

# DO SQL PARA O R USANDO O PACOTE dplyr



# Olá!

## Sou Maria Marinho

- Formada em Processamento de Dados, bacharela e licenciada em Matemática, pós graduada em Educação Matemática
- Cientista de dados na SulAmérica
- Participante das R-Ladies e PyLadies São Paulo

 [mariamarinhos@gmail.com](mailto:mariamarinhos@gmail.com)

 [@mariaeme](https://t.me/mariaeme)

 [maryms](https://github.com/maryms)



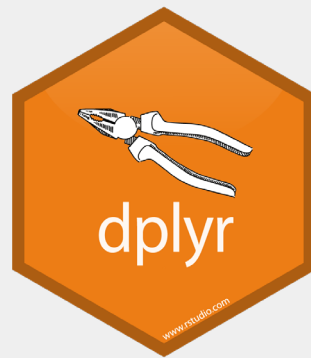
**NerdZão**

## Por que SQL e R?

- Primeiro contato com o **R** Base: **2015**. Quantos colchetes!
- Python surgiu na vida, o R foi para o congelador...
- **R-Ladies** São Paulo: 1º meetup em 08/**2018**
- Ganhei uma bolsa de estudos do **Curso-R**: 10/2018, daí...
- Conheci o outro lado do R com o **Tidyverse**
- **Aprendi** o pacote **dplyr** fazendo **assimilações** com os meus conhecimentos prévios em **SQL**
- **R is back, baby!**

## Objetivo

- A ideia é compartilhar um “de-para” bem legal dos comandos SQL para o R com o pacote dplyr.



# Tidyverse

- É uma coleção de pacotes R projetados para Ciência de Dados. Todos os pacotes compartilham uma mesma filosofia de desenvolvimento, sintaxe e estruturas de dados.



- A ideia do pacote **dplyr** é tornar a manipulação de dados explícita utilizando verbos que indicam a ação a ser realizada.



## Os 6 verbos do dplyr

- `filter()`: seleciona linhas
- `arrange()`: ordena de acordo com uma ou mais colunas
- `select()`: seleciona colunas
- `mutate()`: cria/modifica colunas
- `summarise()`: sumariza/agrega colunas
- `group_by()`: agrupa colunas

## Dataset: Índice Geral de Reclamações recebidas pela ANS

- O IGR têm como base os dados de beneficiários e reclamações recebidos pela ANS e permitem comparar a atuação das empresas que atuam no setor de Saúde Suplementar.

Fonte:

- <http://dados.gov.br/dataset/indice-de-reclamacoes>
- <http://dados.gov.br/dataset/indice-de-reclamacoes/resource/c621fc2d-a4e6-43b3-afb4-9143c36f4d6c>

# No RStudio, no banco SQL

## #Carrega biblioteca e importa o arquivo no R

```
library(tidyverse)
```

```
igr <- read_csv('dados/dados-gerais-das-reclamacoes-por-operadora.csv',  
locale=locale(encoding='Latin1'))
```

```
> igr  
# A tibble: 122,545 x 10  
  registro_ans razao_social beneficiarios numero_demanda data_atendimento classificacao natureza_demanda subtema_demanda competencia data_atualizacao  
  <chr>        <chr>          <int>      <int> <chr>          <chr>      <chr>      <chr>      <int> <chr>  
1 353663      UNIMED NORT~      56931    4092889 01/11/2018 00:1~ INATIVA    Assistencial Reembolso      201910 21/11/2019 08:2~  
2 366871      PETRÓLEO BR~      261887    4092890 01/11/2018 00:3~ INATIVA    Não Assistencial Demitidos, Exo~ 201910 21/11/2019 08:2~  
3 005711      BRADESCO SA~     3356419    4092891 01/11/2018 07:2~ INATIVA    Não Assistencial Suspensão e Re~ 201910 21/11/2019 08:2~  
4 006246      SUL AMERICA~     2637229    4092892 01/11/2018 07:3~ INATIVA    Não Assistencial Mensalidade ou~ 201910 21/11/2019 08:2~
```

- ✓ Para os comandos SQL dos próximos slides, o nome assumido da tabela é **igr**.



**1.**

**SELECT \***



View

## SELECT \* ... View

### SQL

```
SELECT * FROM igr
```

### dplyr

```
View(igr)
```

```
#ou
```

```
igr %>% View
```

%>%

- Pense no Pipe %>% como um operador que efetua as operações à direita nos valores que estão à esquerda.

igr %>% View

- Ou ainda, o operador %>% passa o que está à esquerda como argumento para a operação da direita.
- **Atalho:** CTRL + SHIFT + M

2.

SELECT



select

## SELECT ... select

### SQL

```
SELECT razao_social, beneficiarios  
FROM igr
```

### dplyr

```
igr %>%  
  select(razao_social, beneficiarios)
```

3.

# SELECT DISTINCT



distinct

## SELECT DISTINCT ... distinct

### SQL

```
SELECT DISTINCT razao_social  
FROM igr
```

### dplyr

```
igr %>% distinct(razao_social) %>% View
```



Para visualizar os dados além do console, use `%>% View` ao final das instruções.

4.

SELECT TOP





## SELECT TOP ... head

### SQL

```
SELECT TOP 200 FROM igr
```

### dplyr

```
igr %>% head(200)
```

**5.**

# **OPERADORES**

# Operadores

Operador	SQL	R
Igual	=	==
Diferente	<>	!=
Maior que	>	>
Menor que	<	<
Maior ou igual que	>=	>=
Menor ou igual que	<=	<=
E	AND	&
OU	OR	
Negação	NOT	!

6.

WHERE



filter

## WHERE ... filter

### SQL

```
SELECT razao_social, beneficiarios  
FROM igr  
WHERE beneficiarios > 1000000
```

### dplyr

```
igr %>%  
  select(razao_social, beneficiarios) %>%  
  filter(beneficiarios > 1000000)
```

7.

LIKE



str\_detect

## LIKE ... str\_detect

### SQL

```
SELECT * FROM igr  
WHERE razao_social LIKE "%SAUDE%"
```

### dplyr

```
igr %>%  
  filter(str_detect(razao_social,  
    "SAUDE"))
```

8.

IN...NOT IN



%in% ... %!in%



**IN ... %in%**

## SQL

```
SELECT * FROM igr  
WHERE subtema_demanda IN  
("Reembolso", "Carência")
```

## dplyr

```
igr %>%  
  filter(subtema_demanda %in%  
c("Reembolso", "Carência"))
```

## NOT IN ... %!in%

### SQL

```
SELECT * FROM igr  
WHERE subtema_demanda NOT IN  
("Reembolso", "Carência")
```

### dplyr

#É preciso criar uma função para  
negar o operador in:

```
`%!in%` = Negate(`%in%`)
```

```
igr %>%
```

```
  filter(subtema_demanda %!in%  
c("Reembolso", "Carência"))
```

**9.**

# **FUNÇÕES DE AGREGAÇÃO**

## MIN, MAX, AVG ... summarize(min, max, mean)

summarise também funciona!

### SQL

```
SELECT MIN(beneficiarios) FROM igr
```

```
SELECT MAX(beneficiarios) FROM igr
```

```
SELECT AVG(beneficiarios) FROM igr
```

### dplyr

```
igr %>%  
  summarize(min(beneficiarios))
```

```
igr %>%  
  summarize(max(beneficiarios))
```

```
igr %>%  
  summarize(mean(beneficiarios))
```

## COUNT, COUNT DISTINCT ... n, n\_distinct

### SQL

```
SELECT COUNT(*) FROM igr
```

```
SELECT COUNT  
(DISTINCT(razao_social)) FROM igr
```

### dplyr

```
#Conta todos as observações
```

```
igr %>% summarize(n())
```

```
#Conta valores distintos da coluna
```

```
igr %>%  
summarize(n_distinct(razao_social))
```

**10.**

**IS NULL...IS NOT NULL**



is.na ... !is.na

## IS NULL ... is.na

### SQL

```
SELECT * FROM igr  
WHERE registro_ans IS NULL
```

### dplyr

```
igr %>% filter(is.na(registro_ans))
```

## IS NOT NULL ... !is.na

### SQL

```
SELECT * FROM igr  
WHERE registro_ans IS NOT NULL
```

### dplyr

```
igr %>% filter(!is.na(registro_ans))
```



**11.**

# GROUP BY



group\_by

## GROUP BY ... group\_by

### SQL

```
SELECT razao_social,  
count(razao_social) as num_ben  
FROM igr  
GROUP BY razao_social
```

### dplyr

```
igr %>%  
  group_by(razao_social) %>%  
  summarize(num_ben = n())
```

**12.**

**ORDER BY**



arrange

## ORDER BY ... arrange

### SQL

```
SELECT razao_social,  
count(razao_social) as qtd  
FROM igr  
GROUP BY razao_social  
ORDER BY qtd
```

### dplyr

```
igr %>%  
  group_by(razao_social) %>%  
  summarize(qtd = n()) %>%  
  arrange(qtd)
```

## ORDER BY DESC ... arrange(desc

### SQL

```
SELECT razao_social,  
count(razao_social) as qtd  
FROM igr  
GROUP BY razao_social  
ORDER BY qtd DESC
```

### dplyr

```
igr %>%  
  group_by(razao_social) %>%  
  summarize(qtd = n()) %>%  
  arrange(desc(qtd))
```

**13.**

**INSERT**



add\_row

## UPDATE ... add\_row

### SQL

```
INSERT INTO igr  
VALUES("999999", "dado2", 3...)
```

### dplyr

```
igr <- igr %>%  
add_row(registro_ans = "999999",  
razao_social = "dado2",  
beneficiarios = 3...)
```

registro\_ans e razao\_social  
são *character* e  
beneficiarios é *integer*

**14.**

**UPDATE**



mutate



# UPDATE ... mutate

## SQL

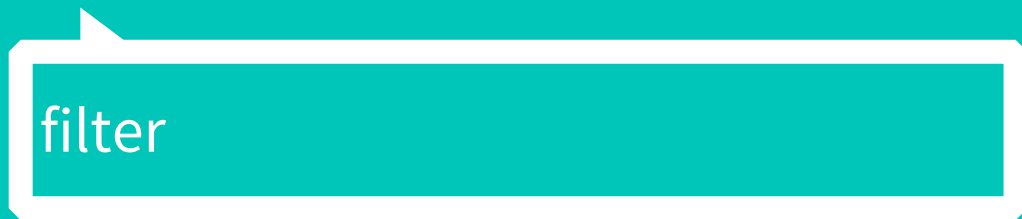
```
UPDATE igr
SET razao_social = "plano_saude_ABC",
    beneficiarios= 101101
WHERE registro_ans = "999999"
```

## dplyr

```
igr <- igr %>%
  mutate(razao_social=if_else(
    registro_ans=="999999","plano_saude_ABC",
    razao_social)) %>%
  mutate(beneficiarios= if_else
    (registro_ans=="999999", 101101,
    beneficiarios))
```

**15.**

**DELETE**



filter

## DELETE ... filter

### SQL

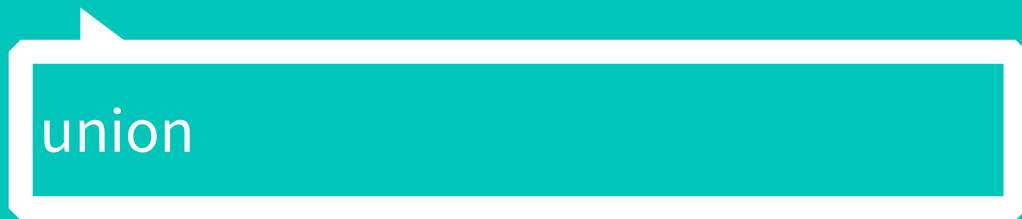
```
DELETE FROM igr  
WHERE subtema_demanda <>  
"Reembolso"
```

### dplyr

```
igr <- igr %>%  
filter(subtema_demanda !=  
"Reembolso")
```

**16.**

**UNION**



## UNION ... union

### SQL

```
SELECT * FROM igr_2018  
UNION  
SELECT * FROM igr_2019
```

### dplyr

#union: colunas iguais, remove  
observações duplicadas.

```
new_igr <- igr_2018 %>%  
  union(igr_2019)
```

## UNION\_ALL ... union\_all

### SQL

```
SELECT * FROM igr_2018  
UNION_ALL  
SELECT * FROM igr_2019
```

### dplyr

#union\_all: colunas diferentes, não  
remove observações duplicadas.

```
new_igr <- igr_2018 %>%  
  union_all(igr_2019)
```

**17.**

**JOIN**

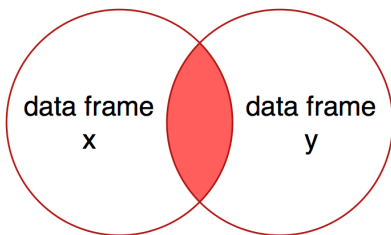
# INNER JOIN ... inner\_join

## SQL

```
SELECT * FROM igr_2018  
INNER JOIN igr_2019 ON  
igr_2018.registro_ans =  
igr_2019.registro_ans
```

## dplyr

```
igr_2018 %>% inner_join(igr_2019,  
by= "registro_ans")
```





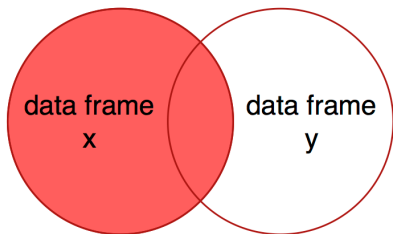
# LEFT JOIN ... left\_join

## SQL

```
SELECT * FROM igr_2018  
LEFT JOIN igr_2019 ON  
igr_2018.registro_ans =  
igr_2019.registro_ans
```

## dplyr

```
igr_2018 %>% left_join(igr_2019,  
by= "registro_ans")
```



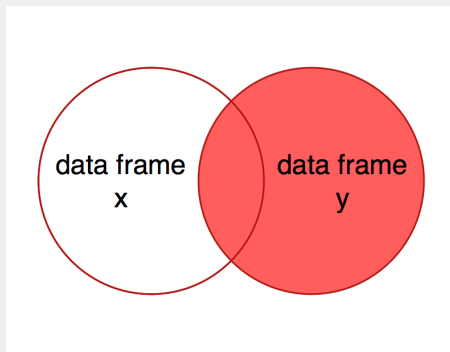
# RIGHT JOIN ... right\_join

## SQL

```
SELECT * FROM igr_2018  
RIGHT JOIN igr_2019 ON  
igr_2018.registro_ans =  
igr_2019.registro_ans
```

## dplyr

```
igr_2018 %>% right_join(igr_2019,  
by= "registro_ans")
```



# Referências

<http://dados.gov.br/dataset/indice-de-reclamacoes>

<http://dados.gov.br/dataset/indice-de-reclamacoes/resource/c621fc2d-a4e6-43b3-afb4-9143c36f4d6c>

[https://github.com/MaryMS/presentations/blob/master/2018\\_11\\_10\\_Oficina\\_R/oficina\\_R\\_respostas.pdf](https://github.com/MaryMS/presentations/blob/master/2018_11_10_Oficina_R/oficina_R_respostas.pdf)

<https://www.linkedin.com/pulse/desapegando-do-sql-com-r-pedro-toledo>

<https://beanumber.github.io/mysql-r-webinar/dplyr.html>

<https://beatrizmilz.github.io/2019-02-R-Interm-R-LadiesSP/#32>

<https://rstudio.com/resources/cheatsheets/>

<https://datacarpentry.org/R-ecology-lesson/03-dplyr.html#mutate>

<http://material.curso-r.com/manip/>

<https://medium.com/@fernando.gama/parte-1-7-passos-para-manipula%C3%A7%C3%A3o-de-dados-com-dplyr-6a1fd092eaff>

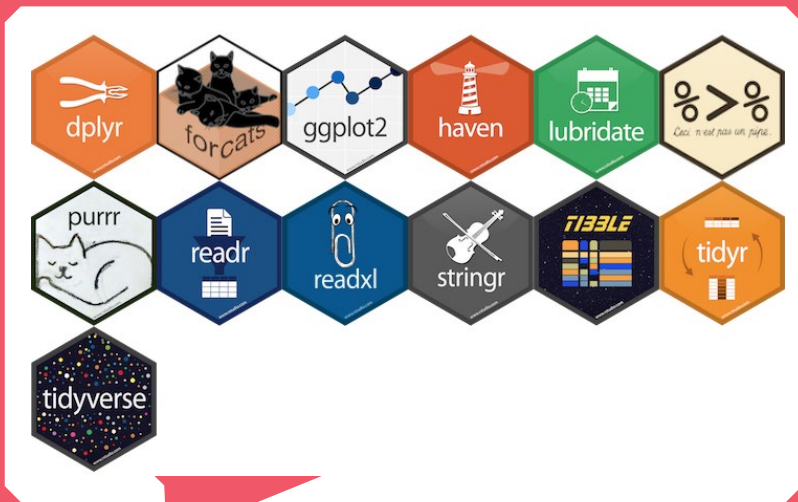
<https://blog.exploratory.io/merging-two-data-frames-with-union-or-bind-rows-a55e79766d0>

[http://www.datasciencemadesimple.com/union-union\\_all-function-r-using-dplyr-union-data-frames/](http://www.datasciencemadesimple.com/union-union_all-function-r-using-dplyr-union-data-frames/)

[http://lindsaydbrin.github.io/CREATE\\_R\\_Workshop/Lesson\\_-\\_dplyr\\_join.html](http://lindsaydbrin.github.io/CREATE_R_Workshop/Lesson_-_dplyr_join.html)

<https://www.w3schools.com/sql/>

A apresentação foi inspirada no modelo Benedick do site: [www.slidescarnival.com/pt-br](http://www.slidescarnival.com/pt-br)



# Linguagem R

Você pode aprender hoje!



**NerdZão**



# Muito obrigada!

## Perguntas?



[mariamarinhos@gmail.com](mailto:mariamarinhos@gmail.com)



[@mariaeme](https://t.me/mariaeme)



[maryms](https://github.com/maryms)