UNIVERSIDADE DE PASSO FUNDO
FACULDADE DE CIÊNCIAS ECONÔMICAS,
ADMINISTRATIVAS E CONTÁBEIS
CENTRO DE PESQUISA E EXTENSÃO DA FEAC
(www.upf.br/cepeac)

Texto para discussão

Texto para discussão Nº 04/2021

Estatística Descritiva e Gráficos no R-Studio

Andre da Silva Pereira
Luan Marca
Edson Jesus de Paiva Silva Filho

Passo Fundo - RS - Brasil

Estatística Descritiva e Gráficos no R-Studio

Andre da Silva Pereira
Luan Marca
Edson Jesus de Paiva Silva Filho

Universidade de Passo Fundo (UPF)

Programa de Pós-Graduação em Administração (**PPGAdm**)

Sumário

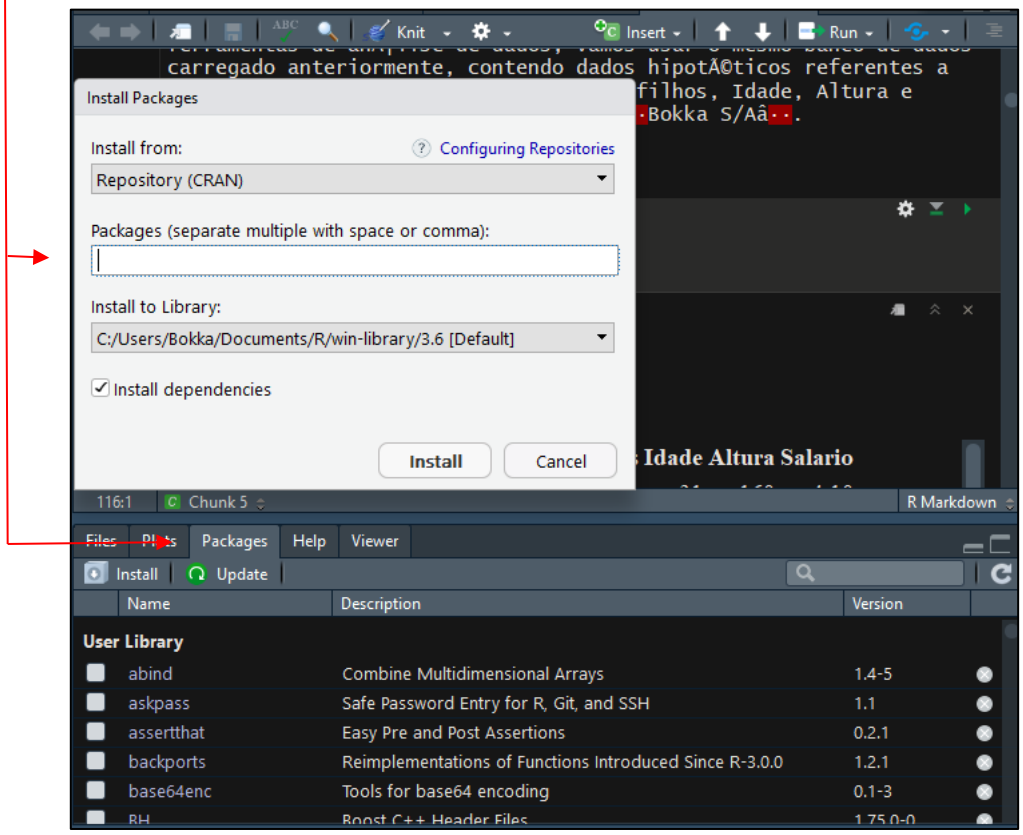
1. Instalação de Pacotes

A linguagem R proporciona grande utilidade aos seus usuários, isso decorre do fato de a mesma possuir uma vasta coleção de pacotes. Um simples pacote do R dispõe de uma coleção de funções, dados e documentação, o que possibilita uma expansão das funcionalidades básicas do software. Esses pacotes são disponibilizados em uma grande comunidade composta por inúmeros programadores.

Existem maneiras diversas de se instalar pacotes, a seguir veremos 2 maneiras mais comuns.

1.1. Instalar pacotes usando o Menu

Umas das maneiras mais fáceis de instalar um pacote no R-Studio é usando o método manual, usando à aba **Packages** localizada no canto inferior direito. Para isso, basta clicar na opção **Packages** -> **Install**, e na sequência, preencher o campo de busca com o nome do pacote desejado.



1.2. Instalação usando Código

Para instalar um pacote do R usando código, usamos a função **install.packages()**.

Vamos baixar agora o primeiro pacote que utilizaremos na sequência, chamado **dplyr**

install.packages("dplyr").

O ***dplyr*** é um pacote capaz de realizar transformação de dados, reunindo facilidade e competência, os scripts que utilizam os verbos dplyr e as habilidades do operador *pipe* proporcionam mais nitidez e organização sem comprometer a velocidade de execução.

As principais funções do dplyr são:

- select() - seleciona colunas
- arrange() - ordena a base
- filter() - filtra linhas
- mutate() - cria/modifica colunas
- group_by() - agrupa a base
- summarise() - sumariza a base

Algumas vantagens de usar o **dplyr** são:

- Manipulação de dados simplificada.
- Codificação intuitiva e de simples leitura.
- O pacote *dplyr* opera C e C++ depois da maior parte das funções o que proporciona mais agilidade ao código.
- Combinação de diferentes fontes de dados: (SQL) e data.table.

1.3. Instalação Condicional

Usando **if()** e **require ()**: Essa função carrega o pacote caso ele já esteja instalado, caso não esteja instalado o código enviará uma instrução para baixá-lo e instalá-lo.

Exemplo:

```
if(! require(dplyr))
  install.packages("dplyr")
```

1.4. Carregando os Pacotes

Baixar um pacote no R não significa que ele estará sempre disponível para uso, ele só estará disponível para uso quando estiver carregado.

Para carregar um pacote usamos a função **library()**

Exemplo:

```
library(dplyr)
```

Na sequência desse módulo vamos baixar e detalhar uma gama de pacotes úteis que irão facilitar muito o trabalho com o R-Staudio.

2. Carregando banco de dados

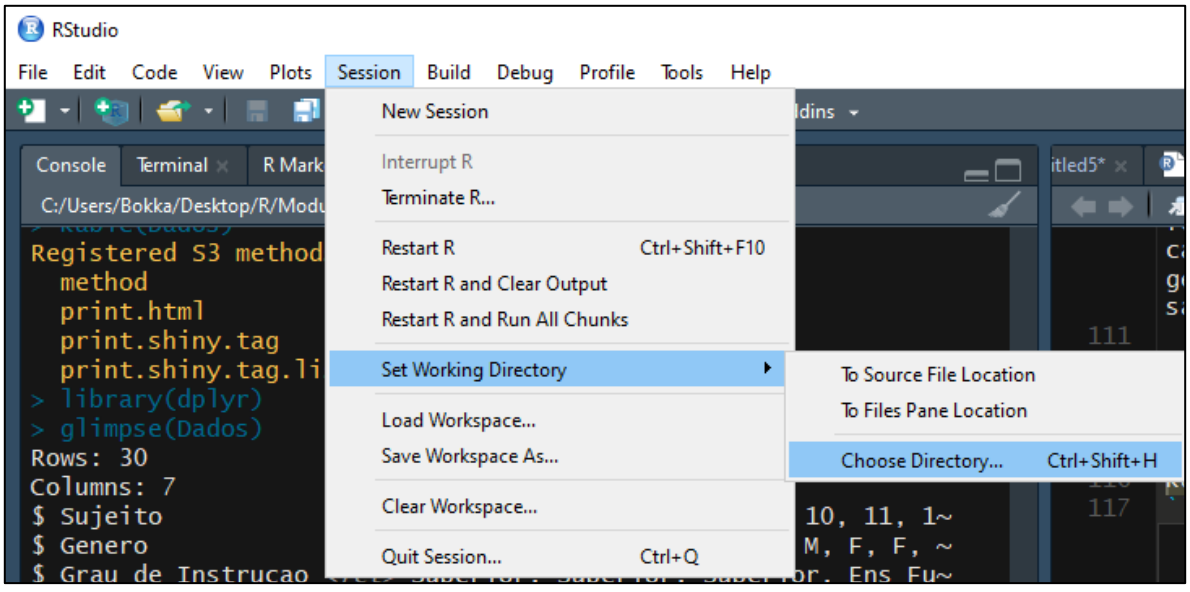
Existem diferentes formas para se carregar dados no R. O formato mais adequado vai depender do tamanho do conjunto de dados, e se os dados já existem em outro formato para serem importados ou se serão digitados diretamente no R.

Como já abordamos no modulo 1 como inserir dados manualmente no R a partir da criação de data frames, a partir de agora veremos como importar dados de diferentes formatos.

2.1. Entrada de dados em arquivos de texto (csv)

Caso os dados já estejam disponíveis em formato eletrônico, isto é, se já foram digitados em outro programa, você pode importar os dados para o R sem a necessidade de digitá-los novamente.

Antes de carregar um arquivo no R é necessário definir o diretório de trabalho, para isso, basta acessar o menu **Session** -> **Set working Directory** -> **Choose Directory**. Na nova aba, você pode escolher a pasta onde o arquivo de texto está armazenado. Quando você enviar uma instrução ao R para ler um arquivo, ele irá procurar nessa pasta de trabalho.



No R, usa-se a função **read.table()** para ler os dados de um arquivo texto e armazenar no formato de um data frame.

Exemplo:

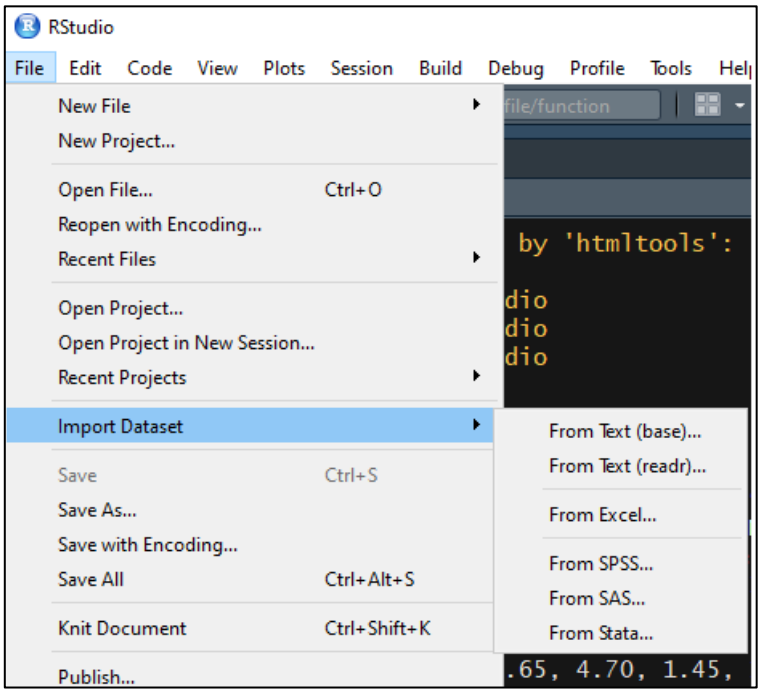
```
Dados <- read.csv2('Banco de Dados 2.csv', stringsAsFactors = T)
```

Para visualizar os dados usamos as funções **View()** e **Glimpse()**:

```
View(Dados)
glimpse(Dados)

## Rows: 30
## Columns: 7
## $ Sujeito         <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ Genero          <fct> M, F, M, M, F, F, M, M, M, M, F, F, F, F, F, M, M, M~
## $ Grau_de_Instruçao <fct> Superior, Superior, Superior, Ens Fundamental, Ensin~
## $ N_Filhos        <int> 1, 0, 0, 0, 0, 2, 0, 0, 1, 2, 0, 3, 0, 0, 1, 1, 0, 0~
## $ Idade           <int> 31, 25, 33, 20, 23, 37, 38, 37, 34, 40, 41, 46, 26, ~
## $ Altura          <int> 160, 160, 157, 163, 163, 155, 165, 168, 163, 170, 17~
## $ Salario         <dbl> 4.10, 2.65, 4.70, 1.45, 1.85, 2.20, 2.35, 2.70, 2.90~
```

2.2. Importando dados de diversos formatos através do menu

Outra maneira simples de carregar dados no R é através do menu **file**. Basta acessar **file** -> **Import Dataset**. Com essa ferramenta, podemos importar planilhas do Excel, SPSS, entre outros.



3. Estatística Descritiva no R-Studio

Nesta sessão veremos alguns comandos do R para fazer uma análise descritiva de um conjunto de dados. Para ilustrar algumas ferramentas de análise de dados, vamos usar o mesmo banco de dados carregado anteriormente, contendo dados hipotéticos referentes a gênero, grau de instrução, número de filhos, Idade, Altura e salário de 30 sujeitos da companhia Bokka S/A.

Base de dados que Vamos utilizar:

```
library(knitr)
kable(Dados)
```

Sujeito	Genero	Grau_de_Instruçao	N_Filhos	Idade	Altura	Salario
1	M	Superior	1	31	160	4.10
2	F	Superior	0	25	160	2.65
3	M	Superior	0	33	157	4.70
4	M	Ens Fundamental	0	20	163	1.45
5	F	Ensino Médio	0	23	163	1.85
6	F	Superior	2	37	155	2.20
7	M	Ens Fundamental	0	38	165	2.35
8	M	Superior	0	37	168	2.70
9	M	Superior	1	34	163	2.90
10	M	Ensino Médio	2	40	170	1.60
11	F	Superior	0	41	170	3.00
12	F	Ensino Médio	3	46	157	3.45
13	F	Ensino Médio	0	26	165	1.00
14	F	Ensino Médio	0	41	160	1.70
15	F	Ensino Médio	1	43	173	1.85
16	M	Superior	1	27	175	2.10
17	M	Superior	0	26	145	3.20
18	M	Superior	0	42	165	5.80
19	F	Ensino Médio	2	43	157	4.30
20	M	Ensino Médio	2	30	163	2.75
21	M	Ens Fundamental	2	35	168	3.40
22	F	Ens Fundamental	0	33	183	2.05
23	F	Ensino Médio	0	30	175	2.30
24	F	Superior	2	32	185	3.30
25	M	Ensino Médio	4	29	173	3.65
26	F	Ens Fundamental	1	40	173	3.65
27	F	Superior	1	36	191	4.15
28	M	Ens Fundamental	0	34	188	1.90
29	M	Superior	0	31	193	3.10
30	F	Superior	1	35	170	4.00

3.1. Frequências Absolutas

É uma terminologia estatística que representa o número de vezes que um determinado dado ou valor específico surge em meio a uma ou mais tentativas. Especificamente a frequência absoluta representa a contagem simples em que determinado valor é observado, esta é expressa em número inteiro e considerada básica, tratando-se de análise estatística.

Para se criar uma tabela simples de frequência absoluta, chamamos uma função nativa do R chama **table()**, dentro dos parênteses () identificamos a base de dados e a variável a ser incluída na tabela. Deve-se usar o **$** para selecionar a variável desejada.

Exemplo:

```
table(Dados$Genero)

##
##  F  M
## 15 15

table(Dados$Grau_de_Instruçao)

##
## Ens Fundamental    Ensino Médio       Superior
##               6              10              14
```

3.2. Tabela de Referência Cruzada

Uma tabela de referência cruzada considera simultaneamente duas variáveis, uma para as colunas e outra para as linhas. Para criar uma tabela cruzada utilizamos a função **table()** e as variáveis separadas por virgulas.

Exemplo:

```
table(Dados$Genero, Dados$Grau_de_Instruçao)

##
##    Ens Fundamental Ensino Médio Superior
##  F               2             7        6
##  M               4             3        8
```

3.2. Frequências Relativas

Mostra a porcentagem de vezes que determinada resposta se manifestou em relação ao todo. Empregamos a frequência relativa para contrapor um dado em relação ao todo.

Para se criar uma tabela de frequência relativa usamos a função **prop.table()**, dentro dessa função indicamos a tabela e a variável.

Exemplo:

```
prop.table(table(Dados$Genero))

##


##   F   M
## 0.5 0.5

prop.table(table(Dados$Grau_de_Instruçao))

##
## Ens Fundamental    Ensino Médio       Superior
##       0.2000000       0.3333333       0.4666667
```

3.3. Variáveis Discretas (Quantitativas)

Uma variável discreta é aquela cujo valor é obtido por contagem. Por exemplo, você pode contar quantos irmãos ou primos você tem. Você pode contar o dinheiro em sua conta bancária. Você também pode contar quantos livros existem em uma biblioteca. Pode levar muito tempo para contar o último, mas o ponto é que ainda é possível contar.

Para se gerar uma tabela com variáveis discretas usamos a função **table()**, dentro dela indicamos a base de dados e a variável a ser inserida na tabela.

Exemplo:

```
table(Dados$N_Filhos)

##
##  0  1  2  3  4
## 15  7  6  1  1

table(Dados$Idade)

##
## 20 23 25 26 27 29 30 31 32 33 34 35 36 37 38 40 41 42 43 46
##  1  1  1  2  1  1  2  2  1  2  2  2  1  2  1  2  2  1  2  1
```

Para pedir uma tabela com frequência relativa chamamos a função **prop.table()**

Exemplo:

```
prop.table(table(Dados$N_Filhos))

##
##         0         1         2         3         4
## 0.50000000 0.23333333 0.20000000 0.03333333 0.03333333
```

3.4. Variáveis Continuas

Variáveis contínuas levariam (literalmente) uma eternidade para serem contadas. Isso acontece porque variáveis contínuas podem assumir quase qualquer valor numérico e podem ser significativamente divididas em incrementos menores, incluindo valores fracionários e decimais. Por exemplo, pegue as horas. Você não pode contá-las de forma exata. Isso levaria literalmente uma eternidade. Por exemplo, você pode contar 5 horas, 4 segundos, 4 milissegundos, 8 nanossegundos, 99 picosendos ... e assim por diante.

Para se montar uma tabela de frequência com variáveis contínuas é necessário criar faixas de valores, para isso, precisamos analisar a amplitude da variável que estamos trabalhando.

Vamos analisar a amplitude da variável salário, qual o menor e qual o maior valor. Para isso utilizamos a função **range()**.

Exemplo:

```
range(Dados$Salario)

## [1] 1.0 5.8
```

Menor valor, **1.0** Salário-mínimo; Maior Valor, **5.8** Salários-Mínimos.

Agora vamos avaliar a quantidade adequada de categorias usando a função **nclass.sturges().**

```
nclass.Sturges(Dados$Salario)

## [1] 6
```

A quantidade adequada de categorias nesse caso é 6.

Criação da tabela com faixas

Para criar uma tabela com faixas de valores usamos a função **table()**, dentro dela chamamos a função **cut()**, essa função divide a variável em faixas de valores, a seguir inserimos a base e identificamos a variável. Além disso, indicamos o valor mínimo e o valor máximo, dentro do parâmetro" **seq"**, por fim indicamos quantas faixas vão existir através do parâmetro **l**.

É recomendado que se coloque uma categoria a mais além da indicada, nesse caso indicaremos 7 categorias no parâmetro **l**.

Exemplo:

```
table(cut(Dados$Salario, seq(0, 6, l = 7)))

##
## (0,1] (1,2] (2,3] (3,4] (4,5] (5,6]
##    1    6   10    8    4    1
```

3.5. Medidas de tendência Central e de Dispersão

A tendência central é uma medida de valores em uma amostra que identifica os diferentes pontos centrais nos dados, muitas vezes referidos coloquialmente como "médias". As medidas mais comuns de tendência central são a **média,** a **mediana** e a **moda.** Identificar o valor central permite que outros valores sejam comparados a ele, mostrando a dispersão ou agrupamento da amostra, o que é conhecido como dispersão ou distribuição. Essas medidas de dispersão são categorizadas em 2 grupos: medidas de dispersão baseadas em **percentis** (quartis superior e inferior) e medidas de dispersão baseadas na média (que é comumente conhecido como **desvio padrão**).

A função mais comum para se obter medidas de posição é a função nativa **summary()**, basta chamar a função, identificar a base e a variável desejada. Ela fornece a média, mediana, valor minimo, máximo, primeiro quartil e terceiro quartil.

Exemplo:

```
summary(Dados$Salario)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.062   2.825   2.905   3.600   5.800
```

Para se obter medidas de dispersão, como desvio padrão, erro padrão e amplitude interquartil, utilizamos a função **describe()** do pacote **psych()**.

O pacote **psych**[1] **é** basicamente uma caixa de ferramentas de propósito geral, desenvolvida primeiramente para teoria psicométrica e psicologia experimental. As funções são principalmente para análise multivariada e construção de escala usando análise de fator, análise de componente principal, análise de cluster e análise de confiabilidade, fornecendo também estatísticas descritivas básicas.

Primeiramente baixamos o pacote utilizando a função **install.packages("psych")**.

Carregando o pacote **psych**:

```
library(psych)
```

Chamando a função, identificamos a base e a variável.

Exemplo:

```
describe(Dados$Salario)
```

```
##    vars  n mean   sd median trimmed  mad min max range skew kurtosis  se
## X1    1 30  2.9 1.09   2.83    2.85 1.19   1 5.8   4.8 0.51    -0.2 0.2
```

[1] Fonte: R Documentation.

A função **describe()** fornece o **n**, **média** (mean), **desvio padrão** (sd), **mediana** (median), **média aparada** (trimmed), **minimo** (min), **máximo** (max), **amplitude** (range), **assimetria** (skew), **curtose** (kurtosis) e **erro padrão** (se).

Além da função **describe()** o pacote "**psych"** oferece a função **describeBy()**, [2]"essa função relata estatísticas básicas de resumo por uma variável de agrupamento. útil se a variável de agrupamento for alguma variável experimental e os dados forem agregados para plotagem."

para isso montamos o script da seguinte forma:

```
describeBy(Dados$Salario, group = Dados$Genero)

##
##  Descriptive statistics by group
## group: F
##    vars  n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 15 2.76 1.01   2.65    2.78 1.19   1 4.3   3.3 0.03     -1.4 0.26
## ------------------------------------------------------------
## group: M
##    vars  n mean   sd median trimmed  mad  min max range skew kurtosis  se
## X1    1 15 3.05 1.18    2.9    2.96 1.11 1.45 5.8  4.35 0.69    -0.25 0.3
```

Veja que as medidas de dispersão da variável "**Salario"** são mostradas por grupo, Femino e masculino. Essa função facilita comparações exploratórias de dados.

Outra vantagem da função **describeBy()** é a possibilidade de se considerar duas variáveis independentes. Para isso montamos a estrutura do script da seguinte forma:

```
describeBy(Dados$Salario, group = Dados$Genero:Dados$Grau_de_Instruçao)

##
##  Descriptive statistics by group
## group: F:Ens Fundamental
##    vars n mean   sd median trimmed  mad  min  max range skew kurtosis  se
## X1    1 2 2.85 1.13   2.85    2.85 1.19 2.05 3.65   1.6    0    -2.75 0.8
## ------------------------------------------------------------
## group: F:Ensino Médio
##    vars n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 7 2.35 1.14   1.85    2.35 0.67   1 4.3   3.3 0.56    -1.33 0.43
## ------------------------------------------------------------
## group: F:Superior
##    vars n mean   sd median trimmed mad min  max range skew kurtosis   se
## X1    1 6 3.22 0.76   3.15    3.22   1 2.2 4.15  1.95 0.02    -1.85 0.31
## ------------------------------------------------------------
## group: M:Ens Fundamental
##    vars n mean   sd median trimmed  mad  min max range skew kurtosis   se
## X1    1 4 2.27 0.84   2.12    2.27 0.67 1.45 3.4  1.95 0.35    -1.93 0.42
## ------------------------------------------------------------
## group: M:Ensino Médio
##    vars n mean   sd median trimmed  mad min  max range  skew kurtosis   se
## X1    1 3 2.67 1.03   2.75    2.67 1.33 1.6 3.65  2.05 -0.08    -2.33 0.59
## ------------------------------------------------------------
```

[2] Fonte: R documentation

```
## group: M:Superior
##    vars n mean   sd median trimmed  mad min max range skew kurtosis   se
## X1    1 8 3.58 1.21   3.15    3.58 1.04 2.1 5.8   3.7 0.58    -1.15 0.43
```

Acima são descritas as medidas de dispersão para cada faixa, sendo elas: gênero feminino, ensino fundamental, médio e superior e gênero masculino, ensino fundamental, médio e superior.

Criando uma tabela usando a função group_by() (dplyr)

Primeiramente carregamos o pacote **dplyr**:

```
library(dplyr)
```

Para criar a tabela devemos fornecer todas as informações necessárias. Primeiramente criamos o objero tabela "**tabela <-"**, essa tabela vai receber a base "**Dados"**. Para agrupar os dados usamos um operador pipe **(%>%)**. Chamamos a função **group_by()** e inserimos as variáveis "**Genero"** e "**Grau_de_instrucao"**. Para indicar e nomear as variáveis que irão compor a tabela, usamos a função **summarise()**.

Para visualizar a tabela, basta escrever "**tabela"** e executar.

Exemplo:

```
tabela <- Dados %>% group_by(Genero, Grau_de_Instruçao) %>%
 summarise(média = mean(Salario),
           DP = sd(Salario),
           mediana = median(Salario))

tabela
```

```
## # A tibble: 6 x 5
## # Groups:   Genero [2]
##   Genero Grau_de_Instruçao média    DP mediana
##   <fct>  <fct>          <dbl> <dbl>   <dbl>
## 1 F      Ens Fundamental    2.85 1.13    2.85
## 2 F      Ensino Médio       2.35 1.14    1.85
## 3 F      Superior           3.22 0.761   3.15
## 4 M      Ens Fundamental    2.28 0.835   2.12
## 5 M      Ensino Médio       2.67 1.03    2.75
## 6 M      Superior           3.58 1.21    3.15
```

4. Gráficos no R-Studio

Os gráficos são mecanismos empregados na representação de fenômenos que possam ser mensurados, quantificados ou ilustrados de modo logico. Tal como os mapas que apresentam uma representação espacial de determinada ocorrência ou local, os gráficos indicam uma dimensão estatística sobre determinado acontecimento.

Para criar gráficos vamos utilizar os pacotes:

ggplt2: É um sistema para criar gráficos declarativamente. Você fornece os dados, diz ao ggplot2 como mapear as variáveis, insere parâmetros para a estatística e ele cuida dos detalhes.[3]

tidyverse: O tidyverse é um conjunto de pacotes que funcionam em harmonia porque compartilham representações de dados comuns e design de API. O pacote tidyverse é projetado para facilitar a instalação e o carregamento de pacotes principais do tidyverse em um único comando.[4]

Carregando os Pacotes:

```
library(ggplot2)

library(tidyverse)
```

A base de dados que vamos tilizer é a mesma da seção estatítistica descritiva:

```
kable(head(Dados, 5))
```

Sujeito	Genero	Grau_de_Instruçao	N_Filhos	Idade	Altura	Salario
1	M	Superior	1	31	160	4.10
2	F	Superior	0	25	160	2.65
3	M	Superior	0	33	157	4.70
4	M	Ens Fundamental	0	20	163	1.45
5	F	Ensino Médio	0	23	163	1.85

[3] Fonte: R documentation

[4] Fonte: R documentation

4.1. Gráficos de Colunas

Do mesmo modo que os gráficos de barra, os gráficos de colunas são bastante utilizados. Estes mostram um dado quantitativo acerca de diferentes variáveis, lugares ou setores e não dependem de proporções. Os dados são apresentados na posição vertical, ao mesmo tempo que as divisões qualitativas aparecem na posição horizontal.

Vamos construir um gráfico com dados referentes ao Grau de instrução, para isso precisamos fornecer as informações necessárias. Começamos chamando a base de dados "**Dados"** e utilizando um operador pipe (**%>%)** para expressar uma sequência de múltiplas ações. Na próxima etapa agrupamos os dados usando a função **group_by()**, indicando nela a variável a ser demonstrada no gráfico, nesse caso a variável "**Grau_de_Instrucao"**, utilizamos novamente um operador pipe (**%>%)** indicando uma sequência.

Para resumir os dados chamamos a função **summarise()**, nomeando como "**contagem"** e chamando a função **n**, indicando que queremos uma contagem de fato.

Exemplo:

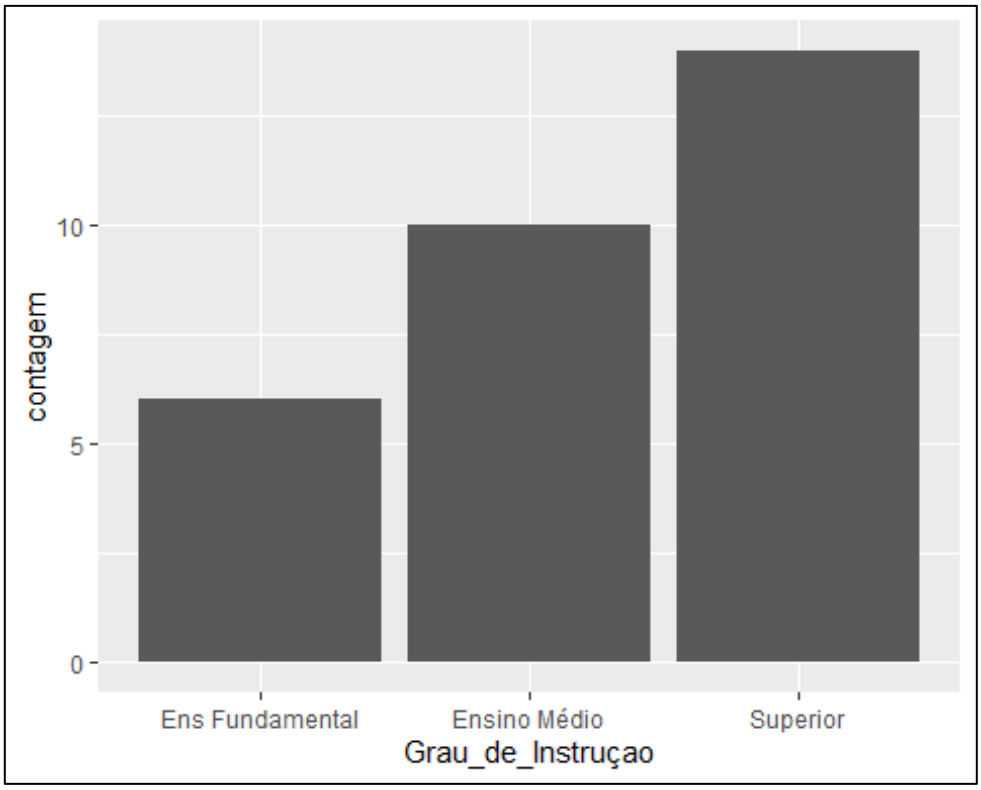
```
Dados %>%
 group_by(Grau_de_Instruçao) %>%
 summarise(
  contagem = n()
 )


##   Grau_de_Instruçao contagem
##   <fct>             <int>
## 1 Ens Fundamental       6
## 2 Ensino Médio         10
## 3 Superior             14
```

Veja que os dados foram classificados de acordo com o Grau de Instrução da amostra.

Para plotar o gráfico, chamamos na sequência de mais um operador pipe (**%>%)** a função **ggplot()**, seguida da função **aes()** apontando o eixo **x** e **y**. Na sequência apontamos o tipo de gráfico a ser plotado, no nosso caso um gráfico de barras, através da função **geom_bar()**. informando através do parâmetro "**stat"** que vamos usar os dados brutos (toda a tabela gerada acima), apontamos o parâmetro "**identity"** como padrão para esse tipo de gráfico.

Exemplo:

```
Dados %>%
 group_by(Grau_de_Instruçao) %>%
 summarise(
  contagem = n()
 ) %>%
 ggplot(aes(x = Grau_de_Instruçao, y = contagem)) +
 geom_bar(stat = "Identity")
```

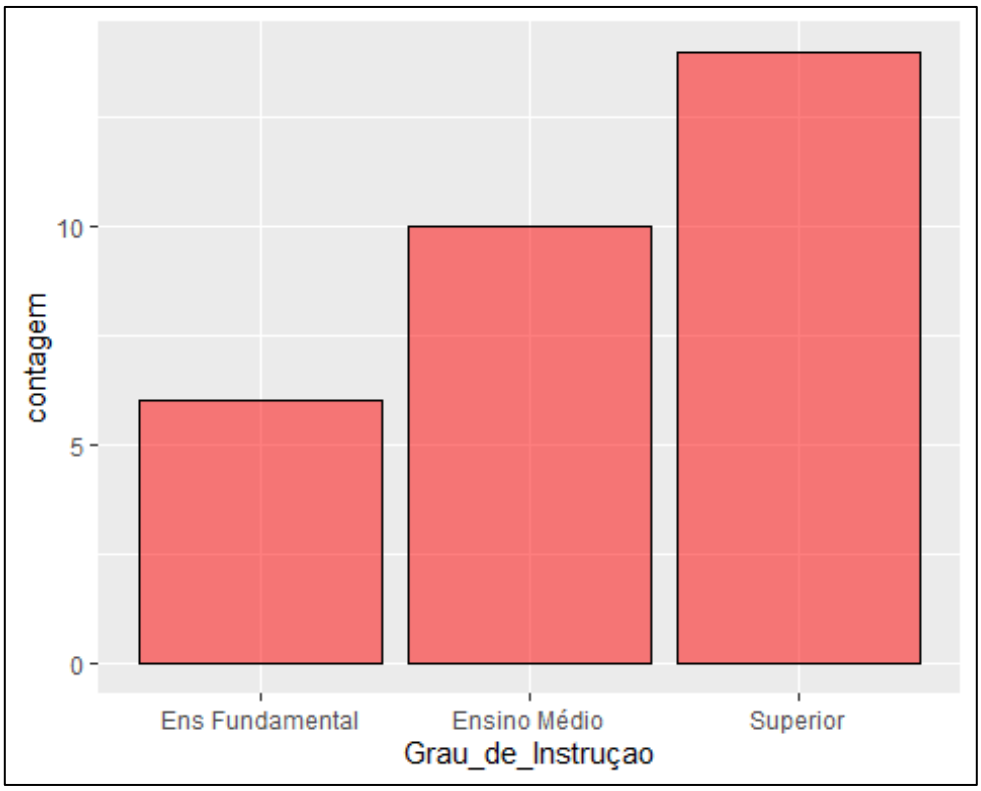


Personalizando o gráfico

Para personalizar nosso gráfico vamos adicionar alguns elementos dentro da função **geom_bar()**. São eles: "**fill**", que adiciona cor, "**color**", que adiciona uma cor as bordas e "**alpha**", que adiciona transparência (quanto mais perto 0 mais transparência).

Exemplo:

```
Dados %>%
 group_by(Grau_de_Instruçao) %>%
 summarise(
  contagem = n()
 ) %>%
 ggplot(aes(x = Grau_de_Instruçao, y = contagem)) +
 geom_bar(stat = "Identity", fill = "red", color = "black", alpha = 0.5)
```

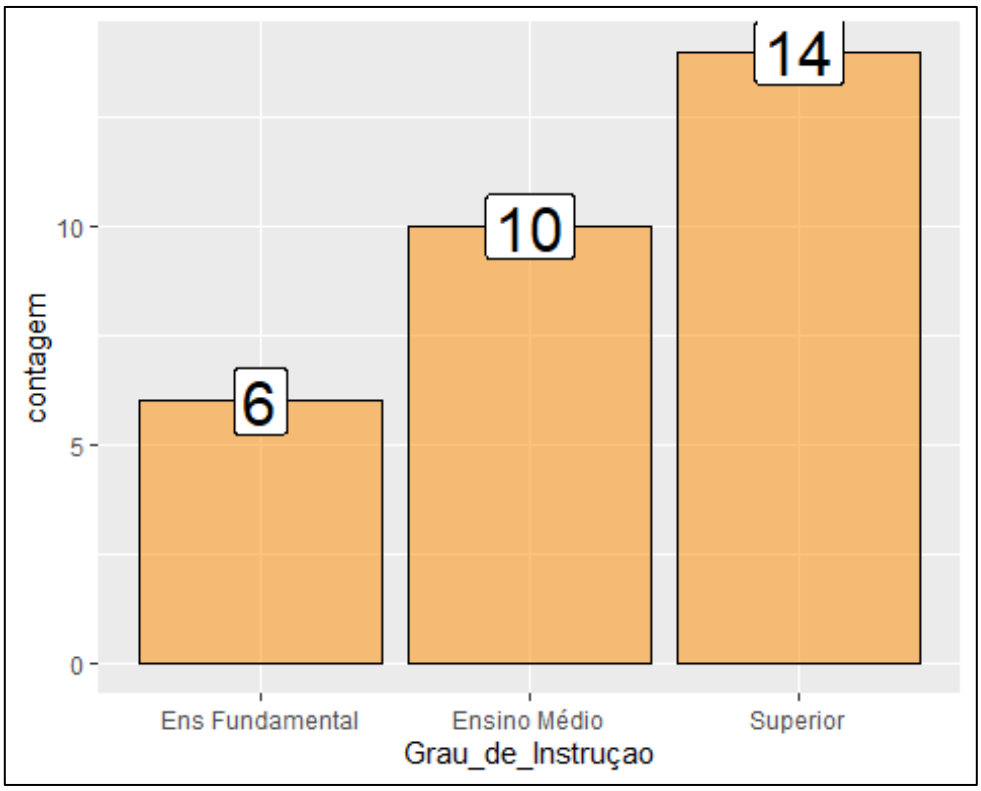


Adicionando Rótulos

Para adicionar rótulos ao gráfico inserimos o elemento "**label"** dentro da função **ggplot()** indicando "**contagem"**. Além disso, inserimos ao final a função **geom_label()**, dentro dessa função podemos indicar o tamanho da fonte usando o elemento "**size"** seguido de um numero, quanto maior o número maior a fonte.

Exemplo:

```
Dados %>%
 group_by(Grau_de_Instruçao) %>%
 summarise(
  contagem = n()
 ) %>%
 ggplot(aes(x = Grau_de_Instruçao, y = contagem, label = contagem)) +
 geom_bar(stat = "Identity", fill = "darkorange", color = "black", alpha = 0.5) +
 geom_label(size = 8)
```



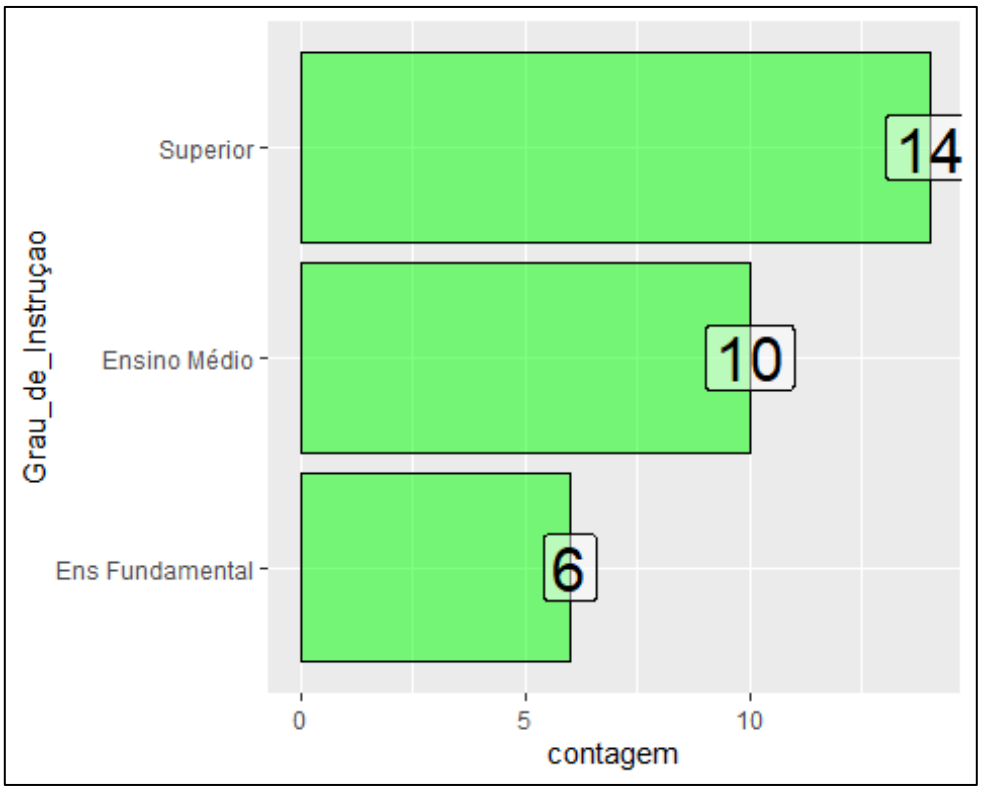
4.2. Gráficos de Barra

Apresentam a mesma funcionalidade dos gráficos de colunas, com os dados de posição horizontal e as informações e divisões na posição vertical.

Para plotar um gráfico de barras usamos exatamente a mesma estrutura de um gráfico de colunas, a única diferença é que ao final adicionamos um elemento a mais chamado de **coor_flip()**.

Exemplo:

```
Dados %>%
 group_by(Grau_de_Instruçao) %>%
 summarise(
  contagem = n()
 ) %>%
 ggplot(aes(x = Grau_de_Instruçao, y = contagem, label = contagem)) +
 geom_bar(stat = "Identity", fill = "green", color = "black", alpha = 0.5) +
 geom_label(size = 8, alpha = 0.5)+
 coord_flip()
```



4.4. Grupos Múltiplos

Agora vamos plotar um gráfico agregando Grau de instrução e Gênero. Para isso definimos a contagem usando a mesma estrutura que já criamos, adicionando as variáveis "**Grau_de_Instrucao"** e "**Genero"** dentro da função **group_by()**.

Exemplo:

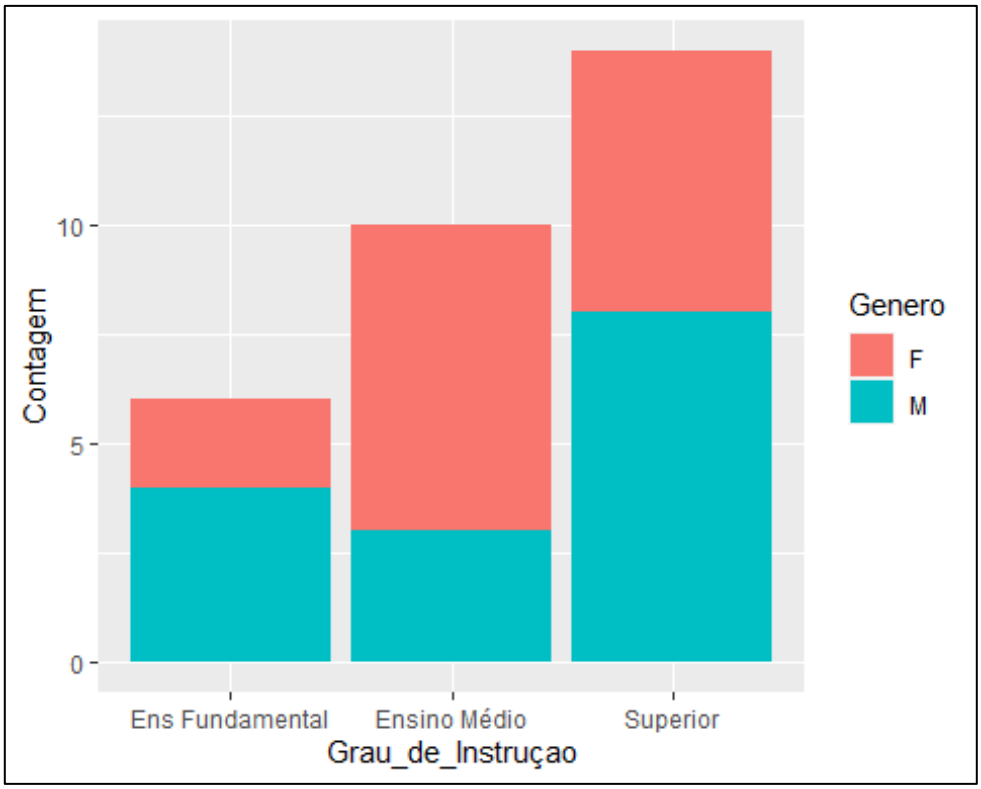
```
Dados %>%
 group_by(Grau_de_Instruçao, Genero) %>%
 summarise(
  Contagem = n()
 )
```

```
## # Groups:   Grau_de_Instruçao [3]
##   Grau_de_Instruçao Genero Contagem
##   <fct>             <fct>    <int>
## 1 Ens Fundamental   F            2
## 2 Ens Fundamental   M            4
## 3 Ensino Médio      F            7
## 4 Ensino Médio      M            3
## 5 Superior          F            6
## 6 Superior          M            8
```

Para agregar as duas variáveis ao gráfico usamos a mesma estrutura já criada, indicando a variável "**Genero"** dentro do elemento "**fill"**.

Exemplo:

```
Dados %>%
 group_by(Grau_de_Instruçao, Genero) %>%
 summarise(
  Contagem = n()
 ) %>%
 ggplot(aes(x = Grau_de_Instruçao, y = Contagem,  fill = Genero)) +
 geom_bar(stat = "Identity")
```

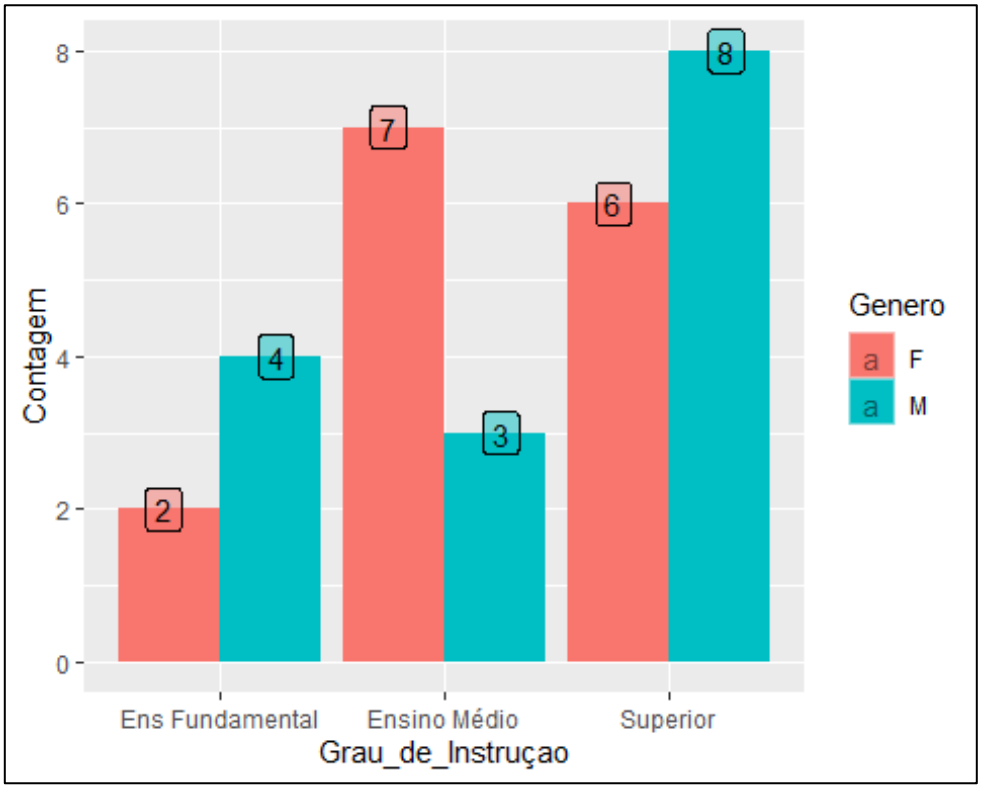


Posicionando as Barras lado a lado

Para posicionar as barras lado a lado usamos o parâmetro **position() - dodge** dentro da função **geom_bar()**. Para inserir os rótulos inserimos o parâmetro "**label = Contagem"** dentro da função **ggplot()** e o parâmetro **pisition = "dodge'** dentro da função **geom_bar()**. Além disso, inserimos dentro da função "**label"** o parâmetro "**position = position_dodge(width = 1)"** como padrão para esse tipo de gráfico.

Exemplo:

```
Dados %>%
 group_by(Grau_de_Instruçao, Genero) %>%
 summarise(
  Contagem = n()
 ) %>%
 ggplot(aes(x = Grau_de_Instruçao, y = Contagem,  fill = Genero, label = Contagem)) +
 geom_bar(stat = "Identity", position = "dodge")+
geom_label(position = position_dodge(width = 1), alpha = 0.5)
```



4.5. Gráfico com uma Barra

Do mesmo modo que no gráfico de pizza, o gráfico de barra é adequado para a expressão de proporcionalidades, no qual o somatório de todos os dados compõe um dado aspecto da realidade.

Para selecionar as informações que vão compor nosso gráfico, usamos a mesma estrutura criada na seção 4.1, referente ao gráfico de barras.

Exemplo:

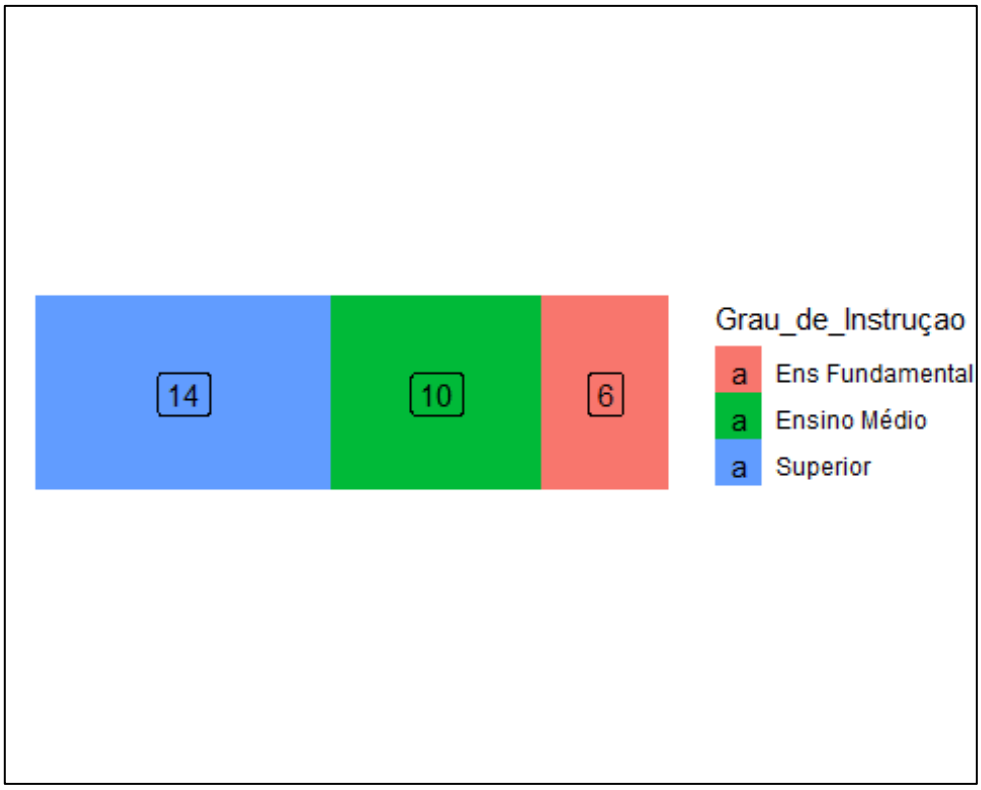
```
Instrucao <- Dados %>%
 group_by(Grau_de_Instruçao) %>%
 summarise(n = n())
```

A estrutura é similar à usada na seção 4.1, usamos aqui os parâmetros "**width = 0.3"** e **theme_void()** para melhorar a visualização.

Por padrão usamos dentro da função **geom_label()** a estrutura "**position = position_stack(vjust = 0.5)"**.

Exemplo:

```
Instrucao %>%
 ggplot(aes(x = "", y = n, fill = Grau_de_Instruçao, label = n)) +
 geom_bar(stat = "Identity", width = 0.3) +
 coord_flip() +
 geom_label(position = position_stack(vjust = 0.5)) +
 theme_void()
```



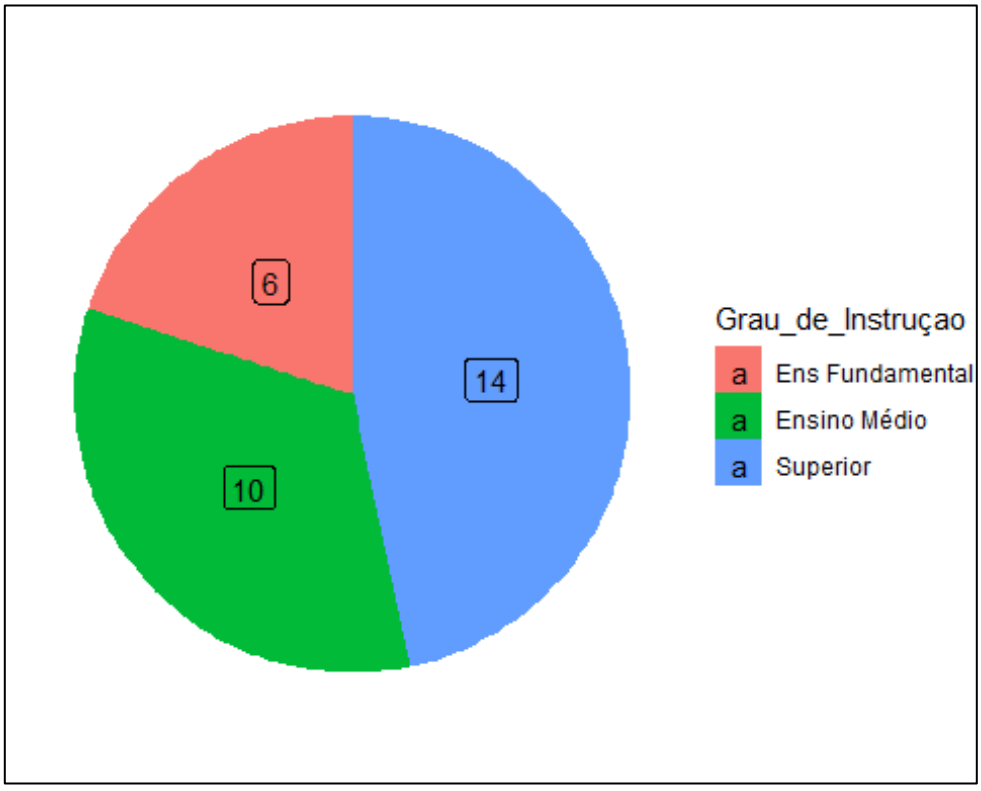
4.6. Gráfico de Pizza

É um gráfico muito utilizado, indicado para expressar uma relação de proporcionalidade, em que todos os dados somados compõem o todo de um dado aspecto da realidade.

Usamos aqui uma estrutura similar à usada no Gráfico de uma barra, a diferença é que inserimos aqui a função **coor_polar()** com os parâmetros "**theta = "y", start = 0**."

Exemplo:

```
Instrucao %>%
 ggplot(aes(x = "", y = n, fill = Grau_de_Instruçao, label = n)) +
 geom_bar(stat = "Identity", width = 0.3) +
 coord_polar(theta = "y", start = 0) +
 geom_label(position = position_stack(vjust = 0.5)) +
 theme_void()
```



4.7. Gráfico de Linha

O gráfico de linha é apropriado para a demonstração de uma sequência numérica de um certo dado ao longo de um período. É empregado na demonstração de evoluções (ou regressões) que acontecem em sequência visando a observação do comportamento dos fenômenos e suas transformações.

Para criarmos um Gráfico de linha usaremos uma base de dados com a evolução do PIB do Brasil, EUA e China de 1970 até 2020.

Importando a base de dados em formato xlsx:

```
library(readxl)
PIB <- read_excel("C:/Users/Bokka/Desktop/R/Modulo 2/GraficodeLinha.xlsx")
```

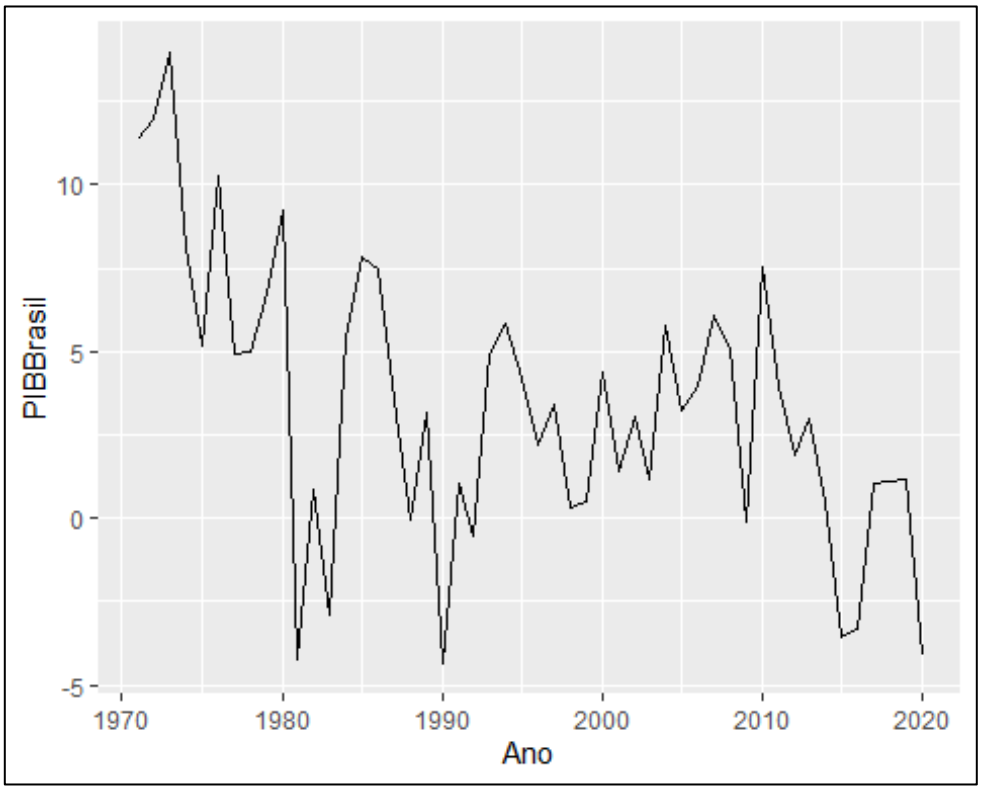
```
kable(PIB)
```

Ano	PIBBrasil	PIBEUA	PIBChina
1971	11.34	3.3	7.1
1972	11.94	5.3	3.8
1973	13.97	5.6	7.8
1974	8.15	-0.5	2.3
1975	5.17	-0.2	8.7
1976	10.26	5.4	-1.6
1977	4.93	4.6	7.6
1978	4.97	5.5	11.7
1979	6.76	3.2	7.6
1980	9.23	-0.3	7.8
1981	-4.25	2.5	5.1
1982	0.83	-1.8	9.0
1983	-2.93	4.6	10.8
1984	5.40	7.2	15.2
1985	7.85	4.2	13.4
1986	7.49	3.5	8.9
1987	3.53	3.5	11.7
1988	-0.06	4.2	11.2
1989	3.16	3.7	4.2
1990	-4.35	1.9	3.9
1991	1.03	-0.1	9.3
1992	-0.54	3.5	14.2
1993	4.92	2.8	13.9
1994	5.85	4.0	13.0
1995	4.22	2.7	11.0
1996	2.21	3.8	9.9
1997	3.39	4.4	9.2
1998	0.34	4.5	7.8
1999	0.47	4.8	7.7
2000	4.39	4.1	8.5
2001	1.39	1.0	8.3
2002	3.05	1.7	9.1
2003	1.14	2.3	10.0
2004	5.76	3.0	10.1
2005	3.20	3.5	11.4
2006	3.96	2.9	12.7
2007	6.07	1.9	14.2
2008	5.09	-0.1	9.7
2009	-0.13	-2.5	9.4
2010	7.53	2.6	10.6
2011	3.97	1.6	9.6
2012	1.92	2.2	7.9
2013	3.00	1.8	7.8
2014	0.50	2.5	7.4
2015	-3.55	3.1	7.0
2016	-3.31	1.7	6.8
2017	1.06	2.3	6.9
2018	1.12	3.0	6.7
2019	1.14	2.2	6.1
2020	-4.10	-3.5	2.3

Chamamos a função **ggplot()** e apontamos a base de dados que vamos utilizar, utilizamos o operador "**+**" para continuar. Em seguida chamamos a função **aes()** e identificamos as variáveis que vão compor o eixo **x** e o eixo **y**. Por fim apontamos qual é o tipo de gráfico que queremos, para um gráfico de linha usamos **geom_line()**.

Exemplo:

```
ggplot(PIB) + aes(x = Ano, y = PIBBrasil) +
 geom_line()
```



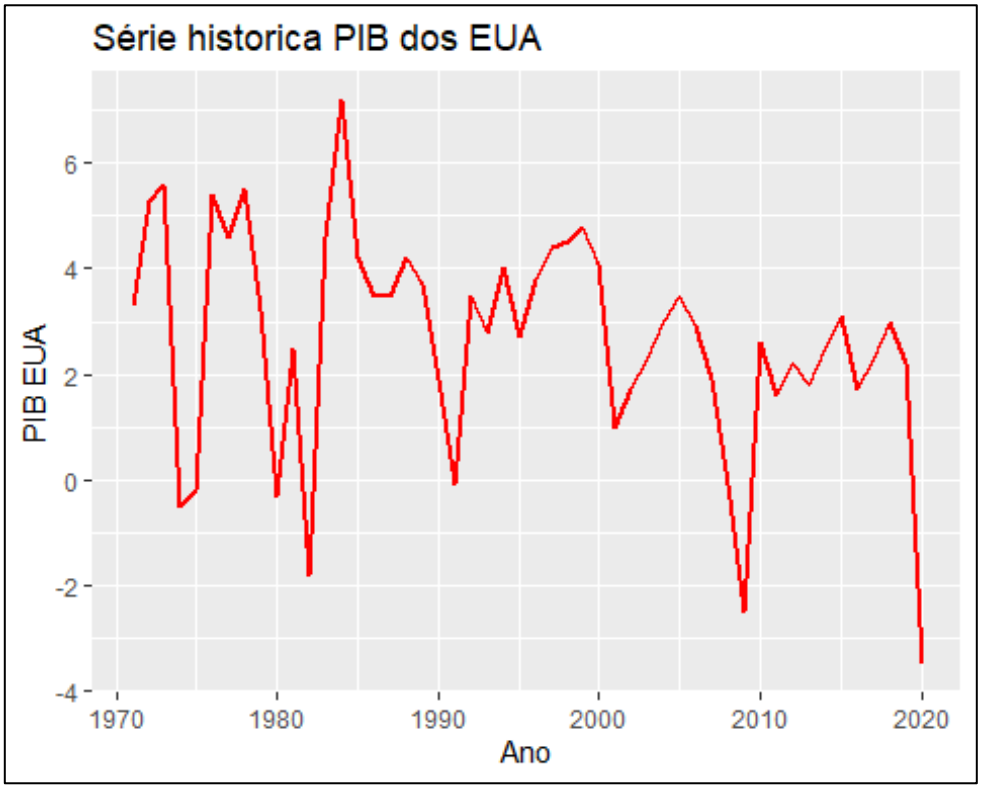
Personalizando

Dentro da função **geom_line()** podemos personalizar o gráfico. Usando parâmetros como "**col**", para colocar outra cor[5]. **"lwd"** para aumentar ou diminuir a espessura da linha; "**labs**" para adicionar as legendas dos eixos **x** e **y**, "**title**" para inserir um título.

[5] O site color in R lista e demonstra todas as cores disponíveis no R. (http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf)

Exemplo:

```
ggplot(PIB) + aes(x = Ano, y = PIBEUA) +
 geom_line(col = "red", lwd = 1)+
 labs(x = "Ano", y = "PIB EUA", title = "Série historica PIB dos EUA")
```



4.8 Criando Gráficos com 2 eixos y

Um gráfico de eixo duplo ou gráfico de múltiplos eixos utiliza-se de dois eixos para ilustrar as relações entre duas variáveis com diferentes magnitudes e escalas de medição.

Para criar um gráfico com dois eixos **y**, primeiramente indicamos que queremos criar um vetor (**V1 <-).** Chamamos a função **ggplot()** e inserindo a base "**PIB**". Com um operador "**+**" para indicar continuidade, definimos o eixo **x** usando a função **aes()**. na sequência indicamos que queremos um gráfico de linha chamando a função **geom_line()**, definindo primeiramente apenas um eixo **y**.

Para adicionar o segundo eixo **y** criamos outra camada no **ggplot()**, chamamos o vetor recém criado (**V2)**, seguido do segundo eixo indicado pelas funções **geom_line()** e **aes()**.

Exemplo:

```
V1 <- ggplot(PIB) +
 aes(x = Ano) +
 geom_line(aes(y = PIBBrasil, col = "PIB Brasil"))

V2 <- V1 + geom_line(aes(y = PIBEUA, col = "PIB EUA"))
V2
```

Editando o Gráfico

Para mudar a cor das linhas chamamos o vetor (**V2)**, usando um operador **"+".** Chamamos a função **scale_color_manual()**, seguido do parâmetro "**values**". Por fim, para definir quais cores queremos chamamos a função **c()** e entre aspas escrevemos (em inglês) as cores desejadas.

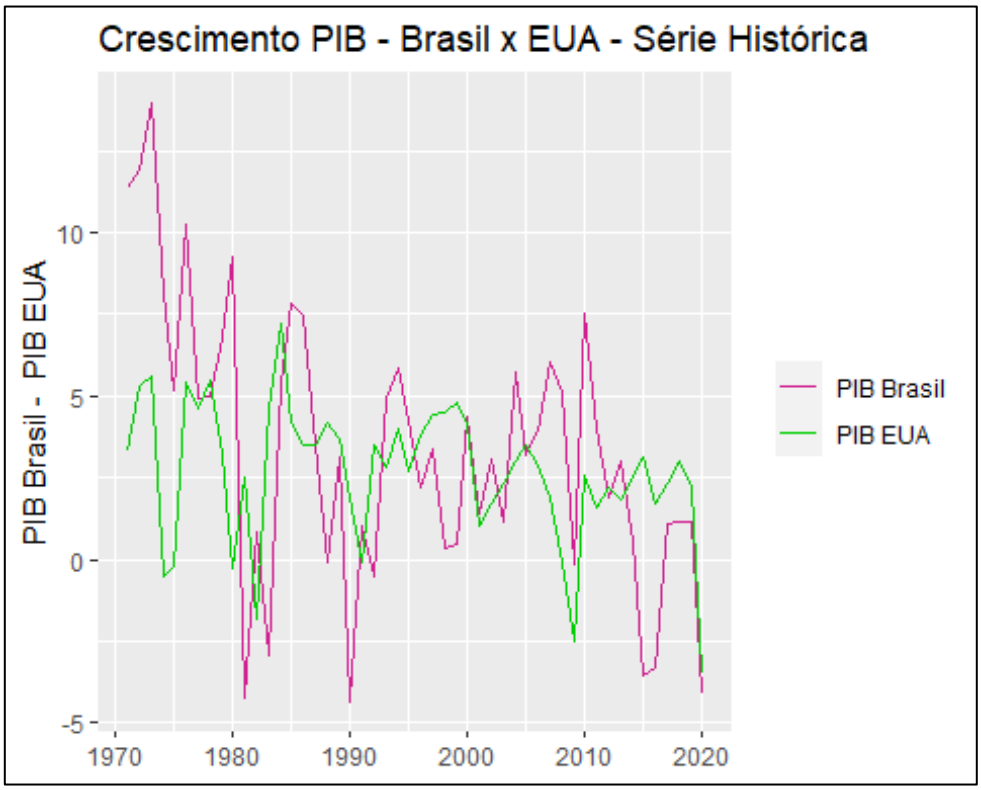
Para inserir um título usamos a função "**labs**", seguida do parâmetro "**title**", onde escreveremos entre aspas o título. Vamos retirar o título do eixo **x** atribuindo a ele a expressão "**NULL**" e no eixo **y** escrevemos o nome das variáveis.

Baixando o pacote **ggthemes()**, temos acesso a diversos temas que podem deixar nosso gráfico mais elegante. Após instalado e carregado, chamamos a função **theme_set()**, em seguida usando o termo "**theme_**" aparecerá diversas opções.

```
library(ggthemes)
```

Exemplo:

```
V2 + scale_color_manual(values = c("violetred", "green3")) +
  labs(title = "Crescimento PIB - Brasil x EUA - Série Histórica", colour = NULL,  x = NULL, y = "
PIB Brasil -  PIB EUA")

theme_set(theme_economist())
```



4.9. Gráficos com Múltiplas Variáveis

Para adicionarmos mais variáveis ao gráfico de linhas, seguimos adicionando camadas, para isso criamos o vetor **"V3".**

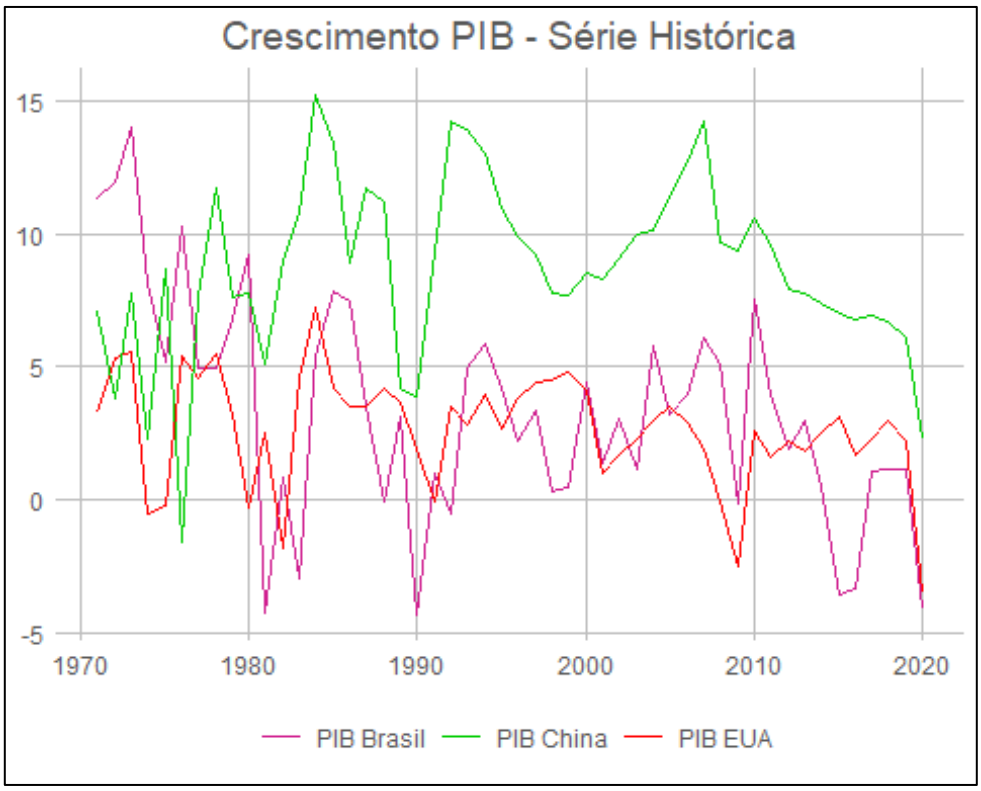
Exemplo:

```
V3 <- V2 + geom_line(aes(y = PIBChina, col = "PIB China"))
```

Na sequência usamos a mesma estrutura usada no Gráfico anterior, especificando as características estéticas.

Exemplo:

```
V3 + scale_color_manual(values = c("violetred", "green3", "red")) +
  labs(title = "Crescimento PIB - Série Histórica", colour = NULL,  x = NULL, y = "PIB") +
theme_excel_new()
```



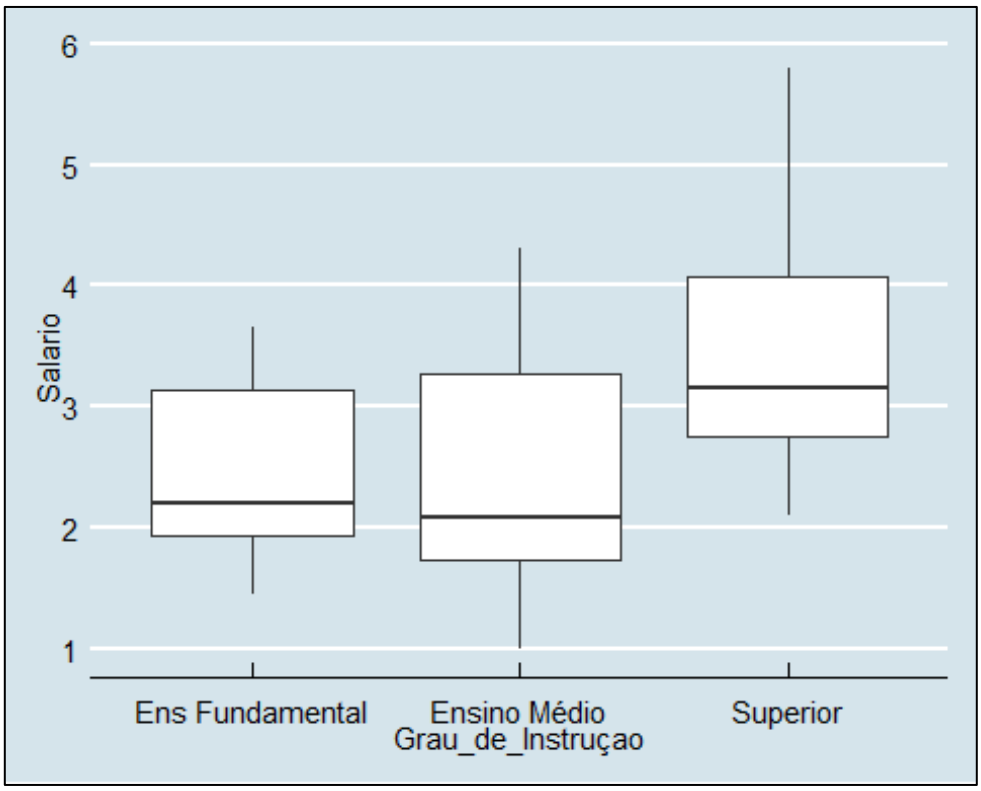
4.10. **Gráfico BoxPlot**

Um boxplot é uma forma padronizada de exibir a distribuição de dados com base em um resumo de cinco números (mínimo, primeiro quartil (Q1), mediana, terceiro quartil (Q3) e máximo). Ele pode informá-lo sobre seus valores discrepantes e quais são seus valores. Ele também pode dizer se seus dados são simétricos, quão firmemente seus dados estão agrupados e se os dados estão distorcidos.

Para construir um BoxPlot indicamos o eixo **x** e **y** através da função **aes()** e indicamos que queremos um gráfico BoxPlot chamando **geom_boxplot()**.

Exemplo:

```
Dados %>%
 ggplot(aes(x = Grau_de_Instruçao, y = Salario)) +
 geom_boxplot()+
 theme_economist()
```

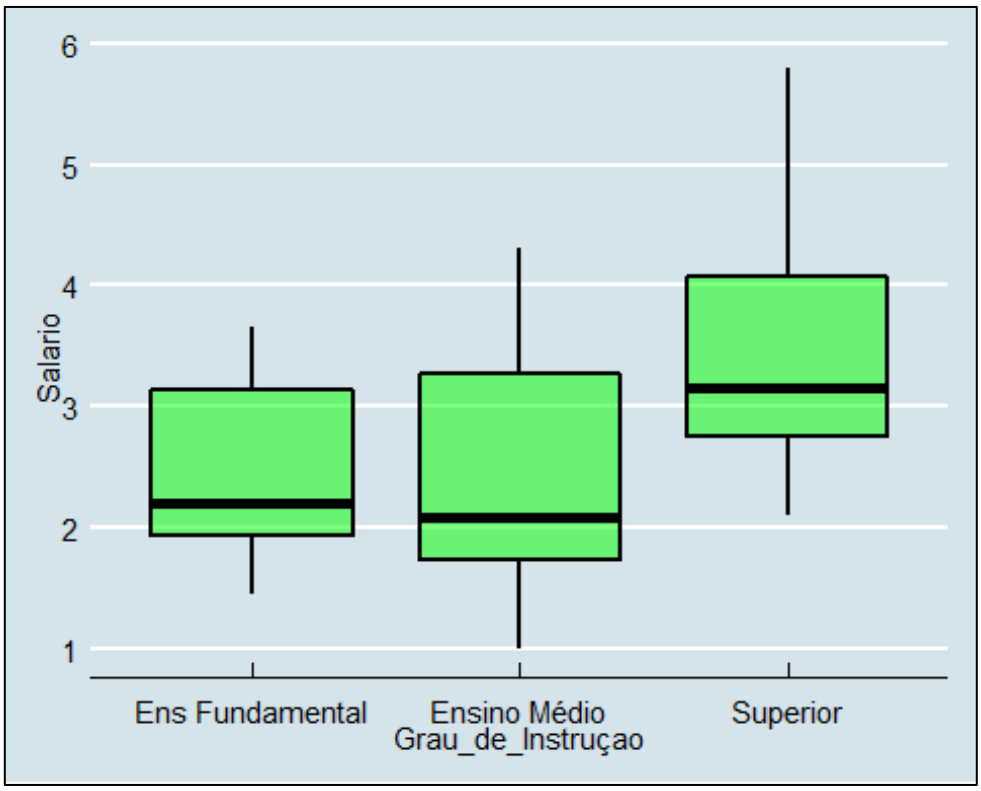


Editando: Inserimos os parâmetros dentro da função **geom_boxplot()**:

"size": Espessura da linha; "**fill**": Cor; **color**: cor das linhas; "**alpha**": Transparência

Exemplo:

```
Dados %>%
 ggplot(aes(x = Grau_de_Instruçao, y = Salario)) +
 geom_boxplot(size = 1, color = "black", fill = "green", alpha = 0.5)+
 theme_economist()
```

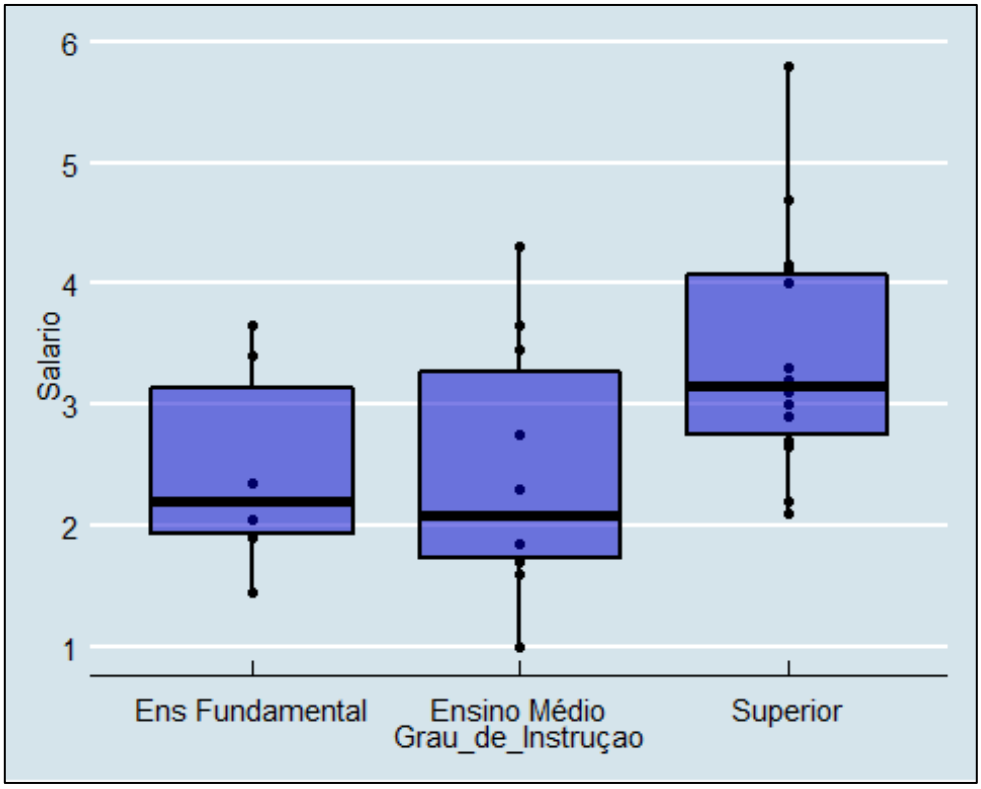


Mostrando os pontos

Para mostrar os pontos inserimos a função **geom_point()**

Exemplo:

```
Dados %>%
 ggplot(aes(x = Grau_de_Instruçao, y = Salario)) +
 geom_point() +
 geom_boxplot(size = 1, color = "black", fill = "blue3", alpha = 0.5) +
 theme_economist()
```



4.11. Histograma

Um histograma é uma representação de dados semelhante a um gráfico de barras que agrupa uma variedade de resultados em colunas ao longo do eixo x. O eixo y representa a contagem do número ou porcentagem de ocorrências nos dados para cada coluna e pode ser usado para visualizar as distribuições de frequência dos dados.

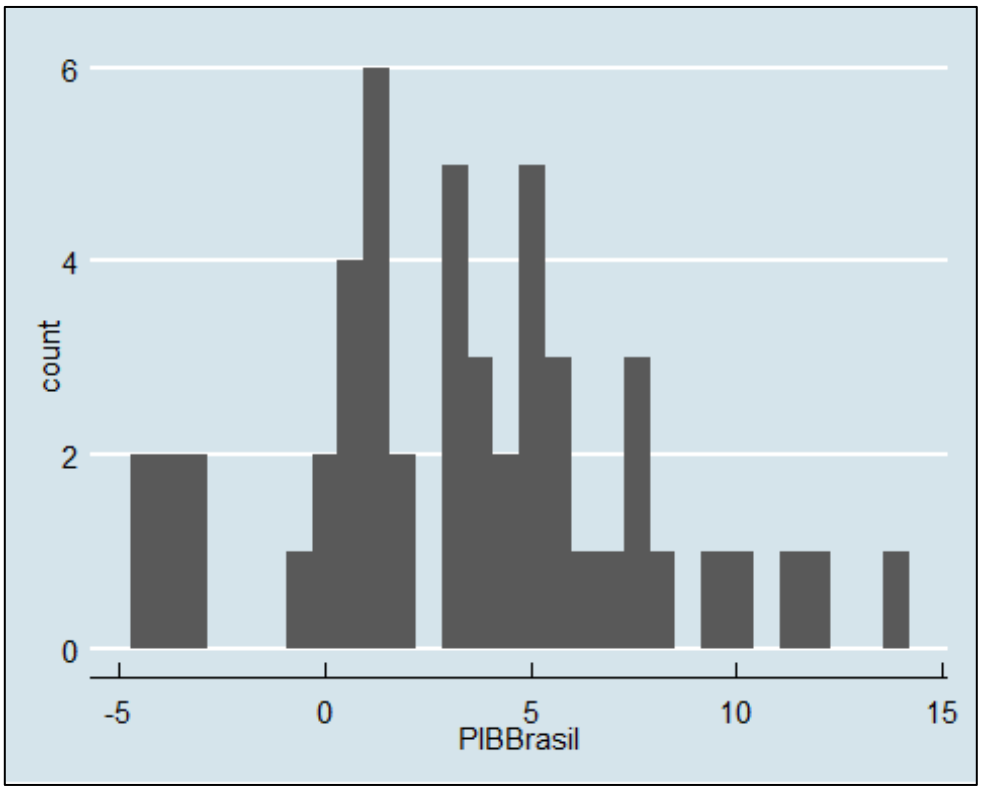
Antes de começarmos a criar um histograma, vamos baixar e carregar mais uma extensão do pacote **ggplot2** chamada **egg()**:

```
library(egg)
```

Mais uma vez, a estrutura que utilizamos para criar um Histograma é similar aos demais gráficos. A diferença é que após apontar a variável de referência através da função **aes()**, devemos indicar que queremos um histograma e chamar a função **geom_histagram()**, indicamos o parâmetro "**bins = 30"** como padrão para esse tipo de gráfico.

Exemplo:

```
PIB %>%
 ggplot(aes(PIBBrasil)) +
 geom_histogram(bins = 30)
```

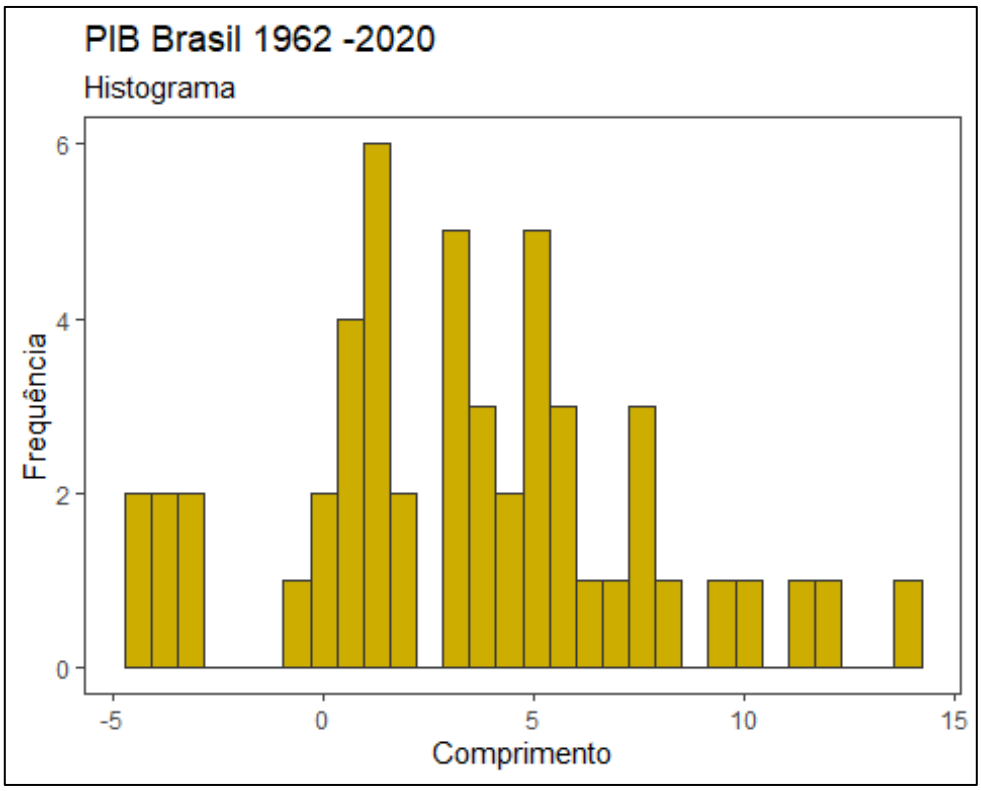


Editando, Vamos colocar alguns detalhes

"color": Borda do Gráfico; "**fill**": Cores; "**Title**": Título; "**Subtitle**": Subtitulo; **Theme_article()**: tema próprio para artigos acadêmicos

Exemplo:

```
PIB %>%
 ggplot(aes(PIBBrasil)) +
 geom_histogram(bins = 30, color = "grey22", fill = "gold3") +
 labs(title = "PIB Brasil 1962 -2020", subtitle = "Histograma", x = "Comprimento", y = "Frequência") +
 theme_article()
```



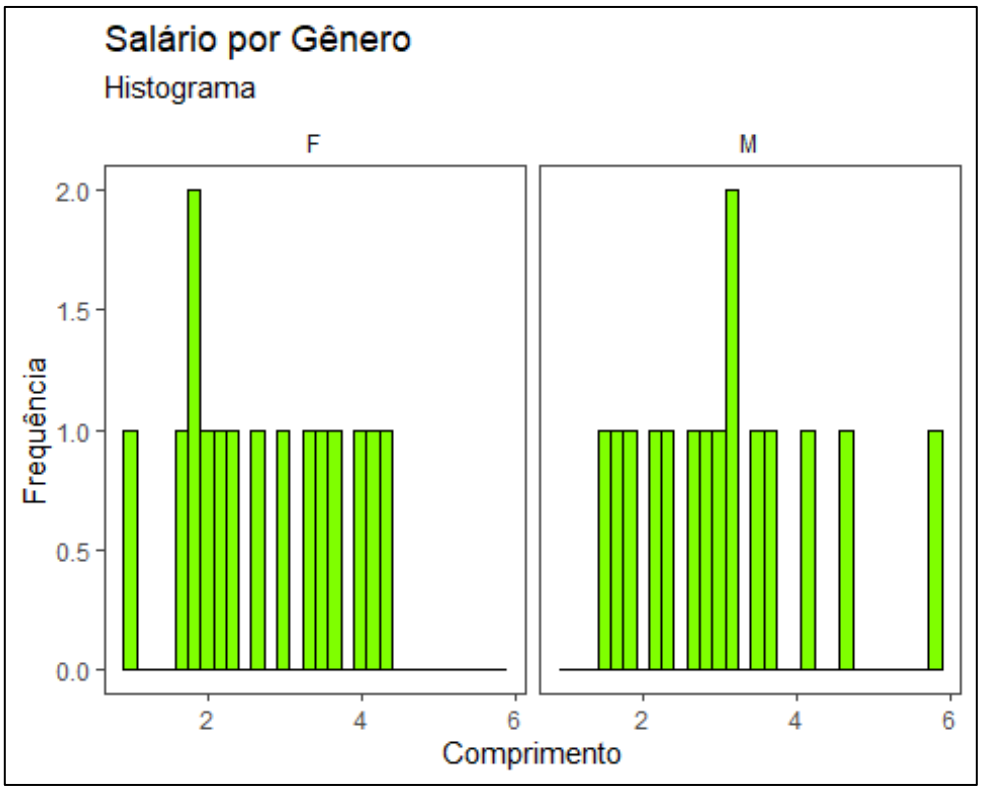
Decompondo os Dados

Agora utilizando o Base "**Dados"** vamos criar dois histogramas com a frequência de salários por gênero.

Para isso, inserimos a função **facet_grid()**, dentro dela usamos "**~"** antes de informar a Variável desejada, que, no nosso caso é "**Genero"**.

Exemplo:

```
Dados %>%
 ggplot(aes(Salario)) +
 geom_histogram(bins = 30, color = "black", fill = "chartreuse") +
 labs(title = "Salário por Gênero", subtitle = "Histograma", x = "Comprimento", y = "Frequênci
a")+
 theme_article() +
 facet_grid(~Genero)
```

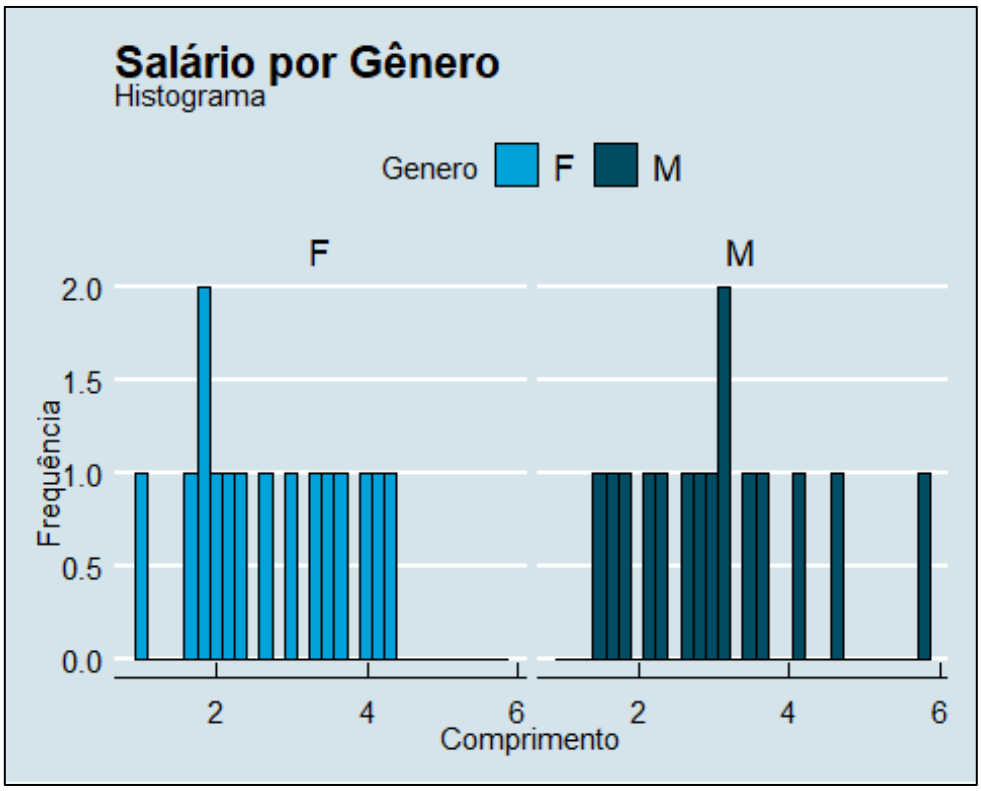


Cores Diferentes

Para que as cores dos Histogramas fiquem diferentes, inserimos o parâmetro "**fill**" dentro da função **aes()**. Aqui utilizamos o tema **theme_economist()** e **scale_fill_economist().**

Exemplo:

```
Dados %>%
  ggplot(aes(Salario, fill = Genero)) +
  geom_histogram(bins = 30, color = "black") +
  labs(title = "Salário por Gênero", subtitle = "Histograma", x = "Comprimento", y = "Frequênci
a")+
  theme_economist() +
  scale_fill_economist() +
  facet_grid(~Genero)
```



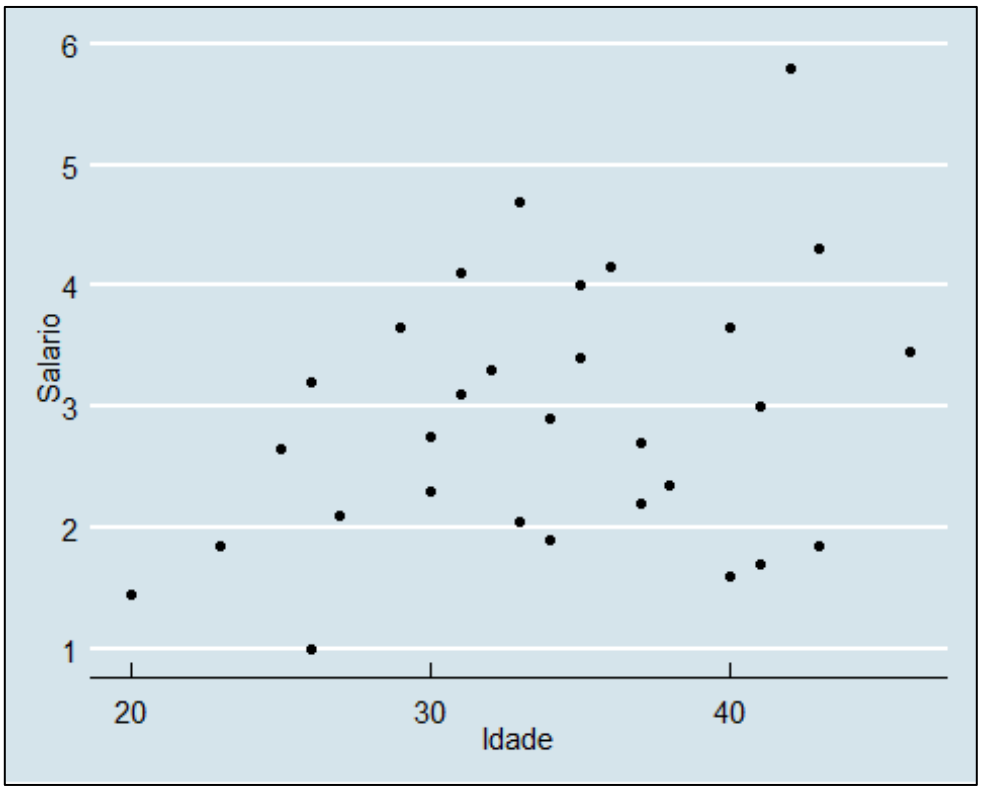
4.12, Gráfico de Dispersão

Um gráfico de dispersão usa pontos para representar valores para duas variáveis numéricas diferentes. A posição de cada ponto nos eixos horizontal e vertical indica valores para um ponto de dados individual. Gráficos de dispersão são usados para observar relações entre variáveis.

Inserimos as variáveis **x** e **y**, na sequencia chamamos a função **geom_point()**.

Exemplo:

```
Dados %>%
 ggplot(aes(x = Idade, y = Salario)) +
 geom_point() +
 theme_economist()
```

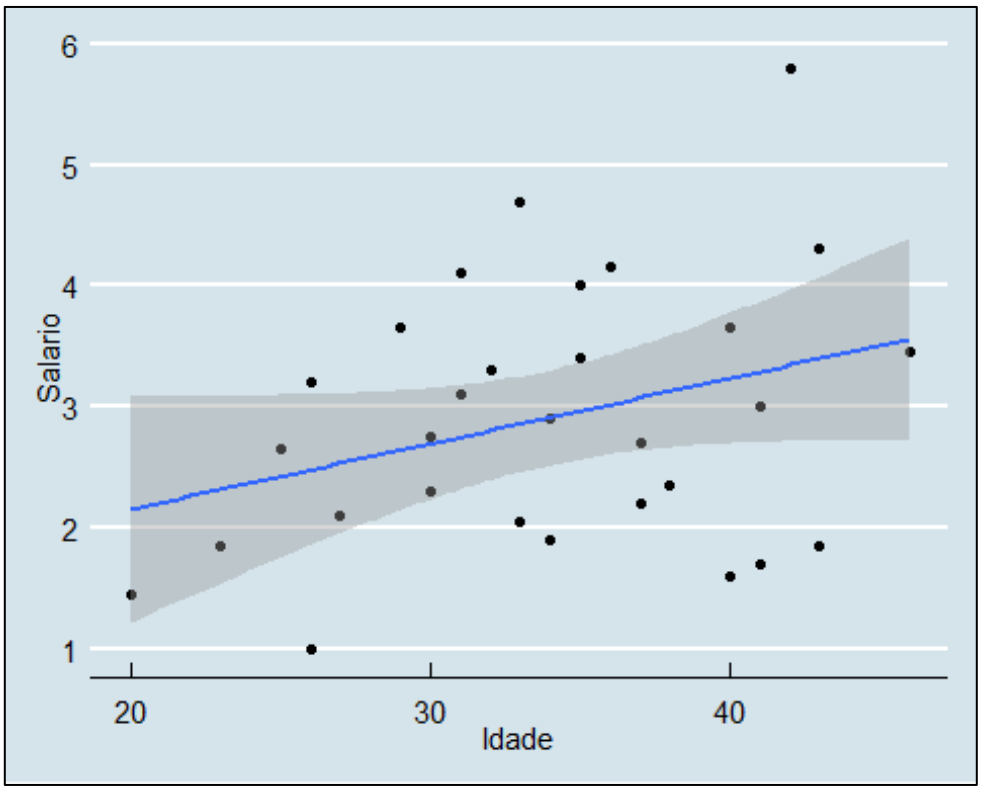


Ajustando a Linha de tendência

Para inserir a reta chamamos a função **geom_smoth()**, indicando através de "**method"** o parâmetro "**lm"** (Linear Model).

Exemplo:

```
Dados %>%
  ggplot(aes(x = Idade, y = Salario)) +
  geom_point() +
  geom_smooth(method = "lm") +
  theme_economist()

## `geom_smooth()` using formula 'y ~ x'
```

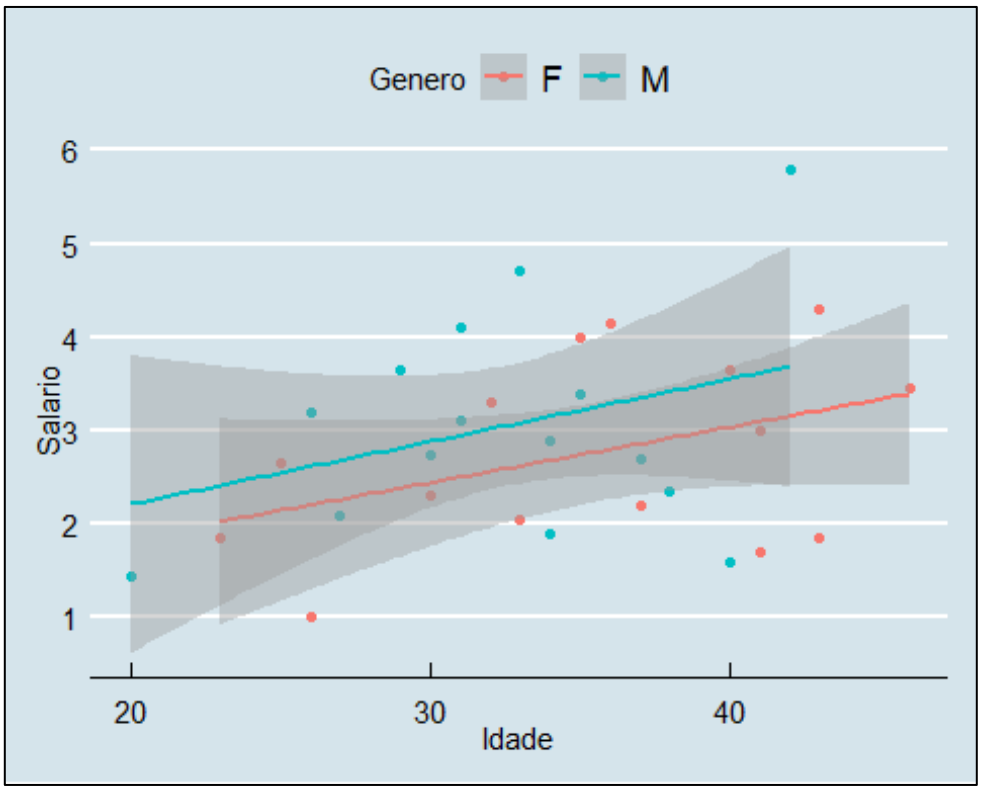


Ajustando o modelo Linear para Tendência

Agora vamos ajustar uma linha para cada Gênero, Feminino e Masculino, para isso usamos o parâmetro "**color"** indicando a variável "**Genero"**, dentro da função **aes()**.

Exemplo:

```
Dados %>%
 ggplot(aes(x = Idade, y = Salario, color = Genero)) +
 geom_point() +
 geom_smooth(method = "lm") +
 theme_economist()

## `geom_smooth()` using formula 'y ~ x'
```



No próximo módulo, vamos nos aprofundar mais sobre a dinâmica que envolve gráficos de dispersão estudando regressão linear simples e Regressão linear Múltipla. Esperamos vocês lá.