

# Estatística e Informática

## Aula 04 - Distribuição de Frequência

Alan Rodrigo Panosso [alan.panosso@unesp.br](mailto:alan.panosso@unesp.br)

Departamento de Engenharia e Ciências Exatas FCAV/UNESP

(06-05-2021)

# Distribuição de Frequência e Visualização de Dados

# 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:

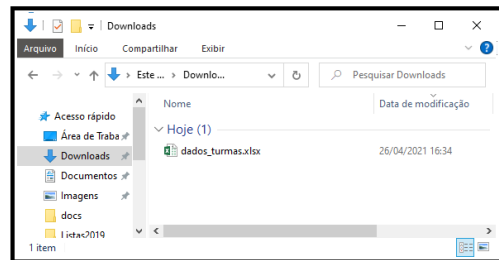
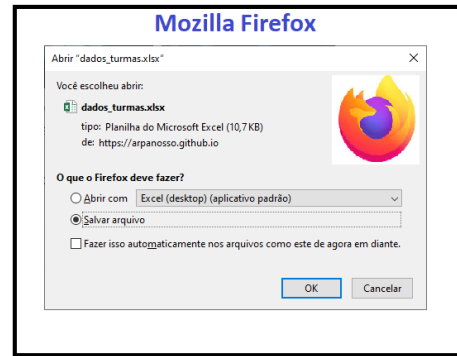
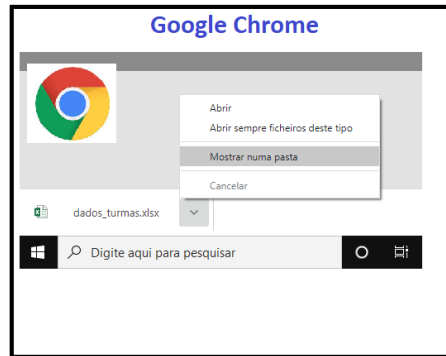
	A	B	C	D	E	F
1	id	sexo	cor_cabelo	GA	altura	idade
2	1	F	CC	mais_social	1.68	19
3	2	F	CE	nao_consomme	1.59	20
4	3	F	CC	pouco	1.7	49
5	4	F	CE	socialmente	1.5	20
6	5	M	CE	mais_social	1.76	23
7	6	M	CC	pouco	1.6	28
8	7	M	L	nao_consomme	1.84	19

# Carregando os dados no R

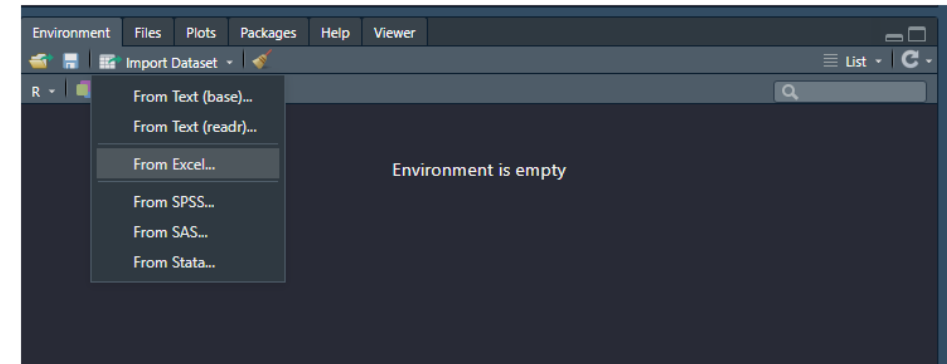
Para carregar o banco de dados da turma no R, siga os passos:

1. Faça o Download dos **dados\_turmas.xlsx**.

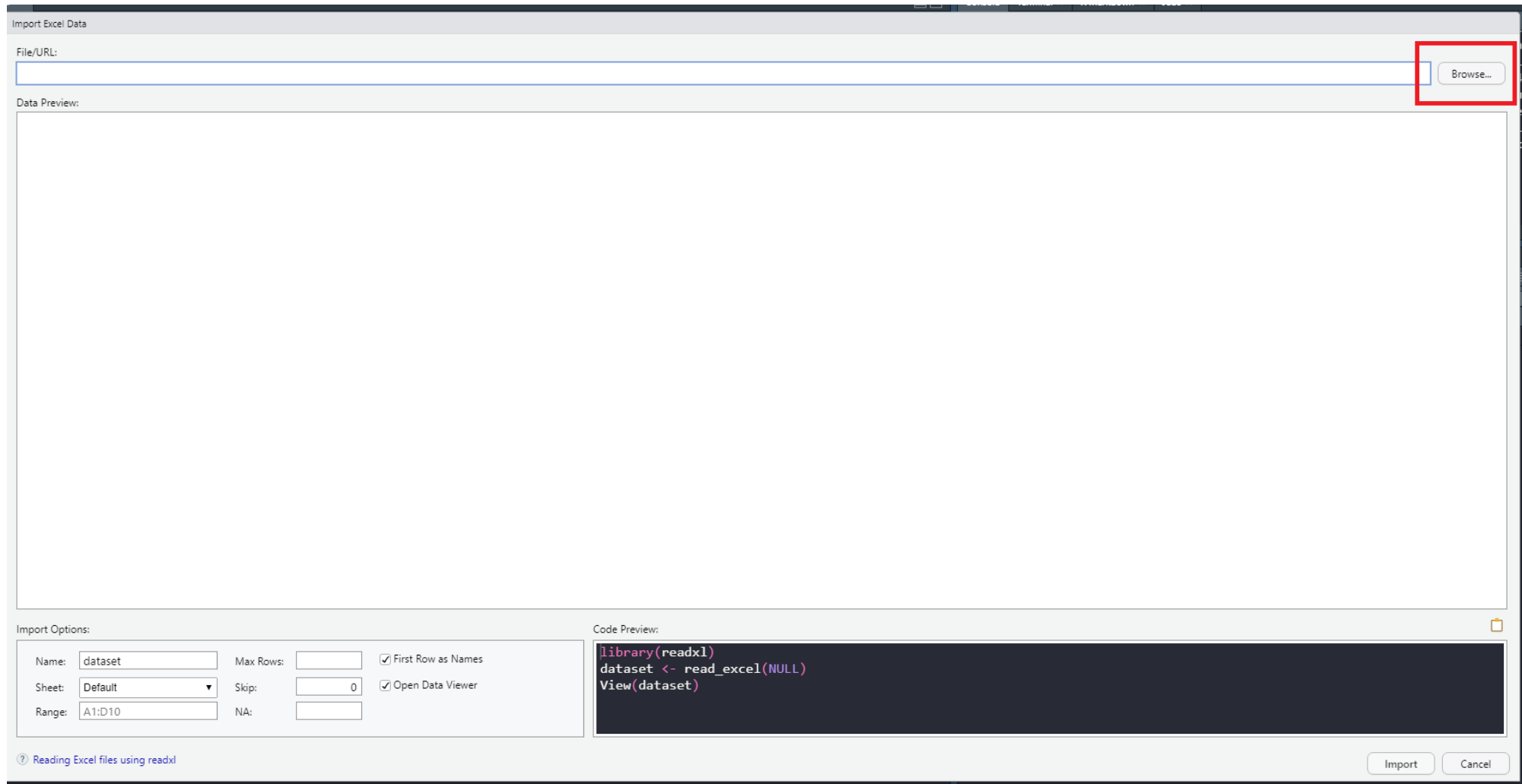
2. Salve em uma pasta do seu computador (no exemplo o documento será salvo na pasta "*Downloads*").



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



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



Import Excel Data

File/URL:

Browse...

Data Preview:

Import Options:

Name: dataset Max Rows: First Row as Names ☒

Sheet: Default Skip: 0 Open Data Viewer ☒

Range: A1:D10 NA:

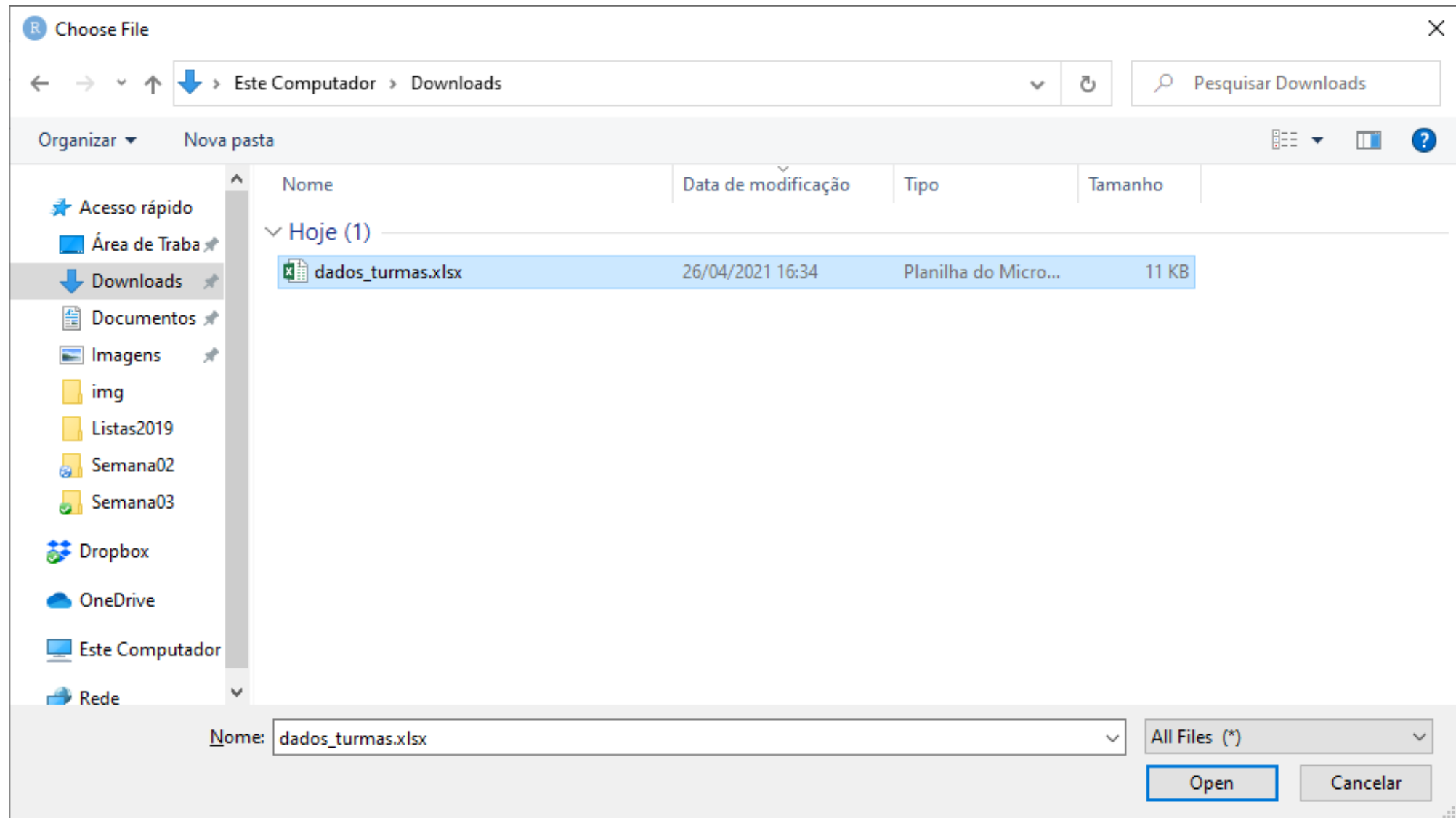
Code Preview:

```
library(readxl)
dataset <- read_excel(NULL)
View(dataset)
```

? Reading Excel files using readxl

Import Cancel

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



6. Na janela serão apresentados os dados, **NÃO CLIQUE EM IMPORT**, ao invés disso, selecione e copie os 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)
```

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

Import Cancel

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("C:/Users/Usuario/Downloads/dados_turmas.xlsx")
View(dados_turmas)
```



## 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) # não esqueça de instalar o tidyverse para ter o glimpse
glimpse(dados_turmas) # vislumbre, resumo rápido dos dados
```

```
#> Rows: 49
#> Columns: 6
#> $ id      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
#> $ sexo    <chr> "F", "F", "F", "F", "M", "M", "M", "M", "M", "M", "M", "M~
#> $ cor_cabelo <chr> "CC", "CE", "CC", "CE", "CE", "CC", "L", "CE", "CC", ~
#> $ GA      <chr> "mais_social", "nao_consomme", "pouco", "socialmente"~
```

## Exemplo da base de dados das turmas

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

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

Para essa tarefa, vamos utilizar o R. Precisaremos, então, fazer algumas operações nos dados das turmas e vamos usar o operador PIPE (`%>%`) feito com o atalho CTRL + SHIFT + M. Vamos utilizar a função `n()` para contar cada ocorrência das diferentes categorias de `sexo`

```
tab <- dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n())  
tab
```

```
#> # A tibble: 2 x 2  
#>   sexo    ni  
#>   <chr> <int>  
#> 1 F      17  
#> 2 M      32
```

- `group_by()` a grupa as categorias da variável `sexo`.
- `summarise()` vai criar o resumo dos dados, ou seja, contará o valor de cada categoria usando a função `n()` e salvara na coluna `ni`.

No R, para facilitar o processo de contagem poderíamos ordenar os dados por sexo:

```
dados_turmas %>%  
  arrange(sexo) +  
  View()
```

No Excel, primeiramente selecionamos os dados, clicamos em **Dados\Classificar** e na janela que se segue clicamos em **Adicionar Nível** e em **Classificar por**, escolhemos sexo e em **OK**.

## Frequência Absoluta ( $n_i$ )

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

Então temos:

$$n_F = 17 \text{ e,}$$

$$n_M = 32$$

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).

$$\sum_{i=1}^k n_i = 17 + 32 = 49$$

## Frequência Relativa ( $f_i$ ) ou proporção

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

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

Vamos, mais uma vez, utilizar o R para esses cálculos.

```
tab <- dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni))  
tab
```

```
#> # A tibble: 2 x 3  
#>   sexo    ni    fi  
#>   <chr> <int> <dbl>  
#> 1 F      17 0.347  
#> 2 M      32 0.653
```

- `mutate()` permitirá criarmos mais uma coluna na nossa tabela de resumo a partir da coluna  $n_i$ , no caso a frequência relativa  $f_i$ .

Portanto:  $f_F = \frac{17}{49} = 0,347$  e  $f_M = \frac{32}{49} = 0,653$

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

$$\sum_{i=1}^k f_i = 1$$

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

## Porcentagem de frequência (*perc* ou %)

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

```
tab<-dados_turmas %>%  
  group_by(sexo) %>%  
  summarise(ni=n()) %>%      # Frequência Absoluta  
  mutate(fi = ni/sum(ni),    # Frequência Relativa  
          perc = fi*100)  
tab
```

```
#> # A tibble: 2 x 4  
#>   sexo    ni    fi  perc  
#>   <chr> <int> <dbl> <dbl>  
#> 1 F      17 0.347  34.7  
#> 2 M      32 0.653  65.3
```

**OBS:** A soma das Porcentagens de frequência é igual a 100%.

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`. Observe que os gráficos serão construídos a partir do objeto `tab` anteriormente definido.

O `ggplot` funciona como 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.

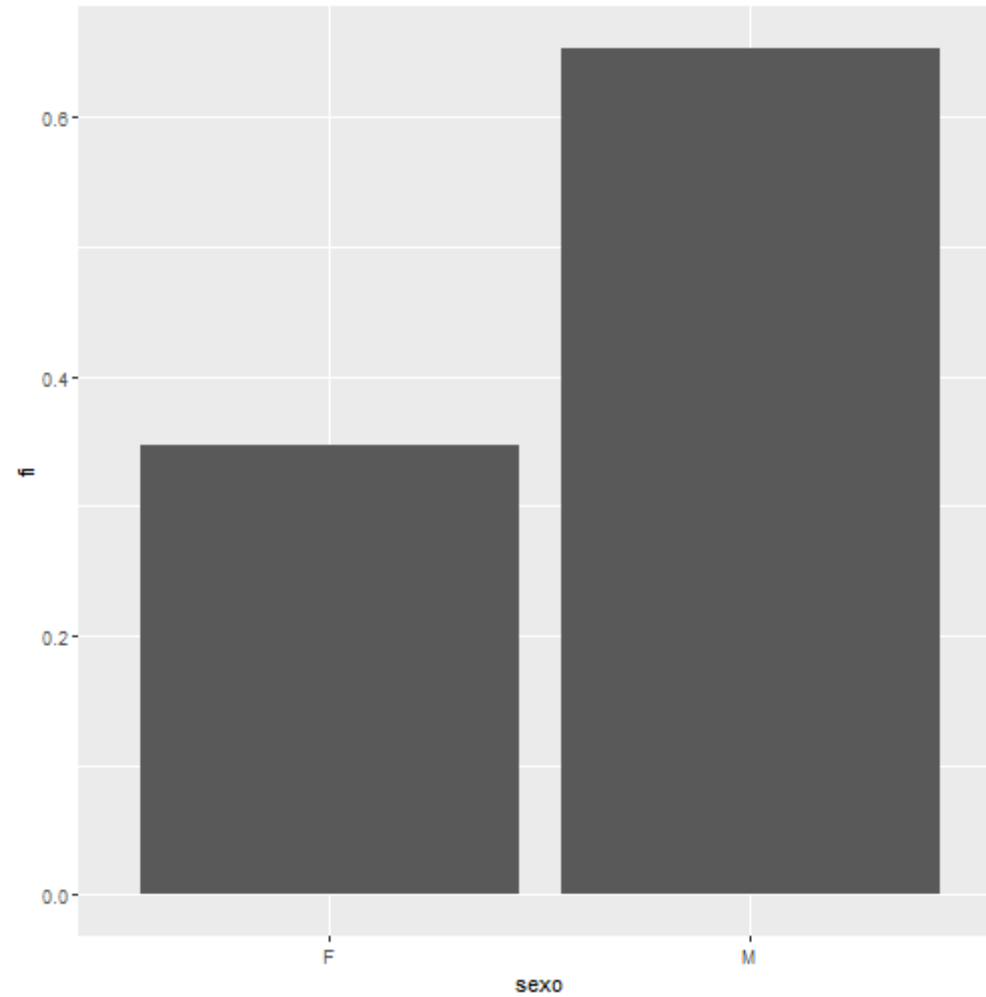
## Gráfico de Colunas para Sexo

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

```
tab %>%  
  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.

```
tab %>%  
  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

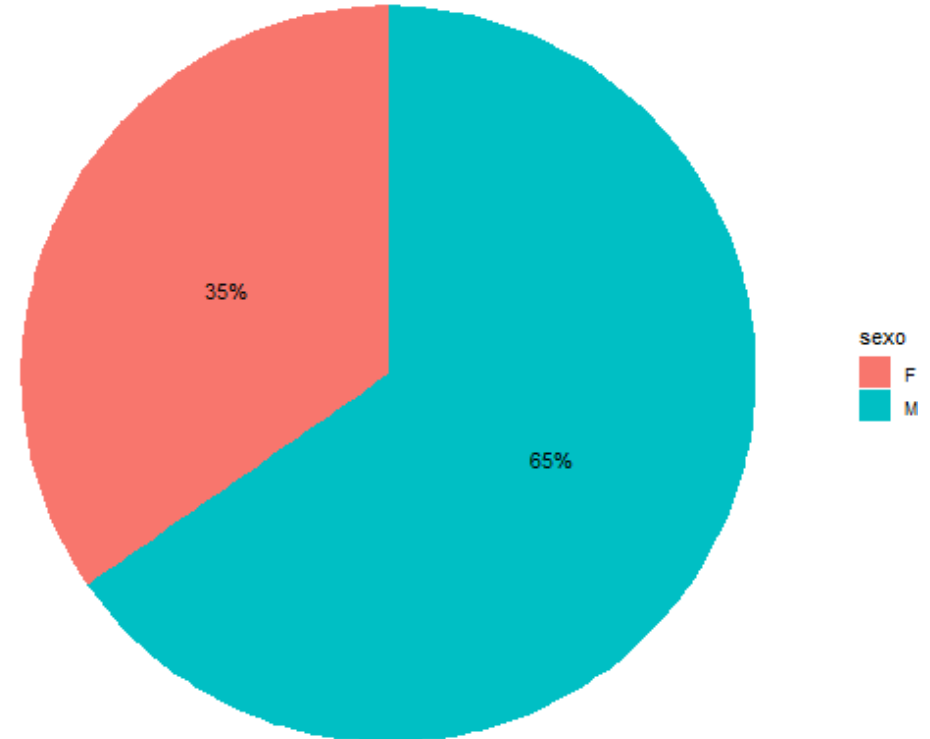
Também conhecido como gráfico de Pizza (ou torta em inglês - *pie*), ele representa cada valor de frequência relativa das diferentes categorias da variável em uma circunferência.

```
tab %>%  
  ggplot(aes(x="",y=fi, fill=sexo)) +  
  geom_bar(stat="identity") +  
  coord_polar("y", start=0) +  
  theme_void()
```

- 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.

## Vamos Adicionar os valores de $f_i$ no gráfico

```
tab %>%  
  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 =
```



# Tabela de Frequência para a variável Cor de cabelo

Vamos, mais uma vez, utilizar o R para conseguirmos as tabelas e a representação gráfica. Observe que esse é o mesmo código daquele apresentado no slide 13, mas com o nome da coluna `cor_cabelo` ao invés de `sexo` dentro da função `group_by()`.

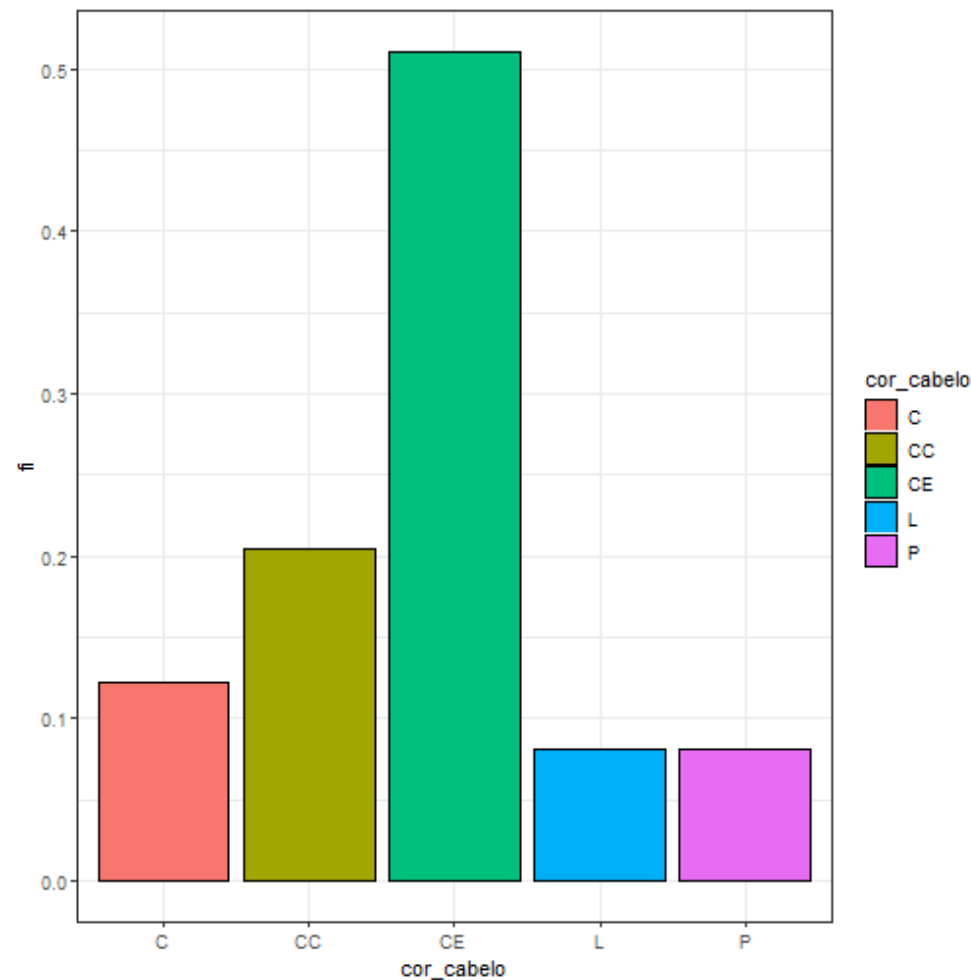
```
tab<-dados_turmas %>%  
  group_by(cor_cabelo) %>%  
  summarise(ni=n()) %>%  
  mutate(fi = ni/sum(ni),  
         perc = fi*100)  
tab
```

```
#> # A tibble: 5 x 4  
#>   cor_cabelo    ni      fi  perc  
#>   <chr>      <int>  <dbl> <dbl>  
#> 1 C          6 0.122  12.2  
#> 2 CC         10 0.204  20.4  
#> 3 CE         25 0.510  51.0  
#> 4 L          4 0.0816  8.16  
#> 5 P          4 0.0816  8.16
```

## Gráfico de Colunas para Cor de cabelo

```
tab %>%  
  ggplot(aes(x=cor_cabelo,y=fi,  
             fill=cor_cabelo)) +  
  geom_col(color="black")+  
  theme_bw()
```

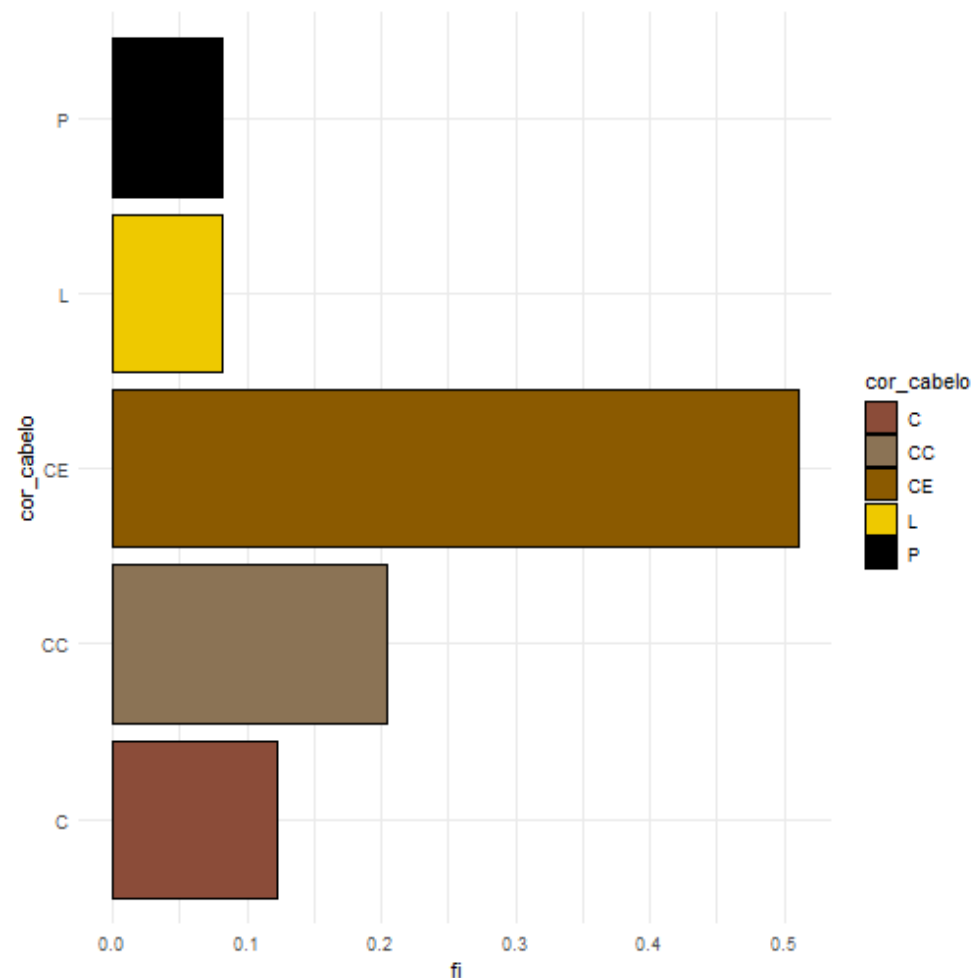
- ao passarmos o argumento `fill=cor_cabelo` dentro da função `aes()` estamos pedindo o mapeamento das cores de cabelo a partir de cores de preenchimento diferentes.
- `aes()` representa a estética do gráfico, ou seja, quem é x, quem é y e quem deve ser mapeado.
- a função `theme_bw()` muda o padrão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



## Gráfico de Barras para Cor de cabelo

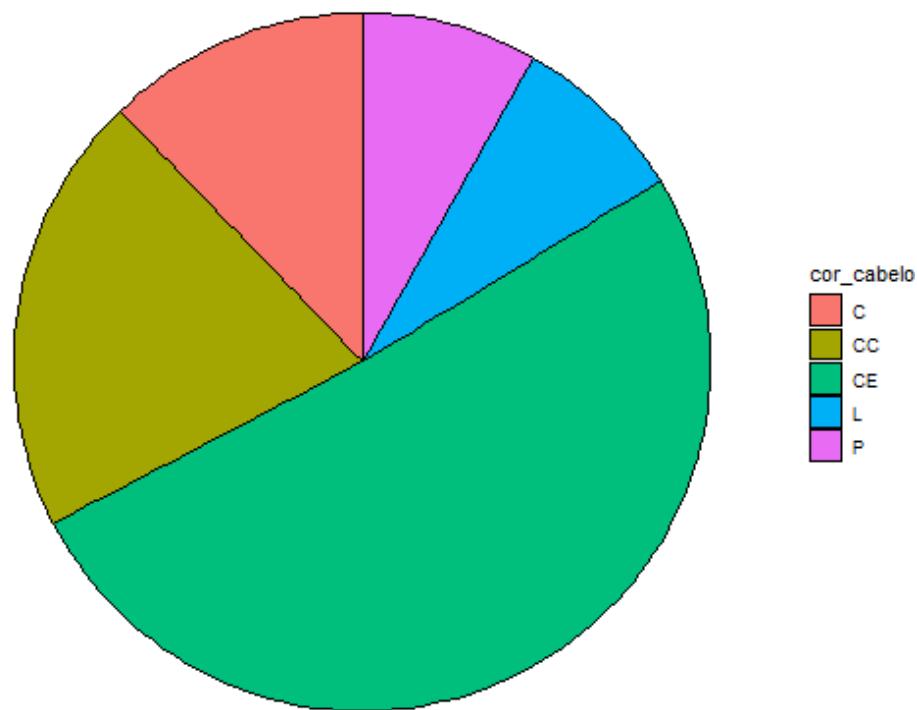
```
tab %>%  
  ggplot(aes(x=fi, y=cor_cabelo,  
             fill=cor_cabelo)) +  
  geom_col(color="black")+  
  scale_fill_manual(values = c("salmon4",  
                               "burlywood4",  
                               "orange4",  
                               "gold2",  
                               "black"))+  
  
  theme_minimal()
```

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



## Gráfico de Setores para cor\_cabelo

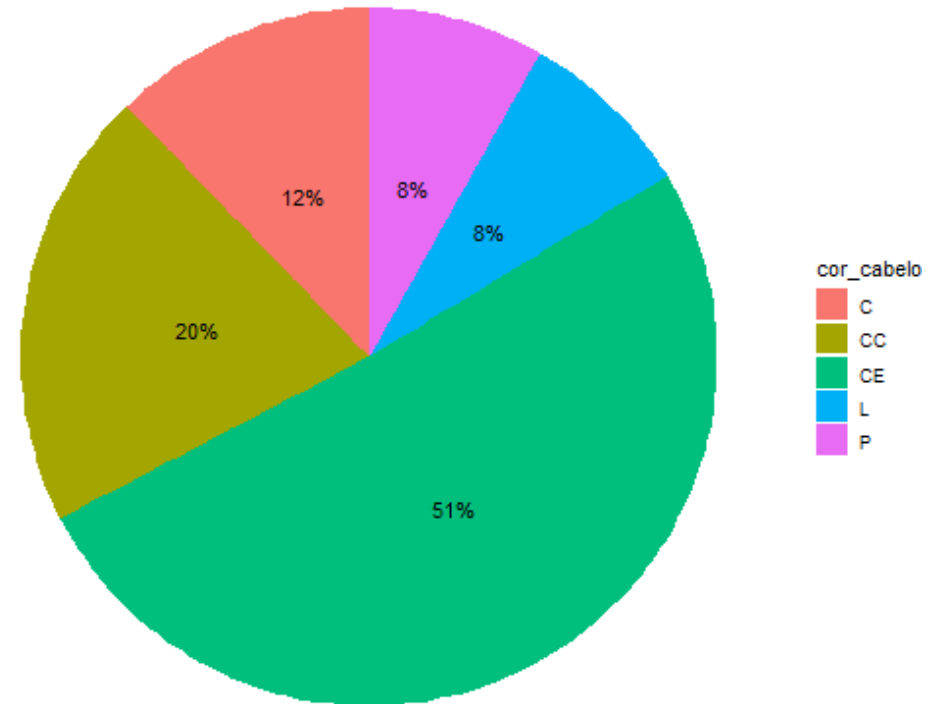
```
tab %>%  
  ggplot(aes(x="", y=fi,  
             fill=cor_cabelo)) +  
  geom_bar(stat="identity", color="black") +  
  coord_polar("y", start=0) +  
  theme_void()
```





## Vamos Adicionar os valores de *perc* no gráfico

```
tab %>%  
  ggplot(aes(x="",y=fi, fill=cor_cabelo)) +  
  geom_bar(stat="identity") +  
  coord_polar("y", start=0) +  
  theme_void() +  
  geom_text(aes(label = paste0(round(perc), "%"),  
                position = position_stack(vjust = 0.5)))
```



# 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;

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

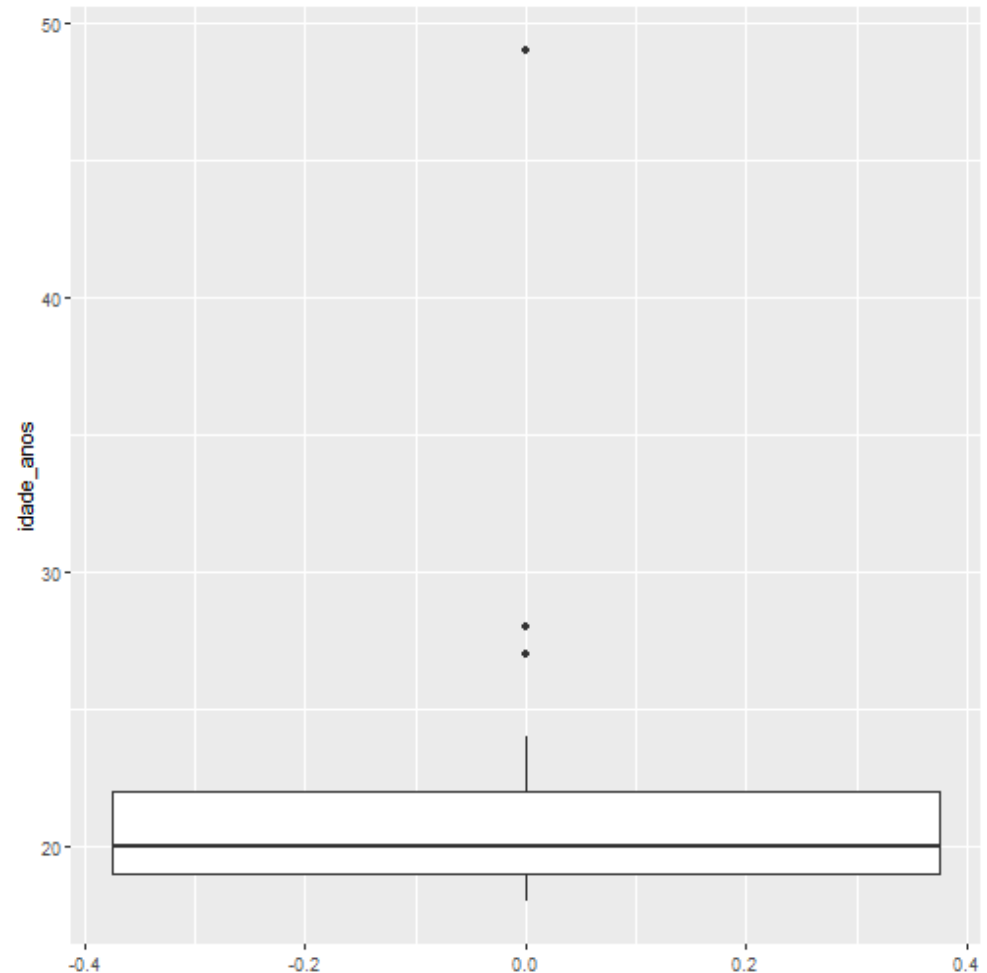
idade_anos	ni	fi	perc
18	8	0.1632653	16.326531
19	14	0.2857143	28.571429
20	11	0.2244898	22.448980
21	3	0.0612245	6.122449
22	3	0.0612245	6.122449
23	6	0.1224490	12.244898
24	1	0.0204082	2.040816
27	1	0.0204082	2.040816
28	1	0.0204082	2.040816
49	1	0.0204082	2.040816

# Gráfico Boxplot

Conhecido como gráfico dos 5 números representa um resumo dos valores *mínimo*, *primeiro quartil*, *mediana*, *terceiro quartil* e *máximo* de uma variável. É construído a partir de `geom_boxplot()`.

Observe que dentro da função `ggplot()` não é necessário passar o `x`, somente é atribuído a `y` a variável discreta `idade_anos`.

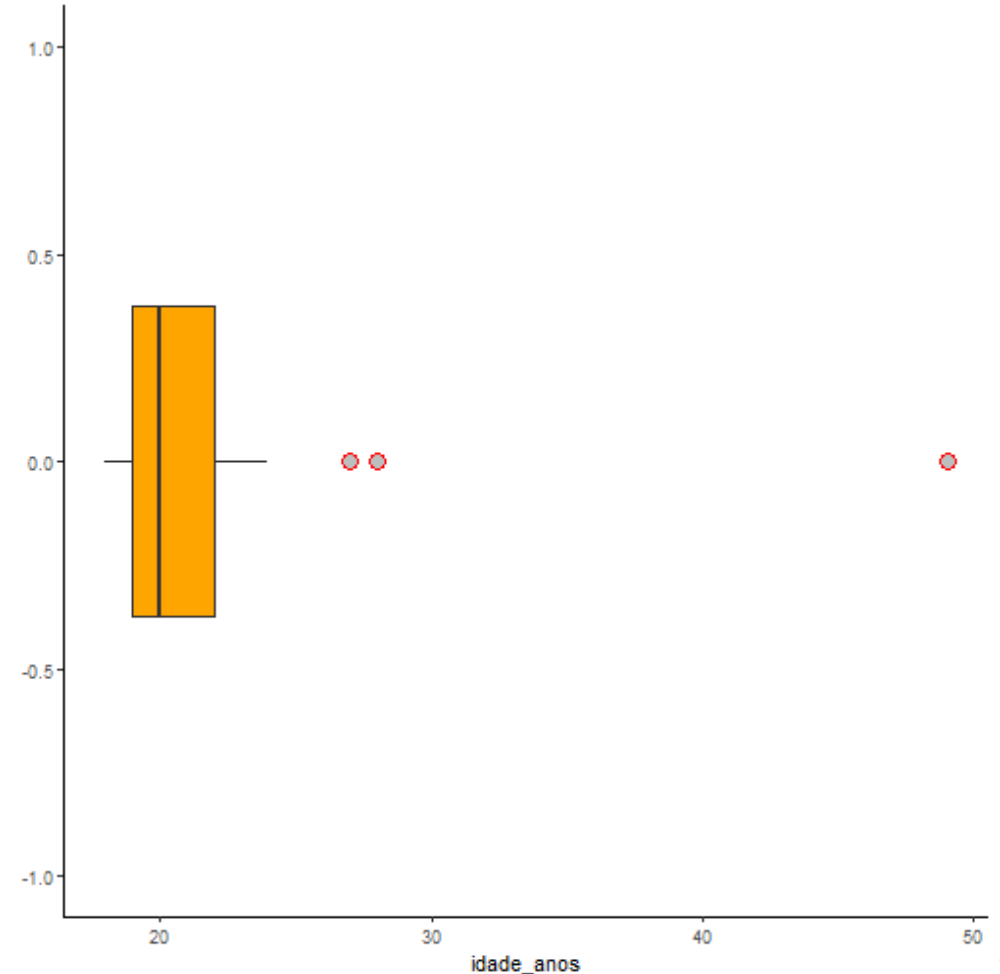
```
dados_turmas %>%  
  ggplot(aes(y=idade_anos)) +  
  geom_boxplot()
```



- O boxplot é uma caixa que vai do 25º percentil ao 75º percentil da distribuição, uma distância conhecida como a **amplitude interquartil** (IIQ).
- No meio da caixa há uma linha que exibe a **mediana**, isto é, 50º percentil, da distribuição. Essas três linhas lhe dão um sentido da dispersão da distribuição e se ela é ou não simétrica sobre a mediana ou enviesada para um lado.
- Pontos visuais que exibem observações são aqueles que caíram em mais do que 1,5 vezes o IIQ de cada limite da caixa. Esses pontos fora da curva, denominados **outliers** são incomuns, então são plotados individualmente.
- Uma linha que se estende de cada lado da caixa e vai até o ponto mais distante da distribuição que não seja um valor incomum.

Suas coordenadas podem ser transpostas trocando y por x no código anterior e o tamanho da caixa, nesse caso, pode ser controlado por `coord_cartesian(ylim=c(-1,1))`. Utilize `fill` para controlar a cor de preenchimento da caixa, e as opções `outlier` para controlar tamanhos, formas, cores e preenchimentos dos outliers.

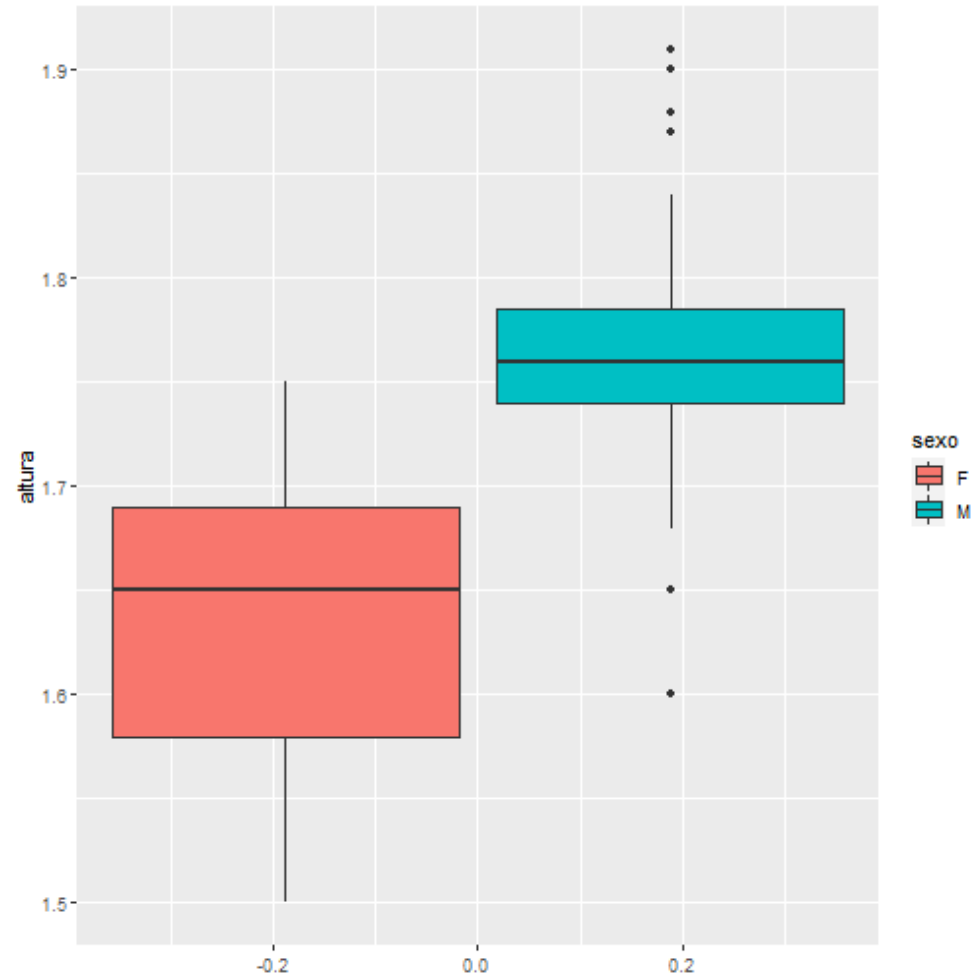
```
dados_turmas %>%  
  ggplot(aes(x=idade_anos)) +  
  geom_boxplot(fill="orange",  
               outlier.size = 4,  
               outlier.shape = 21,  
               outlier.color = "red",  
               outlier.fill = "gray") +  
  coord_cartesian(ylim=c(-1,1)) +  
  theme_classic()
```



Outra alternativa para exibir a distribuição de uma variável quantitativa, no exemplo `altura`, é desmembrá-la por uma variável categórica aqui no boxplot.

```
dados_turmas %>%  
  ggplot(aes(y=altura,  
             fill=sexo)) +  
  geom_boxplot()
```

Para fazer isso basta passarmos como preenchimento o nome da variável categórica, no exemplo, `sexo`.



# Tabela de frequência para variável Altura (Quantitativa Contínua)

Quando a variável quantitativa for contínua, recomendamos a utilização dos gráficos boxplot, 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 mensurarmos uma variável contínua, temos 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()`.

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

classes_altura	ni	fi	perc
(1.5,1.58]	5	0.1020408	10.20408
(1.58,1.66]	8	0.1632653	16.32653
(1.66,1.75]	12	0.2448980	24.48980
(1.75,1.83]	17	0.3469388	34.69388
(1.83,1.91]	7	0.1428571	14.28571



# 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 ( $\Delta$ ), 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$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 ( $\Delta_i$ )

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:

$$\Delta_i = \frac{\Delta}{k} = \frac{0,41 \text{ m}}{5} = 0,082 \text{ m}$$

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

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

Cada um dos 5 intervalos terá uma amplitude de 0,082 m. Ou seja, o cálculo dos limites das classes é feito a partir da adição ao valor Mínimo o valor de  $\Delta_i$   $k$  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 qua a função `cut()` utilizou para calcular os limites de classes, e essa é a metodologia clássica para lidar com dados contínuos e os agrupar em classes.

<b>classes_altura</b>	<b>ni</b>	<b>fi</b>	<b>perc</b>
(1.5,1.58]	5	0.1020408	10.20408
(1.58,1.66]	8	0.1632653	16.32653
(1.66,1.75]	12	0.2448980	24.48980
(1.75,1.83]	17	0.3469388	34.69388
(1.83,1.91]	7	0.1428571	14.28571

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

Agora podemos criar um gráfico de barras para cada uma dessas classes, denominado hitograma:

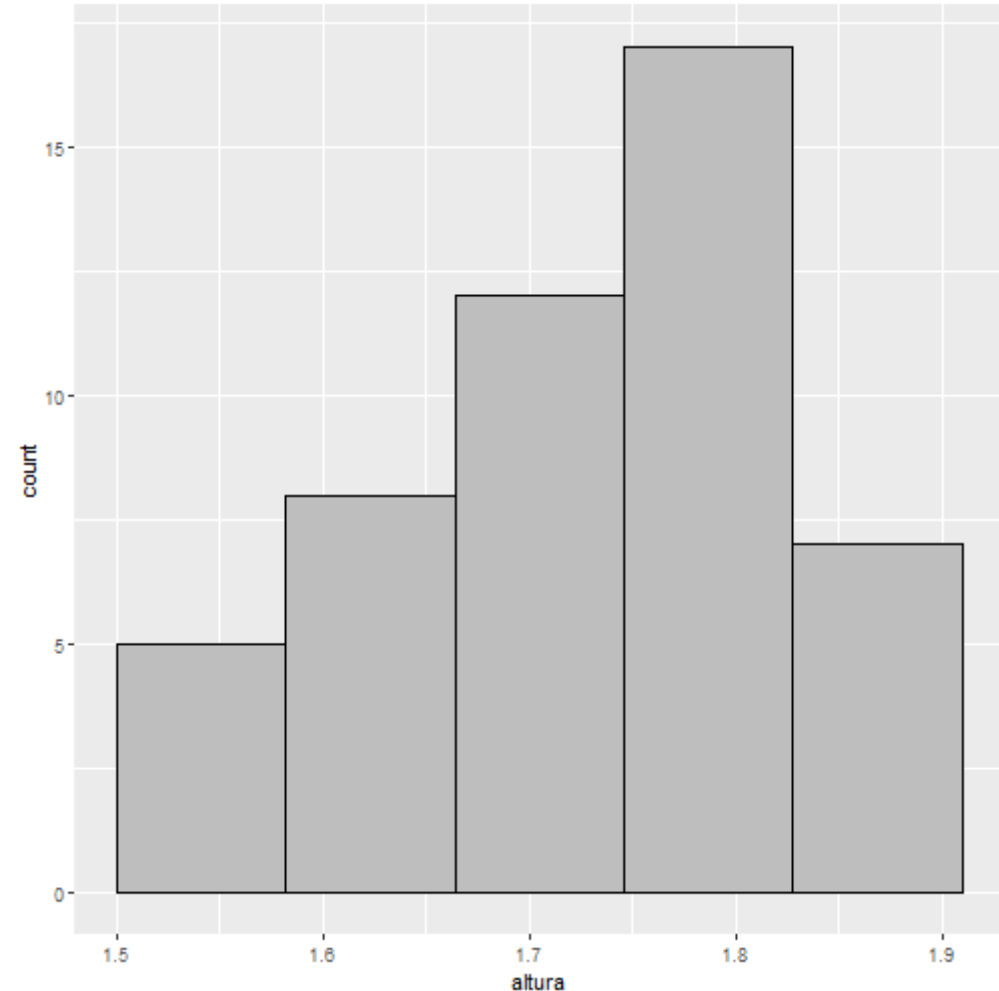
# Gráfico histograma (frequências absolutas)

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.



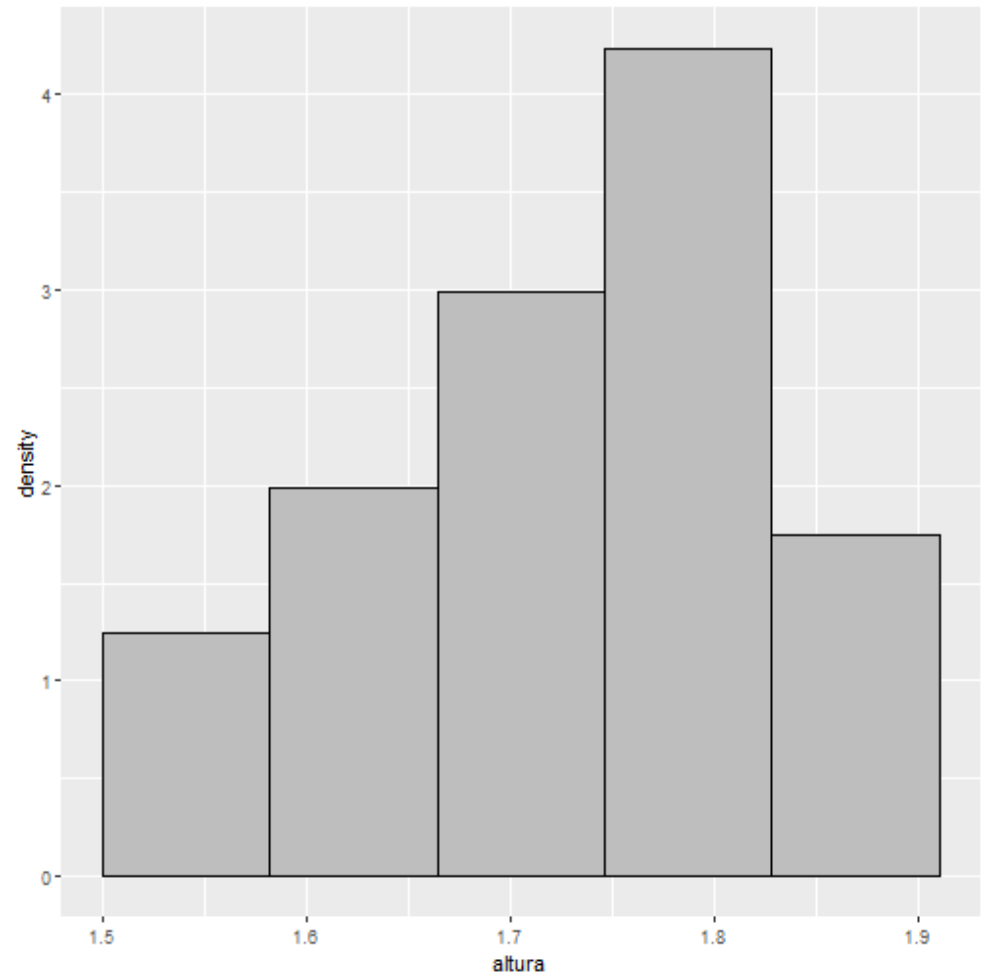
# Gráfico histograma (frequências relativas)

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  $f_i/\Delta_i$  (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:

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

```
tab <- 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)  
View(tab)
```

classes_altura	ni	fi	perc	di
(1.5,1.58]	5	0.1020408	10.20408	1.244400
(1.58,1.66]	8	0.1632653	16.32653	1.991040
(1.66,1.75]	12	0.2448980	24.48980	2.986560
(1.75,1.83]	17	0.3469388	34.69388	4.230961
(1.83,1.91]	7	0.1428571	14.28571	1.742160

# Medidas de Frequências 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.

```
tab <- 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))
```

```
View(tab)
```

# Tabela de Frequência para Altura

classes_altura	ni	fi	perc	di	Ni	Fi	Perc
(1.5,1.58]	5	0.1020408	10.20408	1.244400	5	0.1020408	10.20408
(1.58,1.66]	8	0.1632653	16.32653	1.991040	13	0.2653061	26.53061
(1.66,1.75]	12	0.2448980	24.48980	2.986560	25	0.5102041	51.02041
(1.75,1.83]	17	0.3469388	34.69388	4.230961	42	0.8571429	85.71429
(1.83,1.91]	7	0.1428571	14.28571	1.742160	49	1.0000000	100.00000



# Histograma e Polígono de Frequência

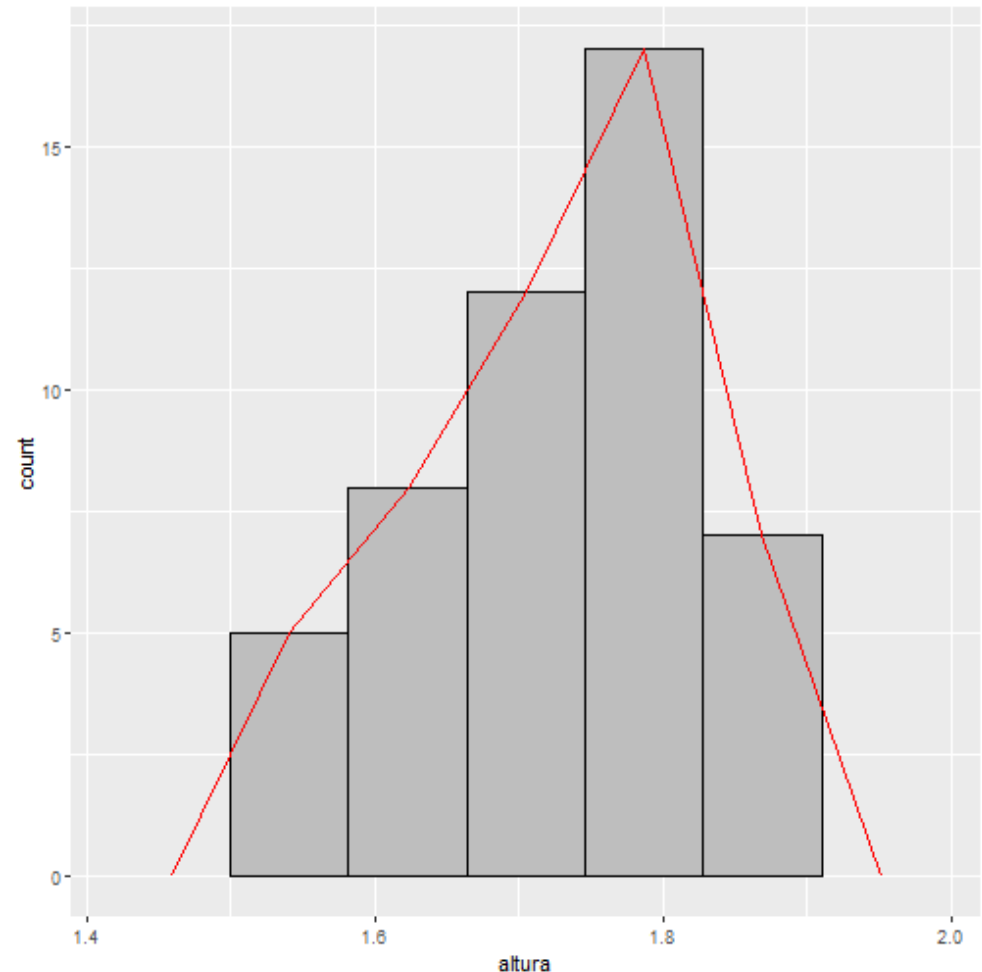
Obtemos o polígono de frequências unindo por uma poligonal (segmentos de retas) os pontos correspondentes às frequências, das classes, centradas nos pontos médios de cada classe.

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")
```

Observe que o histograma foi construído com as frequências absolutas ( $n_i$ ), ou seja,  $y=..count...$

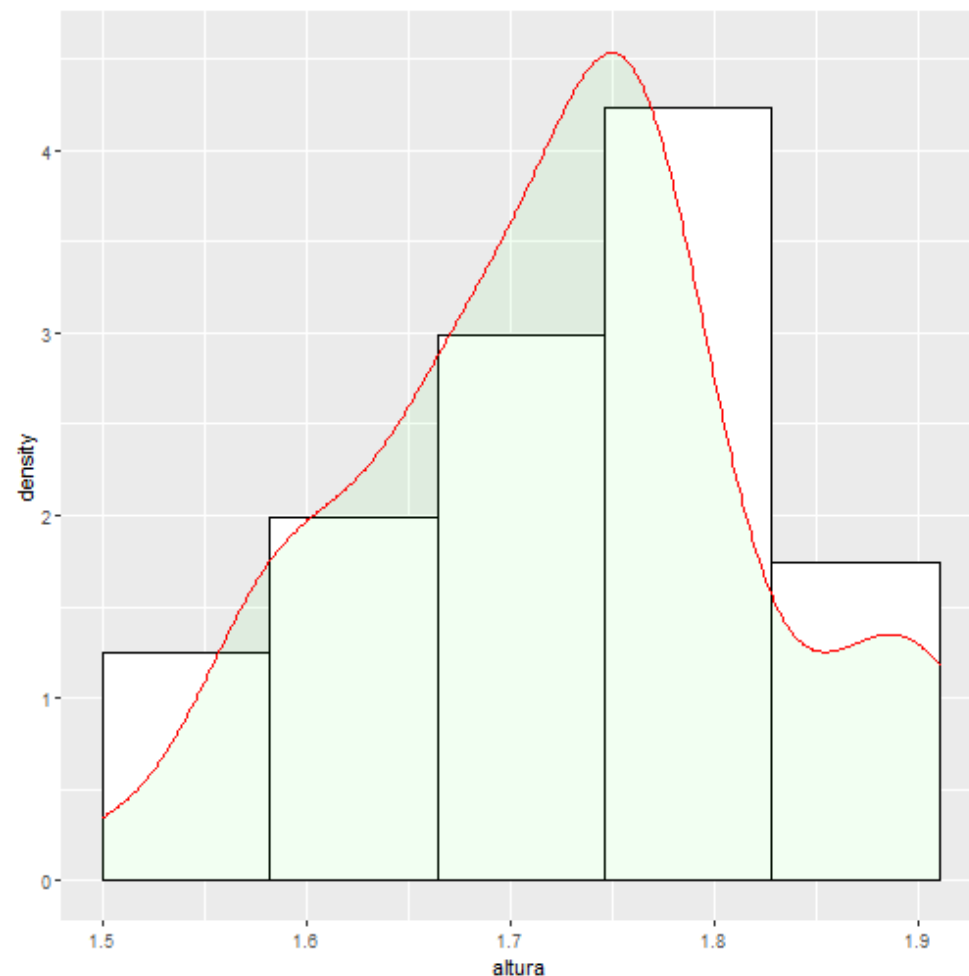


# Histograma e Estimativas de densidade suavizadas

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

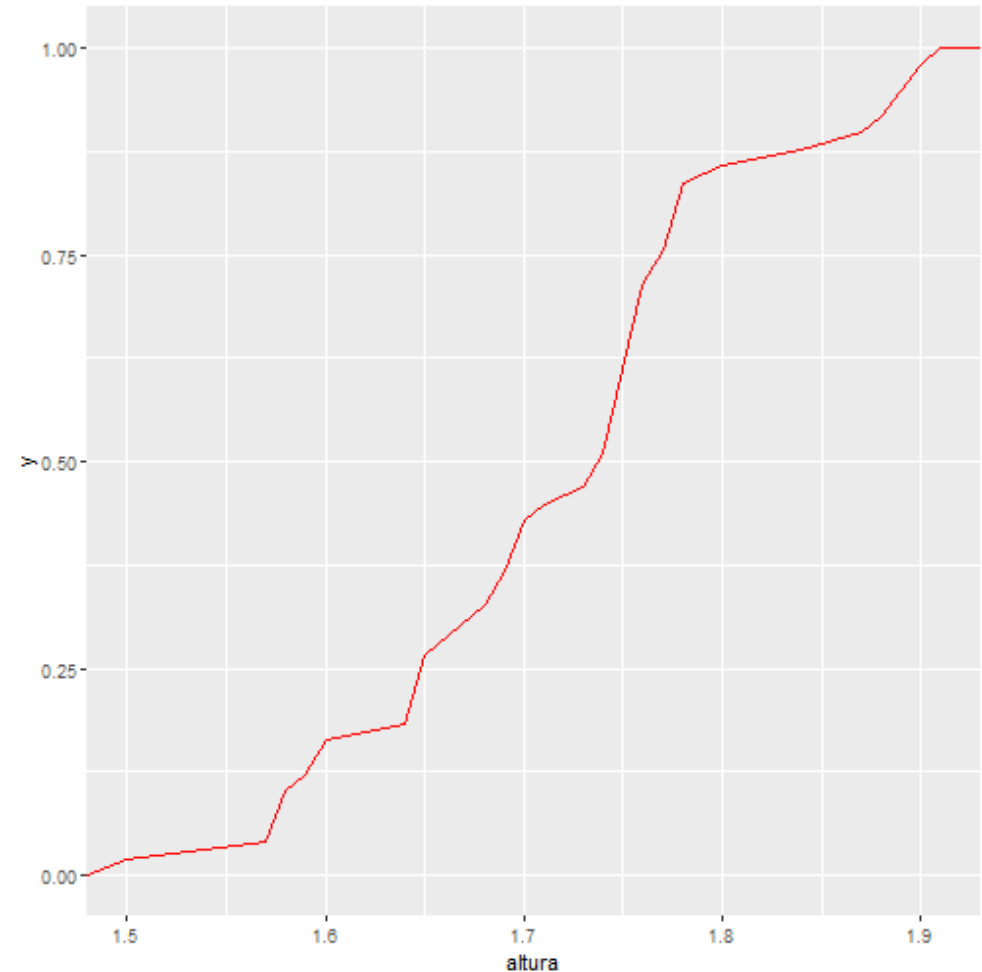
A função de distribuição acumulada descreve como probabilidades são associadas aos valores ou aos intervalos de valores de uma variável aleatória. Em outras palavras, ela representa a probabilidade de uma variável aleatória ser menor ou igual a um valor real qualquer  $x$ .

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

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")
```

