, @Akcakaya2009 = Akcakaya et al. (2009), Akcakaya et al. (2009)`

Legal. Até agora, já adicionamos nossa base de dados de referências, fizemos a citação. Mas como fica a bibliografia?

Basta adicionarmos um capítulo **## References** (com qualquer nível de hierarquia) e pronto! O *RMarkdown* já entenderá que nessa seção deverão constar todos e apenas aquelas referências que constam em nosso *.bib* que tenham sido usadas. E por mais que usemos mais de uma citação para um mesmo artigo, só aparecerá uma referência...

E aí: Vai querer continuar usando o Word?

Outra coisa, pelo fato de o *.bib* ser um doc de texto, o mesmo pode ser versionado. Outros colaboradores poderão adicionar novas referências e fazer citações e tudo será controlado por versionamento...

**A qualidade da referencia se derá conforme a qualidade dos dados disponíveis e organizados no *.bib*. Ou seja, se seu gerenciador de referências não foi capaz de identificar algumas informações do artigo, os mesmos estarão ausentes.**

## 13.1 Adicionando autor, abstract e afiliações

Para adicionar autor e abstract teremos que adicionar à área de configuração do *RMarkdown*:

```
author:  
- Felipe S. M. Barros  
abstract: |  
  Aqui vai o abstract
```

Você pode adicionar quantos autores quiser;

**FALTA VER COMO INFORMAR UNIVERSIDADE DOS AUTORES**





## Chapter 14

# Notações matemáticas

Bom, graças ao *Latex*, podemos também escrever nossas equações e expressões matemáticas. E podemos fazer isso de duas formas:

### 14.1 *Inline*

Para que a expressão seja renderizada na linha em que está sendo escrita, usamos um cifrão (\$) ao início e outro ao fim, e entre eles a expressão matemática em linguagem *Latex*. A linguagem *Latex* não é difícil e sabendo pesquisar fica fácil escrever as expressões matemáticas e equações estatísticas.

**Exemplo:**

Escrevendo uma equação inline é assim  $\sum_{n} = \pi + \sigma^2$  e o texto segue;

**Resultado:**

Escrevendo uma equação inline é assim  $\sum_n = \pi + \sigma^2$  e o texto segue;

### 14.2 *Em destaque*

Agora, quando necessitamos escrever uma equação ou expressão de forma que ela fique em destaque, ou não precisamos dela em nosso texto, podemos usar dois cifrões ao início e ao fim, e entre eles a equação/expressão:

**Exemplo:**

Escrevendo em bloco, seria assim: 
$$\sum_{n} = \pi + \sigma^2$$

**Resultado:**

Escrevendo em bloco, seria assim:

$$\sum_n = \pi + \sigma^2$$

Para mais informações sobre notações matemáticas em Latex, visite este site

## Chapter 15

# Configuração Markdown

Há vários elementos de configuração do *RMarkdown*, que poderão ser buscados pelas referências na seção 15.3.

### 15.1 Numerar as seções/capítulos

Como vocês puderam notar, as seções do meu documento estão numeradas e a de vocês (caso esteja usando o modelo padrão), não. Caso queiram que as seções sejam numeradas, basta inserir na área de configuração do markdown (como vimos na seção 2.5 ) o comando: `number_sections: yes`.

### 15.2 Table of cotent

Basta adicionar à configuração do *RMarkdown*:

```
output:
  pdf_document:
    toc: true
    toc_depth: 4
```

### 15.3 Mais infos

Markdown é um universo, juntando com as possibilidades de programação do R, fica maior ainda. Por isso cobri nesse documento apenas os elementos básicos para compreender como usá-los. Mas para extrair o máximo de todo esse

potencial, sugiro que estude por conta própria. Só o *sheat sheet*, já é um bom começo.

Se quiser algo realmente interessante, veja o livro sobre *RMarkdown*

Mas tem tbm vários artigos de blogs, dos quais, esse aqui ou esse são bem interessantes.

Para mais informações sobre o RMarkdown, clique aqui.

Ou até mesmo esse super livro.

Ou esse artigo de blog.

## Chapter 16

## References



# Bibliography

Akcakaya, H. R., Ferson, S., Burgman, M. a., Keith, D. a., Georgina, M., and Todd, C. R. (2009). Making Consistent IUCN Classifications under Uncertainty. *Conservation biology*, 14(4):1001–1013.