

Estatística e Informática

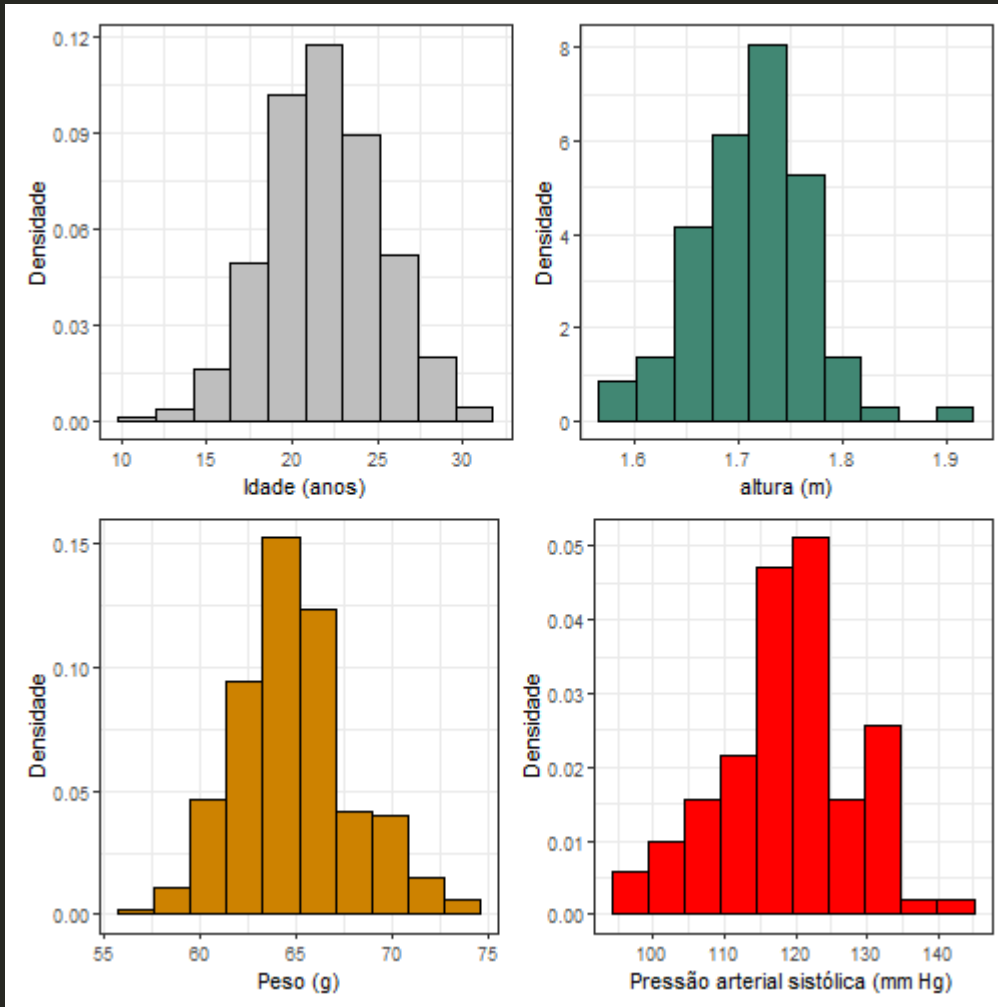
Aula 03 - Distribuição de Frequência

Alan Rodrigo Panosso alan.panosso@unesp.br

Departamento de Ciências Exatas FCAV/UNESP

(14-03-2024)

Distribuição de frequência de uma variável



Distribuição de frequência de uma variável

Quando analisamos uma variável aleatória (qualitativa ou quantitativa), deve-se conhecer a distribuição de frequência dessa variável por meio de suas possíveis realizações (observações).

Nesse sentido, o objetivo dessa aula será apresentar as principais formas e visualização gráfica de variáveis quali e quantitativas.

Exemplo: Considerando os `dados_turmas.xlsx` amostrados das turmas de Estatísticas temos:

	A	B	C	D	E	F	
1	id	sexo	cor_cabelo	cons_alcool	altura	idade_anos	
2	1	F	CC	4	1.68	19	
3	2	F	CE	1	1.59	20	
4	3	F	CC	2	1.7	49	
5	4	F	CE	3	1.5	20	
6	5	M	CE	4	1.76	23	
7	6	M	CC	2	1.6	28	
8	7	M	CC	1	1.84	19	

Sexo		Cabelos		C.A.	
M	Masculino	C	Castanhos	1	Não
F	Feminino	CC	Cast. Claros	2	Pouco
		CE	Cast. Escuros	3	Socialmente
		P	Pretos	4	Mais que social

Tamanho da População (N)

O tamanho da população N é o número total dos elementos alvos da pesquisa. Muitas vezes não conhecemos esse valor.

Em nosso exemplo, poderíamos entender como N o número de todos os alunos da Unesp que estão no segundo ano de sua graduação.

Tamanho da amostra (n)

É o número total de registros de sua base de dados, ou seja o número total de elementos amostrados da população.

O comando `glimpse` permite que veriquemos o tamanho do banco de dados em linhas (Rows - n) e colunas (Columns - variáveis).

Onde `chr` representa variáveis do tipo **strings**, ou seja textos e `dbl` representa variáveis numéricas.



```
library(tidyverse)
dados_turmas <- readxl::read_xlsx("../data/dados_turmas.xlsx")
glimpse(dados_turmas)
```

```
#> Rows: 44
#> Columns: 6
#> $ id          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
#> $ sexo        <chr> "F", "F", "F", "F", "M", "M", "M", "M", "M", "M", "M", "M",
#> $ cor_cabelo   <chr> "CC", "CE", "CC", "CE", "CE", "CC", "CC", "CE", "CC",
#> $ cons_alcool  <dbl> 4, 1, 2, 3, 4, 2, 1, 2, 2, 4, 3, 3, 3, 2, 3, 2, 3, 3,
#> $ altura       <dbl> 1.68, 1.59, 1.70, 1.50, 1.76, 1.60, 1.84, 1.88, 1.90,
#> $ idade_anos   <dbl> 19, 20, 49, 20, 23, 28, 19, 20, 20, 19, 21, 21, 21, 18
```

Exemplo da base de dados das turmas

Construir uma tabela de frequências para a variável `sexo` contendo as frequências absolutas (n_i), as frequências relativas (f_i) e a percentagem (*perc*) para as categoria existentes.

Após isso, realizar a visualização de dados com gráficos de Colunas, Barras e Setores (Pizza ou *Pie*).

Frequência Absoluta (n_i)

É definida como o número de observações no conjunto de dados pertencentes à uma categoria ou classe da variável em estudo.

Então, consideramos 1 para F e 2 para M , temos:

$$n_1 = 15$$

$$n_2 = 29$$

Assim temos a primeira regra da análise de nossa base de dados, a soma da frequência absoluta das classes (k) da variável categoria é igual a n .

$$\sum_{i=1}^k n_i = n$$

Onde k é o número de categorias da variável em questão, no caso do sexo, temos duas categorias (M e F).

$$\sum_{i=1}^k n_i = n_1 + n_2 = 15 + 29 = 44$$

Frequência Relativa (f_i)

É definida como a proporção de cada categoria em relação ao **Total de observações** (n), ou seja:

$$f_i = \frac{n_i}{n}$$

Portanto temos que,

$$f_1 = \frac{15}{44} = 0,3409$$

e

$$f_2 = \frac{29}{55} = 0,6591$$

Assim, temos que a soma das frequências relativas sempre será igual a 1:

$$\sum_{i=1}^k f_i = f_1 + f_2 = 0,3409 + 0,6591 = 1$$

onde k é o número de categorias da variável `sexo`, ou seja, 2, nesse caso.

Porcentagem de frequência (*perc* ou %)

Definida como o resultado da multiplicação da frequência relativa (proporção) por 100.

$$perc_1 = f_1 \times 100 = 0,3409 \times 100 = 34,09\%$$

$$perc_2 = f_2 \times 100 = 0,6591 \times 100 = 65,91\%$$

Para essa tarefa, vamos utilizar o R. Precisaremos, então, fazer algumas operações nos dados das turmas e vamos usar o operador PIPE (`%>%`), cujo atalho é CTRL + SHIFT + M.

Vamos utilizar a função `n()` para contar cada ocorrência das diferentes categorias da variável `sexo`.

- `group_by()` agrupa a variável `sexo` por suas diferentes categorias.
- `summarise()` cria o resumo dos dados, ou seja, contará o valor de cada categoria usando a função `n()` e salvará na coluna n_i .
- `mutate()` é a função utilizada para calcular f_i e suas respectivas porcentagens $perc_i$ a partir da coluna i .

```
dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n()) %>% # frequência Absoluta  
  mutate(fi = ni/sum(ni), # frequência Relativa  
         perc = fi*100) # porcentagem de frequência
```

```
#> # A tibble: 2 × 4  
#>   sexo      ni      fi  perc  
#>   <chr> <int> <dbl> <dbl>  
#> 1 F      15 0.341  34.1  
#> 2 M      29 0.659  65.9
```

A partir da tabela de frequência, poderemos criar representações gráficas que nos auxiliarão na apresentação e interpretação do comportamento dos dados.

Essa etapa é a denominada de **Visualização de Dados**.

VISUALIZAÇÃO DE DADOS

(Variáveis Qualitativas)

Visualização dos dados

Os tipos de gráficos podem variar de acordo com o tipo de variável, geralmente, para as variáveis qualitativas utilizamos os gráficos de Barras, Colunas ou de Setores (Pizza ou Pie).

Para isso, vamos utilizar as funções do pacote `ggplot2` que é carregado junto com o pacote `tidyverse`.

O `ggplot` funciona na forma de camadas representações gráficas e de formatações, que são adicionadas de acordo com a necessidade do usuário por meio do operador de adição `+` digitado ao final de cada linha.

Gráfico de Colunas para Sexo

Deve ser utilizado para variáveis categóricas (qualitativas ordinais ou nominais).

```
dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
          perc = fi*100) %>%  
  ggplot(aes(x=sexo, y=fi)) +  
  geom_col()
```


Gráfico de Colunas para Sexo

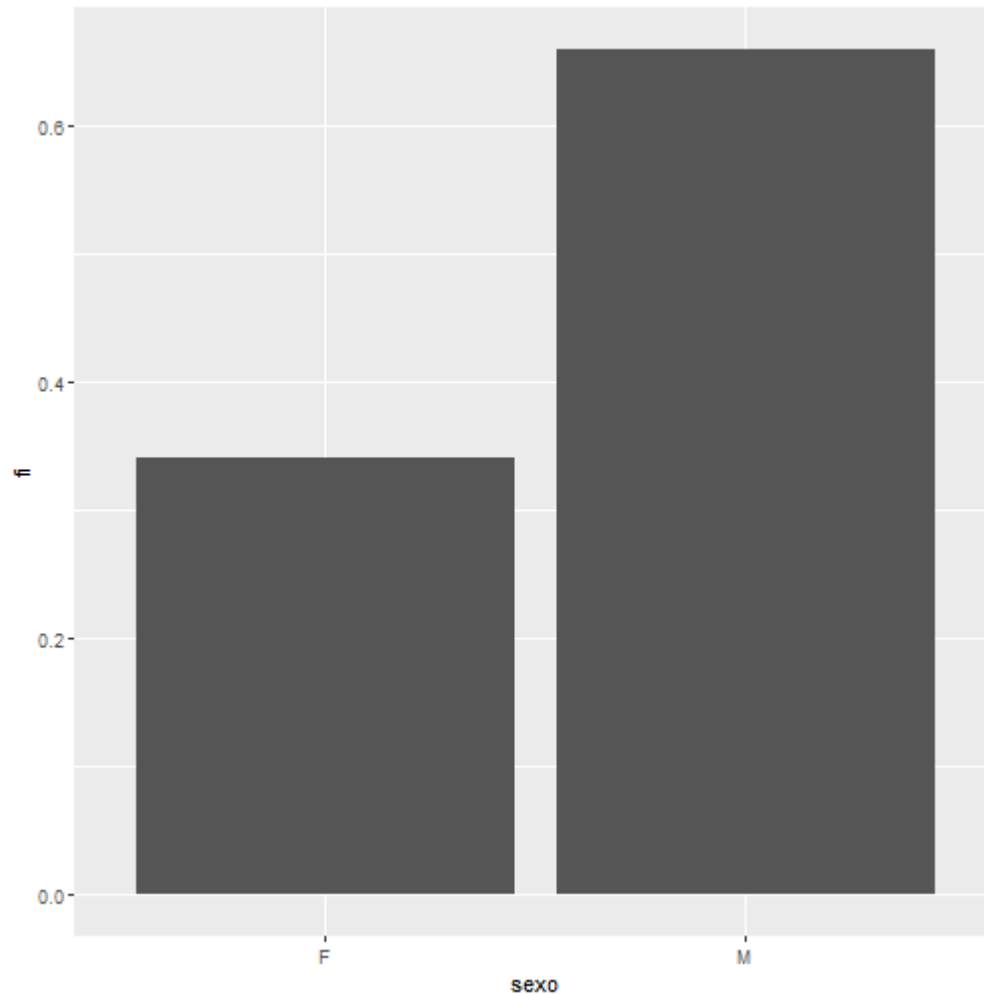
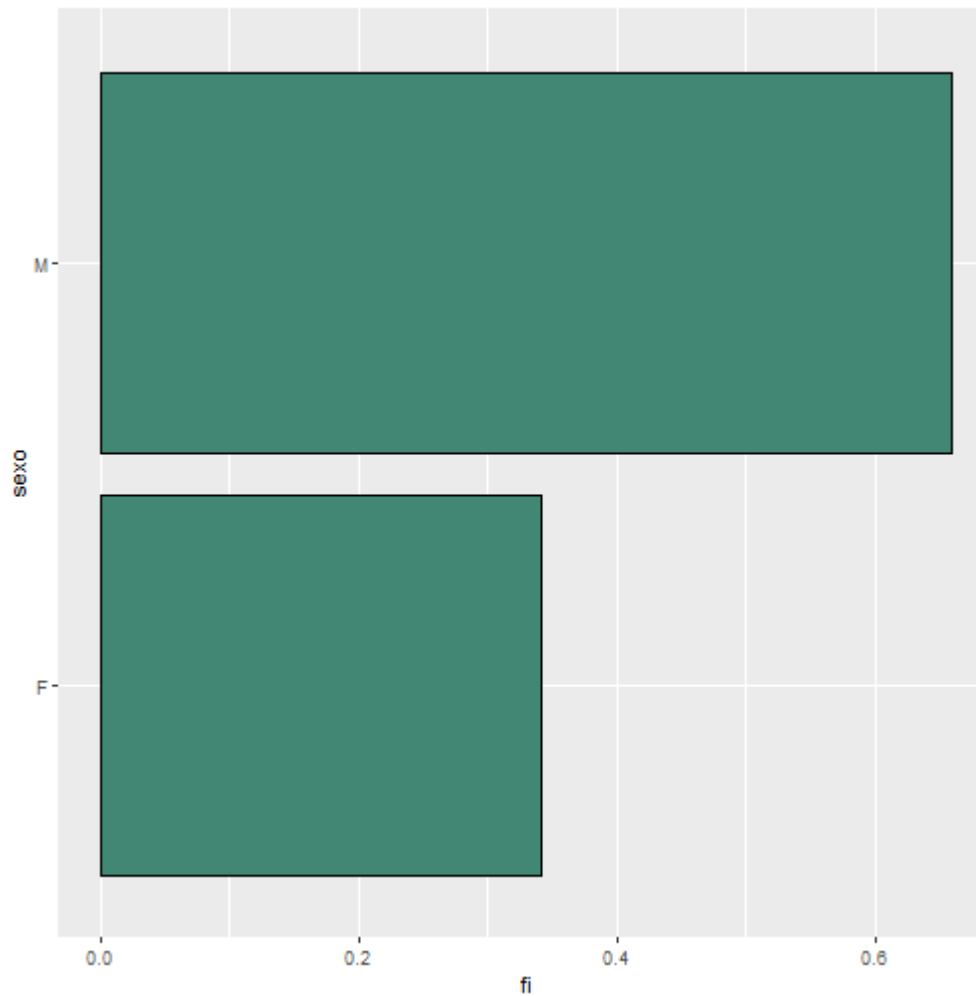


Gráfico de Barras para Sexo

Semelhante ao gráfico de colunas, contudo, com as barras na horizontal, facilita a leitura do nome das categorias.

```
dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
          perc = fi*100) %>%  
  ggplot(aes(x=fi,y=sexo)) +  
  geom_col(fill="aquamarine4",  
           color="black")
```

- o argumento `fill = "aquamarine4"` permite que possamos alterar a cor do preenchimento (*fill*) da barra e o argumento `color="black"` permite a alteração da cor o contorno das barras.



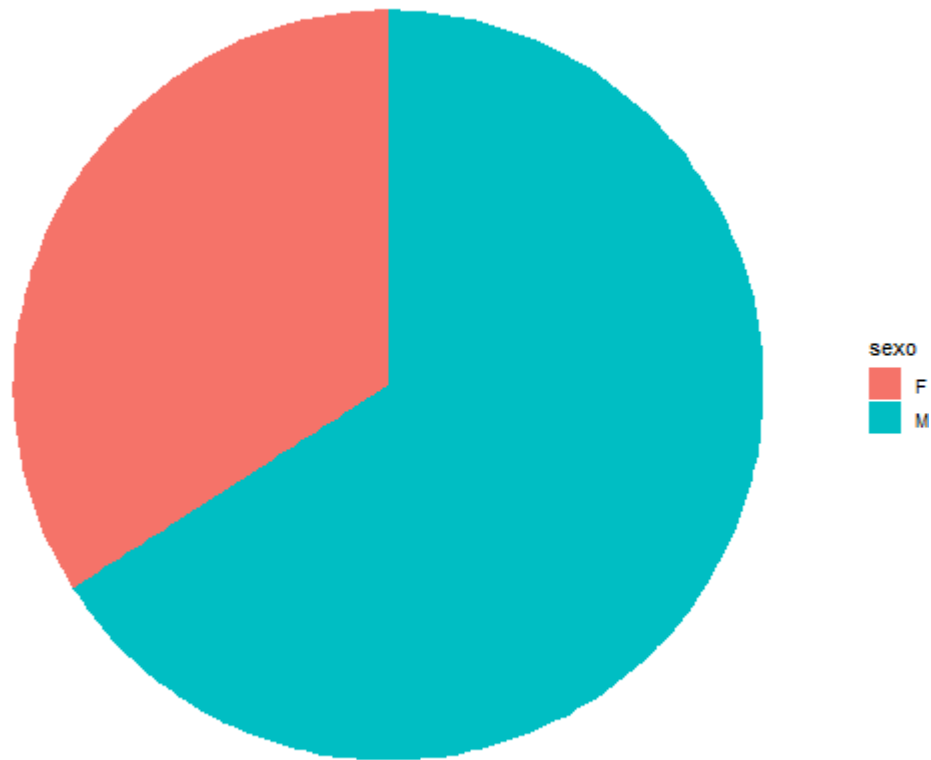
Outras cores são possíveis tente algumas das cores no R.

Gráfico de Setores para Sexo

Também conhecido como gráfico de Pizza (ou torta em inglês - *pie*), ele representa cada valor de frequência relativa das diferentes categorias da variável em uma circunferência.

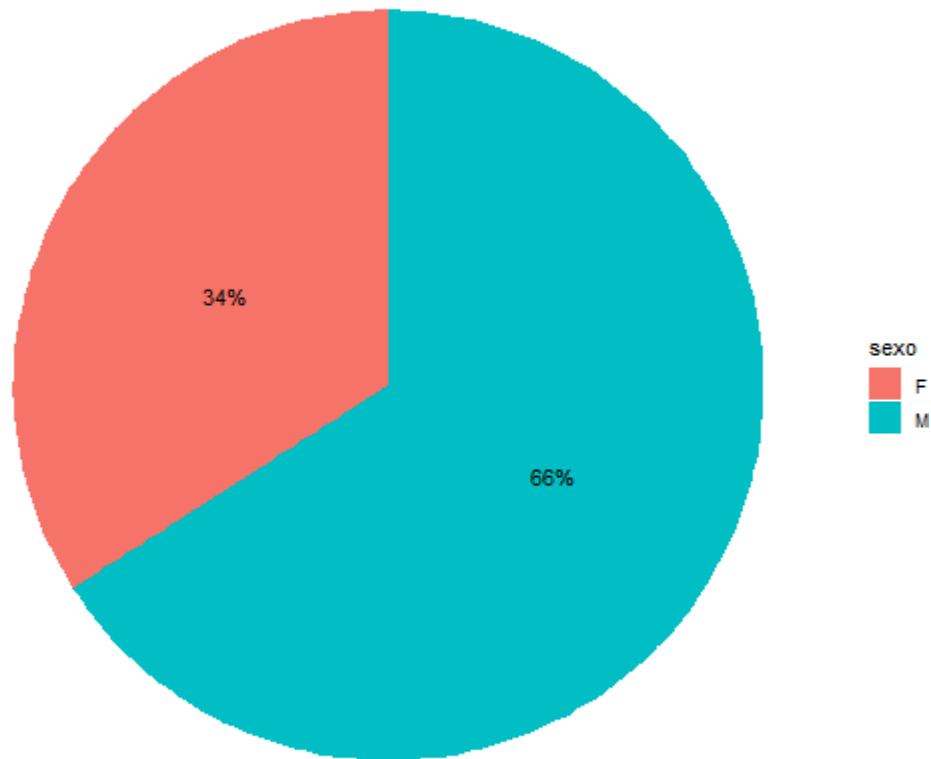
- A função `geom_bar()` associada à `coord_polar()` permite a transformação do gráfico de barras no gráfico de pizza.
- A função `theme_void()` retira elementos como linhas e nomes e números da representação gráfica.

```
dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
          perc = fi*100) %>%  
  ggplot(aes(x="",y=fi, fill=sexo)) +  
  geom_bar(stat="identity") +  
  coord_polar("y", start=0) +  
  theme_void()
```



Adicionando os valores de f_i no gráfico

```
dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
         perc = fi*100) %>%  
  ggplot(aes(x="",y=fi, fill=sexo)) +  
  geom_bar(stat="identity") +  
  coord_polar("y", start=0) +  
  theme_void() +  
  geom_text(  
    aes(  
      label=paste0(round(perc), "%"),  
      position=position_stack(vjust = 0.5))
```



Treemap

É uma técnica de visualização de dados que exhibe hierarquias utilizando retângulos aninhados.

Cada retângulo representa uma hierarquia de dados e é subdividido em retângulos menores que representam subcategorias.

Treemaps pode responder perguntas sobre os dados como: "Quais são as proporções de categorias para o total?"


```
library(treemapify)
dados_turmas %>%
  group_by(sexo) %>%
  summarise(ni=n()) %>%
  mutate(fi = ni/sum(ni),
          perc = fi*100) %>%
  ggplot(aes(area = perc, fill = sexo)) +
  geom_treemap() +
  geom_treemap_text(
    aes(label = paste(sexo,
                      paste0(round(perc, 2), "%"), sep = "\n")),
    colour = "black") +
  theme(legend.position = "none")
```

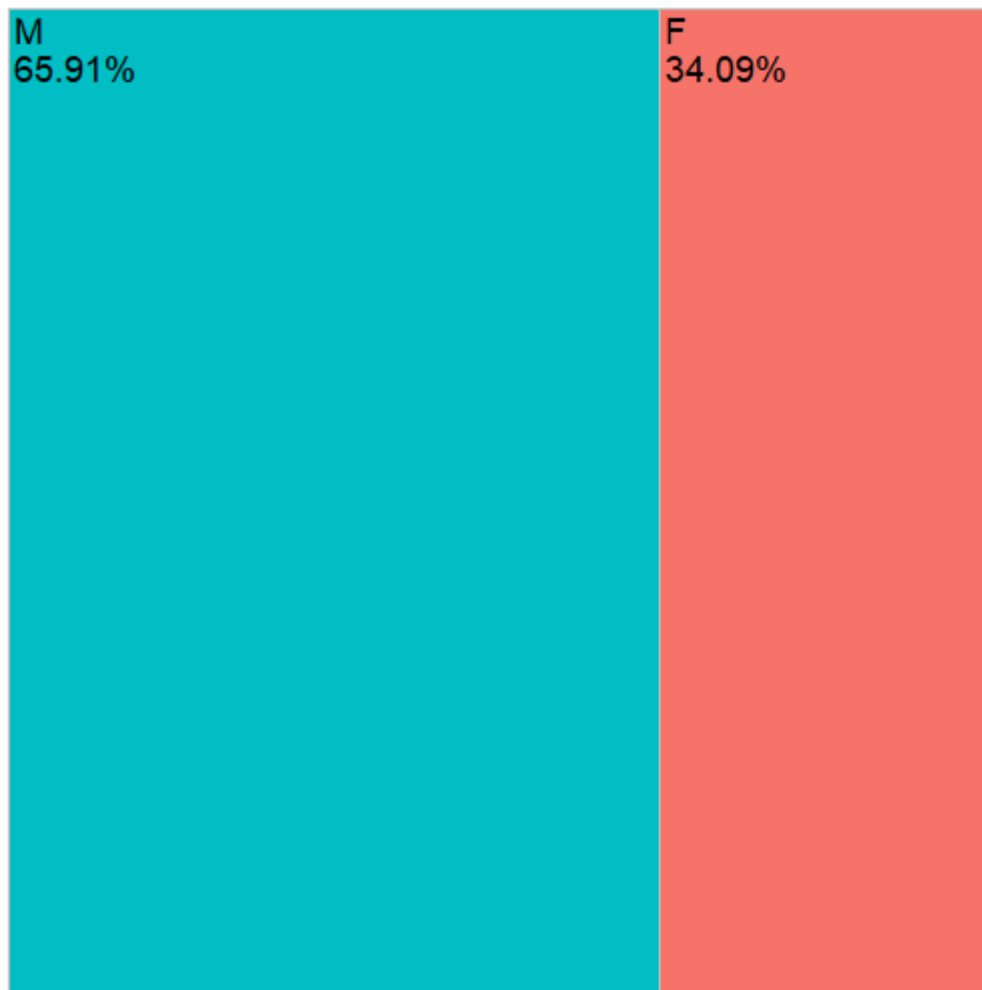


Tabela de Frequência para a variável Cor de cabelo

Vamos, mais uma vez, utilizar o R para conseguirmos as tabelas e a representação gráfica.

```
dados_turmas %>%  
  group_by(cor_cabelo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
         perc = fi*100)
```

```
#> # A tibble: 4 × 4  
#>   cor_cabelo      ni      fi  perc  
#>   <chr>      <int>  <dbl> <dbl>  
#> 1 C          6 0.136  13.6  
#> 2 CC         8 0.182  18.2  
#> 3 CE        29 0.659  65.9  
#> 4 P          1 0.0227  2.27
```

Gráfico de Colunas para Cor de cabelo

```
dados_turmas %>%  
  group_by(cor_cabelo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
          perc = fi*100) %>%  
  ggplot(aes(x=cor_cabelo,y=fi,  
            fill=cor_cabelo)) +  
  geom_col(color="black")+  
  theme_bw()
```

- ao passarmos o argumento `fill=cor_cabelo` dentro da função `aes()` estamos pedindo o mapeamento das cores de cabelo a partir de cores de preenchimento diferentes.
- `aes()` representa a estética do gráfico, ou seja, quem é x, quem é y e quem deve ser mapeado.
- a função `theme_bw()` muda o padrão de cores e de linhas do gráfico. Existem outros padrões como `theme_classic`, `theme_minimal` entre outros.

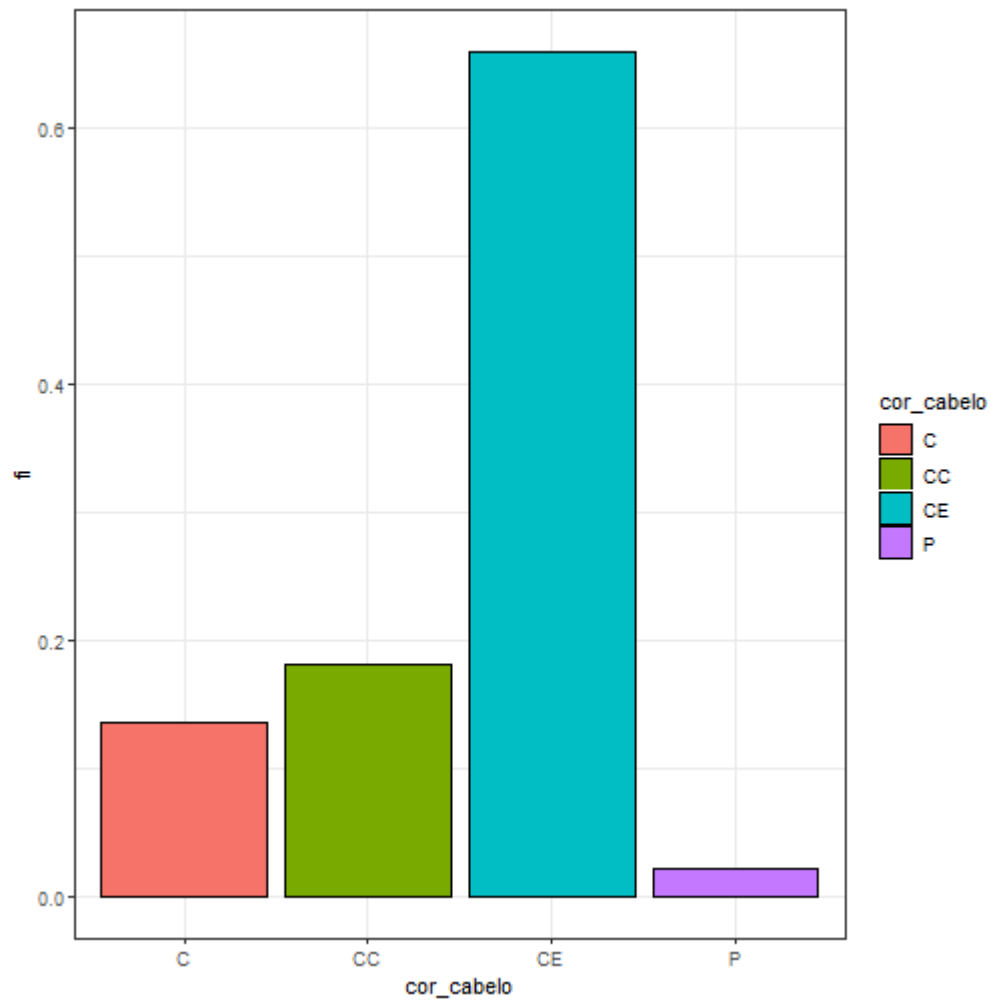


Gráfico de Barras para Cor de cabelo

```
dados_turmas %>%  
  group_by(cor_cabelo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
          perc = fi*100) %>%  
  ggplot(aes(x=fi, y=cor_cabelo,  
            fill=cor_cabelo)) +  
  geom_col(color="black")+  
  scale_fill_viridis_d() + # <<  
  theme_minimal()
```

- utilize a função `scale_fill_manual()` para alterar as cores dos preenchimentos, se necessário.

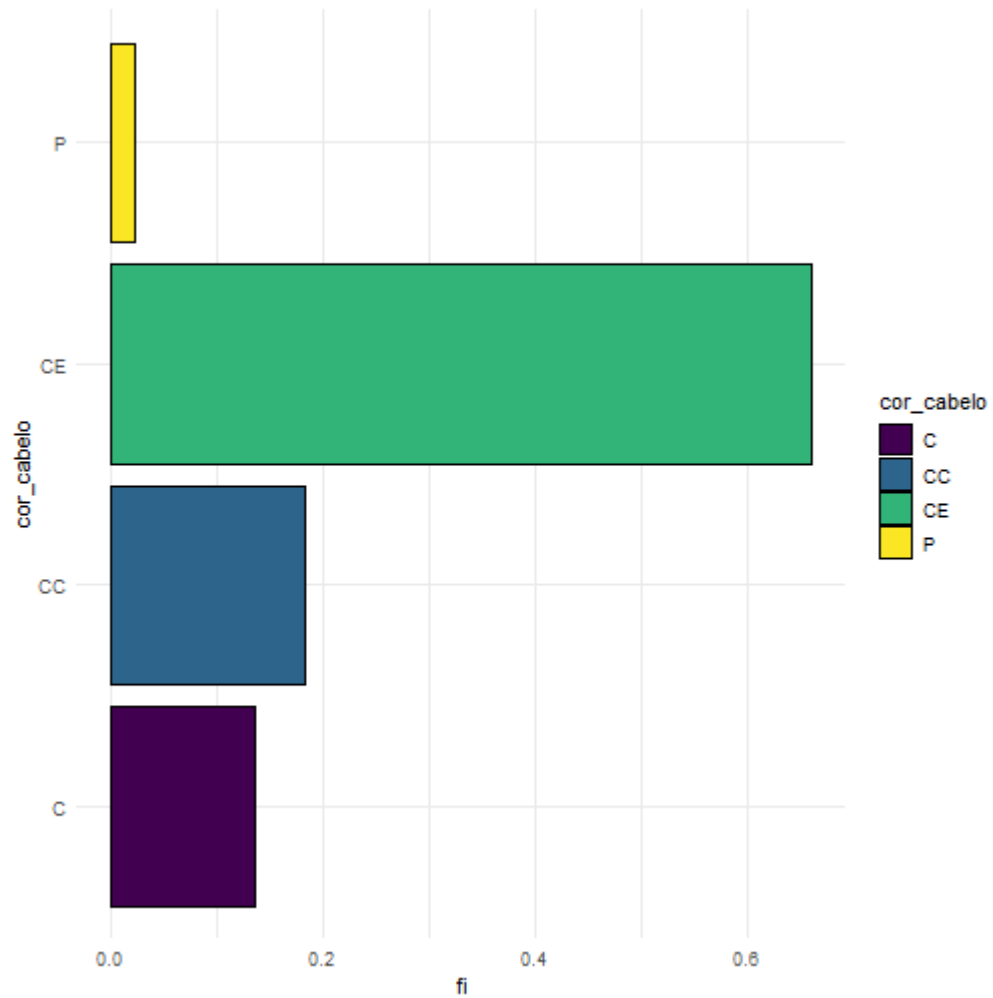
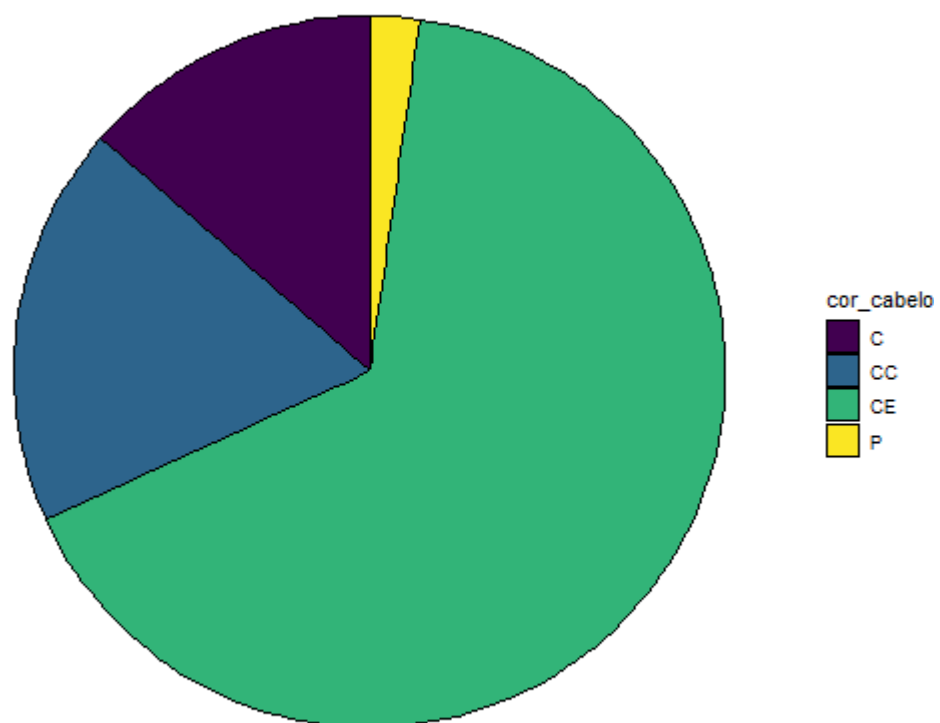


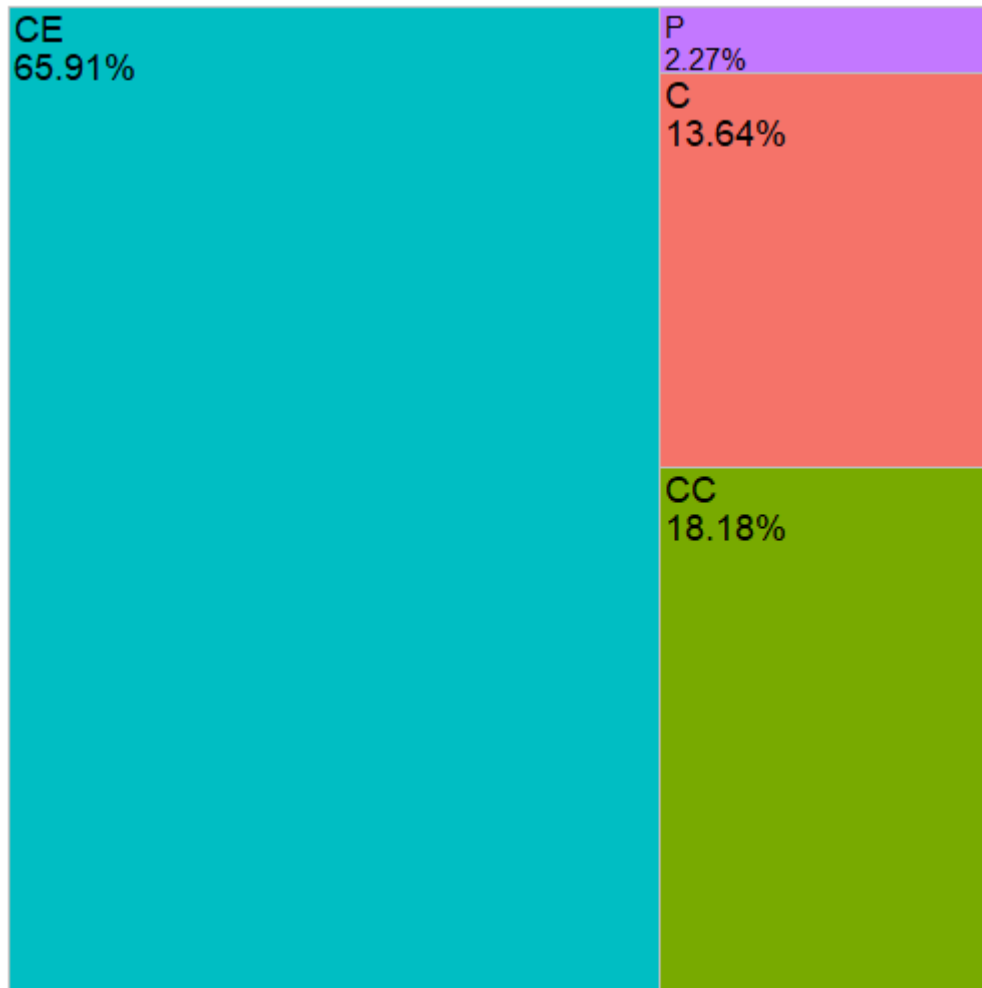
Gráfico de Setores para cor_cabelo

```
dados_turmas %>%  
  group_by(cor_cabelo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
          perc = fi*100) %>%  
  ggplot(aes(x="", y=fi,  
            fill=cor_cabelo)) +  
  geom_bar(stat="identity",color="black") +  
  coord_polar("y", start=0) +  
  scale_fill_viridis_d() + # <<  
  theme_void()
```

Treemap para cor dos cabelos

```
library(treemapify)
dados_turmas %>%
  group_by(cor_cabelo) %>%
  summarise(ni=n()) %>%
  mutate(fi = ni/sum(ni),
         perc = fi*100) %>%
  arrange(ni) %>%
  ggplot(aes(area = perc, fill = cor_cabelo)) +
  geom_treemap() +
  geom_treemap_text(
    aes(label = paste(cor_cabelo,
                      paste0(round(perc, 2), "%"), sep = "\n")),
    colour = "black") +
  theme(legend.position = "none")
```



VISUALIZAÇÃO DE DADOS

(Variáveis Quantitativas)

Tabela de frequência e visualização para Idade em anos (discreta)

Quando a variável for quantitativa discreta, os mesmos gráficos de variáveis qualitativas podem ser utilizados. Porém, também recomendamos a utilização dos gráficos boxplot, histograma e Função distribuição acumulada.

Pra a variável `idade_anos` vamos criar a tabela de frequência;

```
dados_turmas %>%  
  group_by(idade_anos) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
         perc = fi*100)
```

idade_anos	ni	fi	perc
18	6	0.1363636	13.636364
19	14	0.3181818	31.818182
20	9	0.2045455	20.454545
21	3	0.0681818	6.818182
22	3	0.0681818	6.818182
23	5	0.1136364	11.363636
24	1	0.0227273	2.272727
27	1	0.0227273	2.272727
28	1	0.0227273	2.272727
49	1	0.0227273	2.272727

Tabela de frequência para variável Altura (Quantitativa Contínua)

Quando a variável quantitativa for contínua, recomenda-se a utilização dos gráficos histograma e Função de distribuição acumulada.

Devemos, inicialmente construir a tabela de frequência da variável altura. Porém, os valores de uma variável contínua não se repetem, mesmo que isso aparentemente ocorra na base de dados.

Em teoria suas realizações podem assumir qualquer valor dentro da reta dos números reais, portanto, ao mensurar uma variável contínua, obtém-se apenas uma aproximação de seu verdadeiro valor dada pelo instrumento de medida.

Para exemplificar, vamos criar 5 classes de alturas a partir da função `cut()`.

```
dados_turmas %>%  
  mutate(  
    classes_altura = cut(altura,5)  
  ) %>%  
  group_by(classes_altura) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
         perc = fi*100)
```

classes_altura	ni	fi	perc
(1.5,1.58]	4	0.0909091	9.090909
(1.58,1.66]	7	0.1590909	15.909091
(1.66,1.75]	12	0.2727273	27.272727
(1.75,1.83]	14	0.3181818	31.818182
(1.83,1.91]	7	0.1590909	15.909091

Amplitude Total

Para melhor entendermos como a função `cut()` funciona, será necessário conhecermos mais algumas medidas para a construção do histograma. Vamos iniciar com a Amplitude total (Δ), definida como a diferença entre o valor máximo menos o valor mínimo da variável.

$$\Delta = \text{Máximo} - \text{Mínimo}$$

Para os dados de altura temos, $\Delta = 1,91\text{ m} - 1,50\text{ m} = 0,41\text{ m}$

No R podemos calcular a amplitude total com as funções do pacote base, para isso devemos, primeiramente, extrair de `dados_turmas` a variável (coluna) `altura` por meio do operador de acesso de listas `$`.

```
altura <- dados_turmas %>%  
  pull(altura)
```

Agora vamos encontrar o máximo e o mínimo e calcular a diferença.

```
D <- max(altura) - min(altura)  
D
```

```
#> [1] 0.41
```

Número de intervalos de classes (k)

Definiremos k como sendo o número de **sub-intervalos** da Amplitude Total. Uma boa representação apresenta um k **NUNCA** inferior a 5 ou superior a 15, pois com um pequeno número de classes, perde-se informação, e com um grande número de classes, o objetivo de resumir os dados fica prejudicado.

```
k <- 5
```

Amplitude de classe (Δ_i)

É o tamanho de cada um dos $k = 5$ sub-intervalos, dado pela amplitude total dividida pelo número de intervalos.

$$\Delta_i = \frac{\Delta}{k}$$

Para os dados de altura:

$$\Delta_i = \frac{\Delta}{k} = \frac{0.41 \text{ m}}{5} = 0,082 \text{ m}$$

Assim, temos

```
Di = D/k  
Di
```

```
#> [1] 0.082
```

Cada um dos 5 intervalos terá uma amplitude de 0,086 m. Ou seja, o cálculo dos limites das classes é feito a partir da adição ao valor Mínimo o valor de Δ_i k vezes:

```
limites <- min(altura) + 0:k * Di  
limites
```

```
#> [1] 1.500 1.582 1.664 1.746 1.828 1.910
```

Obervem que foi essa a metodologia utilizada pela função `cut()` para calcular os limites de classes, e essa é a metodologia clássica para lidar com dados contínuos e os agrupar em classes.

classes_altura	ni	fi	perc
(1.5,1.58]	4	0.0909091	9.090909
(1.58,1.66]	7	0.1590909	15.909091
(1.66,1.75]	12	0.2727273	27.272727
(1.75,1.83]	14	0.3181818	31.818182
(1.83,1.91]	7	0.1590909	15.909091

```
#> [1] 1.500 1.582 1.664 1.746 1.828 1.910
```

Gráfico histograma (frequências absolutas)

A partir da tabela anterior, pode-se construir o gráfico de frequência de cada classe de valor de altura, denominado **Histograma**.

```
dados_turmas %>%  
  ggplot(aes(x=altura,y=..count..))+  
  geom_histogram(breaks = limites,  
                 color="black",  
                 fill="gray")
```

O código acima gera um histograma com 5 barras onde o eixo y será a frequência absoluta, ou seja, a contagem (`..count..`) de quantos valores de altura estão dentro de uma determinada classe.

A opção `breaks = limites` deixa o histograma igual ao observado na tabela anterior.

Histograma da altura (m)

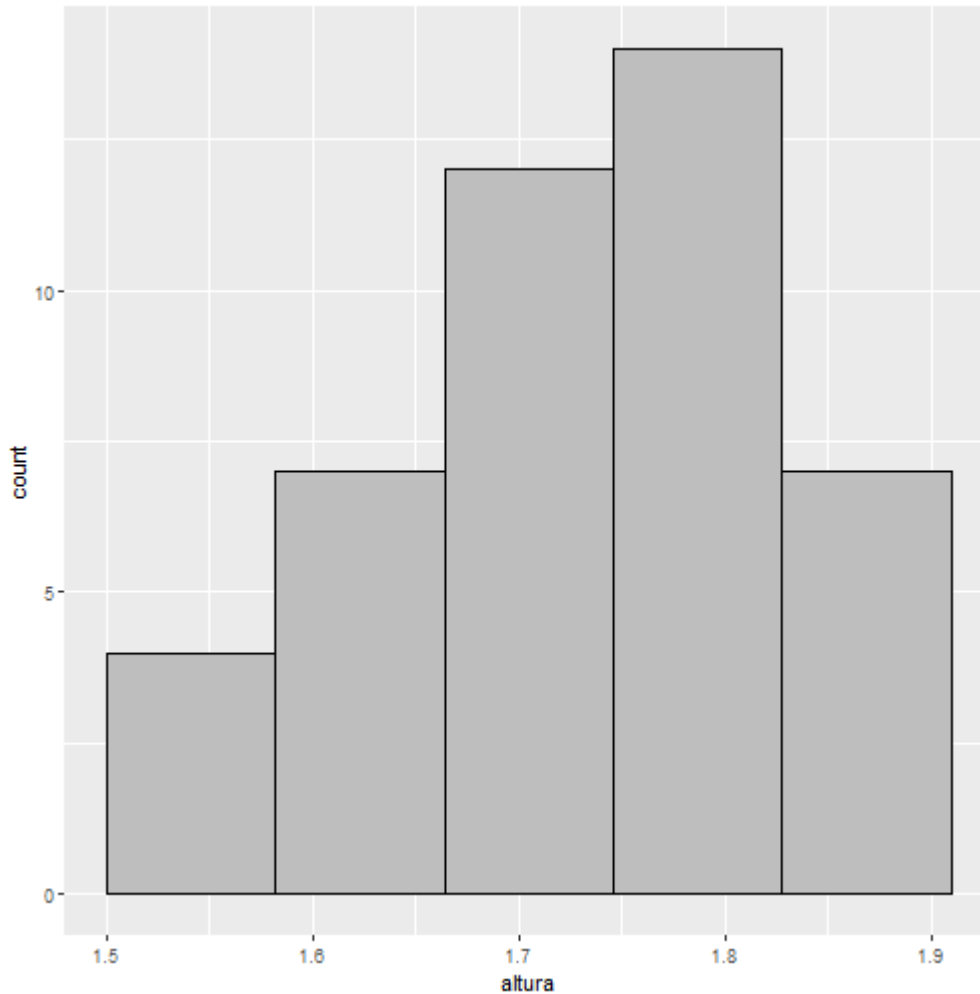


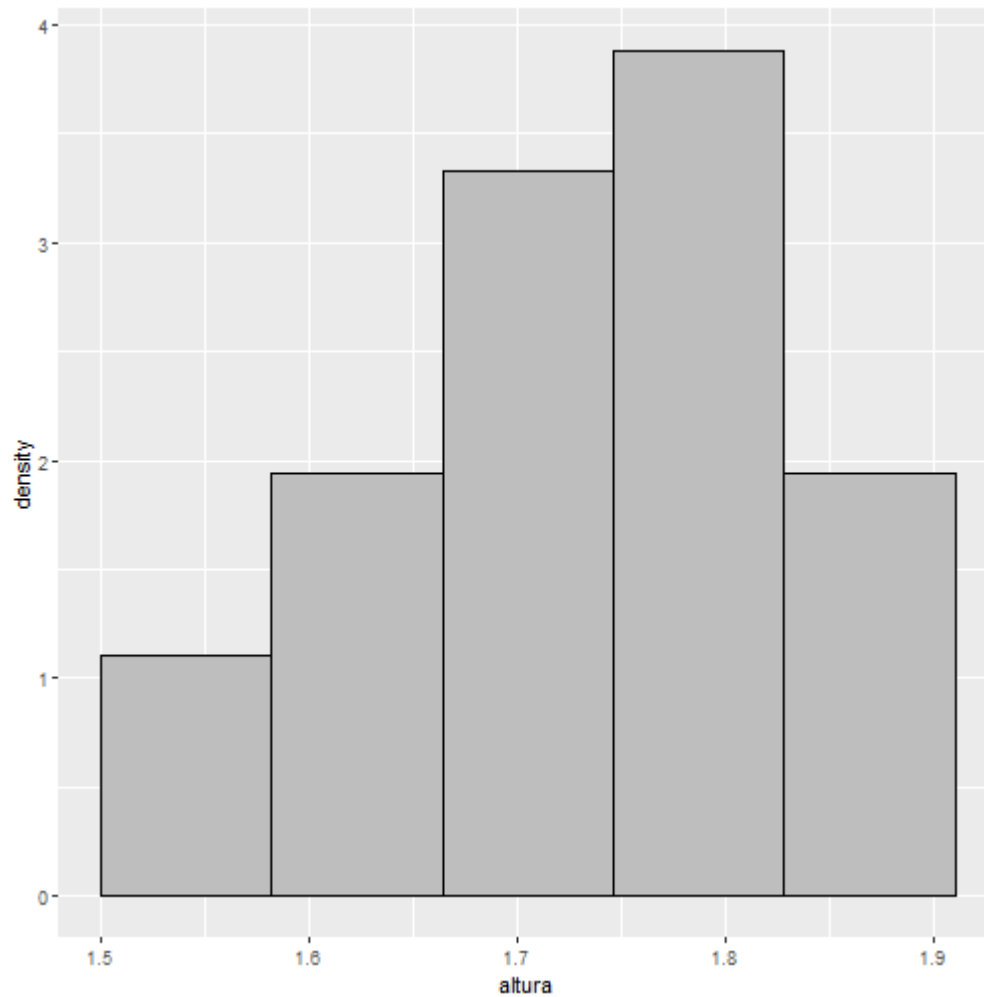
Gráfico histograma (frequências relativas)

Ao longo de nosso curso, vamos estudar que a frequência relativa f_i é uma estimativa empírica da probabilidade $P(X = x_i)$, assim é interessante que a área total da figura do histograma seja igual a 1, correspondendo à soma total das frequências relativas (f_i).

Então, para construção do histograma, sugere-se usar no eixo das ordenadas os valores de f_i/Δ_i (denominado densidade de frequência), ou seja, da medida que indica qual a concentração por unidade da variável.

```
dados_turmas %>%  
  ggplot(aes(x=altura,y=..density..))+  
  geom_histogram(breaks = limites,  
                 color="black",  
                 fill="gray")
```

Para isso utilizamos `y=..density...`



Densidade de frequência (d_i)

Agora vamos atualizar a tabela com o valor de densidade de frequência, dado por:

$$d_i = \frac{f_i}{\Delta_i}$$

```
dados_turmas %>%  
  mutate(  
    classes_altura = cut(altura,5)  
  ) %>%  
  group_by(classes_altura) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
         perc = fi*100,  
         di=fi/Di)
```

classes_altura	ni	fi	perc	di
(1.5,1.58]	4	0.0909091	9.090909	1.108648
(1.58,1.66]	7	0.1590909	15.909091	1.940133
(1.66,1.75]	12	0.2727273	27.272727	3.325942
(1.75,1.83]	14	0.3181818	31.818182	3.880266
(1.83,1.91]	7	0.1590909	15.909091	1.940133

Medidas Acumuladas

As medidas acumuladas são interessantes para compor algumas visualizações:

N_i : Frequência Absoluta Acumulada.

F_i : Frequência Relativa Acumulada.

Perc: Porcentagem de Frequência Acumulada.

```
dados_turmas %>%  
  mutate(  
    classes_altura = cut(altura,5)  
  ) %>%  
  group_by(classes_altura) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
         perc = fi*100,  
         di=fi/Di,  
         Ni = cumsum(ni),  
         Fi = cumsum(fi),  
         Perc = cumsum(perc))
```

Tabela de Frequência para Altura

classes_altura	ni	fi	perc	di	Ni	Fi	Perc
(1.5,1.58]	4	0.0909091	9.090909	1.108648	4	0.0909091	9.090909
(1.58,1.66]	7	0.1590909	15.909091	1.940133	11	0.2500000	25.000000
(1.66,1.75]	12	0.2727273	27.272727	3.325942	23	0.5227273	52.272727
(1.75,1.83]	14	0.3181818	31.818182	3.880266	37	0.8409091	84.090909
(1.83,1.91]	7	0.1590909	15.909091	1.940133	44	1.0000000	100.000000

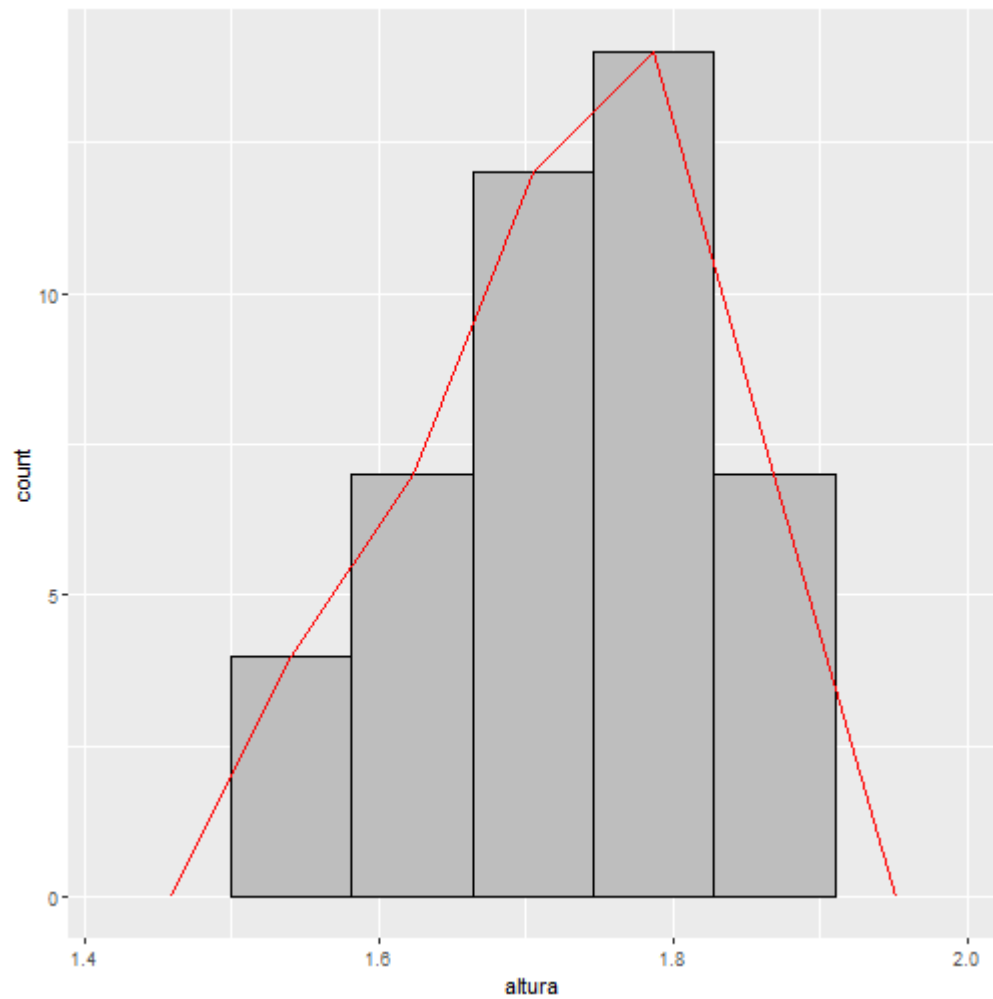
Histograma e Polígono de Frequência

Obtemos o polígono de frequências unindo por uma poligonal (segmentos de retas) os pontos correspondentes às frequências, das classes, centradas nos pontos médios de cada classe.

Para se obter as interseções do polígono com o eixo horizontal, cria-se em cada extremo do histograma uma classe com frequência nula.

No R:

```
dados_turmas %>%  
  ggplot(aes(x=altura, y=..count..))+  
  geom_histogram(breaks = limites,  
                 color="black",  
                 fill="gray") +  
  geom_freqpoly(breaks=limites,color="red")
```

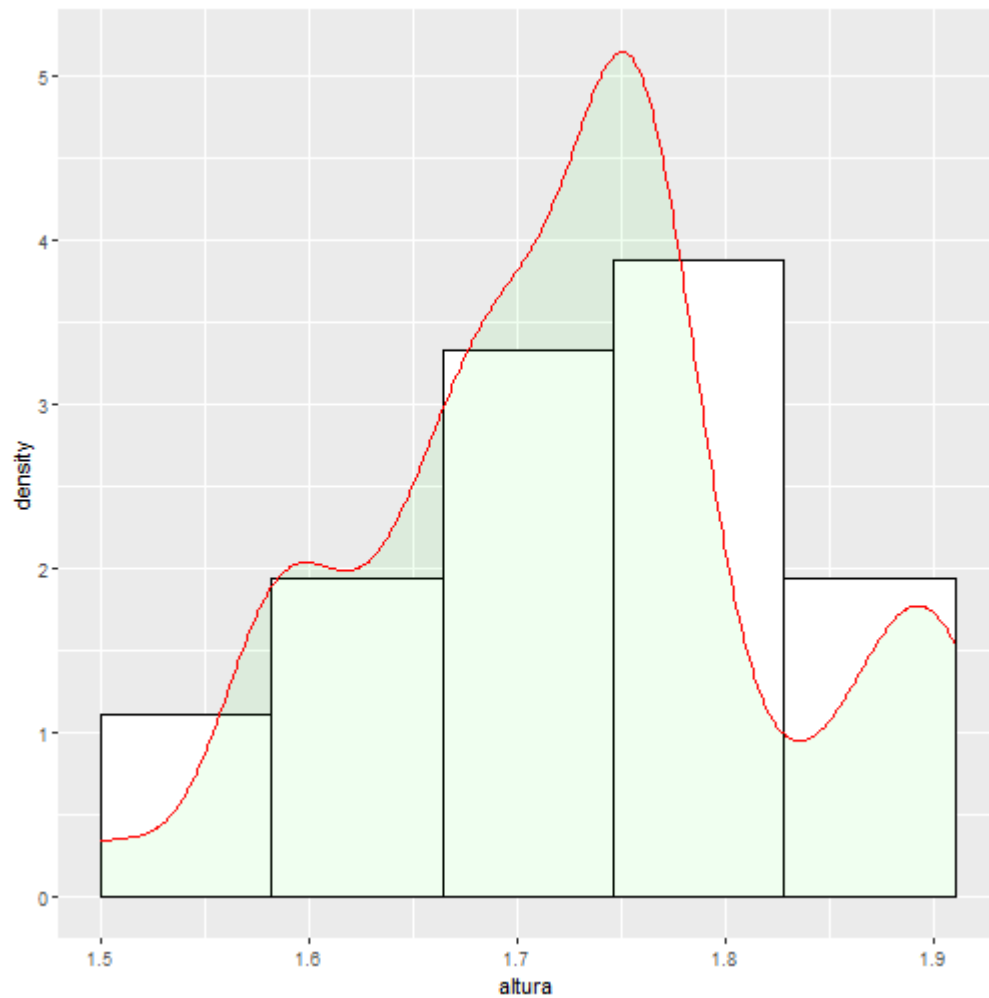


Histograma e Estimativas de densidade suavizadas

Calcula e desenha a estimativa da densidade, que é uma versão suavizada do histograma. Esta é uma alternativa útil para dados contínuos que vêm de uma distribuição suave subjacente.

```
dados_turmas %>%  
  ggplot(aes(x=altura,y=..density..))+  
  geom_histogram(breaks = limites,  
                 color="black",  
                 fill="white") +  
  geom_density(color="red",  
               fill="green",  
               alpha=0.05)
```

- Observe que o histograma foi construído com as frequências absolutas (n_i), ou seja, $y=..density..$. Utilizamos a função `geom_density()`.
- O argumento `alpha=0.05` controla a transparência do preenchimento.



Função de Distribuição Acumulada

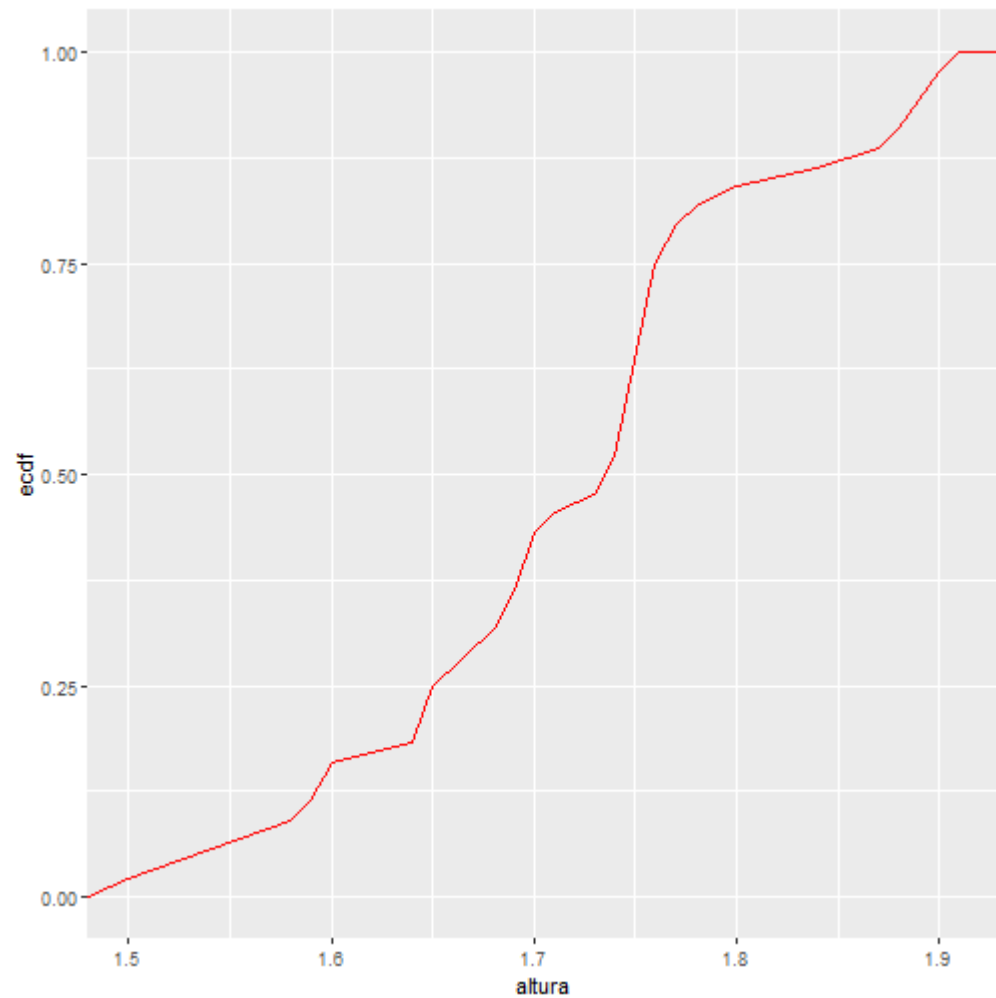
A função de distribuição acumulada descreve como probabilidades são associadas aos valores ou aos intervalos de valores de uma variável aleatória. Em outras palavras, ela representa a probabilidade de uma variável aleatória ser menor ou igual a um valor real qualquer x .

$$F(x) = P(X \leq x) \in [0, 1].$$

Para uma variável aleatória contínua (altura):

$$\int_{-\infty}^x f(x_i) dx$$

```
dados_turmas %>%  
  ggplot(aes(x=altura))+  
  stat_ecdf(geom = "line",color="red")
```

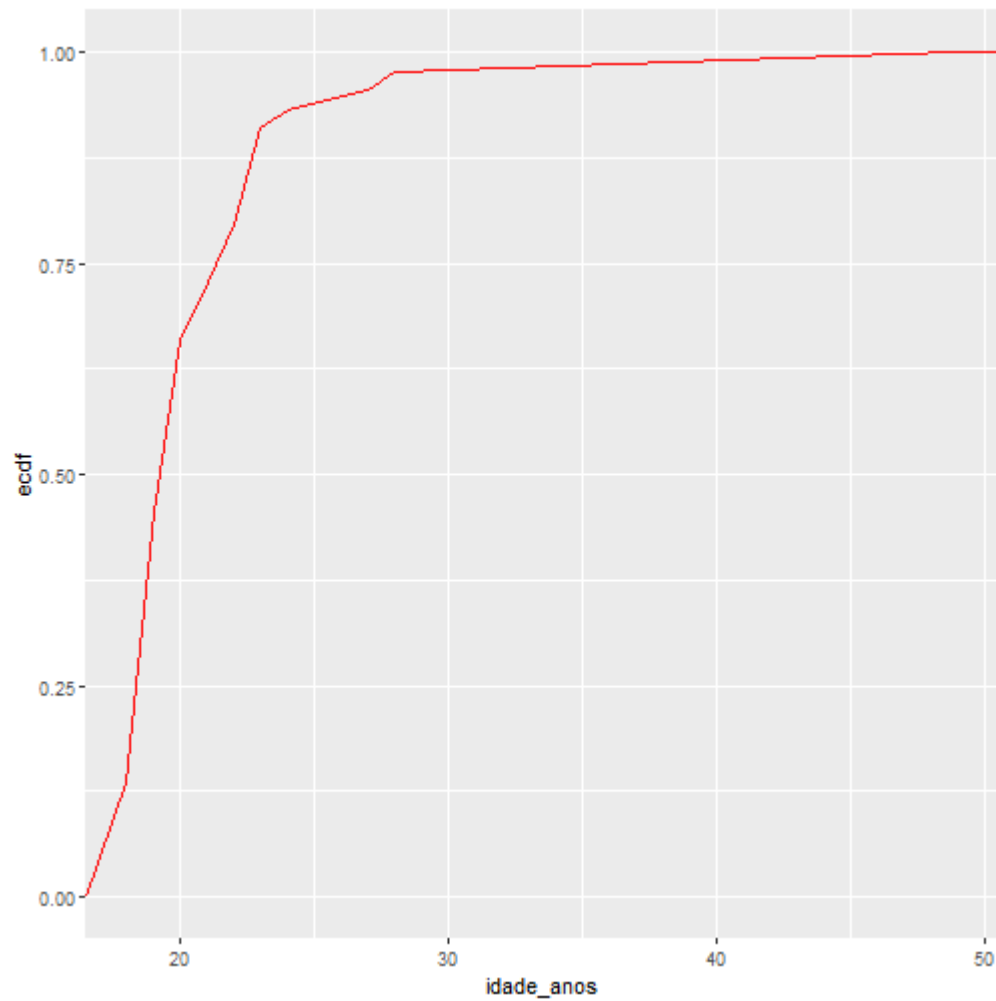


Função de Distribuição Acumulada

No caso da variável aleatória discreta (idade_anos):

$$F(x) = \sum_{x_i < x} f(x_i)$$

```
dados_turmas %>%  
  ggplot(aes(x=idade_anos))+  
  stat_ecdf(geom = "line",color="red")
```



Prática para Casa: Instalação de Pacotes no R

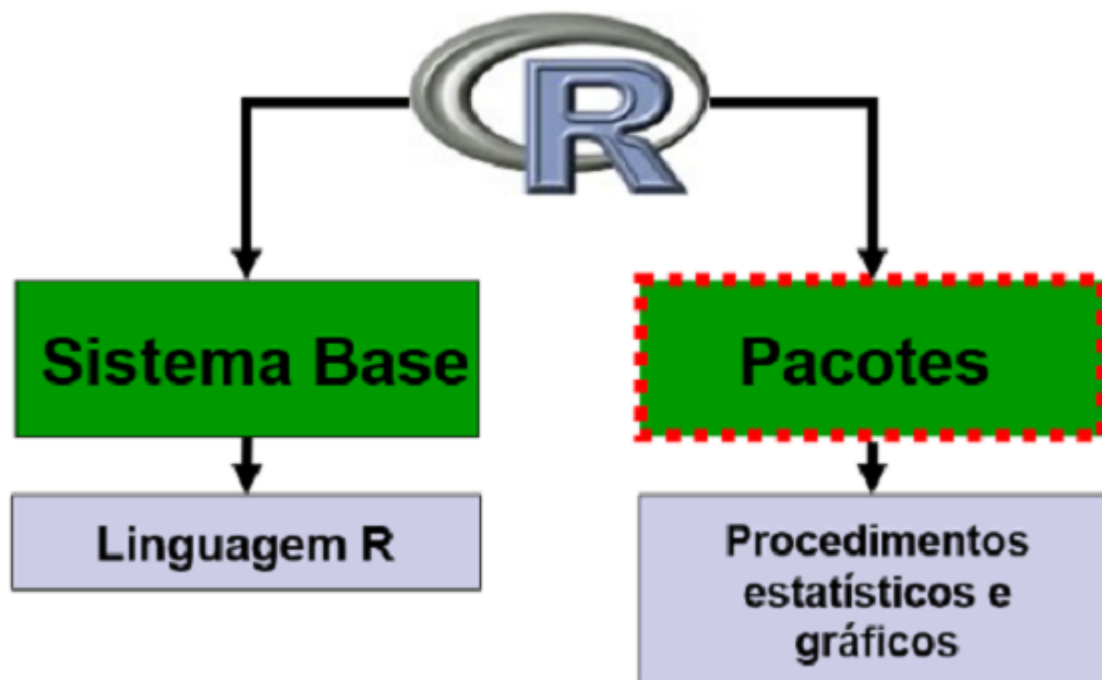
| Aula #4

Pacotes do R

[Link do Video](#)

Pacote em R

Um pacote é uma coleção de funções, exemplos e documentação. A funcionalidade de um pacote é frequentemente focada em uma metodologia estatística especial" (Everitt & Hothorn).



Pacotes no R são coleções de funções, exemplos e documentações, que devem ser previamente instalados e alocados no ambiente pela função `library`.

Pacotes básicos

Liste os pacotes carregados no ambiente com:

```
(.packages())
```

```
#> [1] "treemapify" "patchwork"  "lubridate"  "forcats"    "stringr"
#> [6] "dplyr"       "purrr"      "readr"      "tidyr"      "tibble"
#> [11] "ggplot2"    "tidyverse"  "stats"      "graphics"   "grDevices"
#> [16] "utils"      "datasets"   "methods"    "base"
```

O retorno da função é uma lista de nomes, caracteres (ou strings), na forma de um *objeto* denominado **veter**. Observe que cada pacote (elemento) é referenciado dentro do vetor por um índice, um número inteiro [*i*] apresentado entre colchetes [**i**].

Carregue um pacote chamando a função `library`.

```
library(MASS)
```

Agora, liste novamente os pacotes e observe a diferença no retorno da função.

```
(.packages())
```

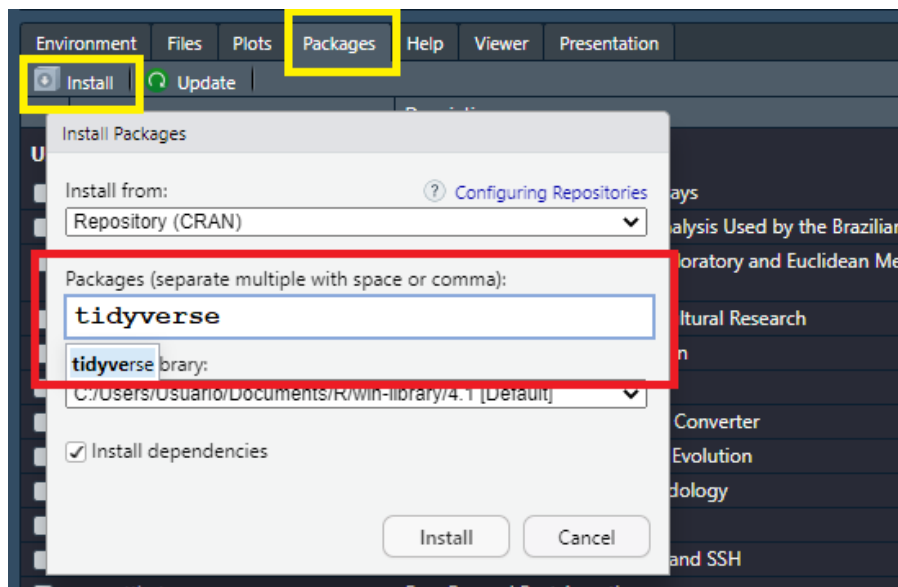
```
#> [1] "MASS"          "treemapify"    "patchwork"     "lubridate"     "forcats"
#> [6] "stringr"       "dplyr"         "purrr"         "readr"         "tidyr"
#> [11] "tibble"        "ggplot2"       "tidyverse"     "stats"         "graphics"
#> [16] "grDevices"     "utils"         "datasets"      "methods"       "base"
```

Observe que o pacote MASS agora está no ambiente de trabalho.

Instalando pacotes

Para a realização de procedimentos estatístico e manipulação de arquivos durante o curso, serão necessários vários pacotes que não fazem parte do base do R, que deverão ser instalados.

Utilizando a opção Packages\Install\nome do pacote



Instale os pacotes:

- tidyverse
- agricolae

Os pacotes também podem ser instalados a partir das linhas de comandos:

```
install.packages("tidyverse")  
install.packages("agricolae")
```

Agora podemos carregar esses pacotes em nosso ambiente de trabalho.

```
library(tidyverse)  
library(agricolae)
```

Vamos agora criar uma "Pipe Line" para visualização de um banco de dados

```
mtcars %>%  
  glimpse()
```

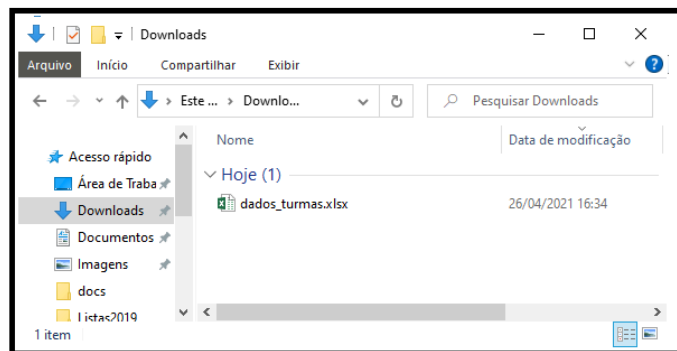
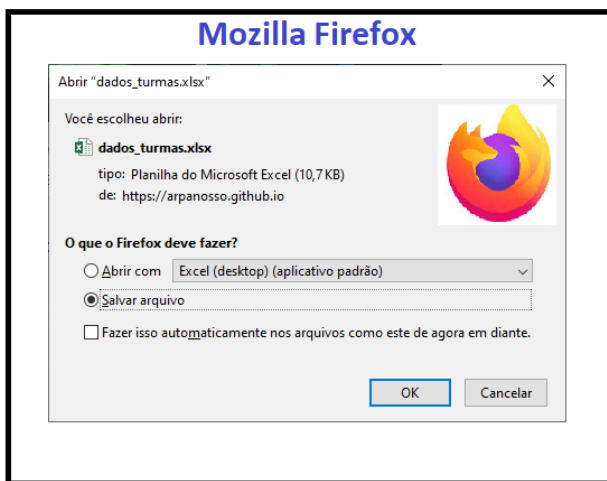
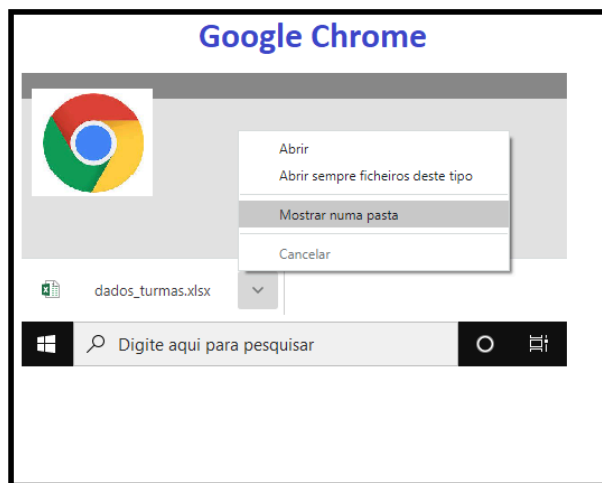
```
#> Rows: 32  
#> Columns: 11  
#> $ mpg   <dbl> 21.0, 21.0, 22.8, 21.4, 18.7, 18.1, 14.3, 24.4, 22.8, 19.2, 1  
#> $ cyl   <dbl> 6, 6, 4, 6, 8, 6, 8, 4, 4, 6, 6, 8, 8, 8, 8, 8, 8, 4, 4, 4, 4  
#> $ disp  <dbl> 160.0, 160.0, 108.0, 258.0, 360.0, 225.0, 360.0, 146.7, 140.8  
#> $ hp    <dbl> 110, 110, 93, 110, 175, 105, 245, 62, 95, 123, 123, 180, 180,  
#> $ drat  <dbl> 3.90, 3.90, 3.85, 3.08, 3.15, 2.76, 3.21, 3.69, 3.92, 3.92, 3  
#> $ wt    <dbl> 2.620, 2.875, 2.320, 3.215, 3.440, 3.460, 3.570, 3.190, 3.150  
#> $ qsec  <dbl> 16.46, 17.02, 18.61, 19.44, 17.02, 20.22, 15.84, 20.00, 22.90  
#> $ vs    <dbl> 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1  
#> $ am    <dbl> 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0
```

Carregando os dados no R

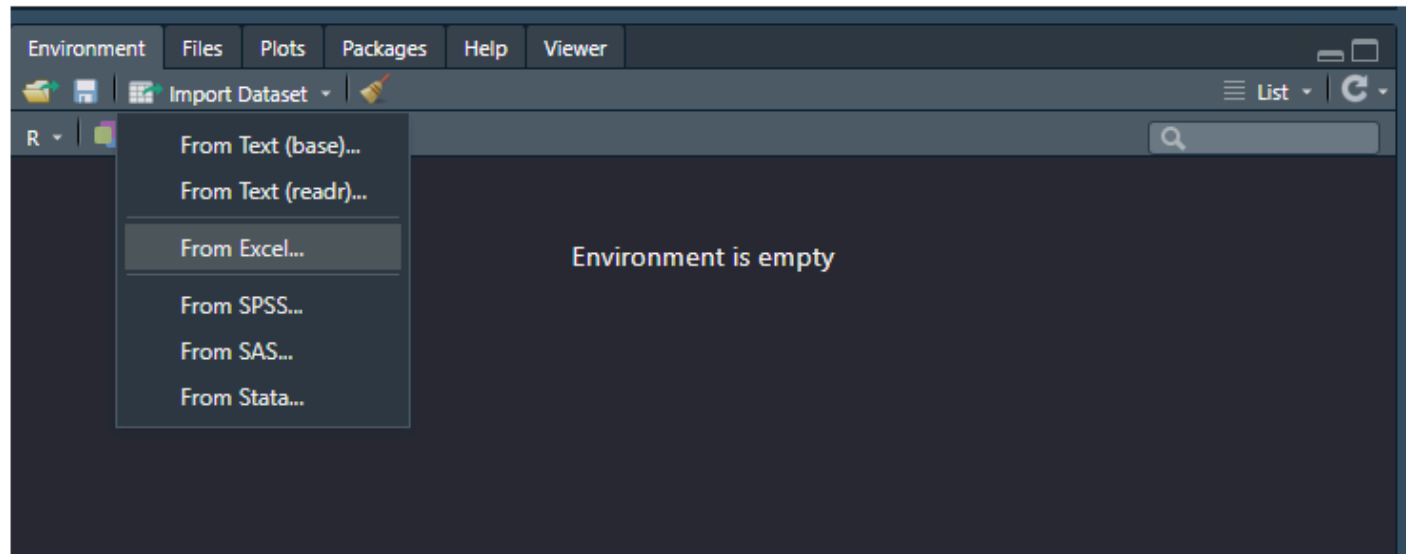
Para carregar o banco de dados da turma no R, siga os passos:

1. Faça o Download dos **dados_turmas.xlsx**.

2. Salve em uma pasta data dentro de um projeto do R previamente criado.

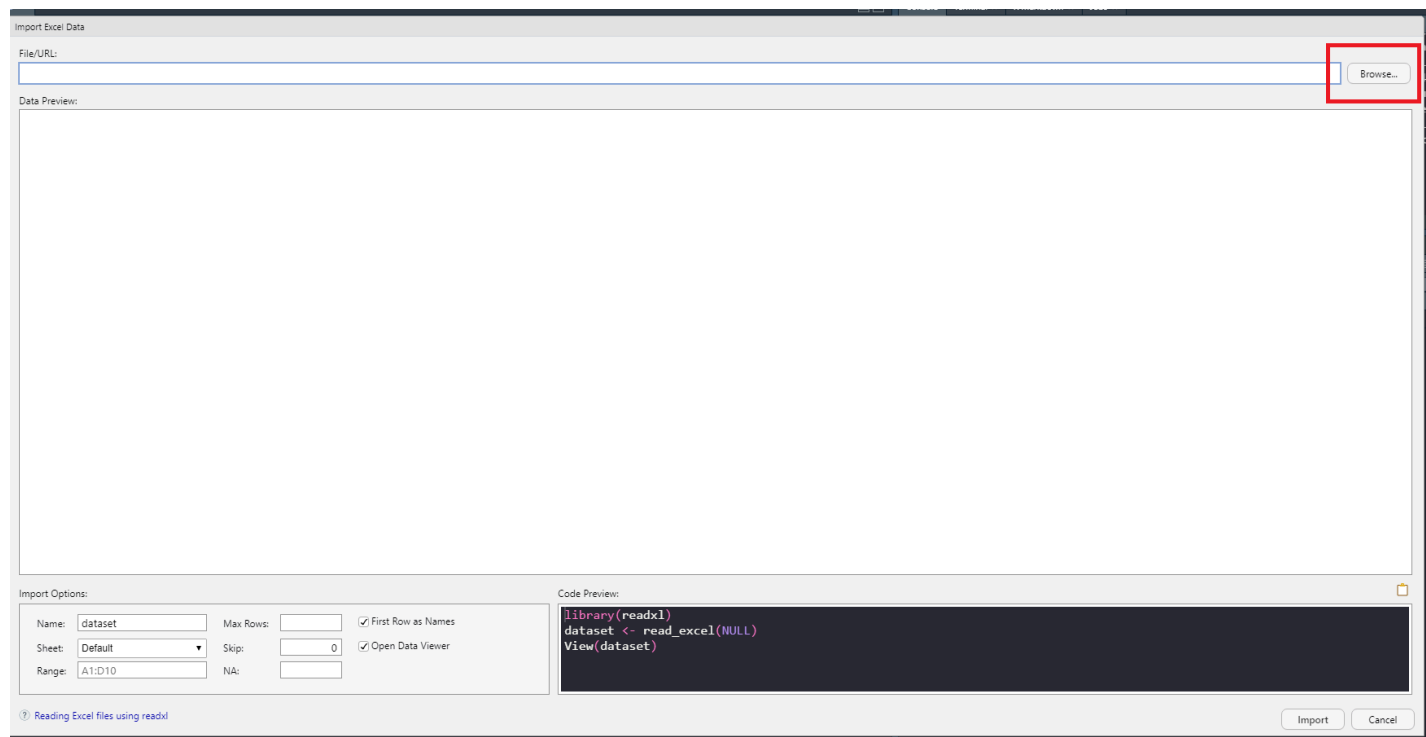


3. Na aba **Environment** do RStudio selecione **Import Dataset/From Excel...** como apresentado abaixo.

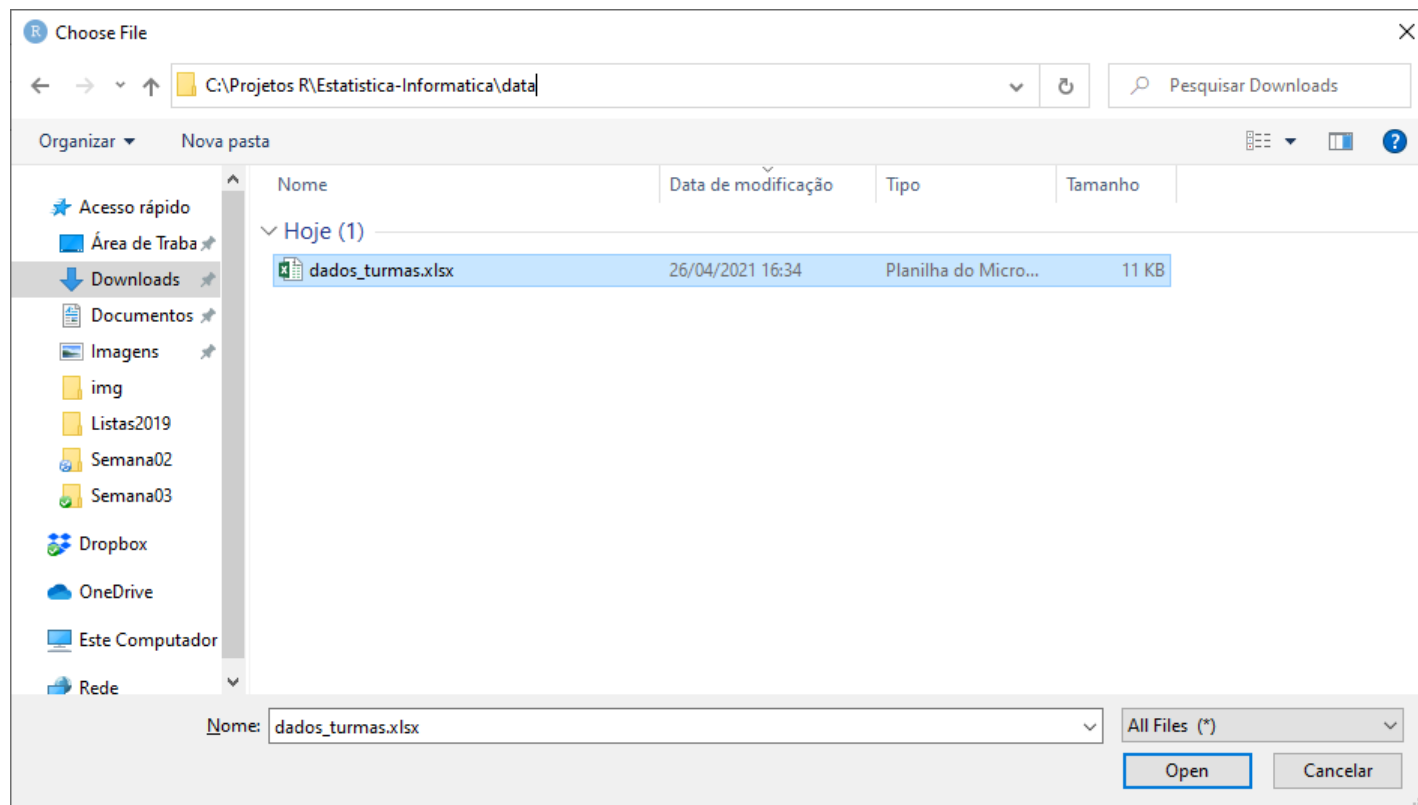


]

4. Selecione **Browse** (destacado em vermelho no canto direito superior).



5. Na próxima janela busque o arquivo da base de dados **dados_turmas.xlsx** que salvamos na pasta "*data*", selecione o arquivo e clique em **Open**.



6. Na janela serão apresentados os dados, **NÃO CLIQUE EM IMPORT**, ao invés disso, **selecione e copie o código** para a importação dos dados. Após isso **CLIQUE EM CANCEL**.

Import Excel Data

File/URL:
C:/Users/Usuario/Downloads/dados_turmas.xlsx

Data Preview:

id (double)	sexo (character)	cor_cabelo (character)	GA (character)	altura (double)	idade_anos (double)
1	F	CC	mais_social	1.68	19
2	F	CE	nao_consome	1.59	20
3	F	CC	pouco	1.70	49
4	F	CE	socialmente	1.50	20
5	M	CE	mais_social	1.76	23
6	M	CC	pouco	1.60	28
7	M	L	nao_consome	1.84	19
8	M	CE	pouco	1.88	20
9	M	CC	pouco	1.90	20
10	M	C	mais_social	1.68	19
11	M	CC	socialmente	1.76	21
12	M	CE	socialmente	1.91	21
13	M	CC	socialmente	1.68	21
14	M	CE	pouco	1.70	18
15	M	CE	socialmente	1.76	19
16	F	CE	pouco	1.65	20
17	F	CC	socialmente	1.58	19
18	M	CE	socialmente	1.87	19
19	M	CE	socialmente	1.75	18
20	F	C	mais_social	1.69	22
21	M	C	socialmente	1.74	19

Previewing first 50 entries.

Import Options:

Name: dados_turmas Max Rows: ☒ First Row as Names

Sheet: Default Skip: 0 ☒ Open Data Viewer

Range: A1:D10 NA:

Code Preview:

```
library(readxl)
dados_turmas <- read_excel("C:/Users/Usuario/Downloads/dados_turmas.xlsx")
View(dados_turmas)
```

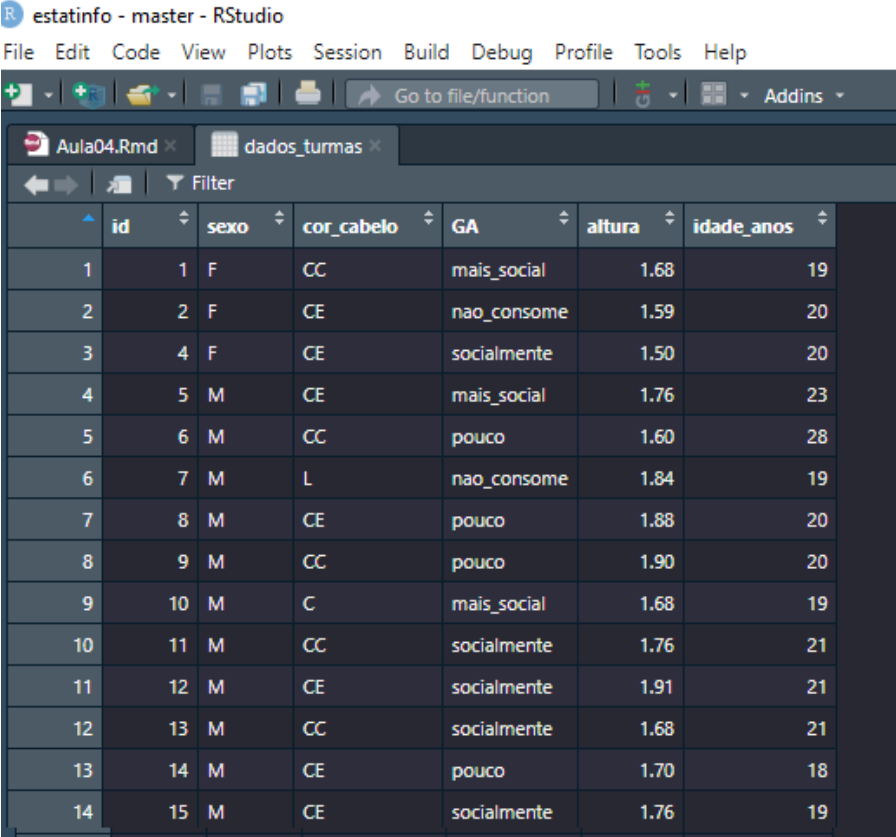
Reading Excel files using readxl

Import Cancel

Copie essas linhas de código, cole no seu script e o execute

7. Cole o código no seu script do R e o execute. Os dados serão salvos no objeto `dados_turmas`. Se necessário, instale o pacote `readxl` com as opções da aba **Packages/Install** ou com o comando `install.packages("readxl")`.

```
library(readxl)
dados_turmas <- read_excel("data/dados_turmas.xlsx")
View(dados_turmas)
```



estatinfo - master - RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function

Aula04.Rmd dados_turmas

Filter

	id	sexo	cor_cabelo	GA	altura	idade_anos
1	1	F	CC	mais_social	1.68	19
2	2	F	CE	nao_consome	1.59	20
3	4	F	CE	socialmente	1.50	20
4	5	M	CE	mais_social	1.76	23
5	6	M	CC	pouco	1.60	28
6	7	M	L	nao_consome	1.84	19
7	8	M	CE	pouco	1.88	20
8	9	M	CC	pouco	1.90	20
9	10	M	C	mais_social	1.68	19
10	11	M	CC	socialmente	1.76	21
11	12	M	CE	socialmente	1.91	21
12	13	M	CC	socialmente	1.68	21
13	14	M	CE	pouco	1.70	18
14	15	M	CE	socialmente	1.76	19