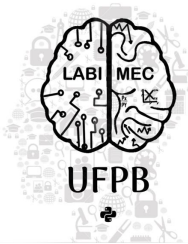





# Curso de programação em R - LABIMEC (2ª parte)

Pedro Milreu Cunha

Universidade Federal da Paraíba (UFPB) - PPGE/UFPB & LABIMEC

06-12-2022

# Revisão

# Curso do prof. Paulo no dia 29/11/2022

1. R, RStudio e instalação de pacotes;

`install.packages()`, `library()`, ...

2. Importação de dados;

`read.csv()`, `read.delim()`, `read.table()`, ...

3. Limpeza e manipulação de dados;

`%>%`, `filter()`, `select()`, `mutate()`, ...

4. Análise de dados;

`group_by()`, `summarise()`, `ggplot()`, ...

# Configurações necessárias para essa aula

# Configurações necessárias para essa aula

```r
# Bibliotecas ----
library(dplyr) # Manipulação de dados
library(readxl) # Importação de arquivos .xlsx

library(ggplot2) # Criação de gráficos
library(ggpubr) # Layout de múltiplos gráficos
library(sf) # Importação e manipulação de dados geográficos (.shp)
library(tmap) # Criação de choropleth maps
library(RColorBrewer) # Paleta de cores

library(plm) # Dados em painel
library(stargazer) # Tabelas bem formatadas

library(showtext)
font_add_google("EB Garamond", "Garamond")
showtext_auto()

# Comando especial ----
`%notin%` <- Negate(`%in%`)

# Função para renderizar figuras em .svg corretamente no LaTeX
show_fig <- function(f) # Fonte: https://stackoverflow.com/a/56044642
  {if (knitr::is_latex_output())
  {
    output = xfun::with_ext(f, 'pdf')
    rsvg::rsvg_pdf(xfun::with_ext(f,'svg'), file = output)
  } else {
    output = xfun::with_ext(f, 'svg')
  }
  knitr::include_graphics(output)
}
```

# Criação de mapas

# Elaboração de mapas utilizando R

- Os mapas são elaborados a partir de polígonos com coordenadas geográficas, permitindo uma visualização espacial dos dados;
- Em particular, têm-se os chamados *choropleth maps*, que utilizam escalas de cores para demonstrar a distribuição espacial de alguma variável de interesse.
- Há várias bibliotecas disponíveis para esse fim: 'spplot', 'tmap' e 'ggplot2' são algumas opções. **Faremos uso do pacote `tmap` nesse curso**.

# Exemplo de mapa elaborado utilizando a biblioteca `tmap`

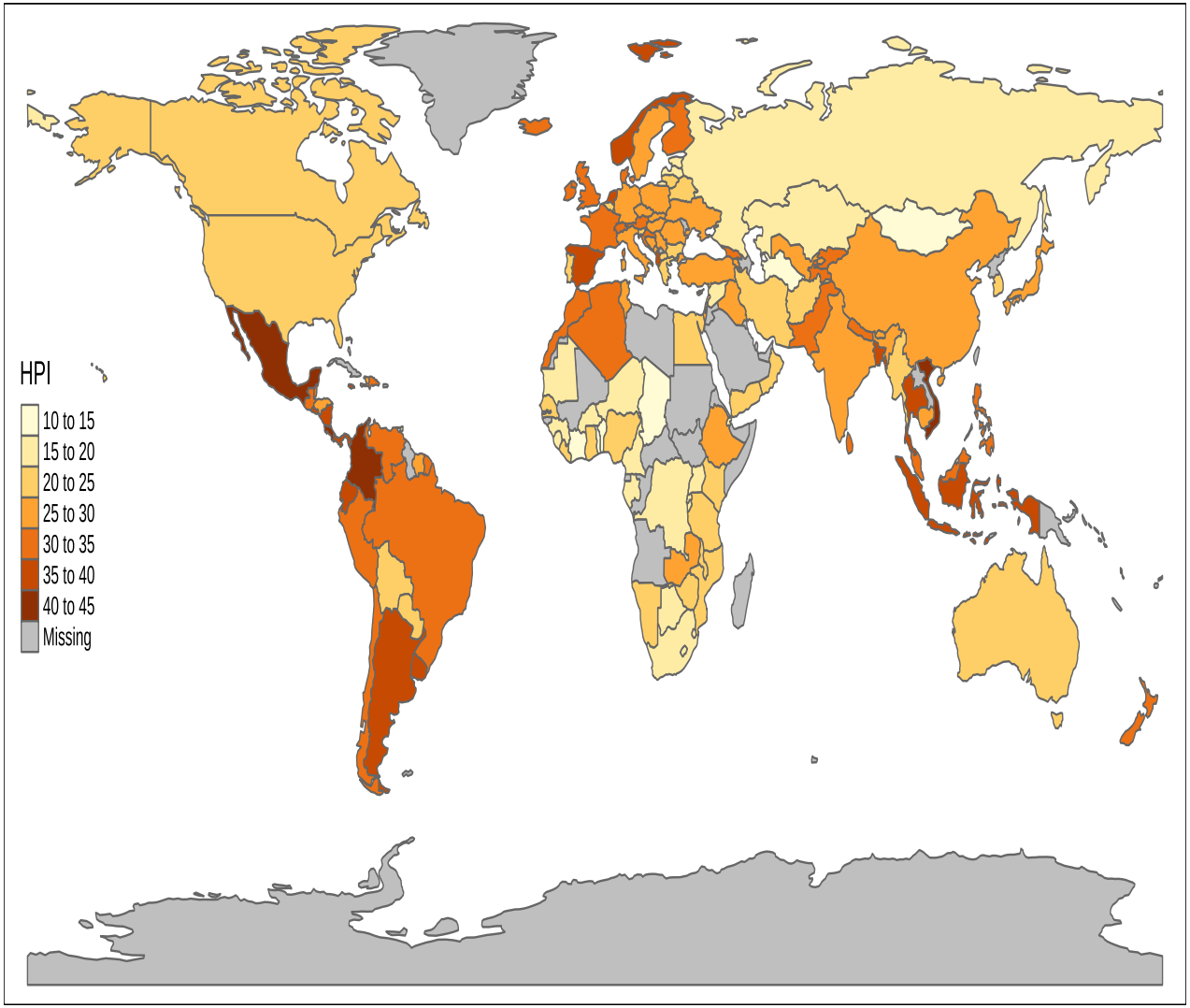



**Figura 1:** Exemplo de mapa criado pela biblioteca 'tmap'.

# O que é necessário para gerar um mapa utilizando R?

1. **Shapefiles**:

as informações absolutamente *necessárias* para criação de mapas são aquelas referentes à geografia e polígonos, ou seja, os valores que mostram como os dados se distribuem espacialmente. São obtidos com os `shapefiles`[1], conjunto de arquivos com as extensões *.cpg, .dbf, .prj, .shp, .shx*;

2. Variáveis distribuídas de acordo com o nível geográfico utilizado para criação do mapa:

dados referentes aos fenômenos cuja distribuição espacial queremos analisar, como, por exemplo: i) a população ou PIB per capita para cada UF brasileira; ii) o nível de abertura comercial para cada país do Mercosul, etc.

[1] Esse não é o único formato possível de representação de dados geográficos.

# Onde obter os *shapefiles* para o Brasil?

- Os *shapefiles* para o Brasil são disponibilizados pelo *Instituto Brasileiro de Geografia e Estatística (IBGE)* em seu site: IBGE;
- É possível obter as malhas territoriais de municípios, UFs, micro e mesorregiões, regiões geográficas imediatas e intermediárias e do país como um todo;
- Para que os arquivos sejam importados corretamente é necessário que todos os arquivos extraídos após download estejam em uma mesma pasta. No caso da malha territorial por estado, tem-se o arquivo comprimido `BR_UF_2021.zip` com os seguintes arquivos dentro: `BR_UF_2021.cpg`, `BR_UF_2021.dbf`, `BR_UF_2021.prj`, `BR_UF_2021.shp`, `BR_UF_2021.shx`.

# Criando um mapa com o pacote `tmap`

***Manipulação dos dados***

1. O primeiro passo é importar a geometria base do mapa. Para tanto, utilizamos o arquivo `BR_UF_2021.shp` e a função `st_read()` da biblioteca `sf`. Além disso, também criamos uma coluna com as siglas das UFs brasileiras no arquivo importado.

```
dados_geo <- st_read("data/shapefiles/BR_UF_2021.shp")
dados_geo$Sigla_UF <- c("RO", "AC", "AM", "RR",
                        "PA", "AP", "TO", "MA",
                        "PI", "CE", "RN", "PB",
                        "PE", "AL", "SE", "BA",
                        "MG", "ES", "RJ", "SP",
                        "PR", "SC", "RS", "MS",
                        "MT", "GO", "DF")
# Passo necessário para a posterior união dos dados
```

# Criando um mapa com o pacote `tmap`

2. Em seguida, lemos o arquivo com os dados cuja distribuição espacial queremos analisar e realizamos as transformações necessárias (nesse caso adicionar as siglas dos estados ausentes, dando o valor `NaN` para as observações em questão). Nesse exemplo vamos utilizar os dados referentes à subnotificação de estupros disponíveis no arquivo `data/sub_estimadas_estupros.xlsx`.

```
dados_sub <- read_excel("data/sub_estimadas_estupros.xlsx") %>%
  select(Sigla_UF, sub_lag_end) %>%
  group_by(Sigla_UF) %>%
  mutate(sub_lag_end = mean(sub_lag_end, na.rm = TRUE)) %>%
  unique()

for (uf in dados_geo$Sigla_UF) { # Adicionar as siglas ausentes
    if (uf %notin% dados_sub$Sigla_UF) {
        dados_sub[nrow(dados_sub) + 1, ] <- list(uf, NaN)
    }
}
```

# Criando um mapa com o pacote `tmap`

3. Para finalizar a manipulação dos dados, unimos as duas bases com o comando `left_join`.

```
dados <- dados_geo %>%
         left_join(dados_sub, by = "Sigla_UF")
str(dados)
```

```
## Classes 'sf' and 'data.frame':   27 obs. of  7 variables:
##  $ CD_UF      : chr  "11" "12" "13" "14" ...
##  $ NM_UF      : chr  "Rondônia" "Acre" "Amazonas" "Roraima" ...
##  $ SIGLA      : chr  "RO" "AC" "AM" "RR" ...
##  $ NM_REGIAO  : chr  "Norte" "Norte" "Norte" "Norte" ...
##  $ Sigla_UF   : chr  "RO" "AC" "AM" "RR" ...
##  $ sub_lag_end: num  0.0175 0.4131 0.3582 0.4839 0.0383 ...
##  $ geometry   :sfc_MULTIPOLYGON of length 27; first list element: List of 1
##   ..$ :List of 1
##   .. ..$ : num [1:8812, 1:2] -62.9 -62.9 -62.9 -62.8 -62.8 ...
##   ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA
##   ..- attr(*, "names")= chr [1:6] "CD_UF" "NM_UF" "SIGLA" "NM_REGIAO" ...
```

***Visualização dos resultados***

- Utilizamos três comandos básicos para criar um *choropleth map*: `tm_shape()`, `tm_borders()` e `tm_fill()`. O primeiro deles cria a base geográfica do mapa, o segundo gera as bordas entre os entes representados e o terceiro preenche o mapa de acordo com o a distribuição de valores de alguma variável. Para mais informações sobre cada um deles consulte `?tm_shape`, `?tm_borders`, `?tm_fill`.

```
mapa_inicial <- tm_shape(dados) +
                tm_borders() +
                tm_fill("sub_lag_end")
```

O resultado obtido com os parâmetros padrões das funções pode ser visto no próximo slide.

- Para além dos funções já apresentadas e seus parâmetros, destacam-se `?tm_credits`, `?tm_logo`, `?tm_text`, `?tm_layout`, com as três primeiras servindo para adicionar créditos (ou fonte), logos e textos ao mapa e a última servindo para customizar o «layout da imagem».[2]

[2] Por «layout da imagem» entenda-se desde a posição da legenda ou título até o tamanho e família das fontes utilizadas.

# Criando um mapa com o pacote `tmap`

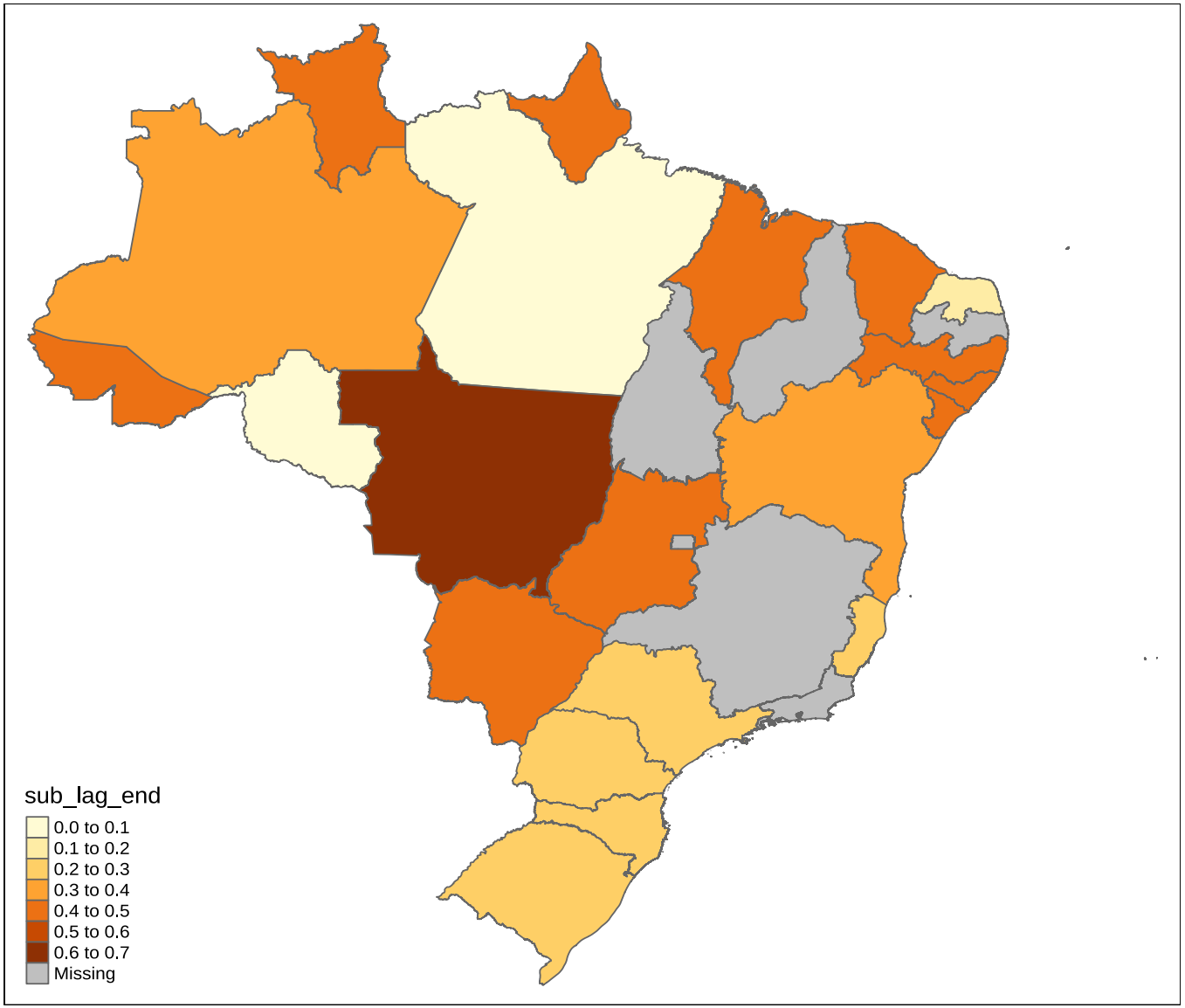



**Figura 2:** Mapa inicial.

# Customizando um mapa com o pacote `tmap`

Visando melhorar o mapa inicial que geramos anteriormente, iremos:

1. Afinar as bordas entre os polígonos:

```
tm_borders(lwd = 0.55)
```

2. Remover o título da legenda e definir seus valores, mudar a paleta de cores e associar uma cor e rótulo específicos para os valores ausentes:

```
tm_fill("sub_lag_end", title = "", textNA = "Dado ausente",
        colorNA = "lightgrey", palette = "GnBu",
        labels = c("< 10%", "10% - 20%",
                   "20% - 30%", "30% - 40%",
                   "40% - 50%", "50% - 60%",
                   "60% - 70%"))
```

# Customizando um mapa com o pacote `tmap`

3. Identificar cada UF no mapa com a sua respectiva sigla:

```
tm_text("Sigla_UF", size = 0.65, fontfamily = "serif",
        fontface = "bold", col = "black")
```

4. Adicionar uma nota e a fonte dos dados ao mapa:

```
tm_credits("Nota: Média do período jan/2012 - dez/2020.",
           fontface = "italic", size = 0.7,
           align = "right")
```

5. Adicionar um título principal ao gráfico, remover o "quadro" envolta do mapa e formatar a fonte da legenda e título:

```
tm_layout(main.title = paste("Distribuição espacial da
subnotificação", "de estupros de mulheres no Brasil",
          sep = "\n"),
          title.size = 1, main.title.position = "center",
          fontfamily = "serif", main.title.fontface = "italic",
          scale = 1, frame = FALSE,
          legend.title.size = 1.5, legend.text.size = 0.8,
          legend.outside.position = c("left", "bottom"))
```

# Customizando um mapa com o pacote `tmap`

- Juntando todos os passos e salvando o mapa ao fim do processo com a função `tmap_save()`, temos:

```
mapa <- tm_shape(dados) +
            tm_borders(lwd = 0.55) +
            tm_fill("sub_lag_end",
                    title = "",
                    textNA = "Dado ausente",
                    colorNA = "lightgrey", palette = "GnBu",
                    labels = c("< 10%", "10% - 20%",
                               "20% - 30%", "30% - 40%",
                               "40% - 50%", "50% - 60%",
                               "60% - 70%")) +
            tm_text("Sigla_UF", size = 0.65, fontfamily = "serif",
                    fontface = "bold", col = "black") +
            tm_credits("Nota: Média do período jan/2012 - dez/2020.",
                       fontface = "italic", size = 0.7,
                       align = "right") +
            tm_credits("Fonte: Elaboração própria a partir dos dados das SSPs.",
                       size = 0.8, align = "right") +
            tm_layout(main.title = paste("Distribuição espacial da subnotificação",
                                         "de estupros de mulheres no Brasil",
                                         sep = "\n"),
                      title.size = 1,
                      main.title.position = "center", fontfamily = "serif",
                      main.title.fontface = "italic", scale = 1, frame = FALSE,
                      legend.title.size = 1.5, legend.text.size = 0.8,
                      legend.outside.position = c("left", "bottom"))

#tmap_save(mapa, "figures/mapa_sub_estupros.svg",
#          width = 12, height = 12, units = "cm")
```

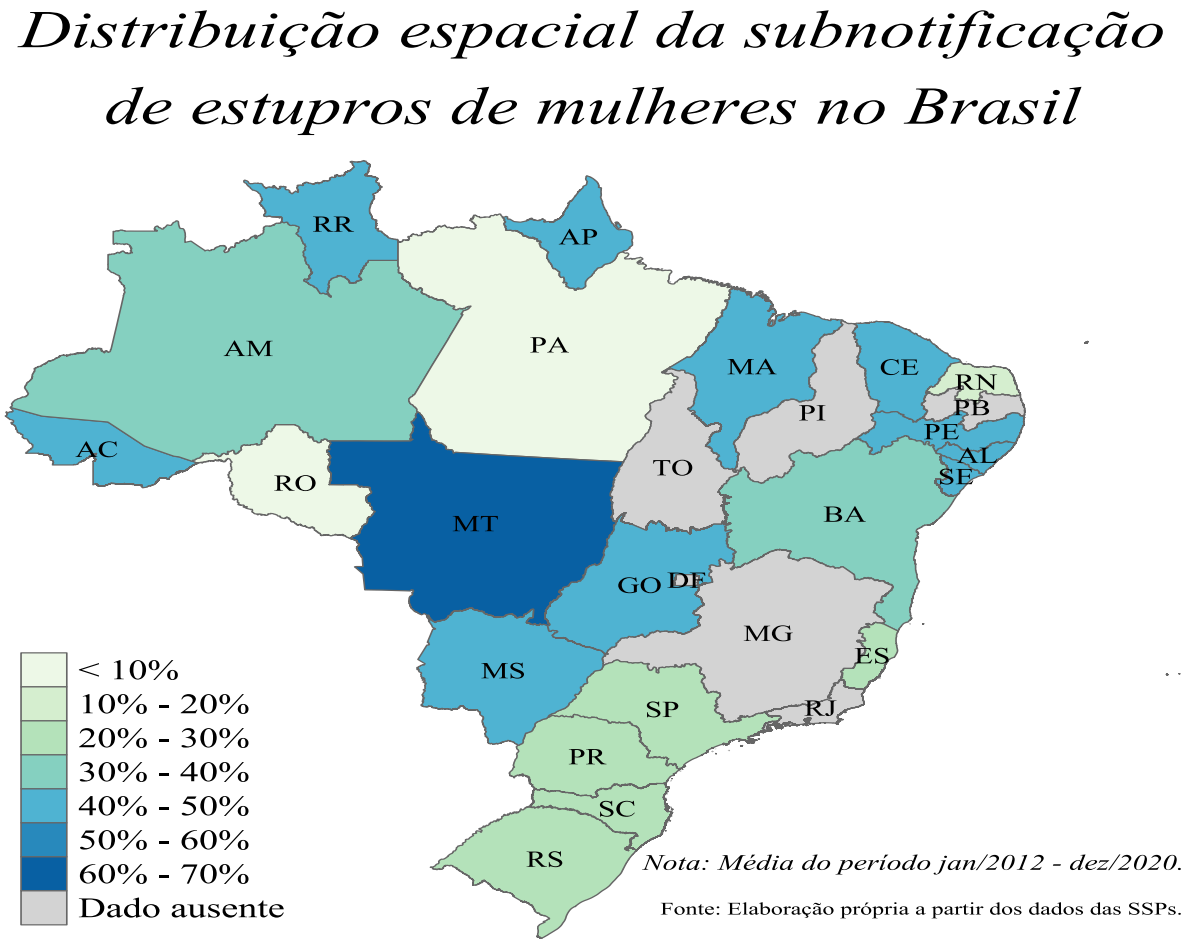



**Figura 3:** Mapa customizado.

# Pacote `stargazer`

# Introdução

- O `stargazer` é um famoso pacote da linguagem R que é utilizado para gerar tabelas bem formatadas contendo resultados de regressões, `data.frames` ou estatísticas descritivas;
- Funciona com regressões estimadas por diversos pacotes, como, por exemplo, modelos lineares (`?lm`), modelos lineares generalizados (`?glm`) e modelos para dados em painel (`?plm`);
- A biblioteca gera tabelas em três formatos: `ASCII` (.txt), `HTML` (.html) e LaTeX(.pdf); iremos focar apenas em `ASCII` e LaTeX nesse curso.

# Utilização

- Toda a funcionalidade do pacote se dá através da função `stargazer`, cujo funcionamento pode ser consultado em `?stargazer::stargazer`. Tem-se abaixo os parâmetros que a função aceita, bem como os valores padrões de cada um.

```
function (..., type = "latex", title = "", style = "default",
    summary = NULL, out = NULL, out.header = FALSE, column.labels = NULL,
    column.separate = NULL, covariate.labels = NULL, dep.var.caption = NULL,
    dep.var.labels = NULL, dep.var.labels.include = TRUE, align = FALSE,
    coef = NULL, se = NULL, t = NULL, p = NULL, t.auto = TRUE,
    p.auto = TRUE, ci = FALSE, ci.custom = NULL, ci.level = 0.95,
    ci.separator = NULL, add.lines = NULL, apply.coef = NULL,
    apply.se = NULL, apply.t = NULL, apply.p = NULL, apply.ci = NULL,
    colnames = NULL, column.sep.width = "5pt", decimal.mark = NULL,
    df = TRUE, digit.separate = NULL, digit.separator = NULL,
    digits = NULL, digits.extra = NULL, flip = FALSE, float = TRUE,
    float.env = "table", font.size = NULL, header = TRUE, initial.zero = NULL,
    intercept.bottom = TRUE, intercept.top = FALSE, keep = NULL,
    keep.stat = NULL, label = "", model.names = NULL, model.numbers = NULL,
    multicolumn = TRUE, no.space = NULL, notes = NULL, notes.align = NULL,
    notes.append = TRUE, notes.label = NULL, object.names = FALSE,
    omit = NULL, omit.labels = NULL, omit.stat = NULL, omit.summary.stat = NULL,
    omit.table.layout = NULL, omit.yes.no = c("Yes", "No"), order = NULL,
    ord.intercepts = FALSE, perl = FALSE, report = NULL, rownames = NULL,
    rq.se = "nid", selection.equation = FALSE, single.row = FALSE,
    star.char = NULL, star.cutoffs = NULL, suppress.errors = FALSE,
    table.layout = NULL, table.placement = "!htbp", zero.component = FALSE,
    summary.logical = TRUE, summary.stat = NULL, nobs = TRUE,
    mean.sd = TRUE, min.max = TRUE, median = FALSE, iqr = FALSE)
```

# Exemplos

- Vamos analisar dois usos mais comuns do pacote `stargazer`:
  1. Visualização dos resultados de análises descritivas;
  2. Visualização dos resultados de estimações de modelos econométricos:

     2.1 Caso mais simples com apenas um modelo e sem customização da tabela;

     2.2 Caso mais complexo com vários modelos e customização da tabela;
- Para isso iremos importar o conjunto de dados `data/analise_enade.Rdata` que contém informações sobre o rendimento de discentes universitários no ENADE:

```
load("data/analise_enade.Rdata")
```

# Exemplos em ASCII (.txt)

- Vamos agora criar uma tabela em ASCII[3] contendo as estatísticas descritivas do `data.frame` *dados_enade* que importamos anteriormente. O código necessário para isso, bem como o resultado obtido podem ser vistos a seguir:

```
stargazer(dados_enade, type = "text")
```

```
##
## ===========================================================
## Statistic            N    Mean    St. Dev.    Min      Max
## -----------------------------------------------------------
## CO_IES             510  977.176  1,307.852    1      5,322
## Ano                510 2,011.500   1.710    2,009    2,014
## idade_media        510  25.630     2.760   18.143   40.271
## idade2_media       510  664.499   150.668  329.163 1,621.748
## nota_media         510  45.971     6.282    2.621   65.930
## prop_brancos       510   0.512     0.212    0.000    0.915
## prop_casados       510   0.132     0.094    0.000    0.542
## prop_homens        510   0.418     0.111    0.077    0.795
## prop_medio_ou_mais 510   0.611     0.153    0.113    0.947
## prop_renda_3SM     510   0.311     0.210    0.000    1.000
## Tempo              510   0.500     0.500      0        1
## log_nota           510   3.815     0.185    0.963    4.189
## -----------------------------------------------------------
```

[3] Esse formato é útil para sessões interativas no R ou estágios iniciais de estimações, momentos nos quais queremos uma visualização rápida e bem formatada das informações presentes nos dados.

# Exemplos em ASCII (.txt)

- Agora um exemplo clássico de formatação de tabelas simples de regressão:

```
stargazer(didreg, type = "text", single.row = TRUE)
```

```
##
## ================================================
##                          Dependent variable:
##                      ---------------------------
##                                log_nota
## ------------------------------------------------
## Tempo                        -0.020 (0.079)
## Tratamento1                  -0.028 (0.054)
## idade_media                 0.315*** (0.099)
## idade2_media                -0.005*** (0.002)
## prop_brancos                  0.060 (0.103)
## prop_casados                -1.427** (0.586)
## prop_homens                  -0.004 (0.275)
## prop_medio_ou_mais           -0.270 (0.262)
## prop_renda_3SM              -0.599*** (0.216)
## Tempo:Tratamento1             0.010 (0.075)
## Constant                     -0.193 (1.375)
## ------------------------------------------------
## Observations                    170
## R2                             0.183
## Adjusted R2                    0.132
## Residual Std. Error       0.239 (df = 159)
## F Statistic           3.563*** (df = 10; 159)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

# Exemplos em ASCII (.txt)

- Por fim, temos agora o exemplo mais complexo. Utilizaremos o seguinte código:

```
stargazer(didreg_fe, didreg_fe_logit, didreg_fe_entropia,
          type = "text", single.row = TRUE,
          title = "Efeito médio de tratamento sobre os tratados - Ciclo 2009-2012",
          decimal.mark = ",", digits = 5,
          column.labels = c("Diff-diff + FE",
                            "Diff-diff + FE + PSM (logit)",
                            "Diff-diff + FE + PSM (entropia)"),
          covariate.labels = c("Tempo", "Idade média", "Idade média²",
                               "Prop. brancos", "Prop. casados", "Prop. homens",
                               "Prop. ensino medio ou mais", "Prop. renda per capita 3S.M.",
                               "Tempo x Tratamento"),
          model.numbers = FALSE,
          initial.zero = TRUE,
          dep.var.labels = "ln(Nota média)", dep.var.caption = "Variável dependente",
          multicolumn = TRUE, model.names = FALSE, header = FALSE)
```

O resultado pode ser visto no próximo slide.

# Exemplo em ASCII (.txt)

```
##
## Efeito médio de tratamento sobre os tratados - Ciclo 2009-2012
## ================================================================================================================
##                                                          Variável dependente
##                                 ---------------------------------------------------------------------------------
##                                                              ln(Nota média)
##                                   Diff-diff + FE     Diff-diff + FE + PSM (logit) Diff-diff + FE + PSM (entropia)
## ----------------------------------------------------------------------------------------------------------------
## Tempo                            -0,02349 (0,09032)       -0,00817 (0,08257)           -0,03741 (0,08719)
## Idade média                     0,60611*** (0,14819)     0,73706*** (0,17083)         0,71491*** (0,16910)
## Idade média²                    -0,00987*** (0,00289)    -0,01299*** (0,00358)        -0,01234*** (0,00358)
## Prop. brancos                   -3,10625*** (0,66278)    -3,36986*** (0,70896)        -3,52649*** (0,71394)
## Prop. casados                   -2,83638*** (0,96541)     -2,07438** (1,01842)        -2,44368** (1,06373)
## Prop. homens                      0,47486 (0,39295)        0,20124 (0,41804)            0,24595 (0,41051)
## Prop. ensino medio ou mais       -0,05687 (0,56586)       -0,27292 (0,55062)           -0,21725 (0,55964)
## Prop. renda per capita 3S.M. -1,11261*** (0,30127)    -1,30979*** (0,28521)        -1,23628*** (0,28013)
## Tempo x Tratamento               -0,04373 (0,05681)       0,00191 (0,05225)            0,00429 (0,05363)
## ----------------------------------------------------------------------------------------------------------------
## Observations                              170                      170                          170
## R2                                      0,50470                  0,48109                      0,48879
## Adjusted R2                            -0,10138                 -0,15389                     -0,13676
## F Statistic (df = 9; 76)             8,60480***               9,29616***                   9,49595***
## ================================================================================================================
## Note:                                                                        *p<0,1; **p<0,05; ***p<0,01
```

# Exemplos em LaTeX(.pdf)

- Vamos agora recriar o exemplo das estatísticas descritivas, porém utilizando o formato LaTeX.

```
stargazer(dados_enade, type = "latex", header = FALSE)
```

**Tabela 1:**

| Statistic | N | Mean | St. Dev. | Min | Max |
|---|---|---|---|---|---|
| CO_IES | 510 | 977.176 | 1,307.852 | 1 | 5,322 |
| Ano | 510 | 2,011.500 | 1.710 | 2,009 | 2,014 |
| idade_media | 510 | 25.630 | 2.760 | 18.143 | 40.271 |
| idade2_media | 510 | 664.499 | 150.668 | 329.163 | 1,621.748 |
| nota_media | 510 | 45.971 | 6.282 | 2.621 | 65.930 |
| prop_brancos | 510 | 0.512 | 0.212 | 0.000 | 0.915 |
| prop_casados | 510 | 0.132 | 0.094 | 0.000 | 0.542 |
| prop_homens | 510 | 0.418 | 0.111 | 0.077 | 0.795 |
| prop_medio_ou_mais | 510 | 0.611 | 0.153 | 0.113 | 0.947 |
| prop_renda_3SM | 510 | 0.311 | 0.210 | 0.000 | 1.000 |
| Tempo | 510 | 0.500 | 0.500 | 0 | 1 |
| log_nota | 510 | 3.815 | 0.185 | 0.963 | 4.189 |

- Em seguida, reproduzimos o exemplo de uma regressão simples:

```
stargazer(didreg, type = "latex", header = FALSE, single.row = TRUE)
```

**Tabela 2:**

| | *Dependent variable:* |
|---|---|
| | log_nota |
| Tempo | −0.020 (0.079) |
| Tratamento1 | −0.028 (0.054) |
| idade_media | 0.315*** (0.099) |
| idade2_media | −0.005*** (0.002) |
| prop_brancos | 0.060 (0.103) |
| prop_casados | −1.427** (0.586) |
| prop_homens | −0.004 (0.275) |
| prop_medio_ou_mais | −0.270 (0.262) |
| prop_renda_3SM | −0.599*** (0.216) |
| Tempo:Tratamento1 | 0.010 (0.075) |
| Constant | −0.193 (1.375) |
| Observations | 170 |
| $R^2$ | 0.183 |
| Adjusted $R^2$ | 0.132 |
| Residual Std. Error | 0.239 (df = 159) |
| F Statistic | 3.563*** (df = 10; 159) |
| *Note:* | *p<0.1; **p<0.05; ***p<0.01 |

# Exemplos em LaTeX(.pdf)

- Por fim, com o código abaixo geramos uma tabela complexa, com diversos modelos e bem formatada.

```
stargazer(didreg_fe, didreg_fe_logit, didreg_fe_entropia,
          type = "latex",
          title = "Efeito médio de tratamento sobre os tratados - Ciclo 2009-2012",
          decimal.mark = ",", digits = 5,
          column.labels = c("Diff-diff + FE",
                            "Diff-diff + FE + PSM (logit)",
                            "Diff-diff + FE + PSM (entropia)"),
          covariate.labels = c("Tempo", "Idade média", "Idade média²",
                               "Prop. brancos", "Prop. casados", "Prop. homens",
                               "Prop. ensino medio ou mais", "Prop. renda per capita 3S.M.",
                               "Tempo x Tratamento"),
          model.numbers = FALSE,
          initial.zero = TRUE,
          dep.var.labels = "ln(Nota média)", dep.var.caption = "Variável dependente",
          multicolumn = TRUE, model.names = FALSE, align = TRUE, header = FALSE,
          table.placement = "H", no.space = TRUE)
```

- É importante destacar que alguns itens específicos não podem ser customizados diretamente através do `stargazer`, como, por exemplo, os rótulos *Observations* e *Adjusted R2*, que não conseguimos traduzir para o português. Apesar disso, uma vez que a função `stargazer::stargazer()` retorna o código em LaTeX necessário para criar a(s) tabela(s) escolhida(s) não é difícil customizar o resultado final.

**Tabela 3:** Efeito médio de tratamento sobre os tratados - Ciclo 2009-2012

| | Variável dependente | | |
|---|---|---|---|
| | ln(Nota média) | | |
| | Diff-diff + FE | Diff-diff + FE + PSM (logit) | Diff-diff + FE + PSM (entropia) |
| Tempo | −0,02349 | −0,00817 | −0,03741 |
| | (0,09032) | (0,08257) | (0,08719) |
| Idade média | 0,60611*** | 0,73706*** | 0,71491*** |
| | (0,14819) | (0,17083) | (0,16910) |
| Idade média$^2$ | −0,00987*** | −0,01299*** | −0,01234*** |
| | (0,00289) | (0,00358) | (0,00358) |
| Prop. brancos | −3,10625*** | −3,36986*** | −3,52649*** |
| | (0,66278) | (0,70896) | (0,71394) |
| Prop. casados | −2,83638*** | −2,07438** | −2,44368** |
| | (0,96541) | (1,01842) | (1,06373) |
| Prop. homens | 0,47486 | 0,20124 | 0,24595 |
| | (0,39295) | (0,41804) | (0,41051) |
| Prop. ensino medio ou mais | −0,05687 | −0,27292 | −0,21725 |
| | (0,56586) | (0,55062) | (0,55964) |
| Prop. renda per capita 3S.M. | −1,11261*** | −1,30979*** | −1,23628*** |
| | (0,30127) | (0,28521) | (0,28013) |
| Tempo x Tratamento | −0,04373 | 0,00191 | 0,00429 |
| | (0,05681) | (0,05225) | (0,05363) |
| Observations | 170 | 170 | 170 |
| $R^2$ | 0,50470 | 0,48109 | 0,48879 |
| Adjusted $R^2$ | -0,10138 | -0,15389 | -0,13676 |
| F Statistic (df = 9; 76) | 8,60480*** | 9,29616*** | 9,49595*** |

*Note:* *p<0,1; **p<0,05; ***p<0,01

# Funcionalidades avançadas

- Dentre as funcionalidades mais avançadas do `stargazer` não utilizadas aqui, destacam-se:
  1. Utilização de coeficientes, erros-padrões, estatísticas-t e p-valores customizados com os parâmetros `coef, se, t, p`, respectivamente;
  2. Substituição dos erros-padrões por intervalos de confiança (parâmetros `ci, ci.custom, ci.level, ci.separator`);
  3. Aplicação de funções nos coeficientes, erros-padrões, estatísticas-t, intervalos de confiança e p-valores (parâmetros `apply.coef, apply.se, apply.t, apply.ci, apply.p`);
  4. Exportar o resultado do comando diretamente para um arquivo `.txt` com o parâmetro `out`.

# Alternativas ao `stargazer`

- Embora ainda seja uma das **referências** no ecossistema do R no que diz respeito a exportação de resultados em tabelas formatadas, especialmente utilizando LaTeX, o `stargazer` tem problemas e não é compatível com todos os tipos de modelos que podem ser gerados com os diversos pacotes existentes na linguagem;
- Boas alternativas existem e é importante procurar qual melhor se adequa em cada ocasião. Algumas opções são:
  1. gtsummary;
  2. flextable;
  3. etable para modelos de efeitos fixos estimados com o pacote `fixest`;
  4. xtable.

# Funções

# O que são funções na linguagem R?

- As funções na linguagem R seguem o mesmo princípio de funções matemáticas, ou seja, recebem uma entrada, realizam alguma operação com ela e retornam algo.
  - Uma diferença importante é que as funções na linguagem R não precisam, **necessáriamente**, retornar alguma coisa ou mesmo receber uma entrada. Por exemplo:

```
saudar <- function() {

  print("Saudações, usuário!")

}

saudar()
```

```
## [1] "Saudações, usuário!"
```

# Como criar funções e quando utilizá-las?

- Criamos as funções como quaisquer outros objetos da linguagem, ou seja, a associando a algum nome com o operador <-. Para tanto tem-se o comando `function(argumentos)` onde `argumentos` são os parâmetros passados à função (vazio caso a função não receba parâmetros).
- É interessante utilizar as funções quando temos algum cálculo ou processamento que queremos replicar diversas vezes ou quando queremos separar as partes do código em pequenas unidades que realizam cada uma um conjunto de tarefas, contribuindo para uma lógica de programação mais fluida e mais simples de se entender.

# Exemplo básico - criação da função

- Função para calcular o custo do consumo energético de um local

```r
custo_energia <- function(tipo_local, consumo_kwh, atraso = 0, desconto = 0) {

  # ENTRADA:
  # tipo_local: string indicando se o local em questão é residencial, comercial ou industrial;
  # consumo_kwh: float com o consumo de energia do local em kW/h;
  # atraso: float com o percentual a ser cobrado por atraso (padrão de 0%);
  # desconto: float com o percentual de desconto a ser aplicado (padrão de 0%);

  # SAÍDA:
  # Não retorna nada.

  custo_kwh <- ifelse(tolower(tipo_local) == "residencial",
              0.72, ifelse(tolower(tipo_local) == "comercial",
              0.85, 1.07))
  total = custo_kwh*consumo_kwh*(1 + (atraso/100) - (desconto)/100)

  print(paste0("Consumo: ", consumo_kwh, "kW/h"))
  print(paste0("Tipo de local: ", tipo_local))
  print(paste0("Custo por atraso: R$", (atraso/100)*custo_kwh*consumo_kwh))
  print(paste0("Desconto: R$", (desconto/100)*custo_kwh*consumo_kwh))
  print(paste0("Custo do kW/h: R$", custo_kwh))
  print(paste0("Total a ser pago: R$", total))

}
```

# Exemplo básico - resultados

- Os resultados podem ser vistos a seguir.

```
custo_energia("Residencial", 412)
```

```
## [1] "Consumo: 412kW/h"
## [1] "Tipo de local: Residencial"
## [1] "Custo por atraso: R$0"
## [1] "Desconto: R$0"
## [1] "Custo do kW/h: R$0.72"
## [1] "Total a ser pago: R$296.64"
```

```
custo_energia(tipo_local = "Comercial", consumo_kwh = 800, atraso = 3)
```

```
## [1] "Consumo: 800kW/h"
## [1] "Tipo de local: Comercial"
## [1] "Custo por atraso: R$20.4"
## [1] "Desconto: R$0"
## [1] "Custo do kW/h: R$0.85"
## [1] "Total a ser pago: R$700.4"
```

# Exemplo aplicado - criação da função

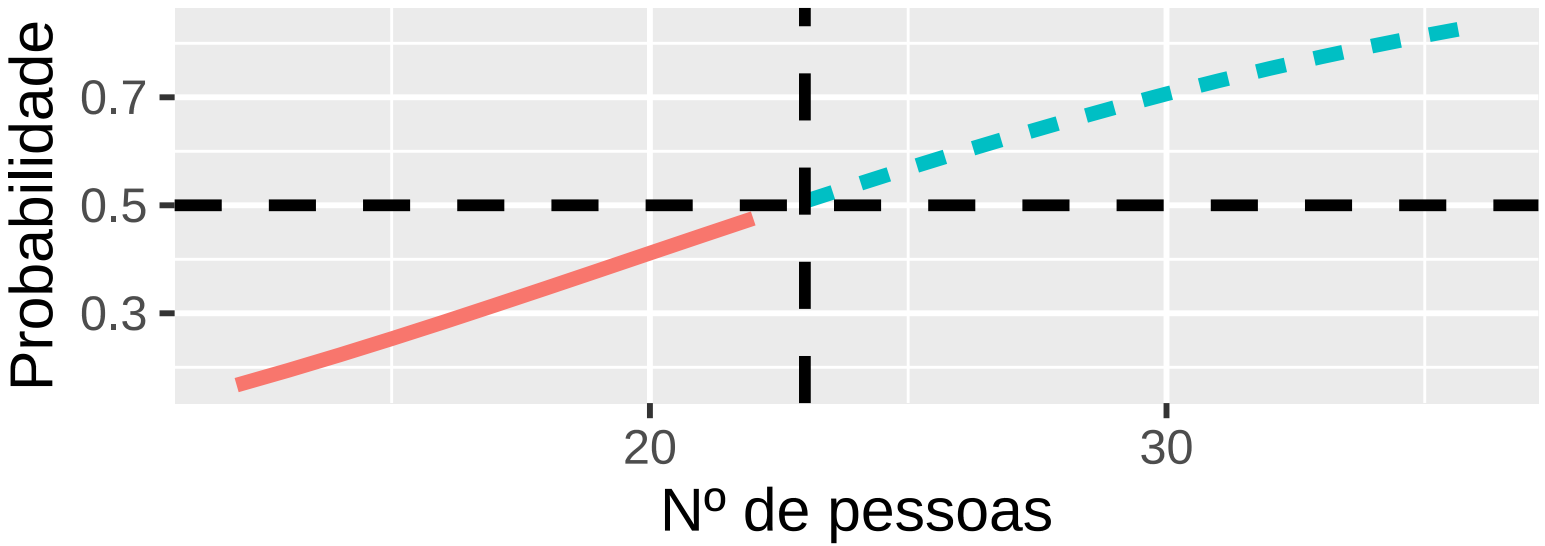
- Aproximar e visualizar graficamente a solução de equações/inequações numéricas. Exemplo baseado no problema do aniversário.

```
simulacao_aniversario <- function(n_pessoas) {
# ENTRADA:
# n_pessoas: sequência de número inteiros;
# SAÍDA:
# resultados: lista contendo um data.frame com os resultados e um gráfico com a visualização deles.

# Simulação
produtorio <- rep(1, length(n_pessoas))
probabilidades <- rep(1, length(n_pessoas))
for (i in 1:length(n_pessoas)) {
  for (j in 1:(n_pessoas[i] - 1)) {
     produtorio[i] <- produtorio[i] * (365- j)/365
  }
}
probabilidades <- 1 - produtorio

# Gráfico da simulação
dados <- data.frame(n = n_pessoas, p = probabilidades)
dados$satisfazem <- ifelse(dados$p >= 0.5,
                           "Sim", "Não")
g <- ggplot(dados, aes(x = n, y = p, color = satisfazem, linetype = satisfazem)) +
geom_line(linewidth = 1.25, show.legend = FALSE ) +
geom_hline(yintercept = 0.5, linetype = "dashed", linewidth = 1, color = "black") +
geom_vline(xintercept = 23, linetype = "dashed", linewidth = 1, color = "black") +
labs(x = "Nº de pessoas", y = "Probabilidade", color = ">= 50%")

colnames(dados) <- c("Nº de pessoas", "Probabilidade", ">=50%")
resultados <- list(dados, g)
return(resultados)
}
resultados <- simulacao_aniversario(seq(12, 36, 1))
```

# Exemplo aplicado - resultados

- Os resultados podem ser vistos a seguir.

```
## [[1]]
```



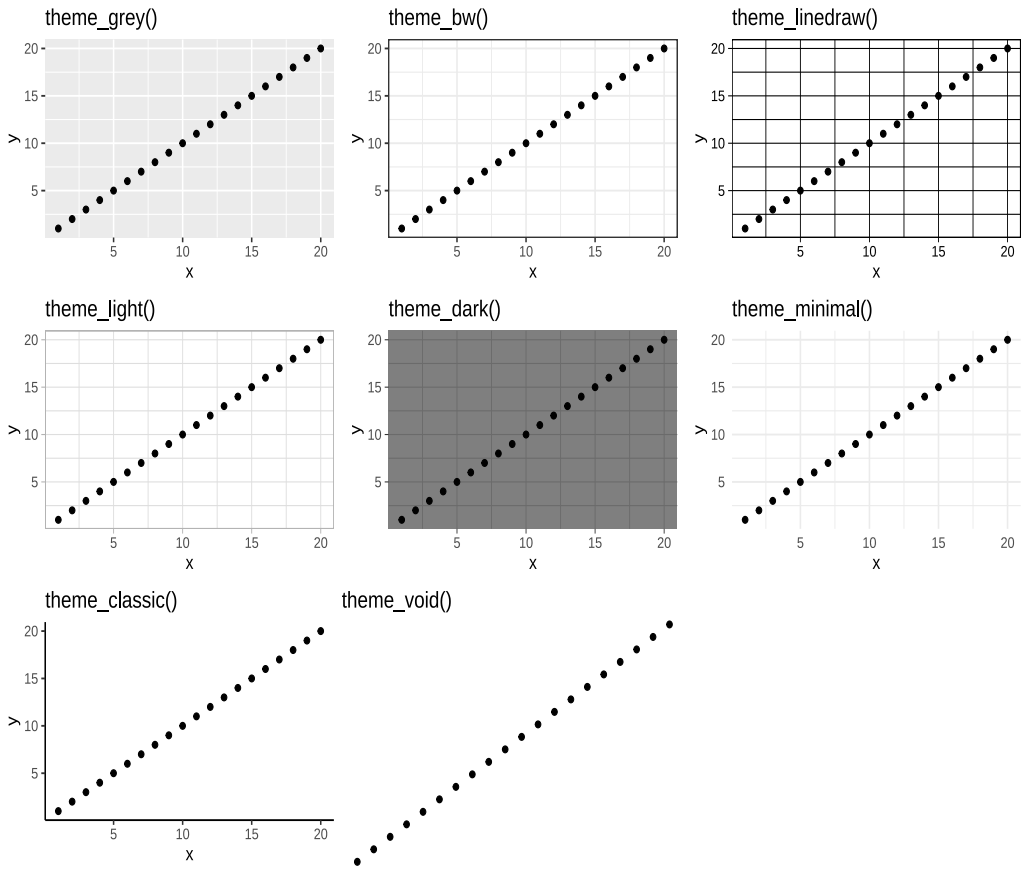
# Temas gráficos customizados

# O que é um tema gráfico?

- Um tema gráfico nada mais é que uma função que é utilizada para configurar as características de um gráfico. Focaremos especificamente na criação e utilização de temas gráficos para **a biblioteca** `ggplot2` **nesse curso**.
- A partir dele podemos definir características padrões para a formatação e posicionamento dos textos, disposição da legenda, existência de "quadriculado" no gráfico, dentre outras diversas coisas.

# Temas básicos disponíveis no `ggplot2`

- A biblioteca `ggplot2` vem com *8 temas gráficos* de fábrica. Seus nomes, bem como suas características podem ser vistas a seguir:



# Criação de temas customizados

- Para gerar temas gráficos, criamos funções com os parâmetros gráficos de interesse dentro delas. Um exemplo extremamente simples mas que já deixa a visualização mais bacana é:

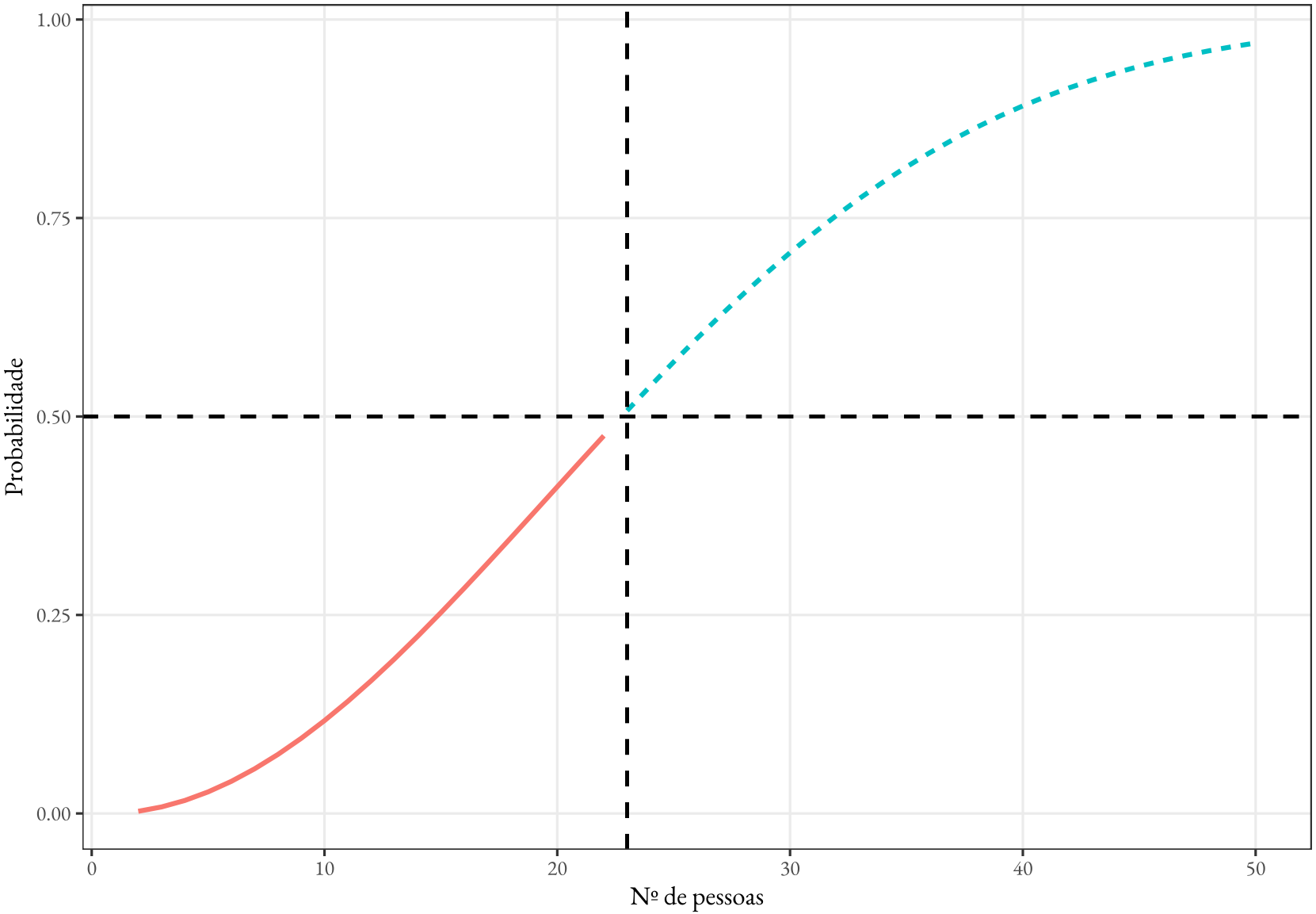
```
novo_tema <- function() {

theme_bw(base_size = 14) +
theme(panel.grid.minor = element_blank(),
      plot.title = element_text(face = "bold"),
      legend.position = "bottom",
      text = element_text(family = "Garamond"))

}
```

no qual alteramos a família e tamanho da fonte[4] dos gráficos, utilizando o tema base `theme_bw` disponível no `ggplot2` como «molde»[5] do nosso tema. No próximo slide temos o resultado dessas mudanças:

[4] Lidar com fontes pode ser complicado! Uma boa discussão sobre o assunto pode ser em: https://stackoverflow.com/questions/34522732/changing-fonts-in-ggplot2.

[5] Ou seja, as características que não forem alteradas pela função serão herdadas do tema clássico.

# Resultado


1.00
0.75
0.50
0.25
0.00
Probabilidade
0
10
20
30
40
50
Nº de pessoas

# Parâmetros importantes

- A seguir, utilizando o comando `?ggplot2::theme`, listo os parâmetros que você pode customizar para deixar o gráfico como quiser. Não deixe de consultar também a vinheta sobre os parâmetros do ggplot2 com `vignette("ggplot2-specs")`.

```
theme(line, rect, text, title, aspect.ratio, axis.title,
  axis.title.x, axis.title.x.top,
  axis.title.x.bottom, axis.title.y, axis.title.y.left, axis.title.y.right,
  axis.text, axis.text.x, axis.text.x.top, axis.text.x.bottom, axis.text.y,
  axis.text.y.left, axis.text.y.right, axis.ticks, axis.ticks.x,
  axis.ticks.x.top, axis.ticks.x.bottom, axis.ticks.y, axis.ticks.y.left,
  axis.ticks.y.right, axis.ticks.length,
  axis.ticks.length.x, axis.ticks.length.x.top,
  axis.ticks.length.x.bottom, axis.ticks.length.y,
  axis.ticks.length.y.left, axis.ticks.length.y.right,
  axis.line, axis.line.x, axis.line.x.top, axis.line.x.bottom, axis.line.y,
  axis.line.y.left, axis.line.y.right, legend.background, legend.margin,
  legend.spacing, legend.spacing.x, legend.spacing.y, legend.key,
  legend.key.size, legend.key.height, legend.key.width, legend.text,
  legend.text.align, legend.title, legend.title.align, legend.position,
  legend.direction, legend.justification, legend.box, legend.box.just,
  legend.box.margin, legend.box.background, legend.box.spacing,
  panel.background, panel.border, panel.spacing, panel.spacing.x,
  panel.spacing.y, panel.grid, panel.grid.major,
  panel.grid.minor, panel.grid.major.x, panel.grid.major.y,
  panel.grid.minor.x, panel.grid.minor.y, panel.ontop,
  plot.background, plot.title, plot.title.position, plot.subtitle,
  plot.caption, plot.caption.position,
  plot.tag, plot.tag.position, plot.margin, strip.background,
  strip.background.x, strip.background.y,
  strip.placement, strip.text, strip.text.x, strip.text.y,
  strip.switch.pad.grid, strip.switch.pad.wrap,
  ...,
  complete = FALSE, validate = TRUE)
```

# Funcionalidade avançada

- Após criar seu tema personalizado, é interessante colocá-lo em um *script* separado e, fazendo uso do comando `source`, importá-lo no início de cada programa. *Eu* costumo ter um *script* chamado `tema_customizado.R` no diretório de trabalho. Assim, simplesmente adiciono

```
# Carregando o tema customizado para os gráficos ----

source("tema_customizado.R")
```

no início de cada *script* e garanto gráficos de alta qualidade e consistência em meus trabalhos.

# RMarkdown (relatórios e slides) e dashboards (*shiny*)

# Exemplos

- Para esses tópicos, em razão da quantidade de assunto a ser discutido e o pouco tempo disponível, optei por disponibilizar 5 exemplos de relatórios (além de um `dashboard` em *shiny*) que eu fiz utilizando R ao longo dos anos. De posse desse material de referência (e bastante pesquisa *online*), os discentes estarão prontos para criar seus próprios materiais quando necessário.
  - Todos se encontram na pasta `Exemplos` do diretório desse curso, inclusive o material necessário para recriar esses *slides* que estou apresentando.

# Dúvidas?

# Encerramento

**Muito obrigado pela atenção e presença. Boa noite!**

*Um evento organizado pelo* ***Laboratório de Inteligência Artificial e Macroeconomia Computacional*** *(LABIMEC) da UFPB.*