Mini curso Introdução prática ao R

Parte 2: Tópicos Diversos

Vinícius M. de Sousa

Universidade do Estado de Santa Catarina

O que é o ggplot2?

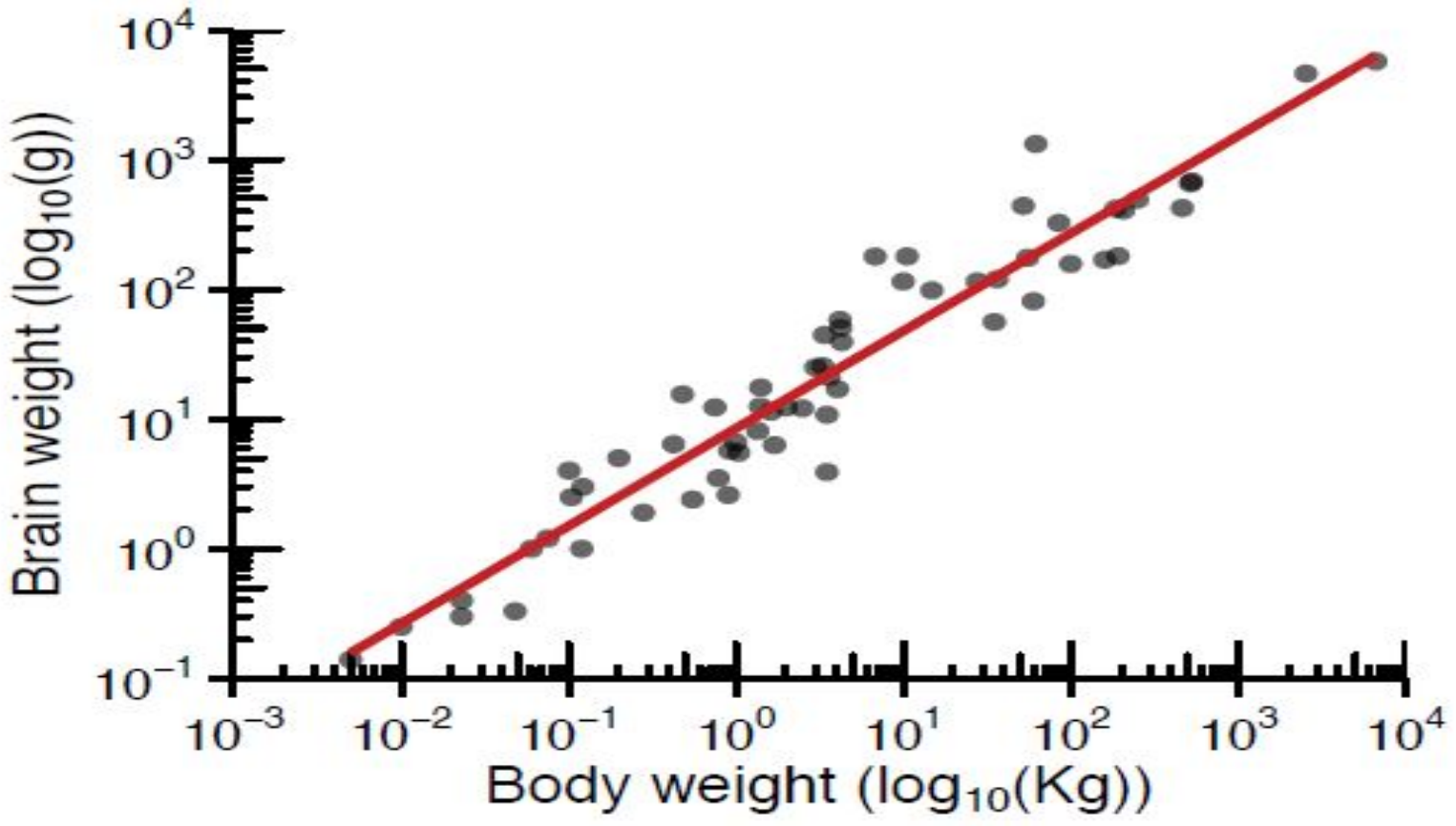
- ggplot2 é um pacote que foi desenvolvido com o objetivo de facilitar a execução de *Data Visualization*.

O que é o ggplot2?

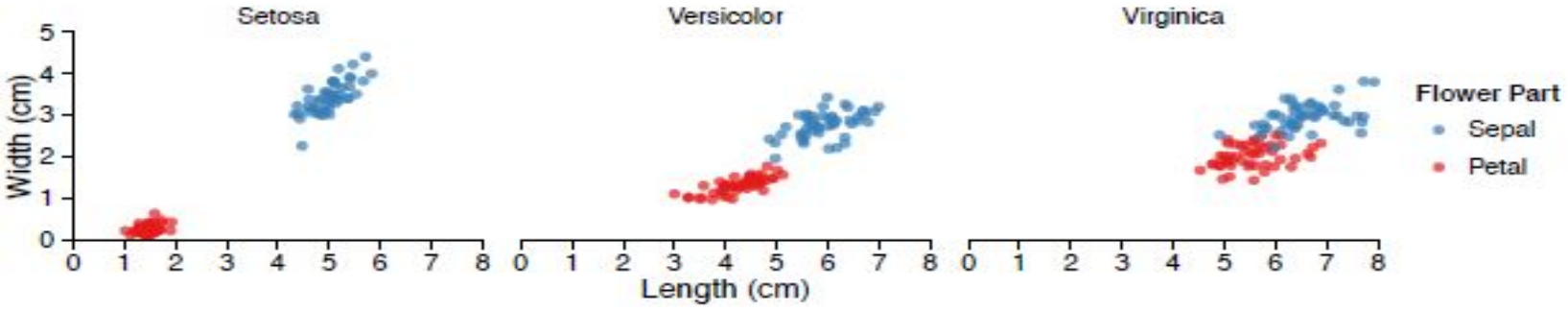
- ggplot2 é um pacote que foi desenvolvido com o objetivo de facilitar a execução de *Data Visualization*.
- Mas o que é exatamente *Data Visualization*?

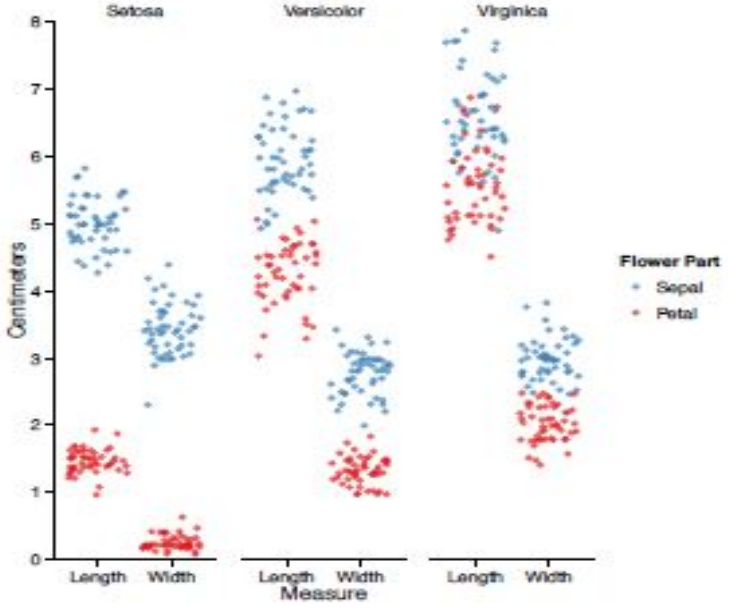

ESTATÍSTICA
DESIGN
SUGAR
E ASSIM SURGIU DATA VISUALIZATION
GERADORMEMES.COM

Gráficos Feitos com ggplot2

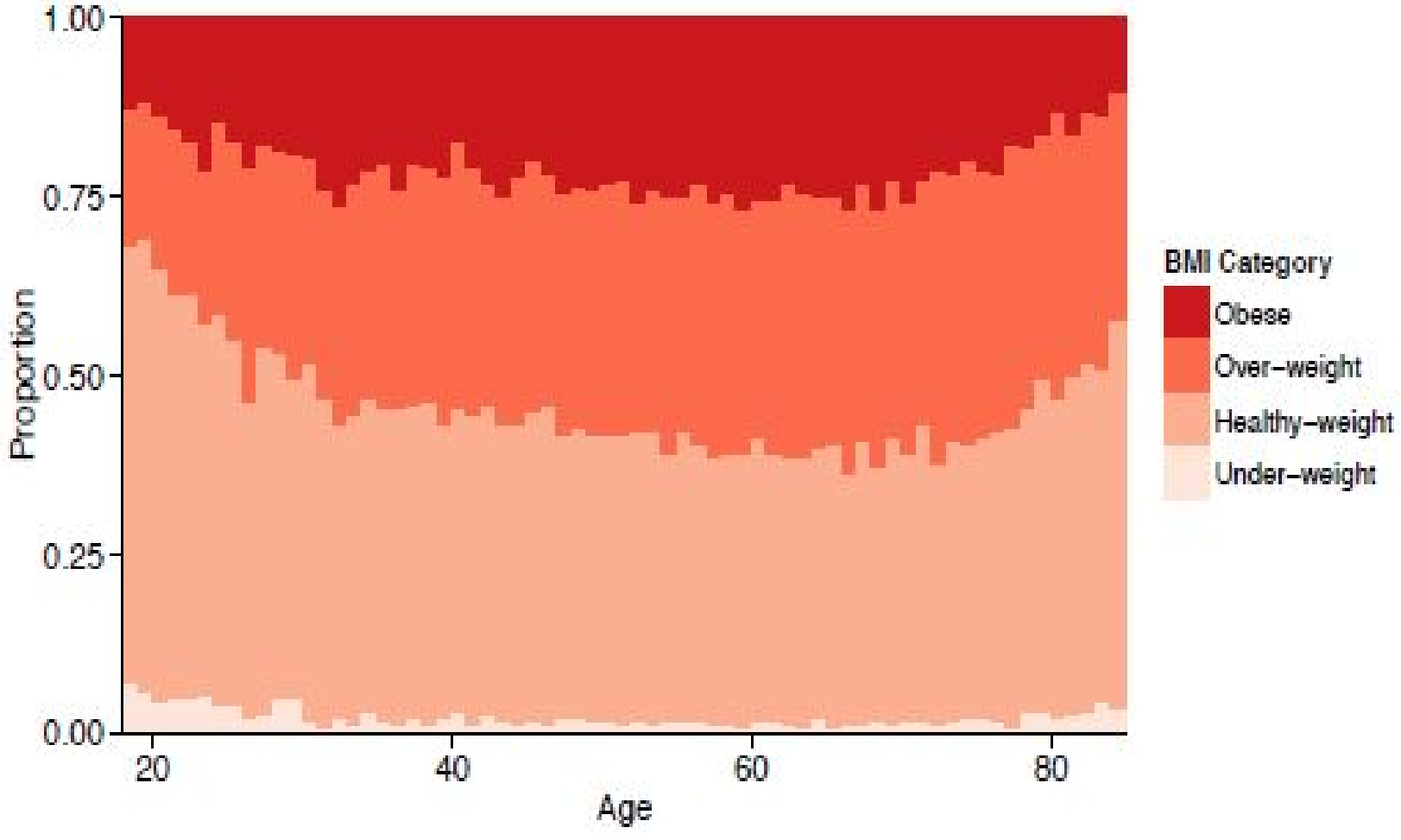


Gráficos Feitos com ggplot2



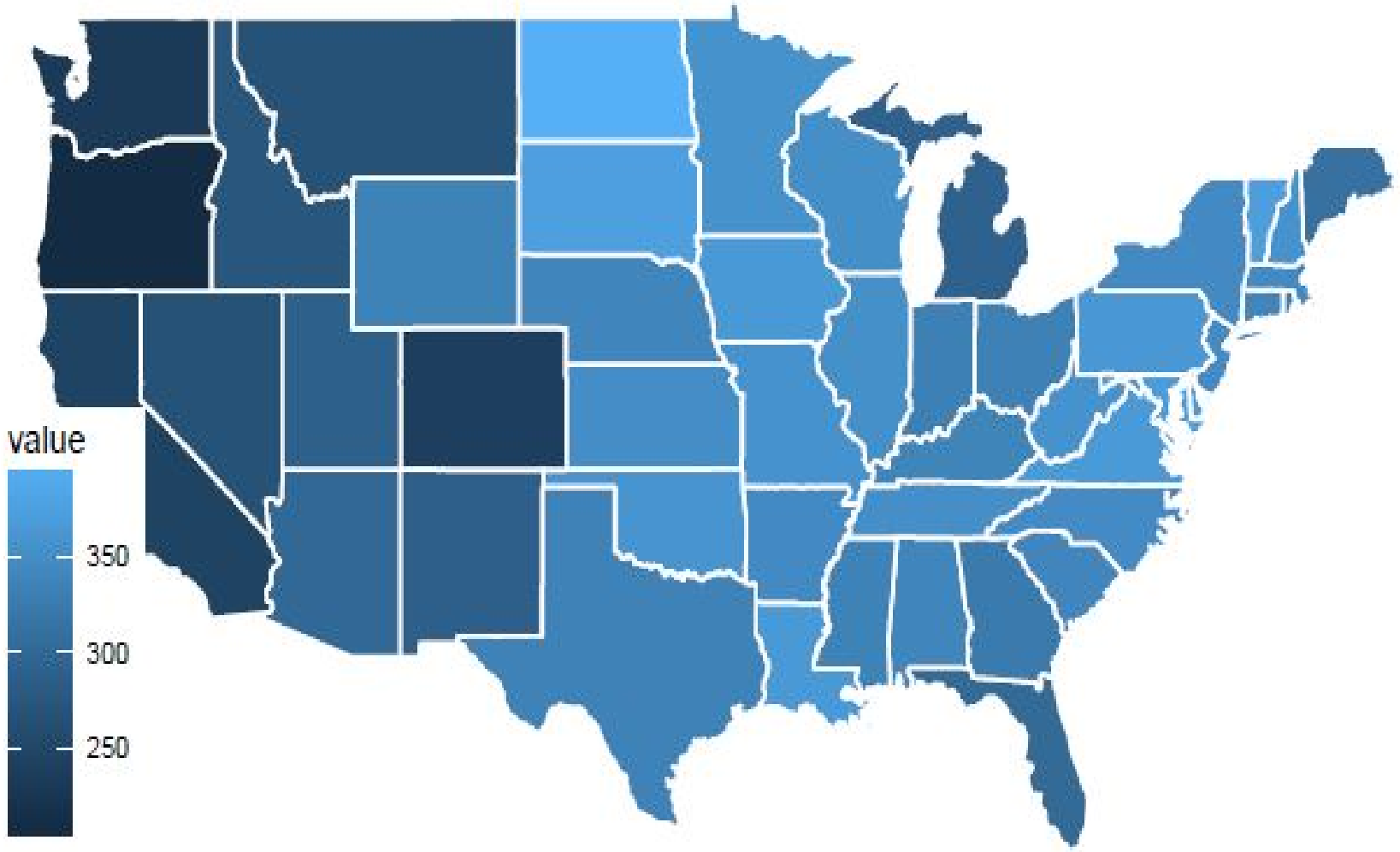


Gráficos Feitos com ggplot2

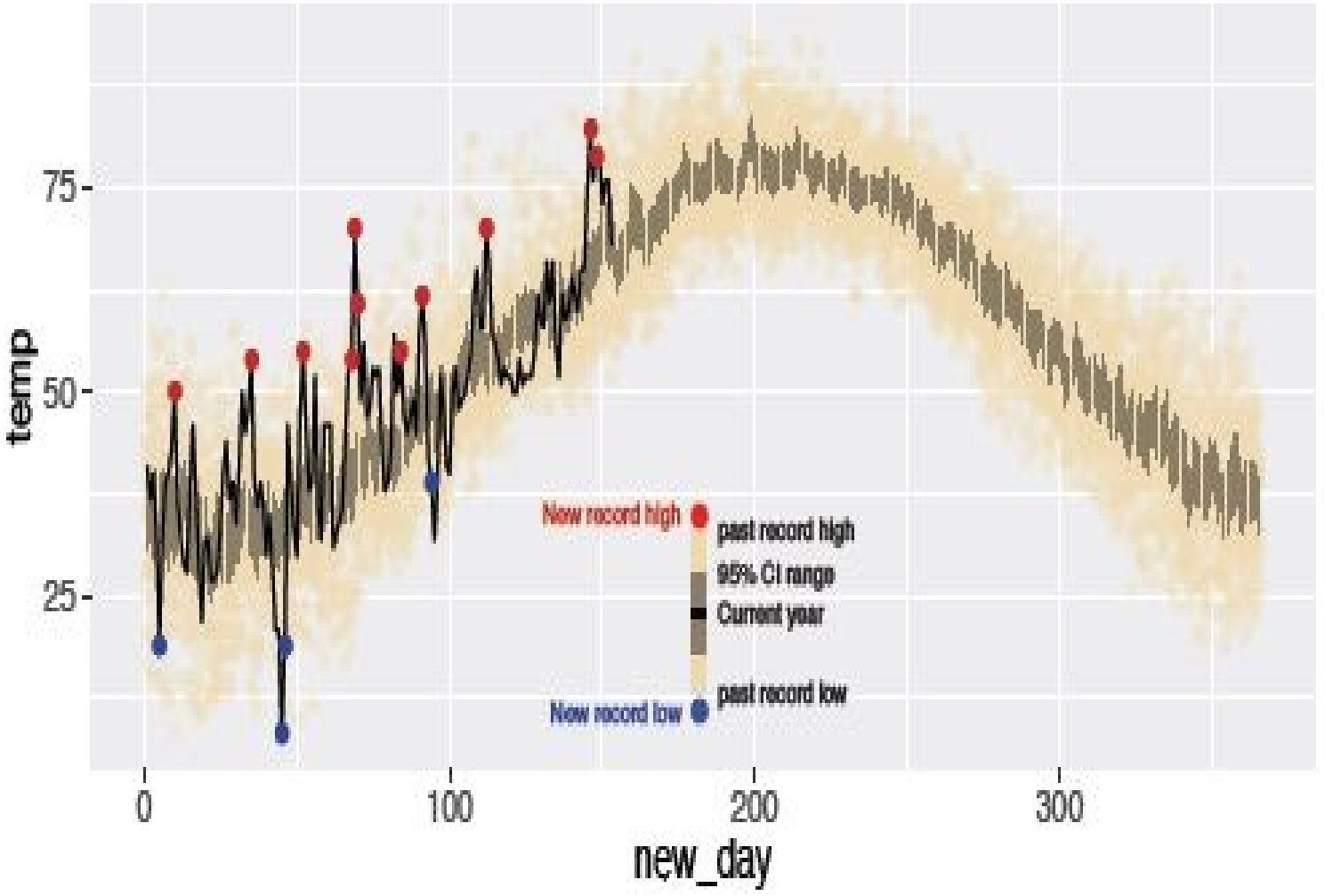


Gráficos Feitos com ggplot2

Gráficos Feitos com ggplot2



Gráficos Feitos com ggplot2



Como usar o ggplot2

- É como um "idioma", onde utilizamos os elementos do idioma para nos comunicar;
- Vamos combinando os elementos (que podem ser entendidas como camadas) para criarmos os plots que desejamos. Como fazer um lanche no subway, onde vamos adicionando o que queremos.

Como usar o ggplot2

Elementos do ggplot...

Element	Description
Data	The dataset being plotted.
Aesthetics	The scales onto which we *map* our data.
Geometries	The visual elements used for our data.
Facets	Plotting small multiples.
Statistics	Representations of our data to aid understanding.
Coordinates	The space on which the data will be plotted.
Themes	All non-data ink.

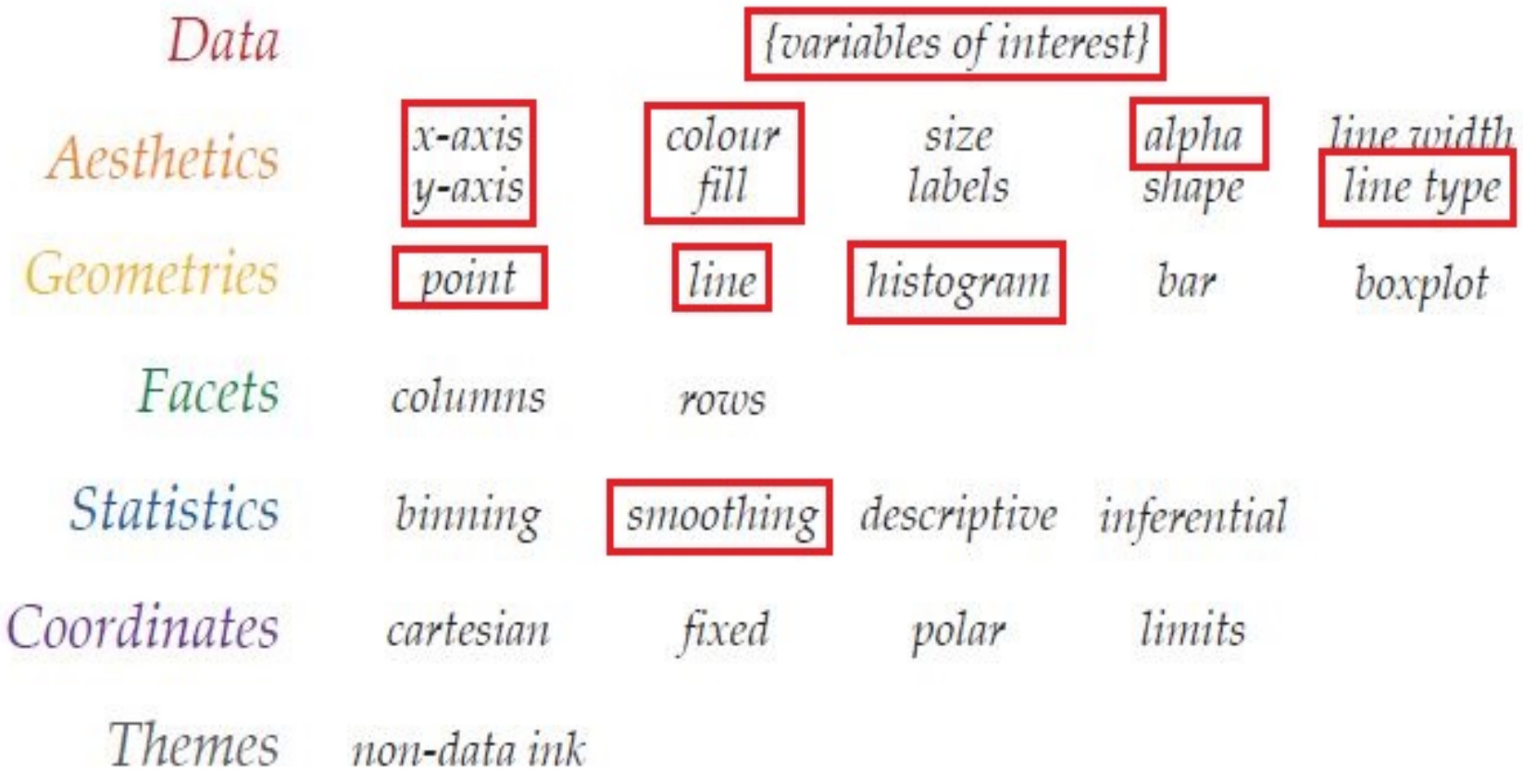
Como usar o ggplot2

Elementos do ggplot que cobriremos...

Element	Description
Data	The dataset being plotted.
Aesthetics	The scales onto which we *map* our data.
Geometries	The visual elements used for our data.
Facets	Plotting small multiples.
Statistics	Representations of our data to aid understanding.
Coordinates	The space on which the data will be plotted.
Themes	All non-data ink.

Como usar o ggplot2

Mais especificamente...



Como usar o ggplot2

Sintaxe dos plots:

```
ggplot(data = <DATA>) +
   <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Como usar o ggplot2

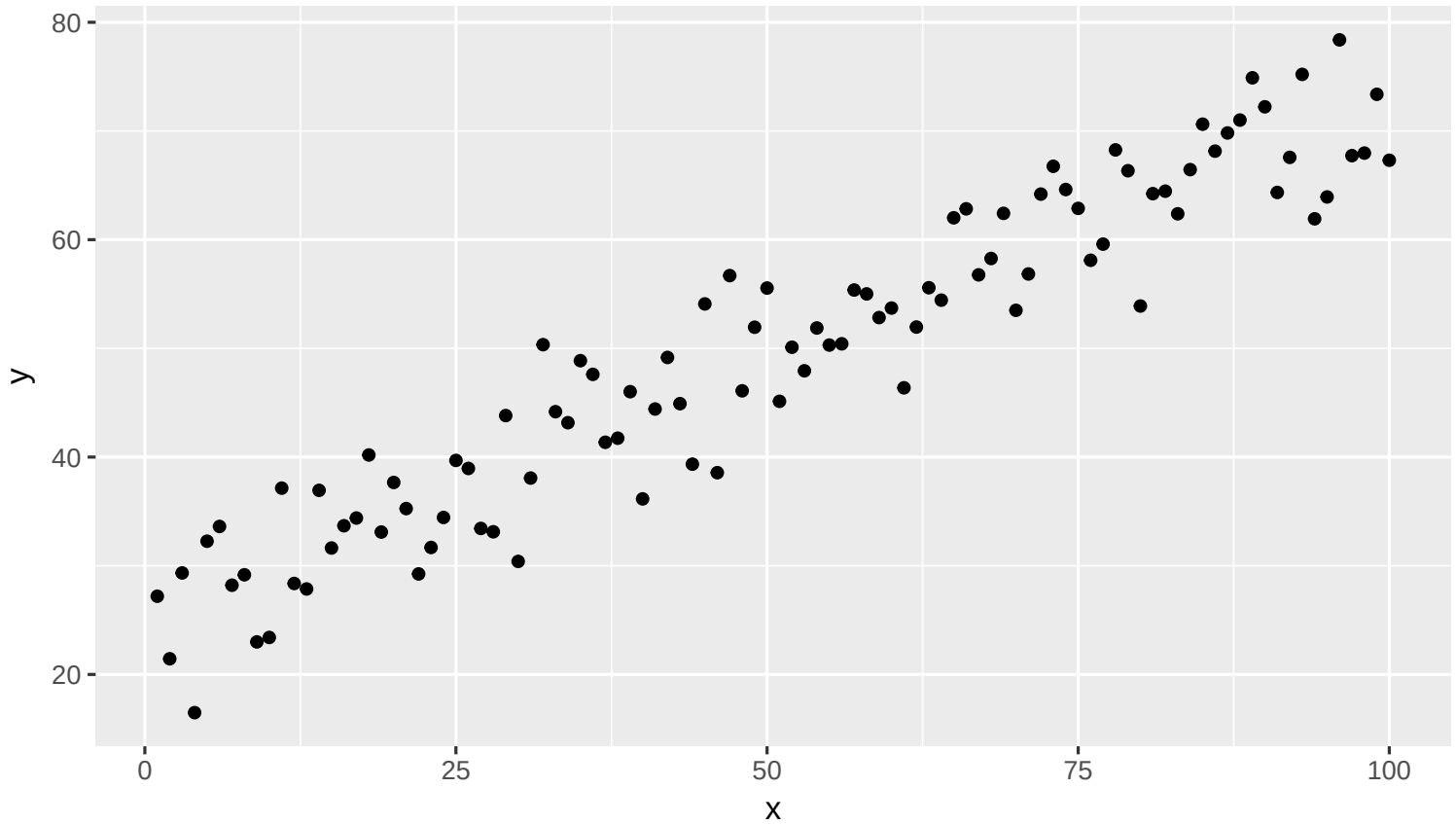
- Exemplo: Scatter Plot simples

```
#criando dados para serem plotados
dados <- tibble(x=1:100,
                y=x*0.5+rnorm(100,25,5))
dados
```

```
## # A tibble: 100 x 2
##        x        y
##    <int>    <dbl>
## 1      1 27.19474
## 2      2 21.44107
## 3      3 29.33936
## 4      4 16.48718
## 5      5 32.25920
## 6      6 33.61547
## 7      7 28.21178
## 8      8 29.16906
## 9      9 22.99460
## 10    10 23.40452
## # ... with 90 more rows
```

Como usar o ggplot2

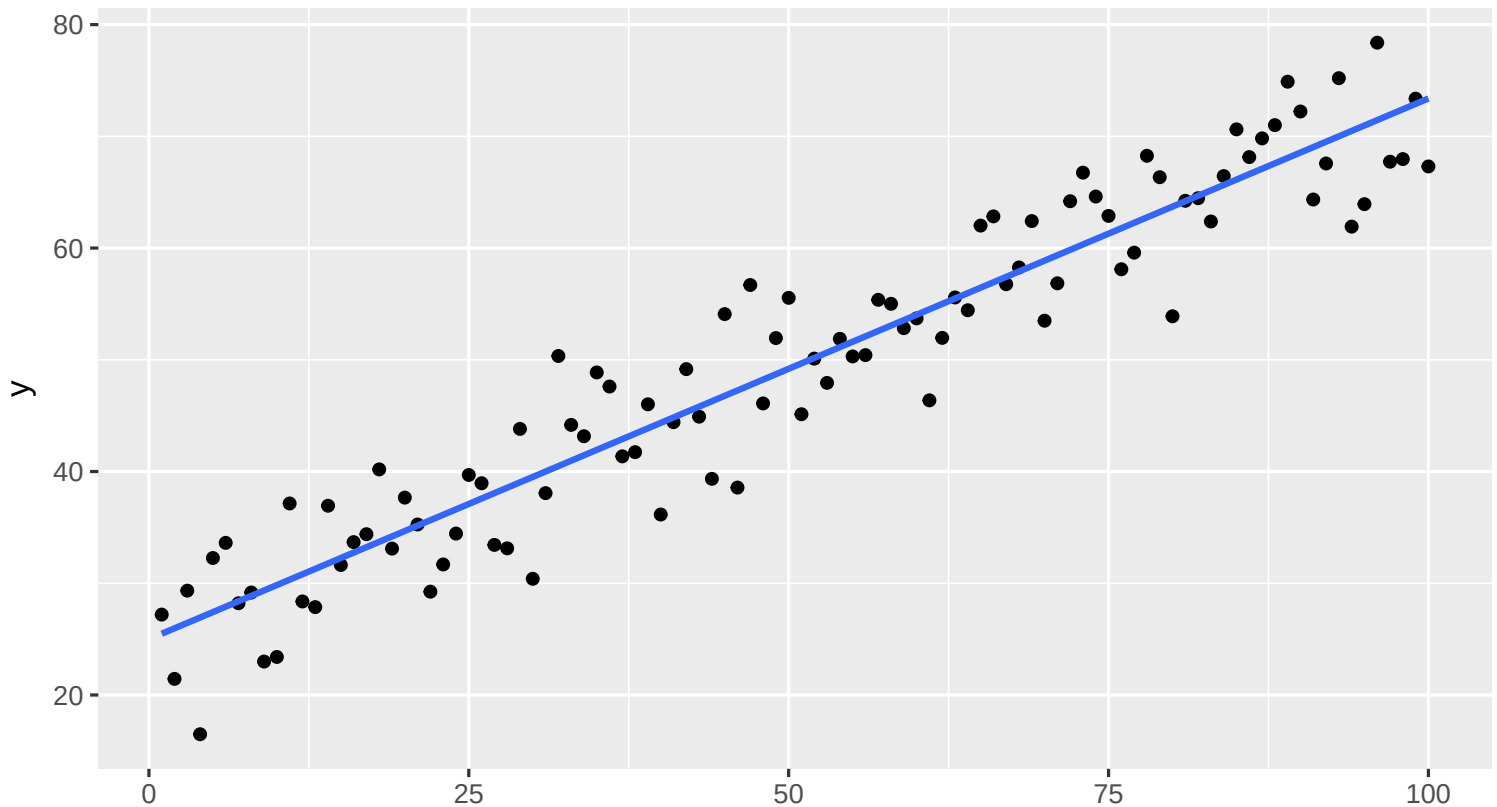
```
# plotando x vs. y
ggplot(data = dados)+geom_point(mapping = aes(x=x,y=y))
```



Como usar o ggplot2

Combinando *geometries* (camadas) para fazer adicionarmos uma linha de regressão

```
ggplot(data = dados)+geom_point(mapping = aes(x=x,y=y))+
  geom_smooth(mapping = aes(x=x,y=y),method = 'lm',se = F)
```

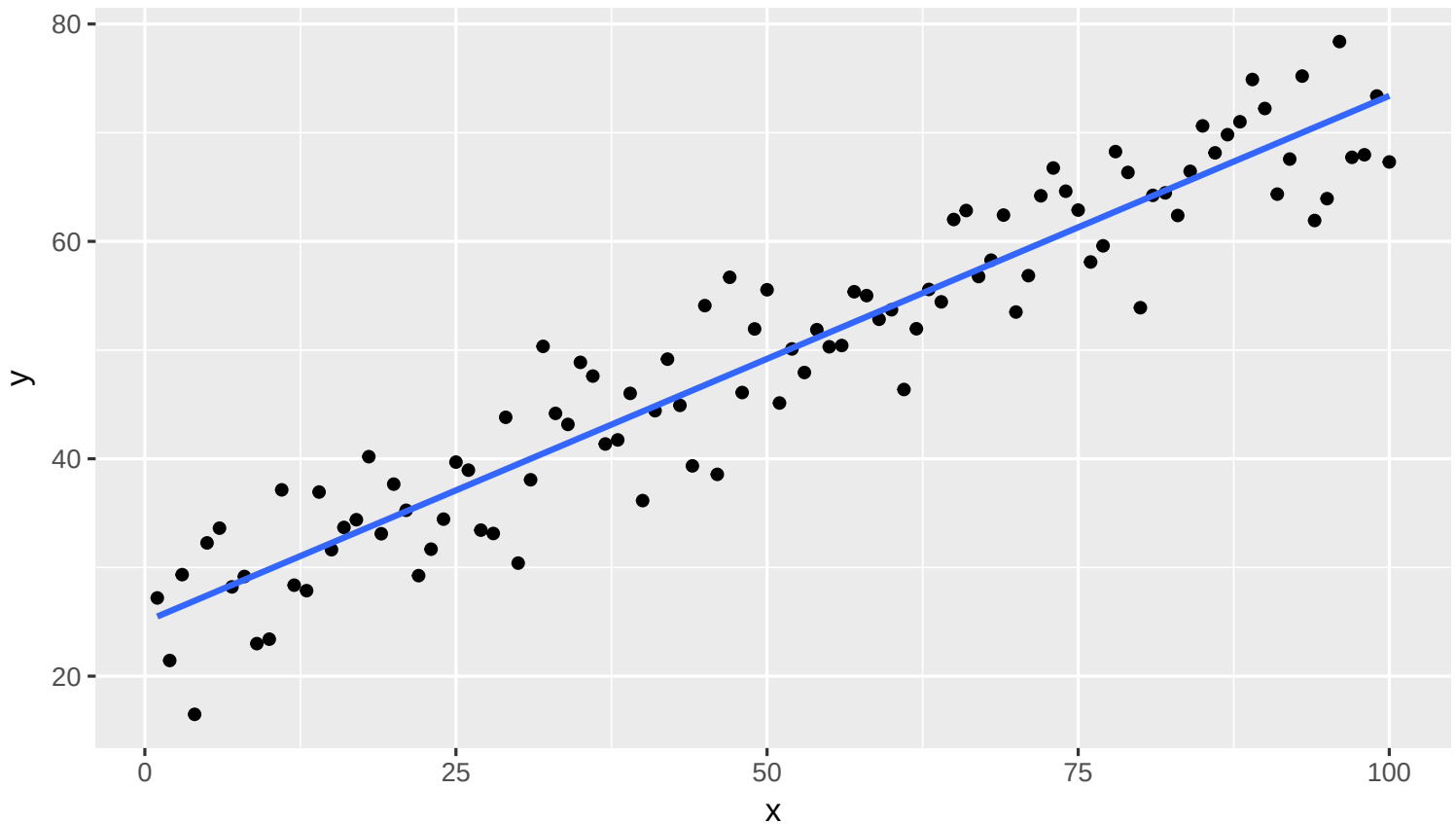


Como usar o ggplot2

Dica: Se sabemos que as variáveis usadas em diferentes *geometries* serão as mesmas, como no exemplo do slide anterior, podemos colocar o argumento **mapping = aes(x=x,y=y)** dentro da chamada **ggplot()**. Deste modo evitamos digitar duas vezes a mesma coisa.

Como usar o ggplot2

```
ggplot(data = dados, mapping = aes(x=x,y=y))+geom_point()+
  geom_smooth(method = 'lm',se = F)
```



Elementos do ggplot2

```
dados_ <- tibble(periodo = 1:500,
                 sexo = rep(c("M","F"),250),
                 peso = rnorm(500,55,11),
                 altura = rnorm(500,1.6,0.20))
```

Elementos do ggplot2

Data: o arquivo (dentro do R) que contem as variáveis que serão utilizadas. Colocamos dentro da função **ggplot(data = "nome do arquivo")**.

Elementos do ggplot2

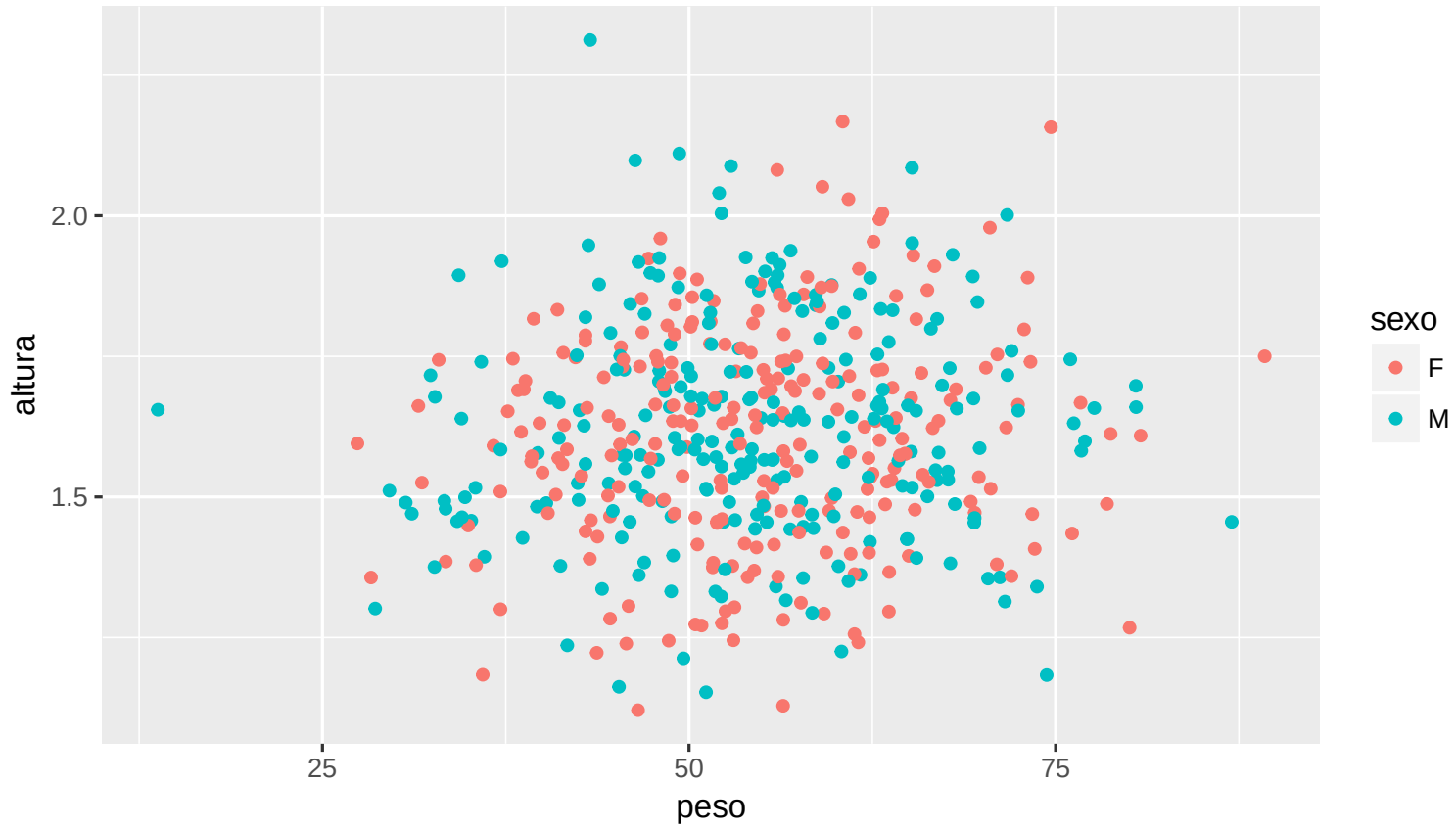
Aesthetics: são as características que serão "mapeadas" para o plot. Sendo alguma delas

- eixos x e y: definem as escalas e comprimentos dos eixos - declaramos no **ggplot(mappings = aes())**;
- colour: separa o elemento *geometries* em diferentes cores por grupos - declaramos no **ggplot(mappings = aes())**;
- fill: cria gráfico de proporção por grupos;
- linetype: cria linhas diferentes por cada grupo;
- alpha: controle a opacidade do *geometries* - - declaramos no **geom_**.

Elementos do ggplot2

color

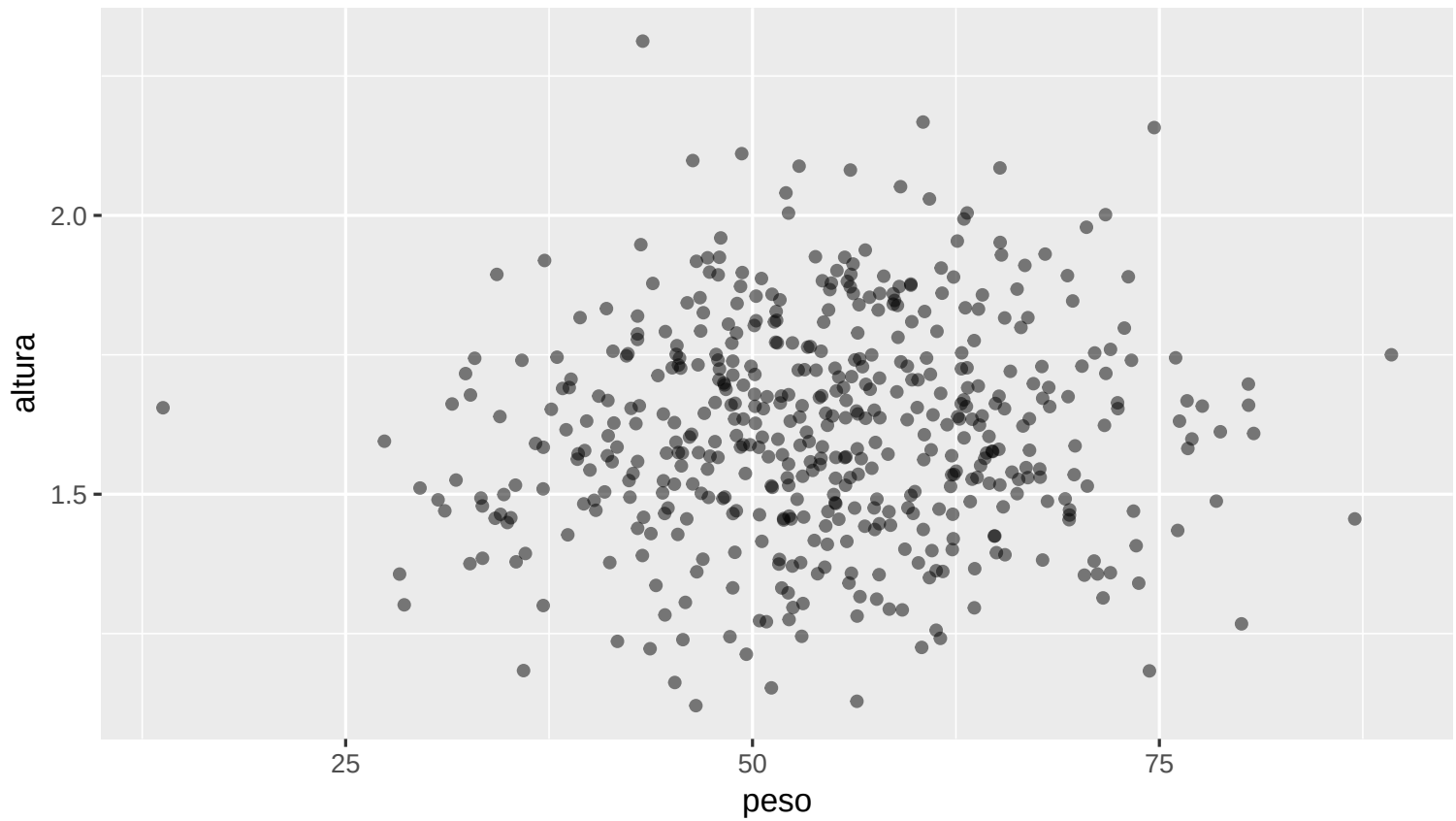
```
ggplot(data = dados_,aes(x=peso,y=altura,color=sexo))+geom_point()
```



Elementos do ggplot2

alpha

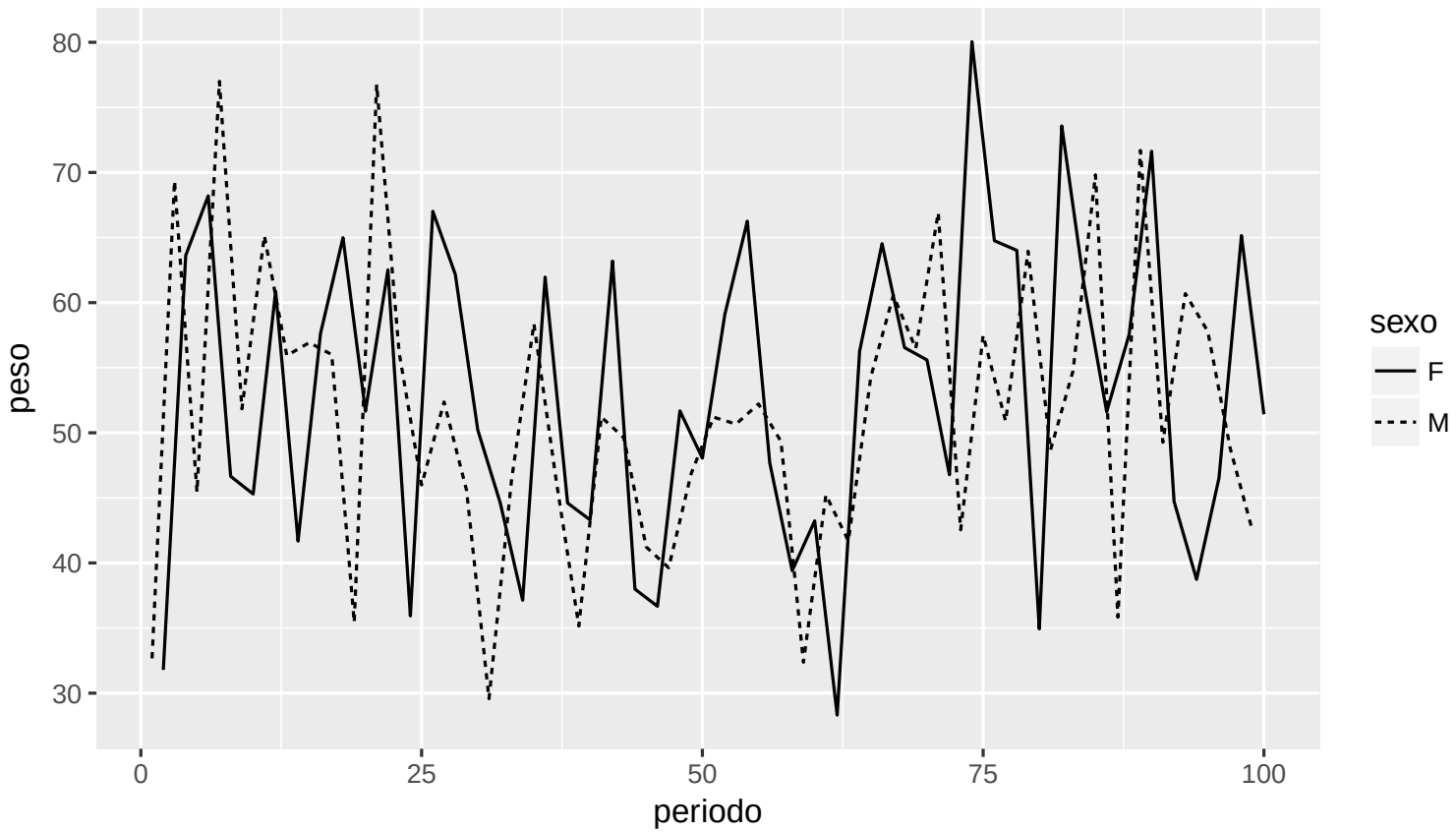
```
ggplot(data = dados_,aes(x=peso,y=altura))+geom_point(alpha=0.5)
```



Elementos do ggplot2

linetype

```
ggplot(data = dados_[1:100,],aes(x=periodo,y=peso,linetype=sexo)
```



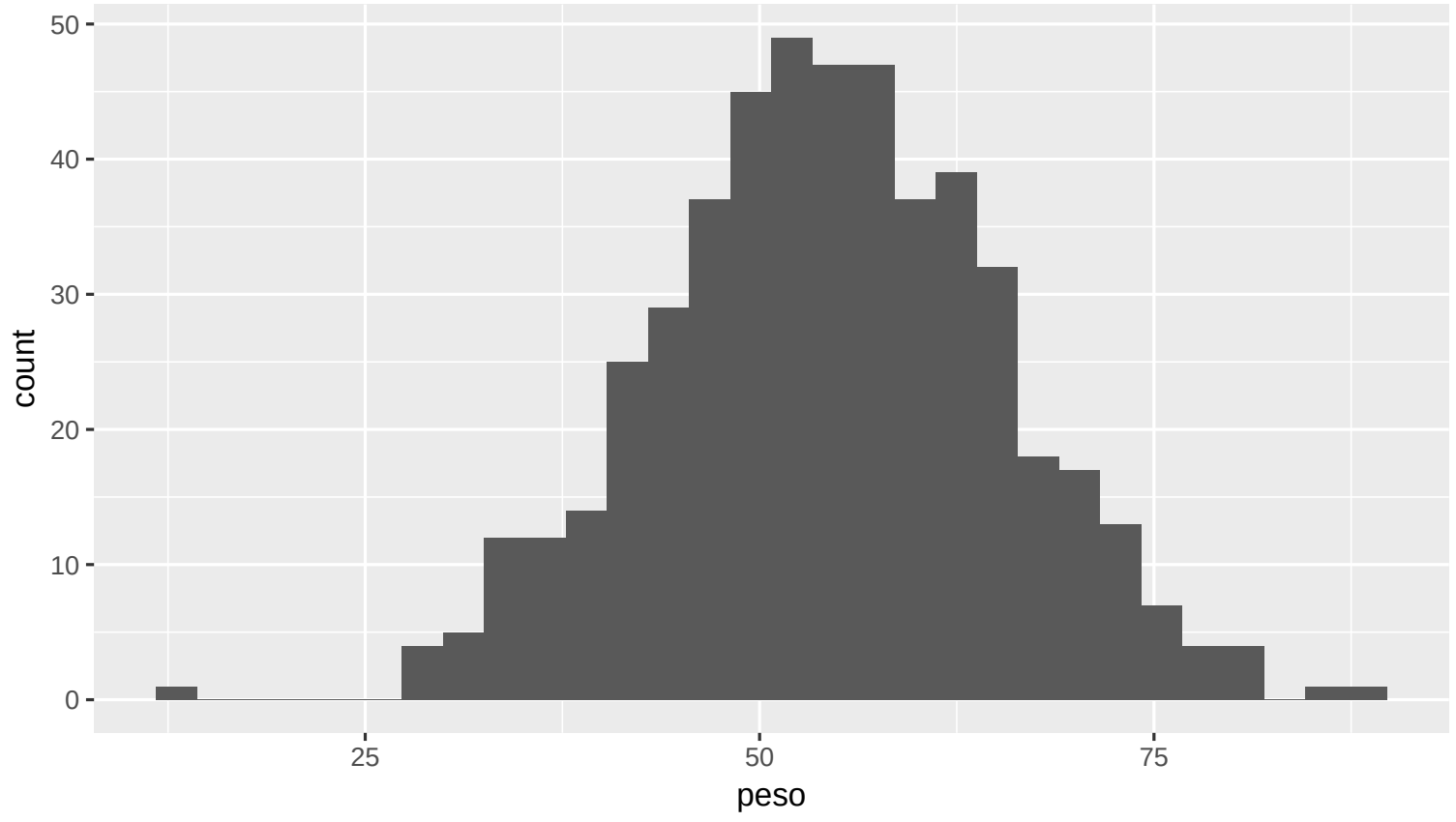
Elementos do ggplot2

Geometries: a forma como queremos representar graficamente as variaveis. e são chamadas através de **geom_point()**, por exemplo. Sendo alguns deles

- point (já vimos);
- line (já vimos);
- histogram.

Elementos do ggplot2

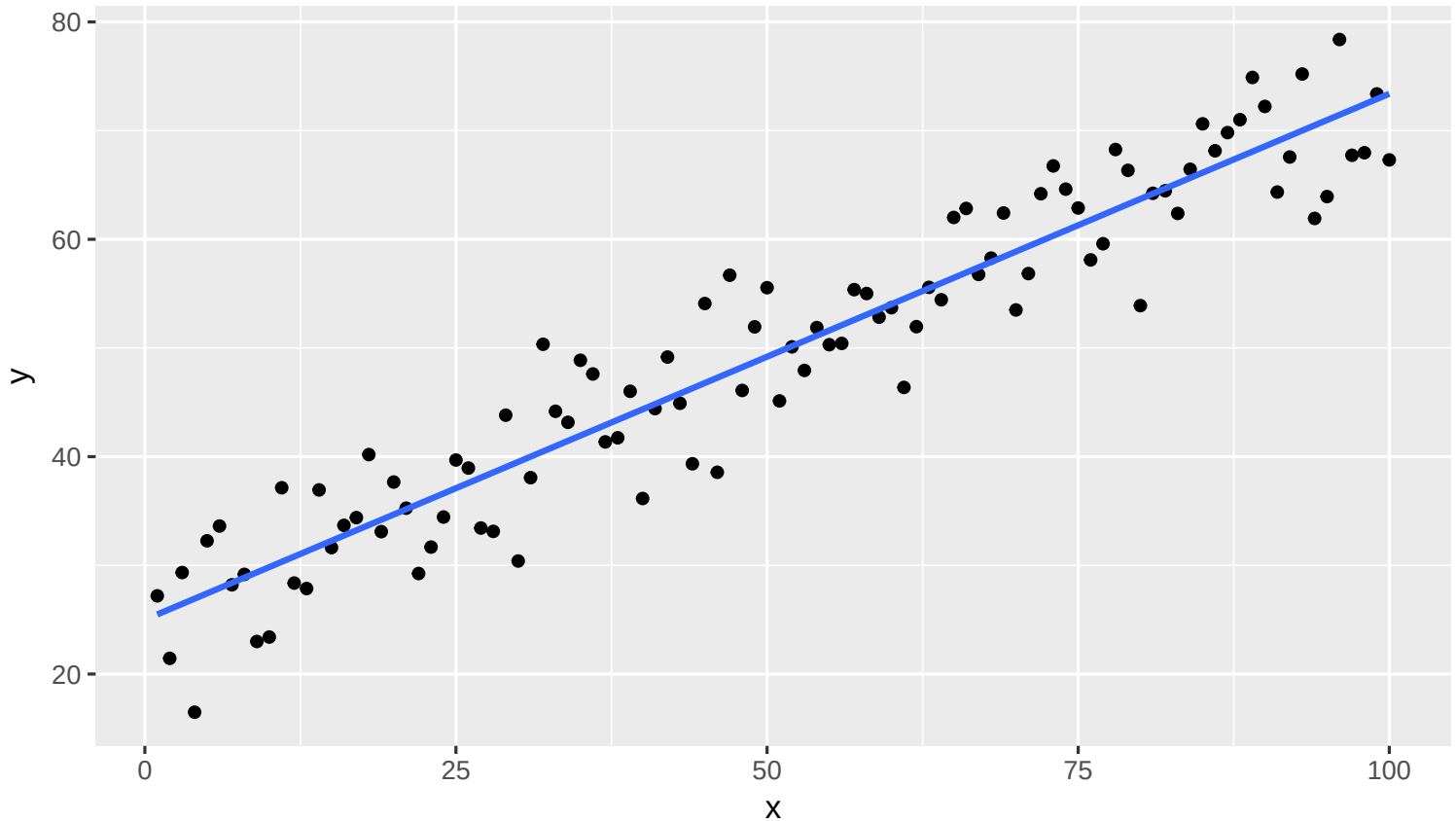
```
ggplot(data = dados_,aes(x = peso))+geom_histogram()
```



Elementos do ggplot2

Statistics: são representações de estimações estatísticas no gráfico. Vamos ver a *lm*, de *Linear model*. Chamamos tal elemento através de **geom_smooth(method=metodo_aqui)**. (refazendo o primeiro exemplo)

```
ggplot(data = dados, mapping = aes(x=x,y=y))+geom_point()+
geom_smooth(method ='lm',se = F)
```



Pacotes Utilizados

```
library(tibble)
library(tidyr)
library(dplyr)
```

Digressão

Tibble vs. Data.frame: dois pontos (principais) a favor do tibble

```
## Visualização
df <- data.frame(id = letters[1:100],
                 idade = sample(x = 0:70,size = 100,replace = T),
                 x = rnorm(100,0,1))
df
```

```
##        id idade           x
## 1       a    12 -0.54372045
## 2       b    62  0.72406463
## 3       c    52 -1.11729560
## 4       d    37 -0.17741012
## 5       e    15  0.14300892
## 6       f     8  0.49249758
## 7       g    33  0.06571036
## 8       h    54 -2.07312104
## 9       i     2  0.27258989
## 10      j    31 -0.37629667
## 11      k     5 -1.08050809
## 12      l    38 -0.43153046
## 13      m    52  0.87705542
## 14      n    24 -1.34198041
```

Digressão

Tibble vs. Data.frame: dois pontos (principais) a favor do tibble

```
## Visualização
tb <- tibble(id = letters[1:100],
                 idade = sample(x = 0:70,size = 100,replace = T),
                 x = rnorm(100,0,1))
tb
```

```
## # A tibble: 100 x 3
##       id idade          x
##    <chr> <int>      <dbl>
## 1      a    43  0.3864901
## 2      b    40  0.1255834
## 3      c    28  0.2521172
## 4      d    48 -0.1424644
## 5      e    24 -0.4926804
## 6      f    22  0.4000943
## 7      g    62  0.8615261
## 8      h    33  0.4210050
## 9      i     7  0.4322694
## 10     j    18 -2.5326386
## # ... with 90 more rows
```

Digressão

Tibble vs. Data.frame: dois pontos (principais) a favor do tibble

```
## criação de variaveis
df <- data.frame(id = letters[1:10],
                 idade = sample(x = 0:70,size = 10,replace = T),
                 x = 0.5*idade+rnorm(10,0,1))

## Error in data.frame(id = letters[1:10], idade = sample(x = 0:70, size
= 10, :  object 'idade' not found
```

Digressão

Tibble vs. Data.frame: dois pontos (principais) a favor do tibble

```
## Criação de variaveis
tb <- tibble(id = letters[1:10],
                 idade = sample(x = 0:70,size = 10,replace = T),
                 x = 0.5*idade+rnorm(10,0,1))
tb
```

```
## # A tibble: 10 x 3
##       id idade         x
##    <chr> <int>     <dbl>
## 1      a    69 35.460038
## 2      b    47 22.948514
## 3      c    69 35.409948
## 4      d    57 29.531807
## 5      e    42 20.642670
## 6      f    12  7.676983
## 7      g    67 31.735749
## 8      h    50 25.442991
## 9      i    27 12.574880
## 10     j    22 13.039402
```

Digressão

Importando dados .csv

```
df <- read.csv(file = "nome_do_arquivo.csv",
               header = "primeira linha são nomes? se sim 'T' se não 'F'",
               sep = "Separador das células (',',';'...)",
               dec = "Separador decimal")
```

Digressão

Importando dados .csv

Atividade

Importar e salvar no *workingspace/environment* as planilhas (i) "100_amostras" e "sexo".

Digressão

```
amostra_100 <- read.csv("100_amostras.csv",
                        header = T,sep = ';',dec = '.')
sexo <- read.csv("sexo.csv",
                        header = T,sep = ';',dec = '.')
```

Introdução ao Conceito de Tidy Data

O que é?

É uma maneira de se armazenar dados estatísticos de maneira a facilitar a análise dos dados. Tem três principios principais:

- Observações como linhas;
- Variaveis como colunas;
- Uma tipo de unidade observacional por matrix.

Introdução ao Conceito de Tidy Data

Dados conforme os princípios

name	age	eye_color	height
Jake	34	Other	6'1"
Alice	55	Blue	5'9"
Tim	76	Brown	5'7"
Denise	19	Other	5'1"

Observation

Variable or Attribute

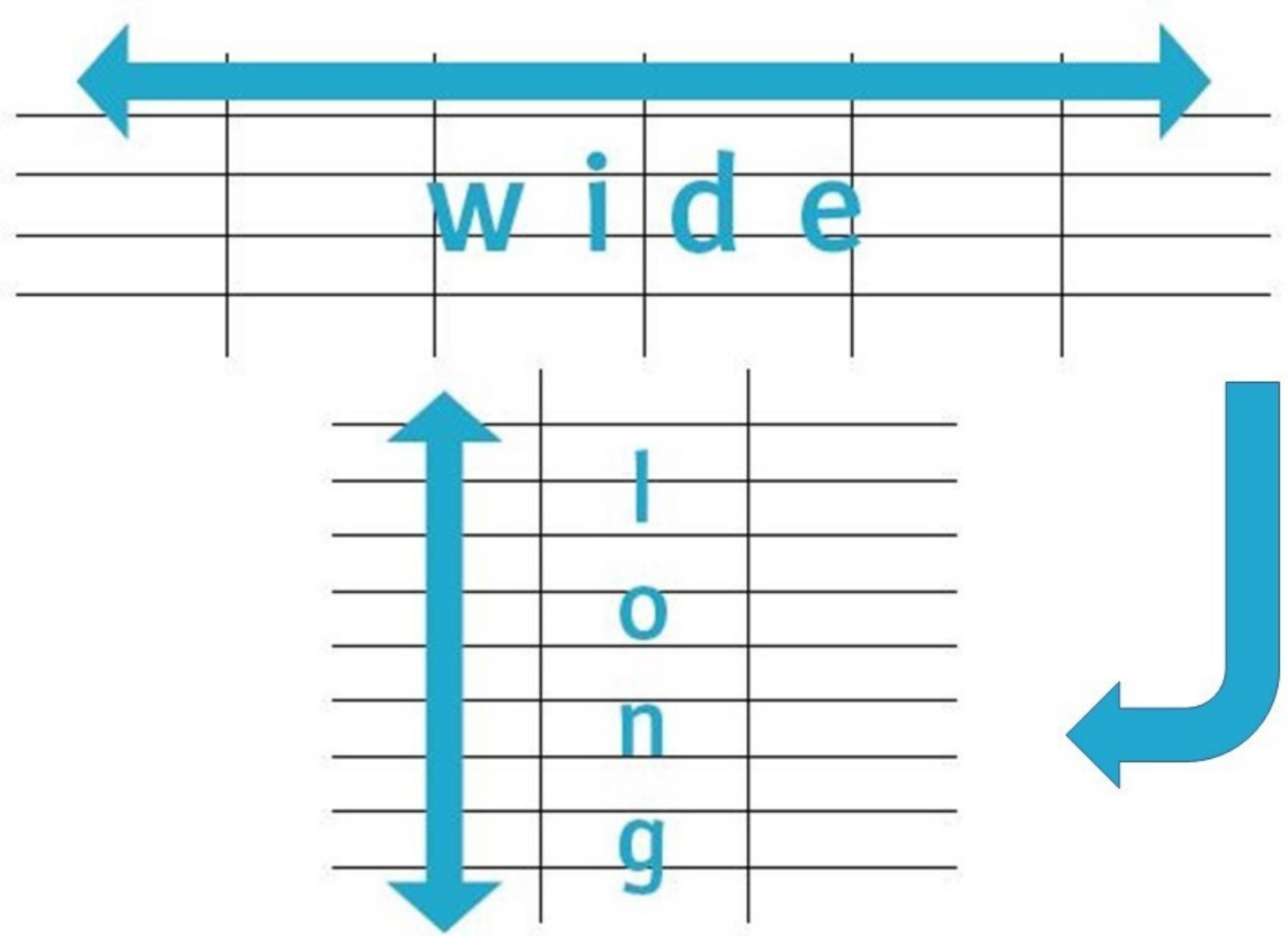
Introdução ao Conceito de Tidy Data

Dados em desconforme com os princípios

name	age	brown	blue	other	height
Jake	34	0	0	1	6'1"
Alice	55	0	1	0	5'9"
Tim	76	1	0	0	5'7"
Denise	19	0	0	1	5'1"

Dois problemas Comuns

Variáveis como nome de Colunas



Dois problemas Comuns

Variáveis como nome de Colunas

```
errado
```

```
## # A tibble: 3 x 4
##      id variavel  2015  2016
##   <chr>    <chr> <dbl> <dbl>
## 1  Zeus    idade    10    11
## 2 Eliot    idade    30    31
## 3   Ela    idade    NA    17
```

```
# filtrando para ter só individuos completos
errado <- errado[complete.cases(errado),]
errado
```

```
## # A tibble: 2 x 4
##      id variavel  2015  2016
##   <chr>    <chr> <dbl> <dbl>
## 1  Zeus    idade    10    11
## 2 Eliot    idade    30    31
```

Dois problemas Comuns

Variáveis como nome de Colunas

```
correto
```

```
## # A tibble: 4 x 3
##      id   ano idade
##   <chr> <dbl> <dbl>
## 1  Zeus  2015    10
## 2  Zeus  2016    11
## 3 Eliot  2015    30
## 4 Eliot  2016    31
```

Dois problemas Comuns

Variáveis como nome de Colunas

Função **gather()** faz isso. obs: todos os argumentos são sem aspas.

```
gather(data = "dados",key = "nome da coluna da variavel que esta espalhada em colunas",
       value = "nome da variavel que esta nas celulas",
       -c("colunas que não serão agrupadas"))
```

```
gather(data = errado,key = ano,value = idade,
       -c(id,variavel))

## # A tibble: 4 x 4
##      id variavel   ano idade
##   <chr>    <chr> <chr> <dbl>
## 1  Zeus    idade  2015    10
## 2 Eliot    idade  2015    30
## 3  Zeus    idade  2016    11
## 4 Eliot    idade  2016    31
```

Dois problemas Comuns

Variáveis como nome de Colunas (inner join)

Atividade

Colocar as planilhas (i) "100_amostra" e (ii) "sexo" de acordo com os princípios de *tidy data*

Dois problemas Comuns

```
amostras_100 <- gather(data = amostras_100,key = amostra,
                       value = consumo,
                       -renda)
as_tibble(amostras_100)

## # A tibble: 10,000 x 3
##    renda   amostra   consumo
##    <int>     <chr>     <dbl>
## 1    100 amostra.1  51.82646
## 2    201 amostra.1 190.20948
## 3    302 amostra.1 167.66289
## 4    403 amostra.1 316.87276
## 5    504 amostra.1 363.15662
## 6    605 amostra.1 420.04058
## 7    706 amostra.1 452.02989
## 8    807 amostra.1 520.10506
## 9    908 amostra.1 625.60375
## 10  1009 amostra.1 659.64036
## # ... with 9,990 more rows
```

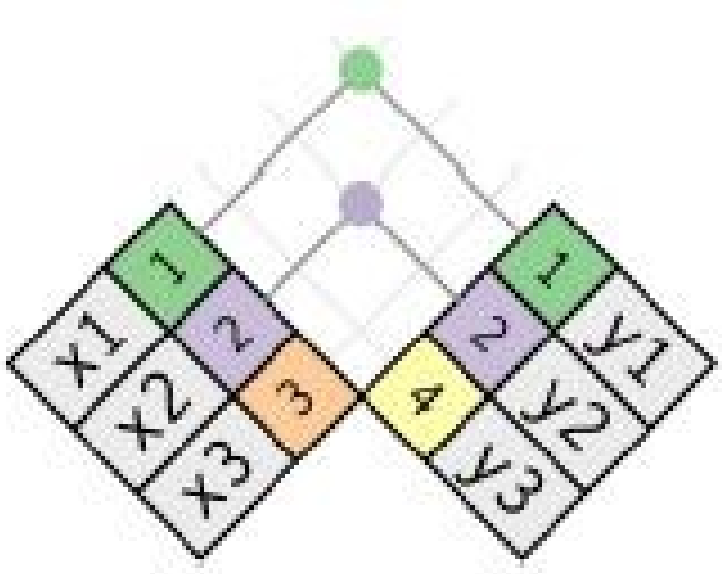
Dois problemas Comuns

```
sexo <- gather(data = sexo,key = amostra,value = sexo,-renda)
as_tibble(sexo)
```

```
## # A tibble: 9,000 x 3
##    renda   amostra   sexo
##    <int>     <chr>  <chr>
## 1    605 amostra.1  Homem
## 2    706 amostra.1  Homem
## 3    807 amostra.1  Homem
## 4    908 amostra.1  Homem
## 5   1009 amostra.1  Homem
## 6   1110 amostra.1  Homem
## 7   1211 amostra.1  Homem
## 8   1312 amostra.1  Homem
## 9   1413 amostra.1 Mulher
## 10  1514 amostra.1  Homem
## # ... with 8,990 more rows
```

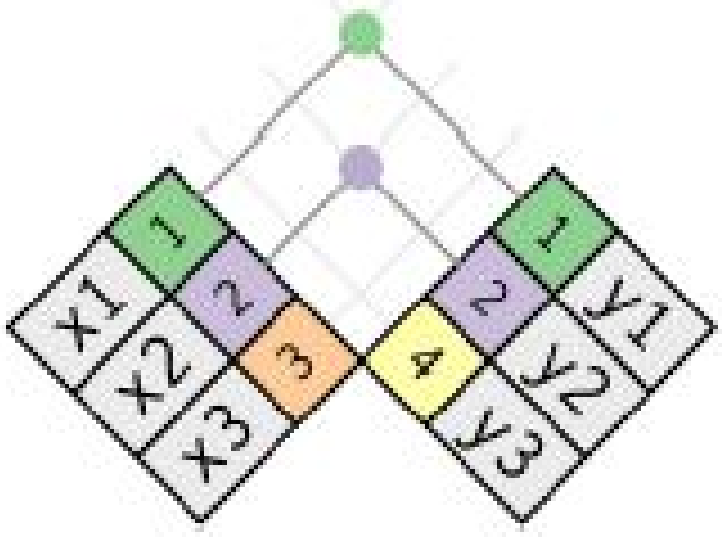
Dois problemas Comuns

Juntar duas Bases de dados (inner join)



Dois problemas Comuns

Juntar duas Bases de dados (inner join)



key	val_x	val_y
1	x1	y1
2	x2	y2

Dois problemas Comuns

Juntar duas Bases de dados (inner join): Exemplo

```
df1
```

```
## # A tibble: 4 x 3
##       id   ano  peso
##    <chr> <dbl> <dbl>
## 1    ana     1    30
## 2    ana     2    29
## 3 daniel     1    60
## 4 daniel     2    63
```

```
df2
```

```
## # A tibble: 5 x 3
##         id   ano altura
##      <chr> <dbl>  <dbl>
## 1      ana     1   1.60
## 2      ana     2   1.60
## 3   Daniel     1   1.78
## 4 Gilberto     1   1.99
## 5 Gilberto     2   1.98
```

Dois problemas Comuns

Juntar duas Bases de dados (inner join): Exemplo

```
df_juntos <- inner_join(x = df1,y = df2,c("id","ano"))
df_juntos

## # A tibble: 2 x 4
##      id   ano  peso altura
##   <chr> <dbl> <dbl>  <dbl>
## 1   ana     1    30    1.6
## 2   ana     2    29    1.6
```

Dois problemas Comuns

Juntar duas Bases de dados (inner join): Exemplo

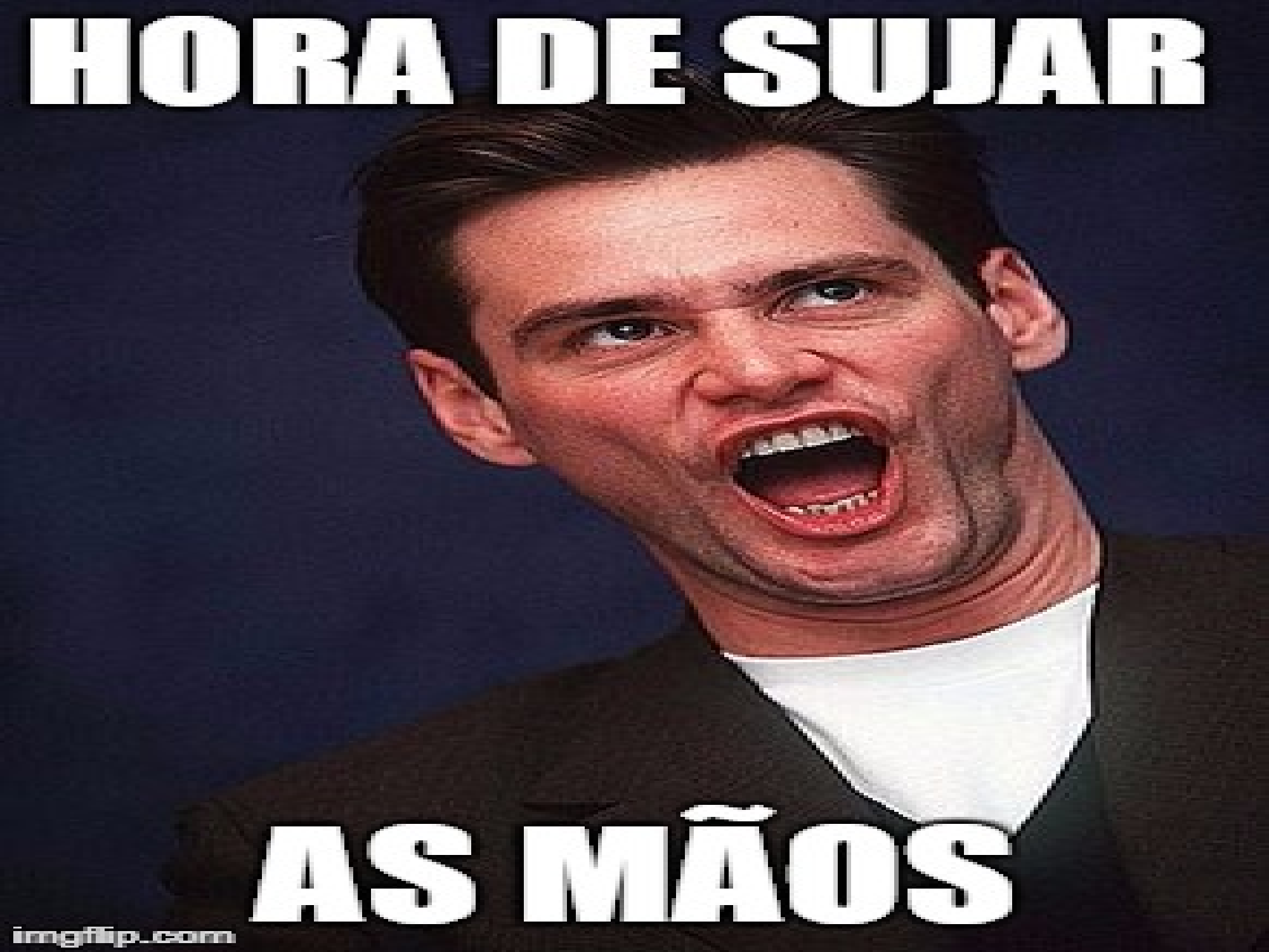
Atividade

Juntar as planilhas "100_amostras" e "sexo" em uma em apenas um data frame através da função **inner_join()**, de acordo com as faixas de renda e o número da amostra.

Dois problemas Comuns

```
dados <- inner_join(x = amostras_100,y = sexo,
                    by = c("amostra","renda"))
as_tibble(dados)
```

```
## # A tibble: 9,000 x 4
##    renda   amostra   consumo   sexo
##    <int>     <chr>     <dbl>  <chr>
## 1    605 amostra.1  420.0406  Homem
## 2    706 amostra.1  452.0299  Homem
## 3    807 amostra.1  520.1051  Homem
## 4    908 amostra.1  625.6038  Homem
## 5   1009 amostra.1  659.6404  Homem
## 6   1110 amostra.1  743.1548  Homem
## 7   1211 amostra.1  815.8679  Homem
## 8   1312 amostra.1  950.7011  Homem
## 9   1413 amostra.1 1012.0082 Mulher
## 10  1514 amostra.1 1025.5946  Homem
## # ... with 8,990 more rows
```


HORA DE SUJAR
AS MÃOS
imgflip.com

Atividade Prática 1

Contextualização

Você, como economista do departamento de desenvolvimento social do Banco Mundial, foi designado para realizar um estudo que busca verificar como a liberdade de imprensa e qualidade na democracia afetam a percepção de currupção ao redor do mundo.

Atividade Prática 1

Contextualização

Para tal, você utilizará os seguites dados:

- Índice de percepção de corrupção (International Transparency): entre 0 e 100, onde 0 é percepção máxma de corrupção e 100 percepção mínima;
- Índice de democracia (Economist Intelligence Unit): entre 0 e 10, onde quanto maior o valor do índice mais democrático é o país;
- Índice de liberdade de imprensa (*Reporters Without Borders*): entre 0 e 100, onde quanto menor o índice maior a liberdade de imprensa;
- Índice de desenvolvimento humano (Nações Unidas): utilizado com controle de diferenças entre os países.

Atividade Prática 1

Estrutura

O trabalho será realizado nas seguintes etapas:

1. Importar os arquivos "democracy_index", "corruption_index","human_development_index" e "press_freedom". Obs: Nessa etapa lembre-se de filtrar os países que tenham observações para todos os ano com a função **complete.cases()**;

Atividade Prática 1

Estrutura

O trabalho será realizado nas seguintes etapas:

1. Importar os arquivos "democracy_index", "corruption_index","human_development_index" e "press_freedom". Obs: Nessa etapa lembre-se de filtrar os países que tenham observações para todos os ano com a função **complete.cases()**;
2. Verificar se eles estão de acordo com os princípios de *tidy data*, se não estiverem arruma-los;

Atividade Prática 1

Estrutura

O trabalho será realizado nas seguintes etapas:

1. Importar os arquivos "democracy_index", "corruption_index","human_development_index" e "press_freedom". Obs: Nessa etapa lembre-se de filtrar os países que tenham observações para todos os ano com a função **complete.cases()**;
2. Verificar se eles estão de acordo com os princípios de *tidy data*, se não estiverem arruma-los;
3. Juntar todos os dados em um mesmo data.frame;

Atividade Prática 1

Estrutura

O trabalho será realizado nas seguintes etapas:

1. Importar os arquivos "democracy_index", "corruption_index","human_development_index" e "press_freedom". Obs: Nessa etapa lembre-se de filtrar os países que tenham observações para todos os ano com a função **complete.cases()**;
2. Verificar se eles estão de acordo com os princípios de *tidy data*, se não estiverem arruma-los;
3. Juntar todos os dados em um mesmo data.frame;
4. Visualisar scatter plot da variáveis de interesse x índice de percepção de corrupção (com uma linha de regressão linear);

Atividade Prática 1

Estrutura

O trabalho será realizado nas seguintes etapas:

1. Importar os arquivos "democracy_index", "corruption_index","human_development_index" e "press_freedom". Obs: Nessa etapa lembre-se de filtrar os países que tenham observações para todos os ano com a função **complete.cases()**;
2. Verificar se eles estão de acordo com os princípios de *tidy data*, se não estiverem arruma-los;
3. Juntar todos os dados em um mesmo data.frame;
4. Visualisar scatter plot da variáveis de interesse x índice de percepção de corrupção (com uma linha de regressão linear);
5. Salvar o data.frame com todos os dados arrumados através da função **write.csv()**.

Digressão

Como criar data no R: pontuais

```
x <- "27/07/2017"
str(x)

##  chr "27/07/2017"

x_data <- as.Date(x,format = "%d/%m/%Y")
str(x_data)

##  Date[1:1], format: "2017-07-27"
```

Digressão

Como criar data no R: sequencias

```
### por dia
seq(as.Date('2017-07-20'),
    as.Date('2017-07-27'),
    by='day')

## [1] "2017-07-20" "2017-07-21" "2017-07-22" "2017-07-23" "2017-07-24"
## [6] "2017-07-25" "2017-07-26" "2017-07-27"

### por mês
seq(from=as.Date('2017-03-20'),
    to=as.Date('2017-07-27'),
    by='month')

## [1] "2017-03-20" "2017-04-20" "2017-05-20" "2017-06-20" "2017-07-20"
```

Digressão

Como popular as colunas de uma tabela com o foor loop

```
# 1º) criar a matriz a ser populada
matriz <- data.frame(matrix(nrow = 11,ncol = 7))
colnames(matriz)[1] <- 'Data'
matriz[,1] <- seq(as.Date("2017-01-01"),length.out = 11,by='month')
```

Digressão

Como popular as colunas de uma tabela com o foor loop

```
# 1º) criar a matriz a ser populada
matriz <- data.frame(matrix(nrow = 11,ncol = 7))
colnames(matriz)[1] <- 'Data'
matriz[,1] <- seq(as.Date("2017-01-01"),length.out = 11,by='month')
matriz
```

```
##          Data X2 X3 X4 X5 X6 X7
## 1  2017-01-01 NA NA NA NA NA NA
## 2  2017-02-01 NA NA NA NA NA NA
## 3  2017-03-01 NA NA NA NA NA NA
## 4  2017-04-01 NA NA NA NA NA NA
## 5  2017-05-01 NA NA NA NA NA NA
## 6  2017-06-01 NA NA NA NA NA NA
## 7  2017-07-01 NA NA NA NA NA NA
## 8  2017-08-01 NA NA NA NA NA NA
## 9  2017-09-01 NA NA NA NA NA NA
## 10 2017-10-01 NA NA NA NA NA NA
## 11 2017-11-01 NA NA NA NA NA NA
```

Digressão

Como popular as colunas de uma tabela com o foor loop

Suponha que queiramos preencher cada uma das colunas com um vetor de numeros aleatórios gerado dentro do loop.

Digressão

Como popular as colunas de uma tabela com o foor loop

Suponha que queiramos preencher cada uma das colunas com um vetor de numeros aleatórios gerado dentro do loop. Fazemos isso da seguinte maneira

Digressão

Como popular as colunas de uma tabela com o foor loop

```
col_names <- paste('col',1:7,sep = '_')
col_names

## [1] "col_1" "col_2" "col_3" "col_4" "col_5" "col_6" "col_7"

# 3º) usar o loop para popular a tabela
for (j in 2:ncol(matriz)){
  col <- rnorm(nrow(matriz)) # criando vetor que será coluna
  col <- round(col,2)        # arredondando casas decimais

  colnames(matriz)[j] <-  col_names[j] # dando nome a coluna
  matriz[,j] <- col           # atribuindo à coluna j o vetor
}
```

Digressão

Como popular as colunas de uma tabela com o foor loop

```
matriz
```

```
##          Data col_2 col_3 col_4 col_5 col_6 col_7
## 1  2017-01-01  0.17 -0.09  0.01 -0.75 -0.48  2.49
## 2  2017-02-01  0.10  0.72 -1.26 -0.95  1.11  1.94
## 3  2017-03-01  1.51  1.08 -0.34 -1.31  0.12 -0.17
## 4  2017-04-01 -0.60  0.00  0.73  0.24 -0.05 -1.08
## 5  2017-05-01  0.06 -0.55  1.38  1.52 -2.01 -0.18
## 6  2017-06-01  0.43  1.32  2.41 -2.65 -1.35  0.60
## 7  2017-07-01 -0.06 -0.63 -2.03 -0.76 -0.60  0.00
## 8  2017-08-01 -0.44  0.63 -1.60  0.61  0.55  0.41
## 9  2017-09-01 -0.32  1.28  0.03  2.59 -0.85 -0.20
## 10 2017-10-01 -0.95 -0.88 -1.21  1.66  0.56  0.33
## 11 2017-11-01 -0.07 -1.25 -0.44 -0.39 -0.56  1.17
```

Apresentação do rbcb

É um pacote que permite pegar as séries do sistema gerenciador de séries temporais do Banco Central do Brasil e importá-los diretamente para o R:

```
install.packages("rbcb")
library(rbcb)
```

Apresentação do rbcb

Principal Função

```
get_series(code = "codigo da série",
           start_date = "inicio da serie",
           end_date = "final da série",
           name = "nome da série (colunm header)",
           as = "'data.frame','tibble'")
```

Apresentação do rbcb

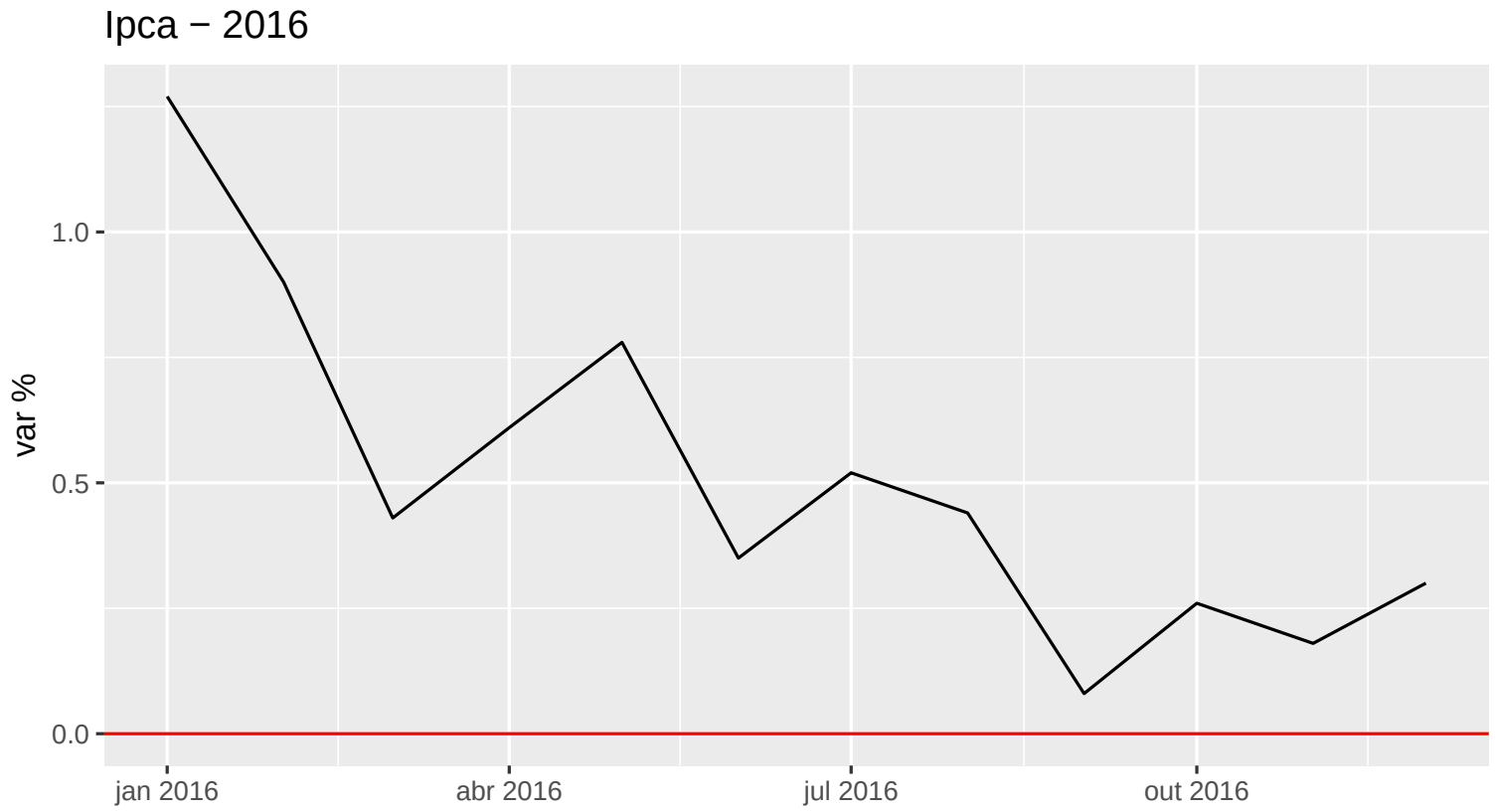
Exemplo 1: Ipca de 2016

```
inicio <- as.Date("2016-01-01")
fim <- as.Date("2016-12-01")
ipca16 <- get_series(code = 433,
                     start_date = inicio,
                     end_date = fim,
                     name = 'ipca',
                     as = "tibble")
```

Apresentação do rbcb

Exemplo 1: Ipca de 2016

```
ggplot(ipca16,aes(x=date,y=ipca))+geom_line()+
labs(x='',title='Ipca - 2016',y="var %")+
geom_hline(yintercept = 0,color='red')
```



Apresentação do rbcb

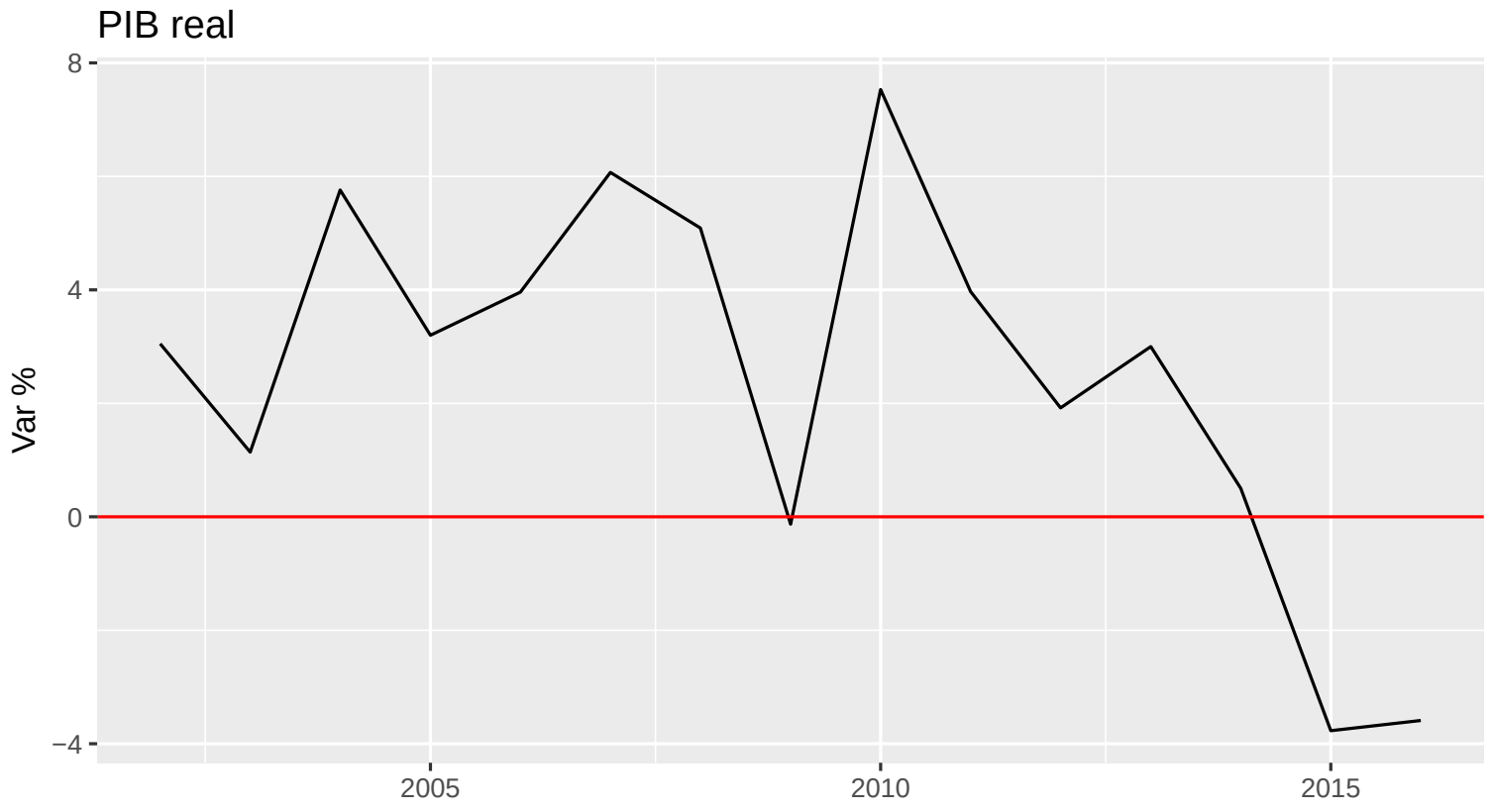
Exemplo 2: Variação real do PIB dos últimos 15 anos

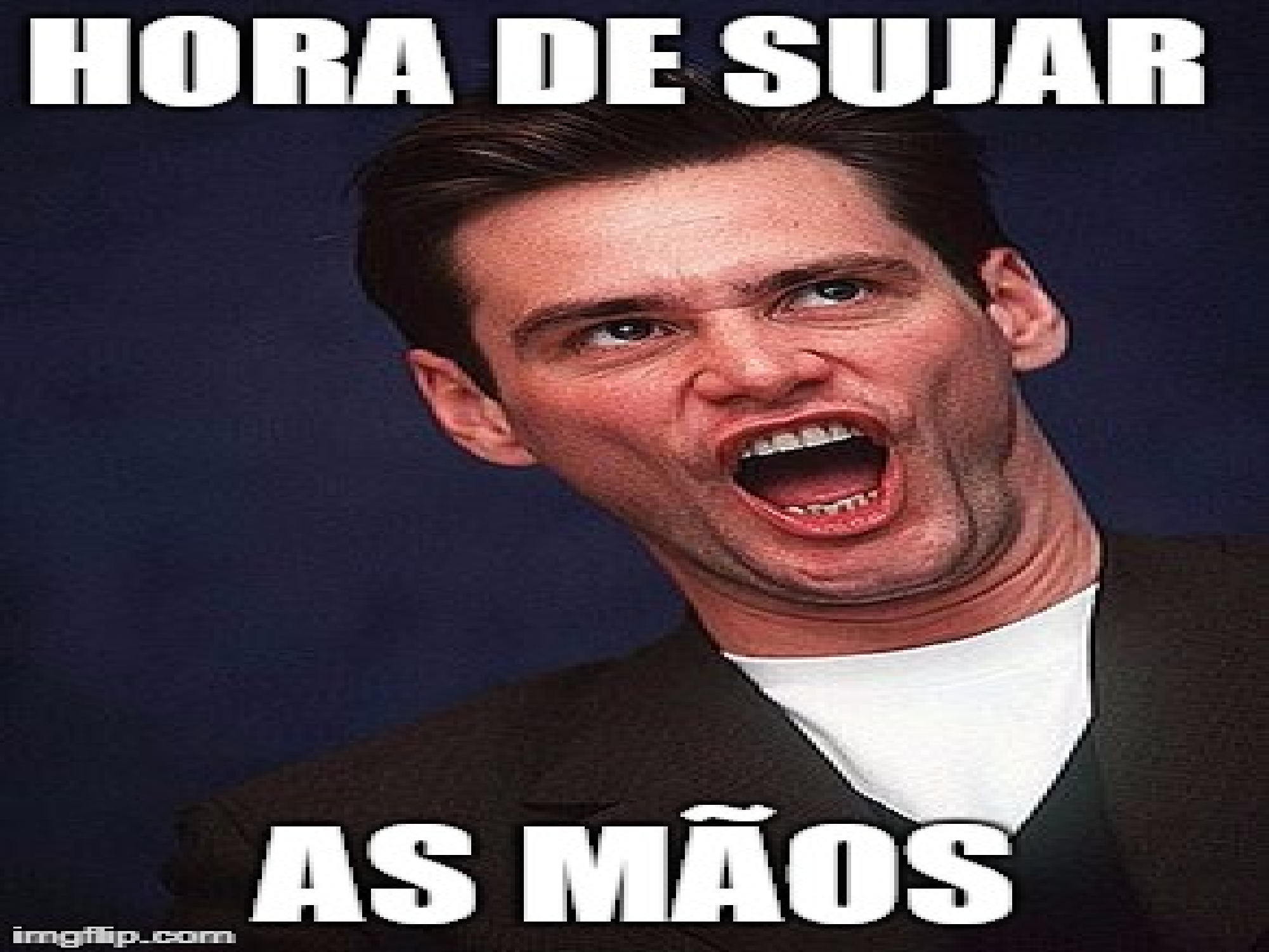
```
pib_real <- get_series(code = 7326,
                       last = 15,
                       name = 'var_real',
                       as = 'tibble')
```

Apresentação do rbcb

Exemplo 2: variação real do PIB dos últios 15 anos

```
ggplot(pib_real,aes(x=date,y=var_real))+geom_line()+
labs(x='',y='Var %',title='PIB real')+
geom_hline(yintercept = 0,color='red')
```




HORA DE SUJAR
AS MÃOS
imgflip.com

Atividade Prática 2

Contextualização

Na condição de analista macroeconômico da *We the Analitics* recebeu a função de medir a volatilidade dos grupos que compõem o IPCA em relação ao IPCA para o período entre 01/01/2002 até 01/06/2017. Porém para que a análise possa ser feita é necessário que você juntos os dados.

Atividade Prática 2

Estrutura da atividade

1. criar um vetor com os números das séries do IPCA e seus componentes de acordo com o sistema do Bacen e um vetor com os nomes do IPCA e seus componentes. Obs: **É importante que a ordem seja a mesma no vetor dos números e dos nomes;**

Atividade Prática 2

Estrutura da atividade

1. criar um vetor com os números das séries do IPCA e seus componentes de acordo com o sistema do Bacen e um vetor com os nomes do IPCA e seus componentes. Obs: **É importante que a ordem seja a mesma no vetor dos números e dos nomes;**
2. Criar matrix (e transformar em data.frame) com o nome de "ipca_por_grupos",que receberá os valores das séries, e já popular a primeira coluna com a data;

Atividade Prática 2

Estrutura da atividade

1. criar um vetor com os números das séries do IPCA e seus componentes de acordo com o sistema do Bacen e um vetor com os nomes do IPCA e seus componentes. Obs: **É importante que a ordem seja a mesma no vetor dos números e dos nomes;**
2. Criar matrix (e transformar em data.frame) com o nome de "ipca_por_grupos",que receberá os valores das séries, e já popular a primeira coluna com a data;
3. Utilizar a função **get_series()** dentro de uma estrutura de loop para pegar cada uma das séries. Quando feito isso salvar o data.frame através da função **write.csv()** para que ele possa ser utilizado depois;

Atividade Prática 2

Estrutura da atividade

1. criar um vetor com os números das séries do IPCA e seus componentes de acordo com o sistema do Bacen e um vetor com os nomes do IPCA e seus componentes. Obs: **É importante que a ordem seja a mesma no vetor dos números e dos nomes;**
2. Criar matrix (e transformar em data.frame) com o nome de "ipca_por_grupos",que receberá os valores das séries, e já popular a primeira coluna com a data;
3. Utilizar a função **get_series()** dentro de uma estrutura de loop para pegar cada uma das séries. Quando feito isso salvar o data.frame através da função **write.csv()** para que ele possa ser utilizado depois;
4. criar um data.frame (através da função **gather()**) chamado de "ipca_por_grupos_plot" que esteja de acordo com o principio de *tidy data* e criar um plot com linhas de cores diferentes para cada um dos grupos;

Atividade Prática 2

Estrutura da atividade

1. criar um vetor com os números das séries do IPCA e seus componentes de acordo com o sistema do Bacen e um vetor com os nomes do IPCA e seus componentes. Obs: **É importante que a ordem seja a mesma no vetor dos números e dos nomes;**
2. Criar matrix (e transformar em data.frame) com o nome de "ipca_por_grupos",que receberá os valores das séries, e já popular a primeira coluna com a data;
3. Utilizar a função **get_series()** dentro de uma estrutura de loop para pegar cada uma das séries. Quando feito isso salvar o data.frame através da função **write.csv()** para que ele possa ser utilizado depois;
4. criar um data.frame (através da função **gather()**) chamado de "ipca_por_grupos_plot" que esteja de acordo com o principio de *tidy data* e criar um plot com linhas de cores diferentes para cada um dos grupos;
5. Estimar a volatilidade dos grupos em relação ao IPCA (próxima parte).


That's all Folks!