



SÉRIE DE NOTAS TÉCNICAS EM ECONOMIA DA UFG

NT N. 11

NOTA TÉCNICA EM ECONOMIA

n. 11

Publicação cujo objetivo é auxiliar na elaboração de aulas e de pesquisas do Curso de Ciências Econômicas da UFG, divulgando demonstrações técnicas e metodológicas ou rotinas computacionais voltadas para a economia.

As opiniões contidas nesta publicação não representam o ponto de vista do Curso ou da FACE/UFG, sendo de inteira responsabilidade de seu(s) autor(es).

Reprodução permitida, desde que citada a fonte.

FACE – Faculdade de Administração, Ciências Contábeis e Ciências Econômicas
Curso de Ciências Econômicas
Direção FACE
Prof^ª. Andrea Freire de Lucena
Vice-Direção FACE
Prof^ª. Daiana Paula Pimenta
Coordenação do Curso de Ciências Econômicas
Prof. Everton Sotto Tibiriçá Rosa

Introdução à análise de dados na linguagem R utilizando o pacote dplyr

Wendell Soares Pachenco
PPGECON/FACE/UFG
Tatiane Ferreira do Nascimento Melo da
Silva
PPGECON e IME/UFG

FACE/UFG
Goiânia – Fevereiro de 2023
Versão 1.0

Endereço

Campus Samambaia, Prédio da FACE – Rodovia
Goiânia/Nova Veneza, km. 0, CEP 74690-900,
Goiânia – GO. Tel. (62) 3521 – 1390

URL

<http://www.eco.face.ufg.br>

**CIÊNCIAS
ECONÔMICAS**

FACE
FACULDADE DE
ADMINISTRAÇÃO, CIÊNCIAS CONTÁBEIS E
CIÊNCIAS ECONÔMICAS



Dados Internacionais de Catalogação na Publicação (CIP)
GPT/BC/UFG

Pachenco, Wendell Soares. Introdução à análise de dados na linguagem R utilizando o pacote dplyr/ Tatiane Ferreira do Nascimento Melo da Silva. - 2023.

23 f.: (Série de Notas Técnicas em Economia, 011)

Universidade Federal de Goiás, Goiânia, 2023. Bibliografia.

Anexos. 1. Software R. 2. dplyr. 3. Dados. I. Título.

Introdução à análise de dados na linguagem R utilizando o pacote dplyr

Wendell Soares Pachenco¹

PPGECON/FACE/UFG

Tatiane Ferreira do Nascimento Melo da Silva²

PPGECON e IME/UFG

Resumo: Esta nota técnica tem como objetivo, apresentar o pacote dplyr, da linguagem de programação R, usado na manipulação e pré-processamento de dados. A base de dados utilizada nos exemplos deste trabalho é a do ranking de competitividade entre os estados brasileiros, elaborada pelo Centro de Liderança Pública (CLP). Serão ilustradas, através de exemplos, algumas formas de filtrar, ordenar, selecionar, elaborar resumos estatísticos, agrupar, inserir e excluir registros deste conjunto de dados. Uma das vantagens do dplyr é a semelhança dos seus comandos com a sintaxe da linguagem de consulta estruturada *SQL* (*Standard Query Language*), o que torna os seus *scripts* mais intuitivos, práticos e elegantes nas rotinas de manipulação de *data.frames*, principalmente quando o volume destes é relativamente grande.

Palavras chave: linguagem R, modelagem de dados, pacote dplyr, *SQL*

Abstract: *This technical note aims to present the dplyr package, from the R programming language, used in data manipulation and pre-processing. The database used in the examples of this work is the ranking of competitiveness among Brazilian states, prepared by the Center for Public Leadership (CLP). Some ways of filtering, ordering, selecting, preparing statistical summaries, grouping, inserting and deleting records from this data set will be illustrated. One of the advantages of dplyr is the similarity of its commands with the syntax of the structured query language SQL (Standard Query Language), which makes its scripts more intuitive, practical and elegant in data.frames manipulation routines, mainly when the volume of these is relatively large..*

Key words: R language, data modeling, dplyr package, *SQL*

¹ wendellpachenco@discente.ufg.br

² tmelo@ufg.br

1. Introdução

Esta nota técnica tem como finalidade apresentar a utilização do pacote `dplyr` na análise de dados. Este pacote está disponível na linguagem de programação R.

O pacote `dplyr`, desenvolvido por WICKHAM et al. (2022), é uma gramática de manipulação de dados, cuja a sintaxe é bem parecida com a linguagem de consulta estruturada *SQL*.

O pacote supracitado possui um conjunto bem amplo de funções, destacando como as principais: i) `mutate()`: cria novas variáveis; ii) `select()`: seleciona colunas; iii) `filter()`: exibe linhas de acordo com o(s) critério(s) informado(s); iv) `summarize()`: produz resumos estatísticos e v) `arrange()`: usado no ordenamento dos dados.

Os exemplos apresentados neste trabalho têm cunho introdutório, e a base de dados utilizada é a do Ranking de Competitividade dos Estados Brasileiros, elaborado pelo Centro de Liderança Pública (CLP).

Os 86 indicadores adotados e avaliados no Ranking de Competitividade dos Estados foram utilizados como forma de mensuração de dois conjuntos de avaliação em sustentabilidade bem conhecidos e validados no mercado: os critérios ESG (sigla em inglês que quer dizer ambiental, social e governança) e os Objetivos de Desenvolvimento Sustentável propostos pela Organização das Nações Unidas (ONU) (CLP, 2022).

Este conteúdo não exige um conhecimento avançado sobre R, entretanto recomendamos que o leitor não avance para o próximo exemplo sem antes ter entendido o anterior.

2. Primeiros passos (instalar pacotes e carregar a base de dados)

Antes de executar os exemplos apresentados nesta nota técnica, precisamos instalar os pacotes: `dplyr` (para manipular dados) e `readxl` (para leitura de arquivos em formato Excel). Na sequência precisamos carregá-los na sessão de trabalho do R e efetuar o *download* da base de dados.

2.1. Instalar os pacotes

Para instalar os pacotes `dplyr` e `readxl` basta utilizar a função `install.packages()`, informando o nome do pacote a ser instalado, conforme o *script* abaixo:

```
# Instalar pacotes
rm(list=ls()) # remover objetos
install.packages("dplyr") # gramática de manipulação de dados
install.packages("readxl") # pacote de leitura arquivos Excel
```

Após a execução do procedimento acima, transforme a linha dos comandos em comentário acrescentando o símbolo # no início da instrução. Isto evitará que você instale os pacotes toda vez que for executar o seu *script*.

2.2. Carregar os pacotes

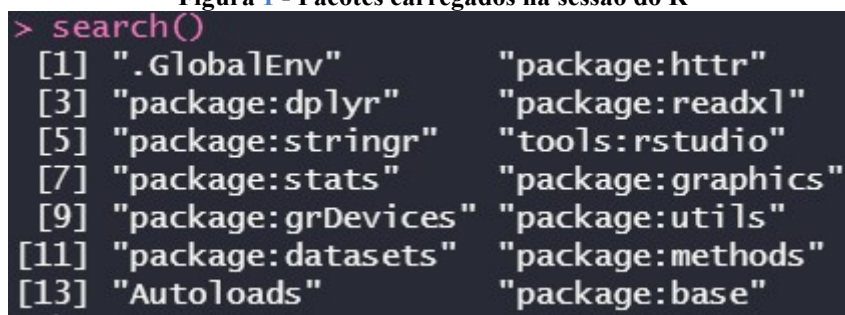
Quando se instala um pacote, este não fica disponível na memória, portanto toda vez que uma nova sessão do R for iniciada o usuário deverá carregá-lo. Se isto não for feito não será possível executar os exemplos constantes nesta nota técnica. Para carregar os pacotes `dplyr` e `readxl` utilize a função `library()` conforme demonstrado abaixo:

```
# Carregar pacotes
library(dplyr)
library(readxl)
```

Para verificar se os pacotes foram carregados execute a função `search()`.

```
# verificar pacotes carregados
search()
```

Figura 1 - Pacotes carregados na sessão do R



```
> search()
[1] ".GlobalEnv"          "package:httr"
[3] "package:dplyr"        "package:readxl"
[5] "package:stringr"      "tools:rstudio"
[7] "package:stats"        "package:graphics"
[9] "package:grDevices"    "package:utils"
[11] "package:datasets"     "package:methods"
[13] "AutoLoads"            "package:base"
```

Fonte: Elaborada pelos autores

Conforme a Figura 1, os pacotes `dplyr` e `readxl` estão carregados na sessão do R. Eles aparecem na 3ª e 4ª posições da lista. Atente-se que estas posições não são fixas, vai variar de acordo com o projeto executado.

2.3. Download da base de dados

A base de dados foi extraída do site Alagoas em dados e informações (<https://dados.al.gov.br/>). Possui uma série histórica do período de 2015 a 2020 do ranking de competitividade entre os estados (elaborado pelo CLP).

Para efetuar o download da base, copie e cole o link abaixo no seu navegador de internet.
https://dados.al.gov.br/catalogo/dataset/636cb783-4565-4ca3-a2a1-0222c8107ed2/resource/332287ac-81b8-4595-b814-64fac8305381/download/ranking_2015_2020.xlsx

Copie o arquivo o arquivo para um diretório de trabalho. Neste exemplo foi criado um diretório chamado NT (nota técnica) em uma unidade de armazenamento externa (*pen-drive*). Fique à vontade para criar o seu diretório no seu local favorito por meio da função `setwd( )`. A unidade e as pastas devem ser separadas por duas barras (`\\`). Para verificar se deu certo utilize a função `getwd( )`, conforme o script abaixo.

```
# informar o diretório de trabalho
setwd("D:\\NT")
# verificar qual é o diretório de trabalho da sessão do R
getwd()
```

Agora iremos criar o *data.frame* (`meus_dados`) com os registros da planilha Excel (utilizando a função `read_xlsx( )`). Basta informar o nome do arquivo já que o diretório de trabalho já foi definido anteriormente.

```
# criar o data.frame da base de dados
arquivo <- "ranking_2015_2020.xlsx"
meus_dados <- read_xlsx(arquivo)
```

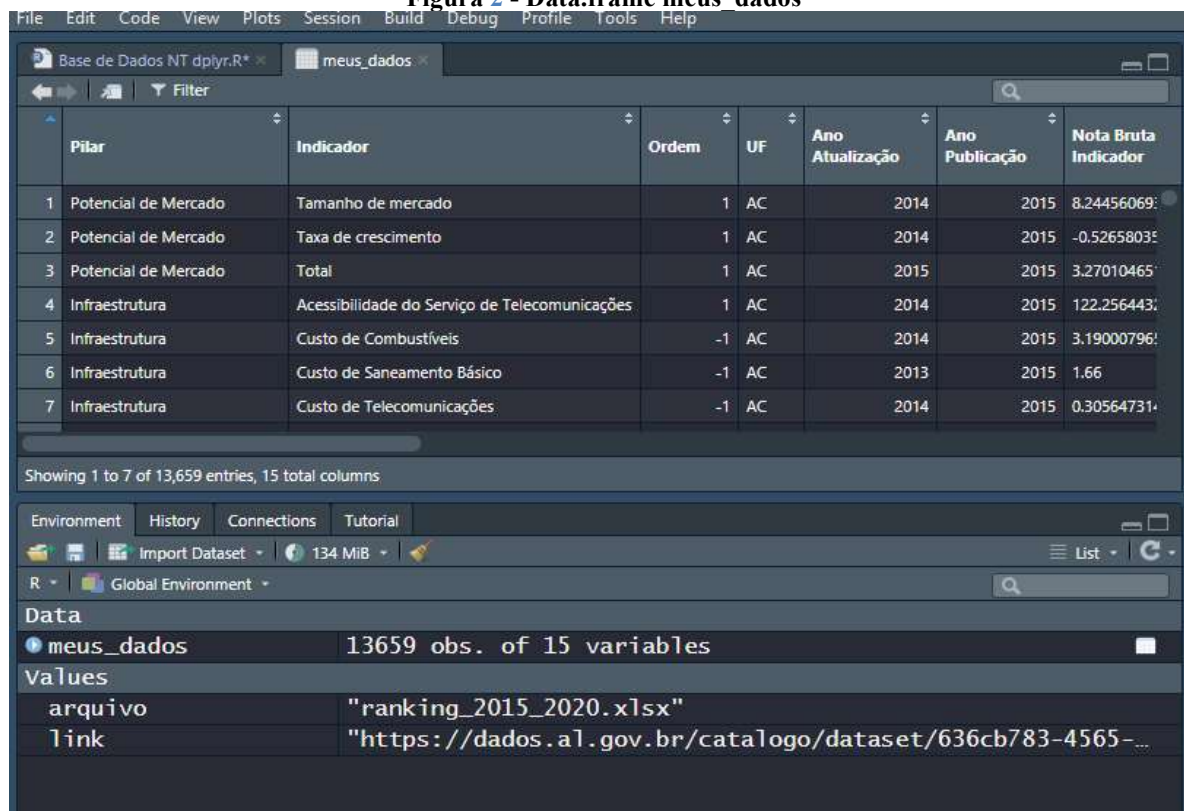
2.4. Converter o data.frame para um objeto específico do pacote dplyr

Ao abrir a base de dados (Figura 2), percebe-se que esta possui 13.659 (registros) linhas e 15 colunas (variáveis). Devido à natureza da sua origem (arquivo Excel) as colunas possuem espaços e acentuação nos seus rótulos, o que não é apropriado para a sua manipulação.

Antes de converter a base para um tipo específico para manipulação do `dplyr` vamos renomear as variáveis. Crie um vetor com os novos rótulos e em seguida aplique a função `names( )` conforme o *script* abaixo:

```
# renomear as colunas (meus_dados)
nomes.colunas <- c(" pilar", " indicador", " ordem", " uf", " ano_atualizacao",
                  " ano_publicacao", " nota_bruta", " peso_indicadores",
                  " peso_pilares", " peso_real", " descricao", " fonte_oficial",
                  " unidade_medida", " detalhes", " nota_normalizada")
names(meus_dados) <- nomes.colunas
```

Figura 2 - Data.frame meus_dados



	Pilar	Indicador	Ordem	UF	Ano Atualização	Ano Publicação	Nota Bruta Indicador
1	Potencial de Mercado	Tamanho de mercado	1	AC	2014	2015	8.24456069
2	Potencial de Mercado	Taxa de crescimento	1	AC	2014	2015	-0.52658035
3	Potencial de Mercado	Total	1	AC	2015	2015	3.27010465
4	Infraestrutura	Acessibilidade do Serviço de Telecomunicações	1	AC	2014	2015	122.256443
5	Infraestrutura	Custo de Combustíveis	-1	AC	2014	2015	3.19000796
6	Infraestrutura	Custo de Saneamento Básico	-1	AC	2013	2015	1.66
7	Infraestrutura	Custo de Telecomunicações	-1	AC	2014	2015	0.30564731

Showing 1 to 7 of 13,659 entries, 15 total columns

Environment | History | Connections | Tutorial

Import Dataset | 134 MiB

R | Global Environment

Data

meus_dados | 13659 obs. of 15 variables

Values

arquivo | "ranking_2015_2020.xlsx"

link | "https://dados.al.gov.br/catalogo/dataset/636cb783-4565-..."

Fonte: Elaborada pelos autores

Abra o objeto `meus_dados` e verifique o resultado das alterações. Se estiver tudo certo, então crie um novo objeto chamado **ranking** e atribua os valores do objeto **meus_dados** convertidos para um tipo mais apropriado para manipulação via `dplyr` utilizando a função `tibble()`.

```
# criar data.frame apropriado para o dplyr
ranking <- tibble(meus_dados)
```

Com a base de dados devidamente preparada avançaremos para a próxima sessão, onde, finalmente, passaremos para a etapa de análise e manipulação dos dados.

3. Análise e manipulação de dados através do pacote `dplyr`

Nesta sessão serão trabalhados alguns exemplos utilizando as principais funções do `dplyr`: `mutate`, `select`, `filter`, `summarize` e `arrange`, apresentadas na introdução deste trabalho. De forma mais secundária também serão apresentadas as funções `glimpse` e `inner_join`, para analisar a natureza das variáveis e juntar (concatenar) base de dados.

3.1. Visualizar os atributos das variáveis da base de dados

A função `glimpse()` exibe os nomes e os tipos das variáveis da base de dados. A sua sintaxe é bem simples, basta informar o nome do *data.frame*, conforme demonstrando a seguir.

```
# visualizar os nomes e atributos das colunas (base de dados)
glimpse(ranking)
```

O output do comando é o seguinte:

```
Rows: 13,659
Columns: 15
$ pilar          <chr> "Potencial de Mercado", ~
$ indicador      <chr> "Tamanho de mercado", "T~
$ ordem          <dbl> 1, 1, 1, 1, -1, -1, -1, ~
$ uf             <chr> "AC", "AC", "AC", "AC", ~
$ ano_atualizacao <dbl> 2014, 2014, 2015, 2014, ~
$ ano_publicacao <dbl> 2015, 2015, 2015, 2015, ~
$ nota_bruta     <chr> "8.2445606938067098", "-~
$ peso_indicadores <lgl> NA, NA, NA, NA, NA, NA, ~
$ peso_pilares   <lgl> NA, NA, NA, NA, NA, NA, ~
$ peso_real      <lgl> NA, NA, NA, NA, NA, NA, ~
$ descricao      <chr> "Produto Interno Bruto (~
$ fonte_oficial  <chr> "IBGE/IBCR/CAGED", "IBGE~
$ unidade_medida <chr> "R$ bilhões", "Porcentag~
$ detalhes       <chr> "O PIB utilizado para cá~
$ nota_normalizada <dbl> 0.0746880, 0.0000000, 0.~
```

O `glimpse` informou que a base de dados possui 13.659 registros distribuídos em 15 colunas. Traz a classe da variável (numérica, caractere e etc.) e exibe alguns dados desta. Este procedimento é importante, pois nos traz elementos para avaliarmos se será necessário algum procedimento para converter tipos ou criar variáveis.

3.2. Visualizar linhas de acordó com um criterio de seleção

Podemos visualizar linhas específicas do *data.frame* utilizando a função `slice()`. Podemos informar apenas uma linha, um conjunto de linhas fora ou dentro de um intervalo, bem como trazer apenas as linhas fora de um intervalo.

A sintaxe da função é:

Nome do data.frame %>% slice(linha(s))

O operador pipe (%>%) serve para ligar o *data.frame* ao comando que vai ser executado. Nas próximas seções utilizaremos o pipe para unir uma sequência de funções que serão executadas conjuntamente.

Como exemplo, faremos quatro seleções com os seguintes critérios: i) exibir apenas a linha 10; ii) exibir apenas as linhas 2 e 7; iii) exibir as linhas 5, 6 e 7 e iv) exibir apenas as linhas fora do intervalo entre 6 e 13.659. Portanto, execute as instruções abaixo.

```
ranking %>% slice(10) # (i) exibir a linha 10
ranking %>% slice(2,7) # (ii) exibir as linhas 2 e 7
ranking %>% slice(5:7) # (iii) exibir intervalo de linhas (5 a 7)
intervalo <- c(6:13659) # criar vetor
ranking %>% slice(-intervalo) # (iv) não exibir as linhas do vetor
```

As saídas apresentadas na Figura 3, demonstram que a função slice exibe apenas partes das colunas, no entanto informa quais não foram exibidas. Caso o leitor queira ver o resultado completo, basta atribuir o resultado a um objeto e abrir o seu *data.frame*, ou seja:

```
resultado <- ranking %>% slice(10) # exibir a linha 10
```

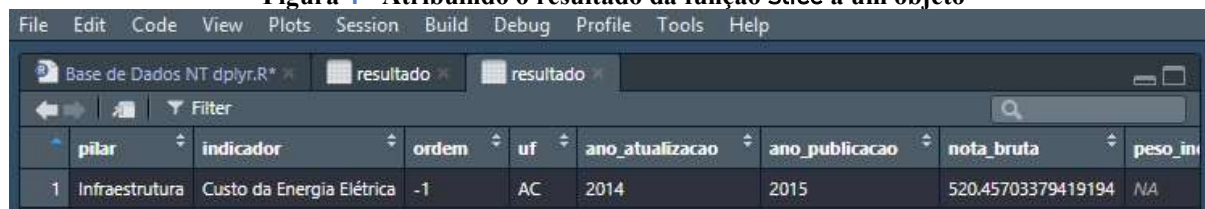
Figura 3 - 1º exemplo da função slice



Fonte: Elaborada pelos autores

O objeto **resultado** (Figura 4) herda as características do *data.frame* **meus_dados** e desta forma é possível avaliar variável por variável da linha selecionada (neste caso a linha 10). Este tipo de operação é muito útil quando se deseja trabalhar com uma amostra da população analisada.

Figura 4 - Atribuindo o resultado da função slice a um objeto



	pilar	indicador	ordem	uf	ano_atualizacao	ano_publicacao	nota_bruta	peso_in
1	Infraestrutura	Custo da Energia Elétrica	-1	AC	2014	2015	520.45703379419194	NA

Fonte: Elaborado pelos autores

Outros recursos interessantes da função *slice*, são os recursos: *head* (exibir a primeira linha), *tail* (exibir a última linha), *min* (exibir a linha que possua o menor valor de uma variável), *max* (exibir a linha que possua o maior valor de uma variável) e *sample* (exibir uma quantidade definida pelo usuário de linhas aleatórias). Execute as instruções abaixo e avalie os resultados.

```
ranking %>% slice_head()           # exibir a 1.a linha
ranking %>% slice_tail()           # exibir a última linha
# exibir linhas que possuam o menor valor para o ano de atualização
ranking %>% slice_min(ano_atualizacao)
# exibir linhas que possuam o maior valor para o ano de atualização
ranking %>% slice_max(ano_atualizacao)
ranking %>% slice_sample(n=5)      # exibir linhas aleatórias (qtde 5)
```

No caso de seleções de linhas aleatórias, ao invés da função *slice()*, podemos utilizar as funções *sample_n()* e *sample_frac()*, sendo que a primeira exibe uma quantidade de linhas informadas pelo usuário e a última exibe um percentual (definido pelo usuário) de linhas.

Por se tratarem de funções randômicas, cada vez que são executadas apresentarão resultados diferentes. Para contornar isto é necessário informar uma semente (*seed*) com um número qualquer escolhido pelo usuário antes de executar as funções.

No exemplo abaixo, após informar uma semente, serão exibidas 6 linhas aleatórias utilizando a função *sample_n* e na segunda execução do *script* serão exibidas 10% dos registros da base de dados. Reforçando, tudo isto de forma aleatória.

```
# Outra alternativa para a função slice_sample
set.seed(10) # informar semente
ranking %>% sample_n(size = 6) # (i) exibir 6 linhas aleatórias
set.seed(10) # informar semente
ranking %>% sample_frac(size = 0.1) # (ii) exibir 10% das linhas
```

Figura 5 - Outputs das funções sample

(i)

```
> set.seed(10) # informar semente
> ranking %>% sample_n(size = 6) # exibir 6 linhas aleatórias
# A tibble: 6 x 15
  pilar indic~1 ordem uf ano_a~2 ano_p~3 nota~4 peso~5 peso~6
<chr> <chr> <dbl> <chr> <dbl> <dbl> <chr> <lgl> <lgl>
1 Sust~ Mortal~ -1 DF 2013 2015 3.5633~ NA NA
2 Infr~ ACESSI~ 1 RO 2015 2016 127.71~ NA NA
3 Infr~ Custo ~ -1 BA 2019 2020 726.43~ TRUE FALSE
4 Sust~ Perda ~ -1 PR 2018 2020 35.24 FALSE TRUE
5 Soli~ Taxa d~ 1 PE 2016 2017 5.4506~ NA NA
6 Sust~ Acesso~ 1 RR 2013 2015 91.891~ NA NA
```

(ii)

```
> set.seed(10)
> ranking %>% sample_frac(size = 0.1) # exibir 10% das linhas
# A tibble: 1,366 x 15
  pilar indic~1 ordem uf ano_a~2 ano_p~3 nota~4 peso~5
<chr> <chr> <dbl> <chr> <dbl> <dbl> <chr> <lgl>
1 Sustentabil~ Mortal~ -1 DF 2013 2015 3.5633~ NA
2 Infraestrut~ ACESSI~ 1 RO 2015 2016 127.71~ NA
3 Infraestrut~ Custo ~ -1 BA 2019 2020 726.43~ TRUE
4 Sustentabil~ Perda ~ -1 PR 2018 2020 35.24 FALSE
5 Solidez Fis~ Taxa d~ 1 PE 2016 2017 5.4506~ NA
6 Sustentabil~ Acesso~ 1 RR 2013 2015 91.891~ NA
```

Fonte: Elaborada pelos autores

3.3. Filtrar registros

Para realizar filtros de seleção onde são informados alguns critérios o dplyr disponibiliza a função `filter()`. A sua sintaxe é:

Nome do data.frame %>% `filter(critério de seleção)`

Junto aos critérios, utiliza os operadores matemáticos e lógicos. Execute o *script* abaixo e analise os resultados.

```
# (i) Filtrar as linhas com a variável pilar = Infraestrutura
ranking %>% filter(pilar == "Infraestrutura")
# (ii) Filtrar as linhas com a variável ano_atualização entre 2018:2020
ranking %>% filter(between(ano_atualizacao, 2018, 2020))
# (iii) Filtrar as linhas com as colunas: Indicador = Total e UF = GO e
```

```

# ano_atualização = 2020
ranking %>% filter(indicador == "Total" &
                    uf == "GO" &
                    ano_atualizacao >= 2020)
#(iv) Filtar as linhas cuja UF sejam igual a DF ou GO
ranking %>% filter(uf == "DF" | uf == "GO")
#(v) ou
criterio <- c("DF","GO")
ranking %>% filter(uf %in% criterio)
#(vi) Filtar as linhas que a UF não contenham os dados do vetor critério
criterio <- c("AC","AL","AM","AP","BA","CE","DF",
              "ES","MA","MT","MS","MG","PA","PB",
              "PR","PE","PI","RJ","RN","RS","RO",
              "RR","SC","SP","SE","TO")
'%ni%' <- Negate("%in%")
ranking %>% filter(uf %ni% criterio)

```

Analisando os *outputs* apresentados na Figura 6, vemos que em (i) foi realizado um filtro onde os valores da variável **pilar** fossem igual ao valor “Infraestrutura”. No item (ii) , por meio do operador *between*, realizou-se um filtro dos registros, cujo valor da variável **ano_atualizacao** estivesse entre 2018 a 2020. No item (iii), o filtro foi realizado para a variável **indicador** igual a “Total”, a variável **uf** igual a “GO” e a variável **ano_atualizacao** maior ou igual a 2020.

Figura 6 - Filtros realizados na base de dados

(i)	(ii)
(iii)	(iv)
(v)	(vi)

Fonte: Elaborada pelos autores

Nos itens (iv) e (v), o filtro apresenta o mesmo resultado, a diferença é que no primeiro código utilizamos o operador lógico “ou” (!) para filtrar os registros, cujos valores da variável **uf** fossem igual a “DF” ou “GO”; já no segundo código utilizamos o operador entre (%in%) para efetuar a mesma consulta. Neste caso, utilizamos um vetor para armazenar os valores do campo **uf**. Já no item (vi), criamos uma variável que nega a condição “entre”, trazendo desta forma apenas as UF’s que que não se encontrem no vetor critério.

3.4. Ordenar registros

O ordenamento dos registros podem ser realizados por meio da função `arrange()`. Este pode ser crescente ou decrescente, bem como, podemos utilizar mais de um critério de ordenamento. A sua sintaxe é:

Nome do data.frame %>% `arrange(critério(s) de ordenamento)`

Nos códigos abaixo, temos em (i) a ordenação dos registros de forma crescente dos valores da variável **nota_bruta**. No item (ii), o ordenamento será de forma decrescente da mesma variável. Já no item (iii), o ordenamento será crescente, mas adotando três variáveis, assim, ordenará primeiro por **nota_bruta**, em seguida por **uf** e por último **ano_atualizacao**. Já no item (iv), a ordenação das variáveis **nota_bruta** e **uf** deverá ser de forma crescente e de forma decrescente a variável **ano_atualizacao**. Execute o *script* abaixo e avalie os resultados apresentados na Figura 7.

```
#(i) ordenar por nota bruta crescente
ranking %>% arrange(nota_bruta)

#(ii) ordenar por nota bruta decrescente
ranking %>% arrange(desc(nota_bruta))

#(iii) ordenar por nota bruta, uf e ano de atualização (crescente)
ranking %>% arrange(nota_bruta,uf,ano_atualizacao)

#(iv) ordenar por nota bruta, uf (crescente) e ano de atualização
#(decrescente)
ranking %>% arrange(nota_bruta,uf,desc(ano_atualizacao))
```

Figura 7 - Exemplos da função arrange

(i)										(ii)									
# A tibble: 13,659 x 15										# A tibble: 13,659 x 15									
pilar	indic~1	ordem	uf	ano_a~2	ano_p~3	nota_~4	peso_~			pilar	indic~1	ordem	uf	ano_a~2	ano_p~3	nota_~4	peso_~		
<chr>	<chr>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<lg1>			<chr>	<chr>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<lg1>		
1	Eficiência ~	Custo ~	-1	DF	2015	2016	-	NA		1	Segurança P~	Segura~	-1	PE	2018	2019	993.60~	FALSE	
2	Segurança P~	Mortes~	-1	DF	2014	2016	-	NA		2	Infraestrut~	Acessi~	1	AM	2015	2016	99.995~	NA	
3	Segurança P~	Mortes~	-1	MG	2014	2016	-	NA		3	Infraestrut~	Acesso~	1	RJ	2018	2019	99.992~	TRUE	
4	Segurança P~	Mortes~	-1	MS	2014	2016	-	NA		4	Infraestrut~	Acesso~	1	AP	2018	2019	99.985~	TRUE	
5	Segurança P~	Mortes~	-1	PB	2014	2016	-	NA		5	Infraestrut~	Acesso~	1	SP	2018	2019	99.980~	TRUE	
6	Segurança P~	Mortes~	-1	RJ	2014	2016	-	NA		6	Infraestrut~	Acesso~	1	PR	2017	2018	99.975~	TRUE	
7	Eficiência ~	Custo ~	-1	RR	2015	2016	-	NA		7	Infraestrut~	Acesso~	1	SP	2017	2018	99.975~	TRUE	
8	Segurança P~	Mortes~	-1	RS	2014	2016	-	NA		8	Infraestrut~	Acesso~	1	RJ	2017	2018	99.974~	TRUE	
9	Segurança P~	Mortes~	-1	SC	2014	2016	-	NA		9	Infraestrut~	Acesso~	1	PB	2017	2018	99.969~	TRUE	
10	Segurança P~	Mortes~	-1	SP	2014	2016	-	NA		10	Infraestrut~	Acesso~	1	SC	2017	2018	99.964~	TRUE	

(iii)										(iv)									
# A tibble: 13,659 x 15										# A tibble: 13,659 x 15									
pilar	indic~1	ordem	uf	ano_a~2	ano_p~3	nota_~4	peso_~			pilar	indic~1	ordem	uf	ano_a~2	ano_p~3	nota_~4	peso_~		
<chr>	<chr>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<lg1>			<chr>	<chr>	<dbl>	<chr>	<dbl>	<dbl>	<chr>	<lg1>		
1	Segurança P~	Mortes~	-1	DF	2014	2016	-	NA		1	Eficiência ~	Custo ~	-1	DF	2019	2020	-	FALSE	
2	Eficiência ~	Custo ~	-1	DF	2015	2016	-	NA		2	Eficiência ~	Custo ~	-1	DF	2017	2018	-	FALSE	
3	Eficiência ~	Custo ~	-1	DF	2016	2017	-	NA		3	Eficiência ~	Custo ~	-1	DF	2016	2017	-	NA	
4	Eficiência ~	Custo ~	-1	DF	2017	2018	-	FALSE		4	Eficiência ~	Custo ~	-1	DF	2015	2016	-	NA	
5	Eficiência ~	Custo ~	-1	DF	2019	2020	-	FALSE		5	Segurança P~	Mortes~	-1	DF	2014	2016	-	NA	
6	Segurança P~	Atuaçã~	1	MA	2018	2020	-	TRUE		6	Segurança P~	Atuaçã~	1	MA	2018	2020	-	TRUE	
7	Segurança P~	Mortes~	-1	MG	2014	2016	-	NA		7	Segurança P~	Mortes~	-1	MG	2014	2016	-	NA	
8	Segurança P~	Mortes~	-1	MS	2014	2016	-	NA		8	Segurança P~	Mortes~	-1	MS	2014	2016	-	NA	
9	Segurança P~	Mortes~	-1	PB	2014	2016	-	NA		9	Segurança P~	Mortes~	-1	PB	2014	2016	-	NA	
10	Segurança P~	Mortes~	-1	RJ	2014	2016	-	NA		10	Segurança P~	Mortes~	-1	RJ	2014	2016	-	NA	

Fonte: Elaborada pelos autores

O dplyr é muito flexível e permite que possamos combinar funções do pacote. No exemplo abaixo, os dados foram agrupados por ano de atualização (com a função `group_by`) e na sequência foi feita uma ordenação crescente por unidade da federação. Note que, neste caso, foram utilizados dois pipes, um para ligar o *data.frame* com a função `group_by` e outro para ligar a função `group_by` com a função `arrange`.

```
# Agrupando os dados por ano de atualização e UF
```

```
ranking %>% group_by(ano_atualizacao) %>%
  arrange(uf)
```

Figura 8 - Combinação da função group_by e arrange

```
# A tibble: 13,659 x 15
# Groups:   ano_atualizacao [10]
  pilar      indic~1 ordem uf      ano_a~2 ano_p~3 nota_~4 peso_~5
  <chr>      <chr>   <dbl> <chr>   <dbl>   <dbl> <chr>   <lgl>
1 Potencial d~ Tamanh~    1 AC     2014     2015 8.2445~ NA
2 Potencial d~ Taxa d~    1 AC     2014     2015 -0.526~ NA
3 Potencial d~ Total    1 AC     2015     2015 3.2701~ NA
4 Infraestrut~ Acessi~    1 AC     2014     2015 122.25~ NA
5 Infraestrut~ Custo ~   -1 AC     2014     2015 3.1900~ NA
6 Infraestrut~ Custo ~   -1 AC     2013     2015 1.66    NA
7 Infraestrut~ Custo ~   -1 AC     2014     2015 0.3056~ NA
8 Infraestrut~ Dispon~    1 AC     2014     2015 2000    NA
9 Infraestrut~ Acesso~    1 AC     2013     2015 93.607~ NA
10 Infraestrut~ Custo ~   -1 AC     2014     2015 520.45~ NA
```

Fonte: Elaborada pelos autores

Conforme a Figura 8, o dplyr, primeiramente realizou o agrupamento dos dados por ano de atualização e em seguida efetuou o filtro da base agrupada por UF.

3.5. Selecionar colunas

A função `select( )` permite selecionar registros por meio dos nomes das colunas do *data.frame*. No *output* só aparece a(s) coluna(s) selecionada(s). A sua sintaxe é:

Nome do *data.frame* %>% `select(nome da(s) coluna(s))`

Além dos nomes das colunas, a seleção pode ser feita pelo número da ordem em que estão dispostas na base de dados.

Os passos dos próximos 4 exemplos são: i) selecionar a variável **pilar**; ii) selecionar as colunas **pilar**, **uf** e **nota bruta**; iii) selecionar colunas pelo número da ordem que aparecem na base de dados; e iv) selecionar as colunas que estão fora de um vetor de restrições.

```
#(i) Selecionar a coluna pilar
ranking %>% select(pilar)

#(ii) Selecionar as colunas pilar, uf e nota bruta
ranking %>% select(pilar, uf, nota_bruta)

#(iii) Selecionar as 5 primeiras colunas da base
ranking %>% select(1:5)

#(iv) Selecionar as colunas que não estão no vetor de restrições
```

```

criterio<-c("ano_publicacao","peso_indicadores",
            "peso_pilares","peso_real",
            "descricao","fonte_oficial",
            "unidade_medida","detalhes",
            "nota_normalizada")
ranking %>% select(-criterio)

```

Repare que conforme a Figura 9, na seleção (i), ao contrário do que vimos nas funções slice e filter, a função select só exibiu a coluna selecionada, neste caso a **pilar**. No item (ii), são exibidas apenas as colunas **pilar**, **uf** e **nota_bruta**. No item (iii), são exibidas as colunas **pilar**, **indicador**, **ordem**, **uf** e **ano_atualizacao** que são as primeiras colunas da base de dados. No item (iv), são exibidas as colunas **pilar**, **indicador**, **ordem**, **uf**, **ano_atualizacao** e **nota_brutas**, colunas estas que estão fora do vetor com as restrições.

Figura 9 - Exemplos de aplicações da função select

(i) <pre> > ranking %>% select(pilar) # A tibble: 13,659 x 1 pilar <chr> 1 Potencial de Mercado 2 Potencial de Mercado 3 Potencial de Mercado 4 Infraestrutura 5 Infraestrutura 6 Infraestrutura 7 Infraestrutura 8 Infraestrutura 9 Infraestrutura 10 Infraestrutura # ... with 13,649 more rows # 1 Use "print(n = ...)" to see more rows </pre>	(ii) <pre> > ranking %>% select(pilar, uf, nota_bruta) # A tibble: 13,659 x 3 pilar uf nota_bruta <chr> <chr> <dbl> 1 Potencial de Mercado AC 8.2445606938067098 2 Potencial de Mercado AC -0.52658035046457807 3 Potencial de Mercado AC 3.2701046518785991E-2 4 Infraestrutura AC 122.25644327913299 5 Infraestrutura AC 3.19000796539691 6 Infraestrutura AC 1.66 7 Infraestrutura AC 0.30564731417464702 8 Infraestrutura AC 2000 9 Infraestrutura AC 93.607305936073104 10 Infraestrutura AC 520.45703379419194 </pre>
(iii) <pre> > ranking %>% select(1:5) # A tibble: 13,659 x 5 pilar indicador ordem uf ano_a-1 <chr> <chr> <dbl> <chr> <dbl> 1 Potencial de Mercado Tamanho de mercado 1 AC 2014 2 Potencial de Mercado Taxa de crescimento 1 AC 2014 3 Potencial de Mercado Total 1 AC 2015 4 Infraestrutura Acessibilidade do S~ 1 AC 2014 5 Infraestrutur~ Custo de Combustive~ - AC 2014 6 Infraestrutur~ Custo de Saneamento~ - AC 2013 7 Infraestrutur~ Custo de Telecomuni~ - AC 2014 8 Infraestrutur~ Disponibilidade de ~ 1 AC 2014 9 Infraestrutur~ Acesso à Energia El~ 1 AC 2013 10 Infraestrutur~ Custo da Energia El~ - AC 2014 </pre>	(iv) <pre> > ranking %>% select(-criterio) # A tibble: 13,659 x 6 pilar indicador ordem uf ano_a-1 nota_~2 <chr> <chr> <dbl> <chr> <dbl> <chr> 1 Potencial de Mercado Tamanho de ~ 1 AC 2014 8.2445~ 2 Potencial de Mercado Taxa de cre~ 1 AC 2014 -0.526~ 3 Potencial de Mercado Total 1 AC 2015 3.2701~ 4 Infraestrutur~ Acessibilid~ 1 AC 2014 122.25~ 5 Infraestrutur~ Custo de Co~ - AC 2014 3.1900~ 6 Infraestrutur~ Custo de Sa~ - AC 2013 1.66 7 Infraestrutur~ Custo de Te~ - AC 2014 0.3056~ 8 Infraestrutur~ Disponibili~ 1 AC 2014 2000 9 Infraestrutur~ Acesso à En~ 1 AC 2013 93.607~ 10 Infraestrutur~ Custo da En~ - AC 2014 520.45~ </pre>

Fonte: Elaborada pelos autores

Também pode-se utilizar o select para selecionar colunas que iniciem ou terminem os seus rótulos com determinada sequência de caracteres, bem como que contenham estas sequências independentemente da sua posição. Execute as instruções abaixo e avalie os resultados.

```

#(i) Selecionar todas as colunas que comecem com peso
ranking %>% select(starts_with("peso"))

#(ii) Selecionar todas as colunas que não comecem com peso
ranking %>% select(-starts_with("peso"))

#(iii) Selecionar as colunas que terminam com "cao"
ranking %>% select(ends_with("cao"))

```

```
#(iv) Selecionar todas colunas que contém a expressão no_
ranking %>% select(contains("no_"))
```

De acordo com a Figura 10, no item (i), foram selecionadas as colunas **peso_indicadores**, **peso_pilares** e **peso_real**, todas começando com a sequência “peso” de acordo com o atributo `starts_with()`. Já no item (ii) aparecem todas as colunas que não começam com sequência “peso”. No item (iii) são exibidas todas as colunas que terminam com “cao”, conforme a cláusula `ends_with()`, a saber: **ano_atualizacao**, **ano_publicacao** e **descricao**. Por fim, no item (iv), são exibidas as colunas **ano_atualizacao** e **ano_publicacao**, de acordo com a instrução `contains()`, diante disto trouxe todos os resultados cujas colunas tenham em alguma posição dos seus rótulos a sequência de caracteres: “no_”.

Figura 10 - Selecionado colunas por seqüências de caracteres

<pre># A tibble: 13,659 x 3 peso_indicadores peso_pilares peso_real <lgl> <lgl> <lgl> 1 NA NA NA 2 NA NA NA 3 NA NA NA 4 NA NA NA 5 NA NA NA 6 NA NA NA 7 NA NA NA 8 NA NA NA 9 NA NA NA 10 NA NA NA # ... with 13,649 more rows</pre>	<pre># A tibble: 13,659 x 12 pilar indic-1 ordem uf ano_a-2 ano_p-3 nota_-4 descr-5 <chr> <chr> <dbl> <chr> <dbl> <dbl> <chr> <chr> 1 Potencia~ Tamanh~ 1 AC 2014 2015 8.2445~ Produ~ 2 Potencia~ Taxa d~ 1 AC 2014 2015 -0.526~ Média ~ 3 Potencia~ Total 1 AC 2015 2015 3.2701~ Nota o~ 4 Infraest~ Acesso~ 1 AC 2014 2015 122.25~ Densid~ 5 Infraest~ Custo ~ -1 AC 2014 2015 3.1900~ Preço ~ 6 Infraest~ Custo ~ -1 AC 2013 2015 1.66 ~ Tarifa~ 7 Infraest~ Custo ~ -1 AC 2014 2015 0.3056~ Custo ~ 8 Infraest~ Dispon~ 1 AC 2014 2015 2000 ~ Número~ 9 Infraest~ Acesso~ 1 AC 2013 2015 93.607~ Propor~ 10 Infraest~ Custo ~ -1 AC 2014 2015 520.45~ Tarifa~</pre>
<pre># A tibble: 13,659 x 3 ano_atualizacao ano_publicacao descricao <dbl> <dbl> <chr> 1 2014 2015 Produto Interno Bruto (PIB) r~ 2 2014 2015 Média Móvel de cinco períodos~ 3 2015 2015 Nota obtida via normalização ~ 4 2014 2015 Densidade de acessos por 100 ~ 5 2014 2015 Preço pago em reais por consu~ 6 2013 2015 Tarifa média praticada pelo s~ 7 2014 2015 Custo dos serviços de telecom~ 8 2014 2015 Número de voos diretos 9 2013 2015 Proporção de domicílios com a~ 10 2014 2015 Tarifa média praticada para o~ # ... with 13,649 more rows</pre>	<pre># A tibble: 13,659 x 2 ano_atualizacao ano_publicacao <dbl> <dbl> 1 2014 2015 2 2014 2015 3 2015 2015 4 2014 2015 5 2014 2015 6 2013 2015 7 2014 2015 8 2014 2015 9 2013 2015 10 2014 2015 # ... with 13,649 more rows</pre>

Fonte: Elaborado pelos autores (2022)

É possível efetuar combinações de funções, utilizando o comando `select`, desde que se use o pipe para conectar as instruções. No exemplo abaixo, será realizado um filtro para o ano, de atualização, igual a 2020 e o valor do **pilar** igual a “Ranking Geral”, em seguida são selecionadas as colunas **ano_atualização**, **uf** e **nota_bruta**. Por fim, os dados são organizados de forma decrescente por **nota_bruta**. O resultado é apresentado na Figura 11.

```
### Combinações de funções
ranking %>% filter(ano_atualizacao == 2020 & pilar == "Ranking Geral") %>%
  select(ano_atualizacao,uf,nota_bruta) %>%
  arrange(desc(nota_bruta))
```

Figura 11 - Seleção com combinação de funções

```
# A tibble: 29 x 3
```

	ano_atualizacao	uf	nota_bruta
	<dbl>	<chr>	<chr>
1	2020	SP	86.408394530638859
2	2020	SC	79.484523978533389
3	2020	DF	70.053543130928972
4	2020	PR	69.937102657976439
5	2020	ES	62.763139554402876
6	2020	MS	61.383627198634031
7	2020	MG	55.818038765491714
8	2020	RS	54.676164345684157
9	2020	MT	51.526158637887775
10	2020	CE	49.508231761427098

Fonte: Elaborada pelos autores

3.6. Criar variáveis

O dplyr permite a modificação dos tipos e criação de novas variáveis na base de dados por meio da função `mutate()` cuja a sintaxe é:

Nome do data.frame %>% `mutate(nome da variável criada = fórmula matemática)`

Antes de detalhar a função `mutate`, exclui-se da base de dados os registros, cuja a variável **uf** possuam os valores NE* e BR*, para isto informe o código abaixo (a avaliação dos resultados ficará por conta do leitor):

```
# Remover os registros cuja uf seja NE* e BR*
ranking <- ranking %>% filter(uf != "NE*" & uf != "BR*" &
                             indicador == "Total") %>%
  select(1:15)
```

Execute a instrução: `glimpse(ranking)`, o campo **nota bruta** é do tipo caractere (chr). Vamos converter esta variável para um tipo numérico usando a função `mutate`.

```
glimpse(ranking)
# Modificar a coluna nota_bruta (chr) para tipo numérico
ranking <- ranking %>% mutate(nota_bruta = as.numeric(nota_bruta))
glimpse(ranking)
```

Figura 12 - Mudança de tipo da coluna `nota_bruta`

Antes da conversão do tipo da nota bruta (i)		Após a conversão do tipo da nota bruta (ii)	
ROWS: 1,782		ROWS: 1,782	
COLUMNS: 15		COLUMNS: 15	
\$ pilar	<chr> "Potencial de Mercado", "Infraestrutu~	\$ pilar	<chr> "Potencial de Mercado", "Infraestrutu~
\$ indicador	<chr> "Total", "Total", "Total", "Total", "~	\$ indicador	<chr> "Total", "Total", "Total", "Total", "~
\$ ordem	<dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1	\$ ordem	<dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
\$ uf	<chr> "AC", "AC", "AC", "AC", "AC", "AC", "~	\$ uf	<chr> "AC", "AC", "AC", "AC", "AC", "AC", "~
\$ ano_atualizacao	<dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2~	\$ ano_atualizacao	<dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2~
\$ ano_publicacao	<dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2~	\$ ano_publicacao	<dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2~
\$ nota_bruta	<chr> "3.2701046518785991E-2", "27.89281496~	\$ nota_bruta	<dbl> 0.03270105, 27.89281497, 45.77492357,~
\$ peso_indicadores	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~	\$ peso_indicadores	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ peso_pilares	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~	\$ peso_pilares	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ peso_real	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~	\$ peso_real	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ descricao	<chr> "Nota obtida via normalização e ponde~	\$ descricao	<chr> "Nota obtida via normalização e ponde~
\$ fonte_oficial	<chr> "Tendências", "Tendências", "Tendênci~	\$ fonte_oficial	<chr> "Tendências", "Tendências", "Tendênci~
\$ unidade_medida	<chr> "Nota de 0 a 100", "Nota de 0 a 100", "~	\$ unidade_medida	<chr> "Nota de 0 a 100", "Nota de 0 a 100", "~
\$ detalhes	<chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~	\$ detalhes	<chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ nota_normalizada	<dbl> 0.000000, 0.000000, 25.184472, 50.177~	\$ nota_normalizada	<dbl> 0.000000, 0.000000, 25.184472, 50.177~

Fonte: Elaborada pelos autores

Na Figura 12, observa-se que, após a execução da função `mutate`, a variável `nota_bruta` é do tipo numérico (`dbl`).

Aproveitando esta modificação, cria-se uma variável chamada `nota_perc` que receberá o valor da variável `nota_bruta` dividida por 100. O resultado é apresentado na Figura 13.

```
# criar a variável nota_perc
ranking <- ranking %>% mutate(nota_perc = nota_bruta/100)
glimpse(ranking)
```

Figura 13 - Variáveis da base de dados (nomes e tipos)

ROWS: 1,782	
COLUMNS: 16	
\$ pilar	<chr> "Potencial de Mercado", "Infraestrutu~
\$ indicador	<chr> "Total", "Total", "Total", "Total", "~
\$ ordem	<dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
\$ uf	<chr> "AC", "AC", "AC", "AC", "AC", "AC", "~
\$ ano_atualizacao	<dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2~
\$ ano_publicacao	<dbl> 2015, 2015, 2015, 2015, 2015, 2015, 2~
\$ nota_bruta	<dbl> 0.03270105, 27.89281497, 45.77492357,~
\$ peso_indicadores	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ peso_pilares	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ peso_real	<lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ descricao	<chr> "Nota obtida via normalização e ponde~
\$ fonte_oficial	<chr> "Tendências", "Tendências", "Tendênci~
\$ unidade_medida	<chr> "Nota de 0 a 100", "Nota de 0 a 100", "~
\$ detalhes	<chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
\$ nota_normalizada	<dbl> 0.000000, 0.000000, 25.184472, 50.177~
\$ nota_perc	<dbl> 0.0003270105, 0.2789281497, 0.4577492~
>	

Fonte: Elaborada pelos autores

3.7. Resumos estatísticos

A função `summarize()` permite elaborar resumos estatísticos das variáveis da base de dados. É possível combiná-la também com as demais funções do `dplyr`. A seguir é apresentado alguns exemplos usando a base de dados descrita anteriormente.

Exemplo 1: Um resumo estatístico da variável **nota_bruta** é apresentado. Serão dados nomes para as medidas calculadas, para que possam ser impressos posteriormente. Além disso, será adicionado o comando `na.rm = TRUE`, cujo objetivo é desconsiderar os valores nulos.

```
# resumo estatístico da variável
ranking %>% summarize(
  media = mean(nota_bruta, na.rm = TRUE),
  mediana = median(nota_bruta, na.rm = TRUE),
  variancia = var(nota_bruta, na.rm = TRUE),
  desvio.padrao = sd(nota_bruta, na.rm = TRUE)
```

Exemplo 2: A base de dados é agrupada por **uf** e em seguida é apresentado um resumo estatístico da mesma. O resultado está na Tabela 2.

```
# resumo estatístico agrupado por uf
ranking %>% group_by(uf) %>%
  summarize(
    minimo = min(nota_bruta, na.rm = TRUE),
    maximo = max(nota_bruta, na.rm = TRUE),
    media = mean(nota_bruta, na.rm = TRUE),
    mediana = median(nota_bruta, na.rm = TRUE),
    variancia = var(nota_bruta, na.rm = TRUE),
    desvio.padrao = sd(nota_bruta, na.rm = TRUE))
```

Tabela 2 - Resumo estatístico por UF (resultados parciais)

uf	minimo	maximo	media	mediana	variancia	desvio.padrao
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
AC	0.0327	73.3	37.9	37.2	241.	15.5
AL	7.37	70.0	39.3	42.8	310.	17.6
AM	11.5	71.2	44.6	45.0	228.	15.1
AP	0.261	66.8	38.4	40.9	261.	16.2
BA	9.90	71.4	43.4	45.3	233.	15.3
CE	15.6	77.5	49.6	49.6	277.	16.6
DF	25.6	89.2	61.7	66.3	276.	16.6
ES	12.0	81.0	55.4	57.8	371.	19.3
GO	10.6	78.7	50.3	52.8	278.	16.7
MA	7.80	68.7	38.0	34.2	295.	17.2

Fonte: Elaborada pelos autores

3.8. Juntar tabelas

Com a função `inner_join()` é possível juntar bases dados, desde que elas tenham algum elemento em comum, ou seja, uma chave primária (*primary key*). No próximo exemplo, esta chave será a variável `uf`.

Exemplo 3: Crie um *data.frame* com dados do Produto Interno Bruto (PIB) do ano de 2019. Execute as instruções abaixo.

```
# unidades federativas
uf<- ("AC", "AL", "AP", "AM", "BA", "CE", "DF", "ES", "GO", "MA", "MT", "MS", "MG", "PR",
      "PB", "PA", "PE", "PI", "RJ", "RN", "RS", "RO", "RR", "SC", "SE", "SP", "TO")
# anoano <- rep(2019,27)
# PIB UF's do ano de 2019
pib <-c(15630,58964,17497,108181,293241,163575,273614,137346,208672,
        97340,142122,106943,651873, 466377,67986,178377,197853,52781,
        779928,71337,482464,47091,14292,323264,44689,2348338,39356)

# data.frame PIB.2019
PIB.2019<- tibble(uf,ano,PIB)
```

Agora, crie um *data.frame* que irá receber a junção dos filtros e seleções do *ranking* com a junção destes com o PIB 2019. O resultado é apresentado na Figura 14.

```
# juntar base de dados
ranking.2 <- ranking %>% filter(ano_publicacao == 2019 &
                              pilar == "Ranking Geral") %>%
  select(uf,nota_bruta) %>%
  inner_join(PIB.2019)
# visualizar base de dados
print(ranking.2, n=27)
```

Figura 14 – Resultado da função inner join

```
> print(ranking.2, n=27)
# A tibble: 27 x 4
  uf      nota_bruta    ano    PIB
<chr>    <dbl>    <dbl>    <dbl>
1 AC      30.2    2019    15630
2 AL      48.2    2019    58964
3 AM      43.3    2019   108181
4 AP      34.4    2019    17497
5 BA      36.9    2019   293241
6 CE      49.6    2019   163575
7 DF      71.8    2019   273614
8 ES      62.4    2019   137346
9 GO      49.0    2019   208672
10 MA      31.2    2019    97340
11 MG      54.3    2019   651873
12 MS      64.3    2019   106943
13 MT      52.8    2019   142122
14 PA      34.1    2019   178377
15 PB      49.6    2019    67986
16 PE      42.5    2019   197853
17 PI      35.0    2019    52781
18 PR      71.0    2019   466377
19 RJ      50.2    2019   779928
20 RN      47.0    2019    71337
21 RO      42.1    2019    47091
22 RR      36.6    2019    14292
23 RS      55.1    2019   482464
24 SC      74.4    2019   323264
25 SE      35.4    2019    44689
26 SP      87.4    2019  2348338
27 TO      41.7    2019    39356
```

Fonte: Elaborada pelos autores

4. Considerações finais

Neste trabalho, foi ilustrado comandos do pacote dplyr, através dos dados referentes ao ranking de competitividade dos estados brasileiros. Observou-se que, este pacote é flexível e robusto nos procedimentos de filtros, seleções, resumos estatísticos e junções de tabelas.

Entretanto, esta nota técnica não deve ser vista como um manual ou uma apostila sobre o dplyr. O seu caráter é meramente introdutório, visto que o pacote possui algumas dezenas de funcionalidades que não foram exploradas aqui. Cabe ao leitor buscar fontes complementares de aprendizagem. Apesar de ser um pacote relativamente jovem, o R possui diversas publicações (livros, artigos e etc) e grupos de discussões, que podem ajudar nesta empreitada.

Referências bibliográficas

Hadley Wickham, Romain François, Lionel Henry and Kirill Müller (2022). dplyr: A Grammar of Data Manipulation. R package version 1.0.10. Disponível em: <https://CRAN.R-project.org/package=dplyr>. Acesso em 21 nov. 2022.

IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Produto Interno Bruto. Disponível em: <https://www.ibge.gov.br/explica/pib.php>. Acesso em 21 nov. 2022.

Ranking competitividade entre os estados. Disponível em : https://dados.al.gov.br/catalogo/dataset/636cb783-4565-4ca3-a2a1-0222c8107ed2/resource/332287ac-81b8-4595-b814-64fac8305381/download/ranking_2015_2020.xlsx. Acesso em 21 nov. 2022.

Sobre o Ranking CLP. Disponível em: <https://www.rankingdecompetitividade.org.br/quem-somos>. Acesso em 21 nov. 2022.