Análise Exploratória II

Jenifer Soares Souza

Roteiro 03 - Análise Exploratória II

Partindo dos dados e das justificativas apresentadas no Roteiro 02, este roteiro tem como objetivo introduzir novas funções para realizar uma análise exploratória utilizando o pacote `tidyverse`, que são organizadas com base em princípios fundamentais que buscam facilitar a análise de dados.

Instalando e Importando Bibliotecas

Para instalar pacotes no R, podemos usar a função `install.packages()`. As utilizadas nesse reletório são:

- `tinytex`: permite compilar documentos R Markdown em PDF;
- `rmarkdown`: fornece ferramentas para criar documentos em diversos formatos, como PDF, HTML e Word;
- `tidyverse`: uma coleção de pacotes do R que compartilham funções para manipulação de dados (`dplyr`), visualização (`ggplot2`), e outras operações de análise de dados;
- `readxl`: usado para importar dados do Excel diretamente para o R;
- `openxlsx`: também é usado para importar e exportar dados do Excel no R.

```
install.packages("tinytex")
install.packages("rmarkdown")
install.packages("tidyverse")
install.packages("readxl")
install.packages("openxlsx")
```

Além de instalar, precisamos carregar as bibliotecas em nosso ambiente de trabalho. Para isso, podemos usar a função `library()`.

```
library(readxl)
library(tinytex)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(rmarkdown)
library(openxlsx)
```

Importando a base de dados

Nesta seção, apresentamos o processo de importação da base de dados relacionada aos indicadores brasileiros para análise utilizando a linguagem R.

Primeiramente, foi feito o download de uma planilha em formato Excel a partir do Painel Saneamento Brasil.

Esses mesmos dados foram tratados e alterados anteriormente, e a versão 2 foi baixada diretamente do GitHub, onde o novo arquivo foi armazenado. Para acessar esses dados na nuvem, foi necessário baixá-los como um arquivo temporário em memória, devido à extensão do arquivo. Em seguida, esses dados foram transformados em um dataframe e utilizados em nosso ambiente de trabalho com o R.

```
head(data)
```

```
## # A tibble: 6 x 12
##   Nome           Tipo      Código Parcela da população ~1 Parcela da população~2
##   <chr>          <chr>     <chr>  <chr>                   <chr>
## 1 Ariquemes      Município 110002 0.087                   0.98
## 2 Cacoal         Município 110004 0.212                   0.493
## 3 Jaru           Município 110011 0.527                   1.0
## 4 Ji-Paraná      Município 110012 0.336                   1.0
## 5 Porto Velho    Município 110020 0.647                   0.952
## 6 Rolim de Moura Município 110028 0.293                   1.0
## # i abbreviated names:
## #   1: 'Parcela da população total que mora em domicílios sem acesso à água tratada\r\n(% da populaç
## #   2: 'Parcela da população total que mora em domicílios sem acesso ao serviço de coleta de esgoto\
## # i 7 more variables:
## #   'Volume de esgoto não tratado (água consumida - esgoto tratado)\r\n(mil m3) (SNIS)' <chr>,
## #   'Razão entre volume de esgoto tratado e volume de água consumida\r\n(%) (SNIS)' <chr>,
## #   'Internações por doenças associadas à falta de saneamento\r\n(Número de internações) (DATASUS)' <
```

Além disso, para uma análise futura sobre estados ou regiões, foram baixados os identificadores dos municípios dos dados de saneamento através do portal do IBGE.

```
file_path = "municipios.xlsx"
municipalities_data <- read_excel(path = file_path)

head(municipalities_data)
```

```
## # A tibble: 6 x 13
##   UF    Nome_UF  'Região Geográfica Intermediária' Nome Região Geográfica Inte~1
##   <chr> <chr>    <chr>                             <chr>
## 1 11    Rondônia 1102                              Ji-Paraná
## 2 11    Rondônia 1102                              Ji-Paraná
## 3 11    Rondônia 1101                              Porto Velho
## 4 11    Rondônia 1102                              Ji-Paraná
## 5 11    Rondônia 1101                              Porto Velho
## 6 11    Rondônia 1101                              Porto Velho
## # i abbreviated name: 1: 'Nome Região Geográfica Intermediária'
```

```
## # i 9 more variables: ‘Região Geográfica Imediata‘ <chr>,
## #   ‘Nome Região Geográfica Imediata‘ <chr>, ‘Mesorregião Geográfica‘ <chr>,
## #   Nome_Mesorregião <chr>, ‘Microrregião Geográfica‘ <chr>,
## #   Nome_Microrregião <chr>, Município <chr>,
## #   ‘Código Município Completo‘ <chr>, Nome_Município <chr>
```

Explorando os dados

Na sessão anterior, criamos um DataFrame, uma estrutura de dados que organiza informações em linhas e colunas, similar a uma planilha no Excel ou, de forma mais simples, a uma tabela. Cada coluna em um DataFrame possui um cabeçalho que indica qual informação ela representa, fornecendo uma visão dos dados. Para o nosso novo conjunto de dados, baixado do portal do IBGE, temos:

```
cat(paste(names(municipalities_data), collapse = "\n\n"))
```

```
## UF
##
## Nome_UF
##
## Região Geográfica Intermediária
##
## Nome Região Geográfica Intermediária
##
## Região Geográfica Imediata
##
## Nome Região Geográfica Imediata
##
## Mesorregião Geográfica
##
## Nome_Mesorregião
##
## Microrregião Geográfica
##
## Nome_Microrregião
##
## Município
##
## Código Município Completo
##
## Nome_Município
```

Operador Pipe

O **operador pipe (%>%)** é a principal contribuição do `tidyverse` à análise de dados. Ele é uma ferramenta poderosa para **expressar uma sequencia de operações**.

Mutate A função `mutate()` permite criar ou alterar colunas em um dataframe, adicionando novas variáveis calculadas com base em operações, funções ou lógica aplicada aos dados existentes nas colunas do dataframe.

Para o relatório atual, vamos utilizar a função `mutate()` para, inicialmente, renomear as colunas para nomes mais significativos e fáceis de manipular. Os nomes originais das colunas podem conter acentos e quebras de linha, o que pode dificultar e até mesmo gerar erros ao selecionar os dados necessários.

Para evitar o uso dos longos nomes das colunas, foi necessário referenciá-los de maneira dinâmica. Para isso, os nomes dos cabeçalhos foram transformados em uma lista e, a partir dessa lista, utilizou-se `!!sym(headers[index])`. Essas expressões avaliam dinamicamente os nomes contidos em `headers[index]` como símbolos, permitindo que sejam utilizados como nomes de novas colunas ou parte de operações de mutação dentro da função `mutate()`.

Além disso, para selecionar apenas as colunas necessárias, foi utilizado o método `select(-any_of(headers))`, que permite escolher um conjunto de colunas em um dataframe, excluindo aquelas que estão listadas no vetor `headers`.

```
headers <- names(data)

data2 <- data %>%
  mutate(
    NOME_MUN = !!sym(headers[1]),
    TIPO_MUN = !!sym(headers[2]),
    CODIGO_MUN = !!sym(headers[3]),
    SEM_AGUA_TRATADA = !!sym(headers[4]),
    SEM_COLETA_ESGOTO = !!sym(headers[5]),
    ESGOTO_NAO_TRATADO = !!sym(headers[6]),
    ESGOTO_TRATADO_AGUA_CONSUMIDA = !!sym(headers[7]),
    INTERNACOES = !!sym(headers[8]),
    OBITOS = !!sym(headers[9]),
    RENDIMENTO_COM_SANEAMENTO = !!sym(headers[10]),
    RENDIMENTO_SEM_SANEAMENTO = !!sym(headers[11]),
    TEVE_INTERNACOES = !!sym(headers[12])
  ) %>%
  select(-any_of(headers))

head(data2)
```

```
## # A tibble: 6 x 12
##   NOME_MUN       TIPO_MUN  CODIGO_MUN SEM_AGUA_TRATADA SEM_COLETA_ESGOTO
##   <chr>          <chr>     <chr>      <chr>            <chr>
## 1 Ariquemes      Município 110002     0.087            0.98
## 2 Cacoal         Município 110004     0.212            0.493
## 3 Jaru           Município 110011     0.527            1.0
## 4 Ji-Paraná      Município 110012     0.336            1.0
## 5 Porto Velho    Município 110020     0.647            0.952
## 6 Rolim de Moura Município 110028     0.293            1.0
## # i 7 more variables: ESGOTO_NAO_TRATADO <chr>,
## #   ESGOTO_TRATADO_AGUA_CONSUMIDA <chr>, INTERNACOES <dbl>, OBITOS <dbl>,
## #   RENDIMENTO_COM_SANEAMENTO <dbl>, RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   TEVE_INTERNACOES <chr>
```

```
headers_municipalities <- names(municipalities_data)

municipalities_data2 <- municipalities_data %>%
  mutate(
    NOME_UF = !!sym(headers_municipalities[2]),
    NOME_MUN = !!sym(headers_municipalities[13]),
  ) %>%
  select(-any_of(headers_municipalities))
```

```r
head(municipalities_data2)
```

```
## # A tibble: 6 x 2
##   NOME_UF  NOME_MUN
##   <chr>    <chr>
## 1 Rondônia Alta Floresta D'Oeste
## 2 Rondônia Alto Alegre dos Parecis
## 3 Rondônia Alto Paraíso
## 4 Rondônia Alvorada D'Oeste
## 5 Rondônia Ariquemes
## 6 Rondônia Buritis
```

Join Como supõe o nome, a união serve para unir duas bases de dados a partir de um identificador.

No tidyverse, a união pode ser feita usando uma de quatro funções, a depender do seu objetivo:

- `left_join()`: Adiciona à primeira base as variáveis da segunda base que possuem correspondência ao identificador.
- `right_join()`: Adiciona à segunda base as variáveis da primeira base possuem correspondência ao identificador.
- `inner_join()`: O resultado é uma base de dados que exclui as observações sem correspondência ao identificador.
- `full_join()`: O resultado é uma base de dados com todas as observações, da primeira e segunda base, adicionando valores faltantes (NA) quando não há correspondência ao identificador.

Neste relatório, optamos pelo left join porque queremos mostrar a variação média de renda em determinados estados. Para isso, precisamos de todos os dados de saneamento, incluindo a identificação do município. Inicialmente, escolhemos usar o `CODIGO_MUN`, mas isso resultou em algumas cidades sem correspondência devido a divergências nos códigos. Para contornar essa situação, optamos por usar a chave `NOME_MUN`.

Para contornar o problema de municípios com nomes iguais em diferentes estados, foram removidas as linhas duplicadas.

```r
full_data <- data2 %>%
  left_join(municipalities_data2, by = "NOME_MUN") %>%
  distinct(NOME_MUN, .keep_all = TRUE)

head(full_data)
```

```
## # A tibble: 6 x 13
##   NOME_MUN       TIPO_MUN  CODIGO_MUN SEM_AGUA_TRATADA SEM_COLETA_ESGOTO
##   <chr>          <chr>     <chr>      <chr>            <chr>
## 1 Ariquemes      Município 110002     0.087            0.98
## 2 Cacoal         Município 110004     0.212            0.493
## 3 Jaru           Município 110011     0.527            1.0
## 4 Ji-Paraná      Município 110012     0.336            1.0
## 5 Porto Velho    Município 110020     0.647            0.952
## 6 Rolim de Moura Município 110028     0.293            1.0
## # i 8 more variables: ESGOTO_NAO_TRATADO <chr>,
## #   ESGOTO_TRATADO_AGUA_CONSUMIDA <chr>, INTERNACOES <dbl>, OBITOS <dbl>,
## #   RENDIMENTO_COM_SANEAMENTO <dbl>, RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   TEVE_INTERNACOES <chr>, NOME_UF <chr>
```

Group By e Summarize **Agrupar e resumir** são duas etapas da análise de dados, geralmente aplicadas juntas, para calcular estatísticas básicas em subconjuntos.

Na aula teórica, aprendemos a calcular o intervalo de confiança.

Para explorar a desigualdade de rendimentos entre pessoas que possuem ou não saneamento, vamos primeiro filtrar os valores faltantes (`drop_na()`), agrupar por regiões (`group_by()`), resumir estatísticas básicas (`summarize()`) e calcular o intervalo de confiança para cada região (`mutate()`) com um intervalo de confiança de 95% (escore-z da curva normal é igual a 1,96). O resultado será salvo como o objeto de nome `income_data`.

```
income_data <- full_data %>%
  drop_na(RENDIMENTO_SEM_SANEAMENTO) %>%
  drop_na(RENDIMENTO_COM_SANEAMENTO) %>%
  group_by(NOME_UF) %>%
  summarize(
    n_obs = n(),
    MEDIA_RENDIMENTO_SEM_SANEAMENTO = mean(RENDIMENTO_SEM_SANEAMENTO),
    MEDIA_RENDIMENTO_COM_SANEAMENTO = mean(RENDIMENTO_COM_SANEAMENTO),
    DP_RENDIMENTO_SEM_SANEAMENTO = sd(RENDIMENTO_SEM_SANEAMENTO),
    DP_RENDIMENTO_COM_SANEAMENTO = sd(RENDIMENTO_COM_SANEAMENTO),
  ) %>%
  mutate(
    ERRO_RENDIMENTO_SEM_SANEAMENTO = (
      1.96 * DP_RENDIMENTO_SEM_SANEAMENTO/sqrt(n_obs)
    ),
    LS_RENDIMENTO_SEM_SANEAMENTO = (
      MEDIA_RENDIMENTO_SEM_SANEAMENTO + ERRO_RENDIMENTO_SEM_SANEAMENTO
    ),
    LI_RENDIMENTO_SEM_SANEAMENTO = (
      MEDIA_RENDIMENTO_SEM_SANEAMENTO - ERRO_RENDIMENTO_SEM_SANEAMENTO
    ),
    ERRO_RENDIMENTO_COM_SANEAMENTO = (
      1.96 * DP_RENDIMENTO_COM_SANEAMENTO/sqrt(n_obs)
    ),
    LS_RENDIMENTO_COM_SANEAMENTO = (
      MEDIA_RENDIMENTO_COM_SANEAMENTO + ERRO_RENDIMENTO_COM_SANEAMENTO
    ),
    LI_RENDIMENTO_COM_SANEAMENTO = (
      MEDIA_RENDIMENTO_COM_SANEAMENTO - ERRO_RENDIMENTO_COM_SANEAMENTO
    ),
  )


head(income_data)
```

```
## # A tibble: 6 x 12
##   NOME_UF  n_obs MEDIA_RENDIMENTO_SEM_SANEAMENTO MEDIA_RENDIMENTO_COM_SANEAMENTO
##   <chr>    <int>                           <dbl>                           <dbl>
## 1 Acre         2                            847.                            4565.
## 2 Alagoas     18                            687.                            1909.
## 3 Amapá        2                            807.                            2865.
## 4 Amazonas    15                            533.                            1725.
## 5 Bahia       53                            558.                            1968.
## 6 Ceará       42                            508.                            1738.
```

```
## # i 8 more variables: DP_RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   DP_RENDIMENTO_COM_SANEAMENTO <dbl>, ERRO_RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   LS_RENDIMENTO_SEM_SANEAMENTO <dbl>, LI_RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   ERRO_RENDIMENTO_COM_SANEAMENTO <dbl>, LS_RENDIMENTO_COM_SANEAMENTO <dbl>,
## #   LI_RENDIMENTO_COM_SANEAMENTO <dbl>
```

GG Plot Para análise gráfica e evitar a poluição visual com muitos dados, selecionamos os 6 estados com o maior número de observações não nulas de rendimentos com ou sem saneamento. Para isso, reordenamos o dataframe anterior com base na variável `n_obs`, de maneira decrescente, selecionando os 6 primeiros estados. Esses estados foram armazenados em uma nova lista, que será usada para filtrar nosso dataframe contendo os dados necessários.

```
ufs_for_analysis <- income_data %>%
  arrange(desc(n_obs)) %>%
  head(6) %>%
  pull(NOME_UF)

ufs_for_analysis
```

```
## [1] "São Paulo"         "Minas Gerais"      "Bahia"
## [4] "Rio Grande do Sul" "Paraná"            "Santa Catarina"
```

```
data_for_analysis <- full_data %>%
  filter(NOME_UF %in% ufs_for_analysis)

head(data_for_analysis)
```

```
## # A tibble: 6 x 13
##   NOME_MUN          TIPO_MUN  CODIGO_MUN SEM_AGUA_TRATADA SEM_COLETA_ESGOTO
##   <chr>             <chr>     <chr>      <chr>            <chr>
## 1 Alagoinhas        Município 290070     0.126            0.676
## 2 Araci             Município 290210     0.504            0.661
## 3 Barra             Município 290270     -                -
## 4 Barreiras         Município 290320     0.0              0.136
## 5 Bom Jesus da Lapa Município 290390     0.0              0.301
## 6 Brumado           Município 290460     0.085            0.924
## # i 8 more variables: ESGOTO_NAO_TRATADO <chr>,
## #   ESGOTO_TRATADO_AGUA_CONSUMIDA <chr>, INTERNACOES <dbl>, OBITOS <dbl>,
## #   RENDIMENTO_COM_SANEAMENTO <dbl>, RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   TEVE_INTERNACOES <chr>, NOME_UF <chr>
```

Dentre os componentes do tidyverse, está o pacote `ggplot2`, que permite a criação de gráficos a partir de uma linguagem universal entre os programadores e designers, chamada de a gramática dos gráficos. Não é o objetivo deste roteiro explorar a visualização de dados em detalhes, mas os comandos abaixo mostram como construir alguns tipos de gráficos.

Com o pacote `ggplot2`, qualquer gráfico bidimensional pode ser construído partindo da função `ggplot()`, que segue a seguinte lógica:

$\text{ggplot}(\text{data} = \textit{base\_de\_dados}, \text{aes}(\text{x} = \textit{codigo\_variavel}, \text{y} = \textit{codigo\_variavel})) + \text{geom}\_\textit{tipo\_de\_geometria}()$

Em que os argumentos obrigatórios são:

- `data` = base de dados (tabela)

- `aes()` = estética. É aqui que você vai especificar os eixos x e y (se houver), assim como atributos adicionais como cor, tamanho, espessura e formato dos pontos/linhas/polígonos

Alguns dos tipos de geometria mais comuns são:

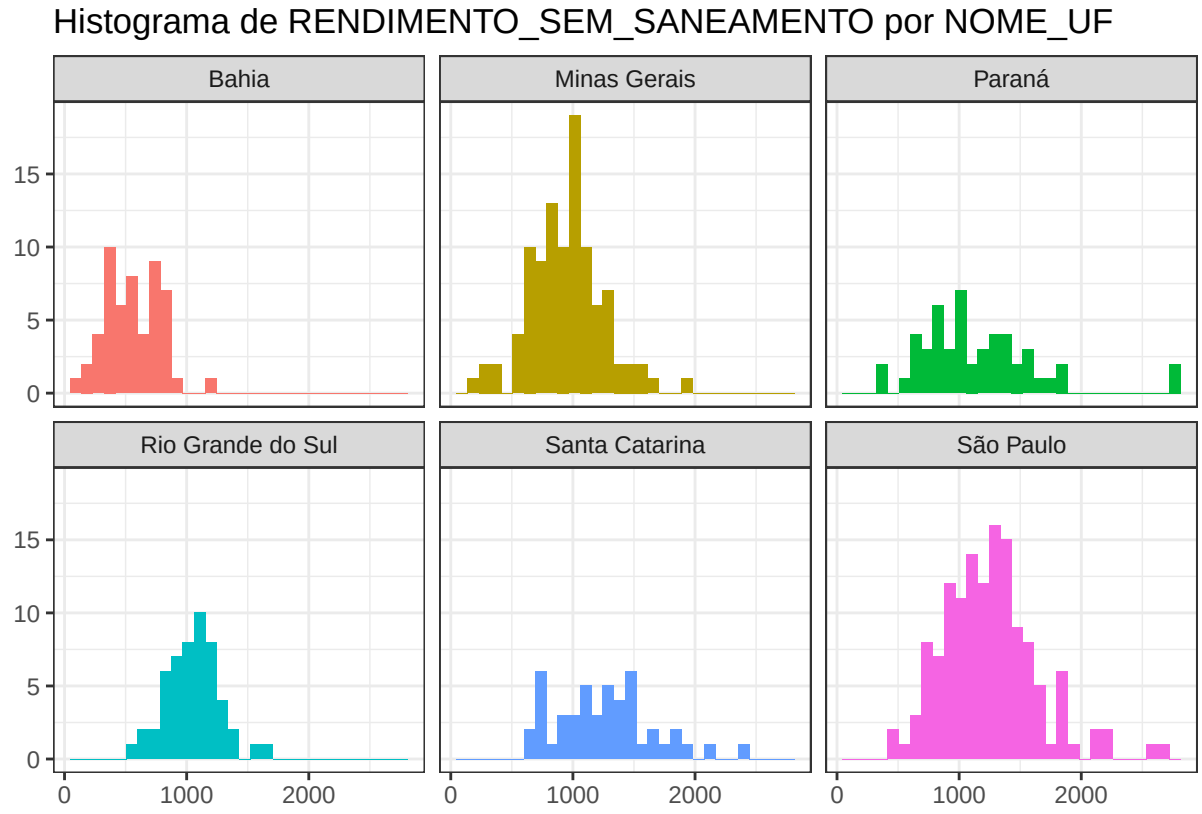
- `+ geom_histogram()` = histograma
- `+ geom_boxplot()` = box-plot
- `+ geom_point()` = gráfico de dispersão
- `+ geom_line()` = gráfico de linhas
- `+ geom_abline()` = reta de tendência (correlação)
- `+ geom_smooth()` = reta de tendência (regressão)
- `+ geom_bar()` = gráfico de barras (pré-tabulação/contagem)
- `+ geom_col()` = gráfico de barras (pós-tabulação/contagem)
- `+ geom_errorbar()` = barras de erro

Alguns argumentos adicionais que podem ser explorados são:

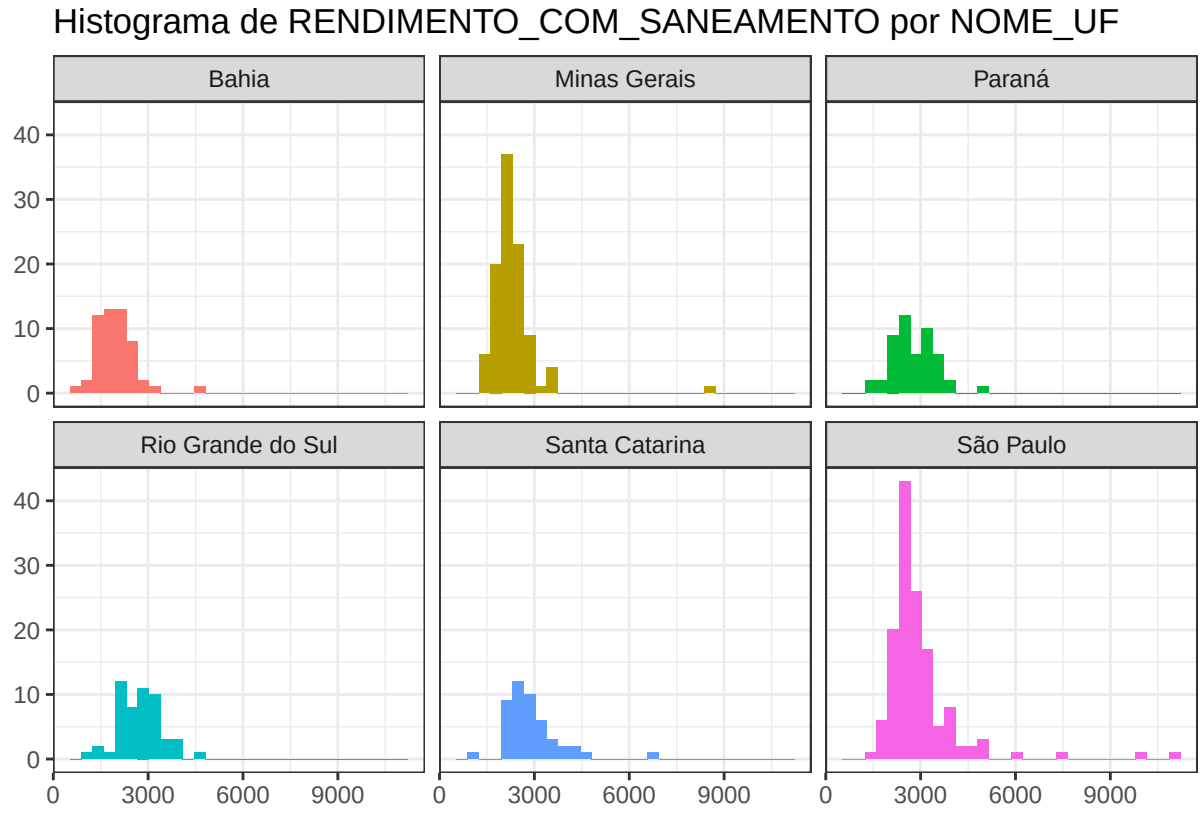
- `+ facet_wrap(~*codigo_variavel*)` = replica um gráfico para diferentes classes
- `+ coord_flip()` = inverte as coordenadas x e y
- `+ ggtitle("Título em parênteses")` = adiciona um título
- `+ theme()` = para customizar os componentes não relacionados aos dados

Vamos usar a função `ggplot()` para explorar com maior profundidade as desigualdades regionais relacionadas à Rendimentos com e sem Saneamento.

```
ggplot(
  data = data_for_analysis,
  aes(x = RENDIMENTO_SEM_SANEAMENTO, fill = NOME_UF)
) +
  geom_histogram(bins = 30) +
  facet_wrap(~NOME_UF) +
  ggtitle("Histograma de RENDIMENTO_SEM_SANEAMENTO por NOME_UF") +
  xlab("") +
  ylab("") +
  theme_bw() +
  theme(legend.position = "none")
```

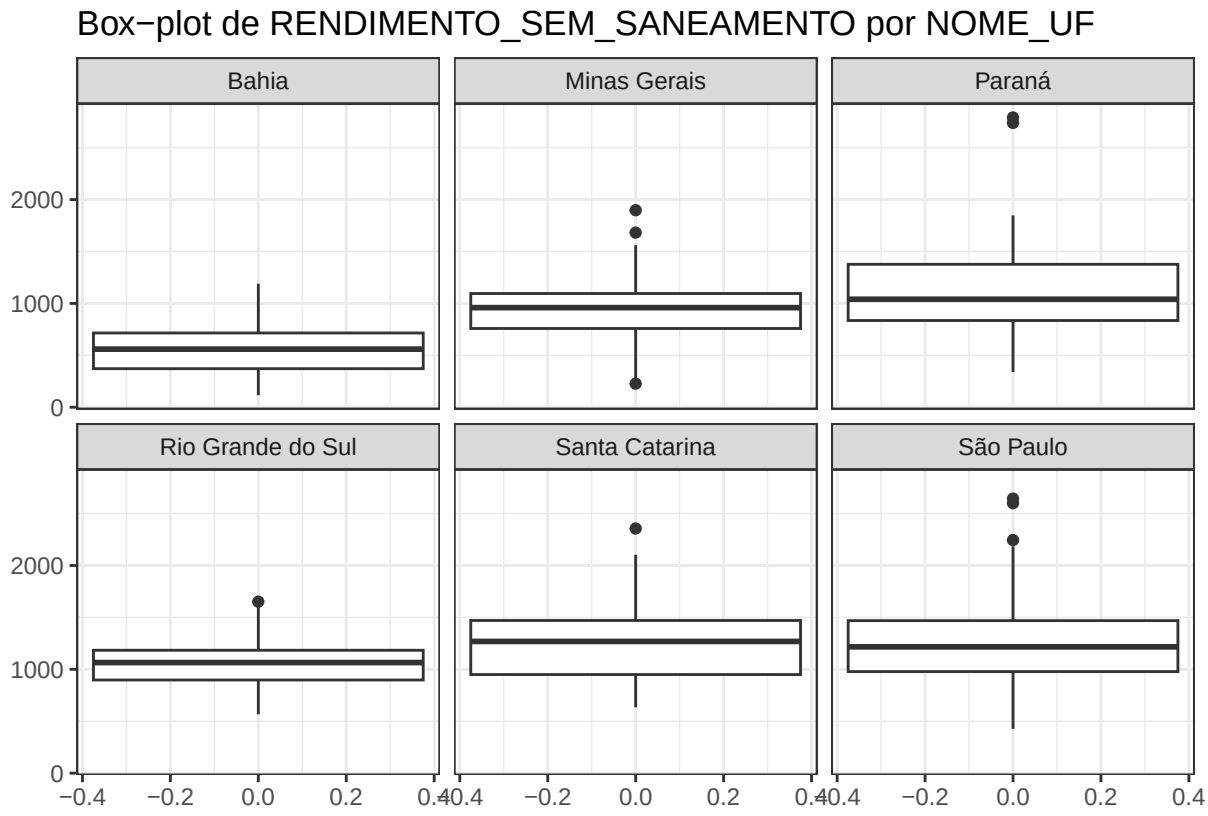



```
ggplot(
  data = data_for_analysis,
  aes(x = RENDIMENTO_COM_SANEAMENTO, fill = NOME_UF)
) +
  geom_histogram(bins = 30) +
  facet_wrap(~NOME_UF) +
  ggtitle("Histograma de RENDIMENTO_COM_SANEAMENTO por NOME_UF") +
  xlab("") +
  ylab("") +
  theme_bw() +
  theme(legend.position = "none")
```

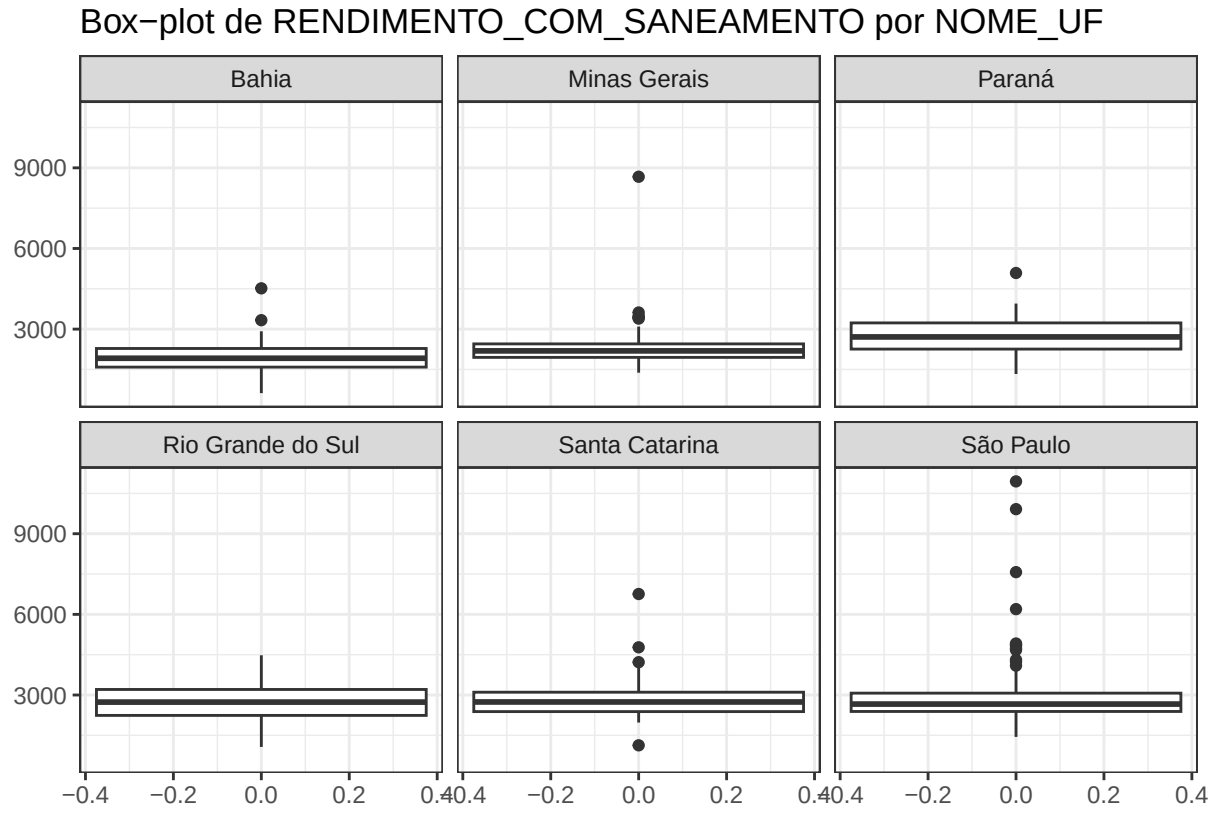


O código abaixo desenha um box-plot que compara Rendimento com ou sem Saneamento entre os municípios.

```
ggplot(
  data = data_for_analysis,
  aes(y = RENDIMENTO_SEM_SANEAMENTO)
) +
  geom_boxplot() +
  facet_wrap(~NOME_UF) +
  ggtitle("Box-plot de RENDIMENTO_SEM_SANEAMENTO por NOME_UF") +
  xlab("") +
  ylab("") +
  theme_bw() +
  theme(legend.position = "none")
```




```
ggplot(
  data = data_for_analysis,
  aes(y = RENDIMENTO_COM_SANEAMENTO)
) +
  geom_boxplot() +
  facet_wrap(~NOME_UF) +
  ggtitle("Box-plot de RENDIMENTO_COM_SANEAMENTO por NOME_UF") +
  xlab("") +
  ylab("") +
  theme_bw() +
  theme(legend.position = "none")
```



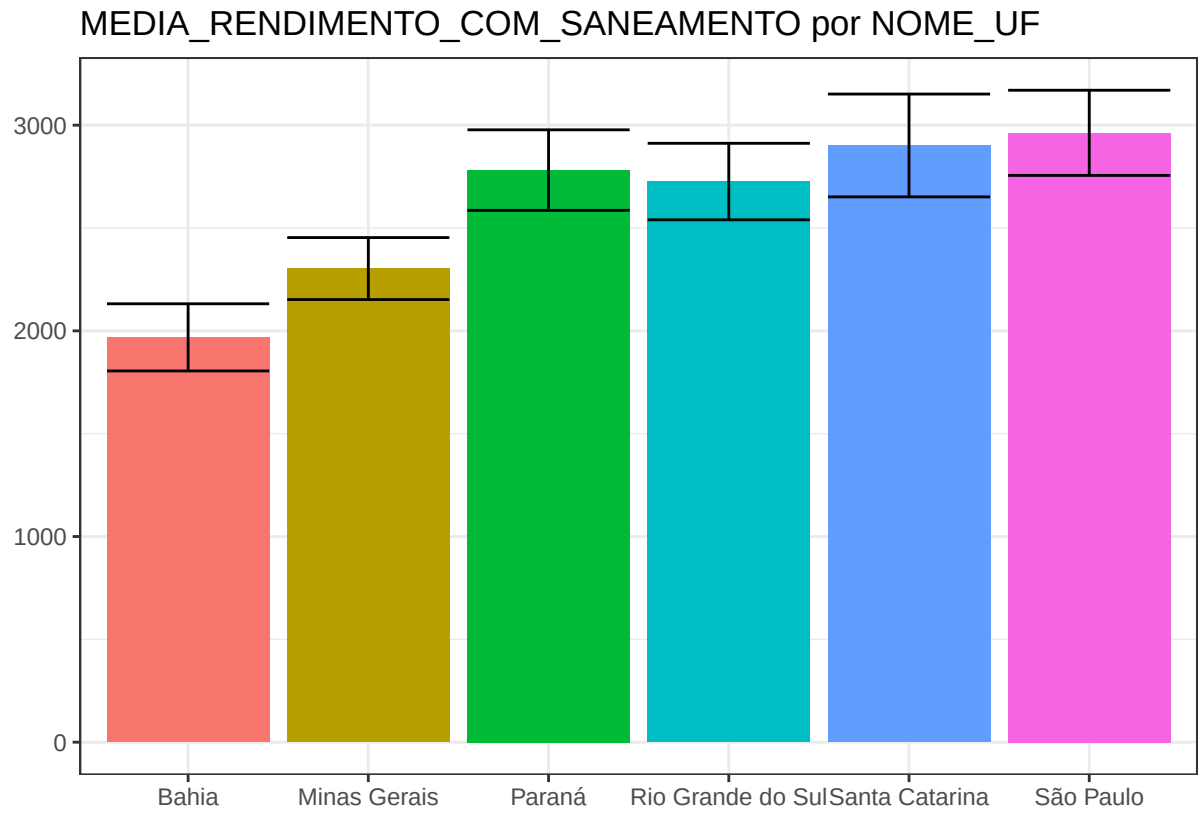

```
income_data_for_analysis <- income_data %>%
  filter(NOME_UF %in% ufs_for_analysis)

income_data_for_analysis
```

```
## # A tibble: 6 x 12
##   NOME_UF           n_obs MEDIA_RENDIMENTO_SEM_SANEAMENTO MEDIA_RENDIMENTO_COM~1
##   <chr>             <int>                           <dbl>                  <dbl>
## 1 Bahia                53                            558.                  1968.
## 2 Minas Gerais        101                            946.                  2303.
## 3 Paraná               50                           1155.                  2782.
## 4 Rio Grande do Sul    52                           1050.                  2726.
## 5 Santa Catarina       47                           1255.                  2901.
## 6 São Paulo           137                           1248.                  2963.
## # i abbreviated name: 1: MEDIA_RENDIMENTO_COM_SANEAMENTO
## # i 8 more variables: DP_RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   DP_RENDIMENTO_COM_SANEAMENTO <dbl>, ERRO_RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   LS_RENDIMENTO_SEM_SANEAMENTO <dbl>, LI_RENDIMENTO_SEM_SANEAMENTO <dbl>,
## #   ERRO_RENDIMENTO_COM_SANEAMENTO <dbl>, LS_RENDIMENTO_COM_SANEAMENTO <dbl>,
## #   LI_RENDIMENTO_COM_SANEAMENTO <dbl>
```

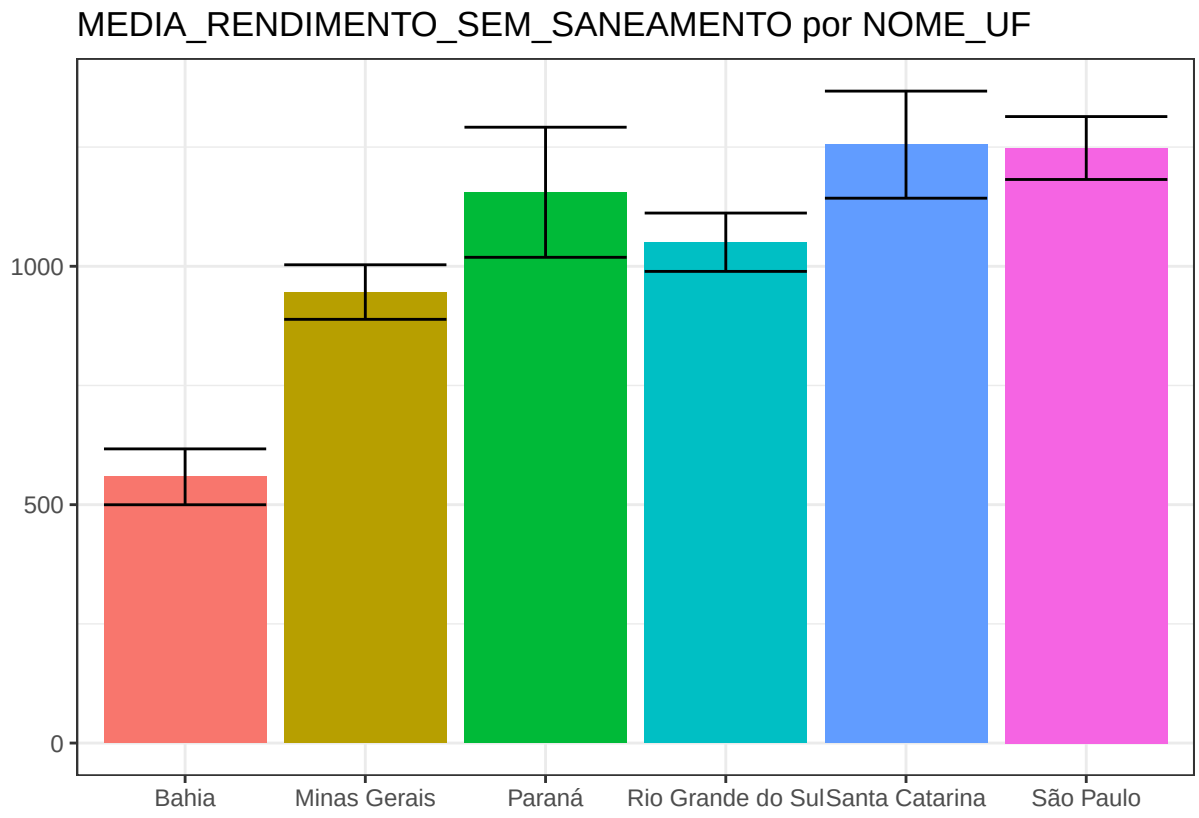
Por fim, o gráfico abaixo compara a média e intervalo de confiança de Rendimento com ou sem Saneamento por região.

```
ggplot(
  data = income_data_for_analysis,
  aes(x = NOME_UF, y = MEDIA_RENDIMENTO_COM_SANEAMENTO, fill = NOME_UF)
) +
  geom_col() +
  ggtitle("MEDIA_RENDIMENTO_COM_SANEAMENTO por NOME_UF") +
  geom_errorbar(
    aes(
      ymin = LI_RENDIMENTO_COM_SANEAMENTO,
      ymax = LS_RENDIMENTO_COM_SANEAMENTO
    )
  ) +
  xlab("") +
  ylab("") +
  theme_bw() +
  theme(legend.position = "none")
```




```
ggplot(
  data = income_data_for_analysis,
  aes(x = NOME_UF, y = MEDIA_RENDIMENTO_SEM_SANEAMENTO, fill= NOME_UF)
) +
  geom_col() +
  ggtitle("MEDIA_RENDIMENTO_SEM_SANEAMENTO por NOME_UF") +
  geom_errorbar(
    aes(
```

```
      ymin = LI_RENDIMENTO_SEM_SANEAMENTO,
      ymax = LS_RENDIMENTO_SEM_SANEAMENTO
    )
  ) +
  xlab("") +
  ylab("") +
  theme_bw() +
  theme(legend.position = "none")
```



Conclusões

De maneira geral, pode-se dizer que:

- Em todos os estados analisados, o rendimento médio das pessoas que possuem saneamento básico é maior do que o das pessoas que não possuem;
- Há uma disparidade no rendimento médio entre os estados. Paraná, Santa Catarina e São Paulo geralmente têm rendimentos mais altos em comparação com Bahia e Minas Gerais, tanto para pessoas com quanto sem saneamento básico.

Exportando a base de dados

Após executar a rotina, é recomendado que o usuário salve as alterações feitas na base de dados para acesso futuro. Para exportar os dados, o procedimento envolve utilizar a biblioteca `openxlsx`.

É possível exportar o dataframe para um arquivo Excel (.xlsx) usando a função `write.xlsx(dataframe, file = "nome_do_arquivo.xlsx")`.

```
output_file_path <- "dados_saneamento_v3.xlsx"
write.xlsx(full_data, file = output_file_path)

output_file_path <- "dados_rendimento.xlsx"
write.xlsx(income_data, file = output_file_path)
```

** Essa aula foi baseada em: Roteiro 03. E os arquivos gerados podem ser visualizados em: Semana 03.