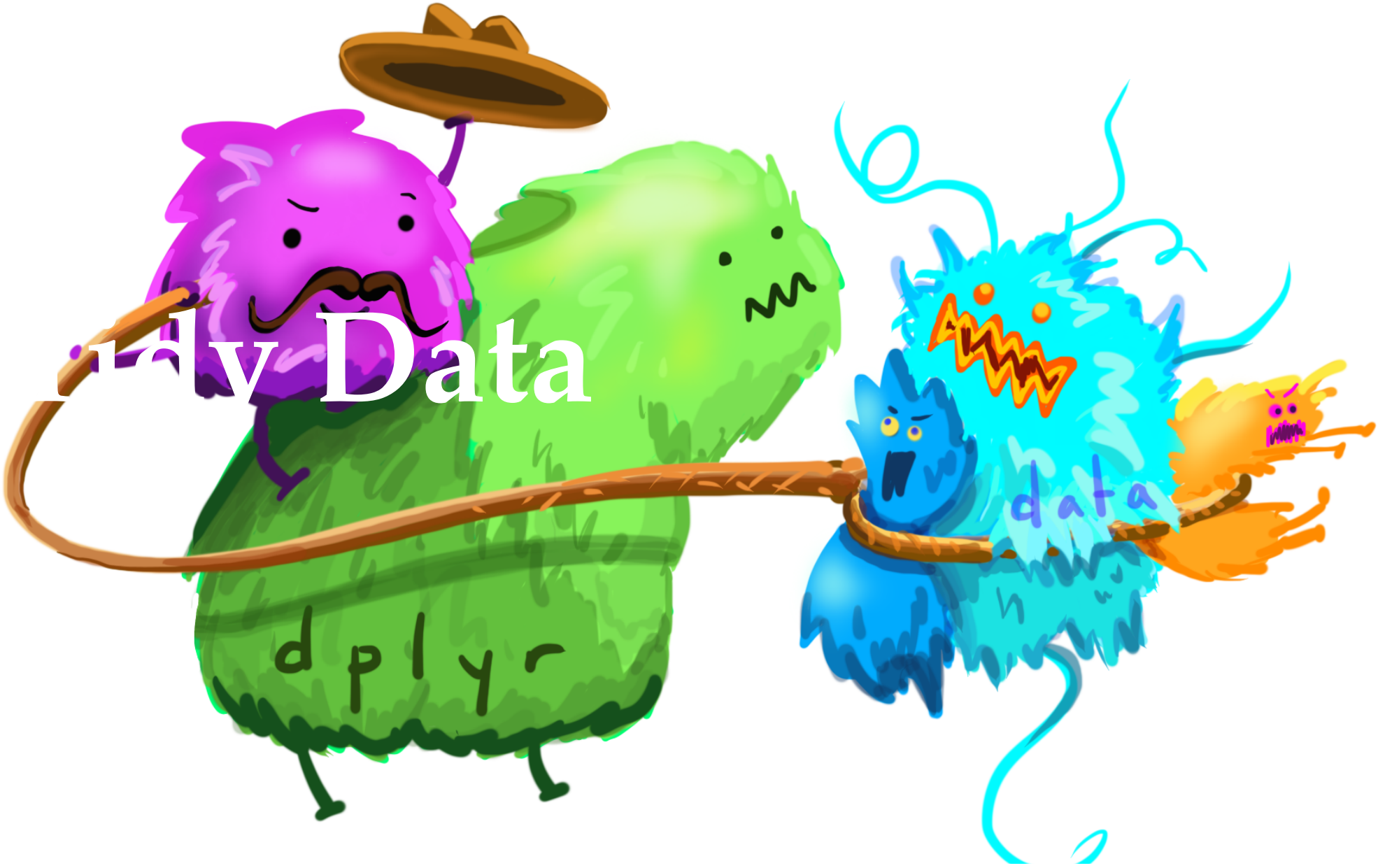
Análise dos Dados com R

Data Munging e o Tidyverse

James R. Hunter, PhD
Retrovirologia, EPM, UNIFESP

2023-10-03

dplyr : go wrangling



Resumo de um *Data Frame/Tibble*

- Estrutura geral dos dados
  - Quantas variáveis
  - Quais tipos
- Utilize ou `str()` ou `glimpse()`
  - `str()` - Base R
  - `glimpse()` - `tibble`

`soro` como Exemplo

```
spc_tbl_ [99 × 10] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ pacid     : chr [1:99] "b6d668e4f818f7b3643ed593b8fb902bf9d2501e" "a090625661c06e9c25ab67b16576ce23b1b0526f"
"0a67cd063da4bfade9be8e0e4fa0144ffc4f2d0b" "b4d0a3ec53a085589a222d3c2f6b6ee02c7f7333" ...
 $ dt_collect: chr [1:99] "28/05/2020" "11/05/2020" "16/06/2020" "10/06/2020" ...
 $ analysis  : chr [1:99] "IgM, COVID19" "IgG, COVID19" "IgG, COVID19" "COVID IgM Interp" ...
 $ result    : chr [1:99] "0.74" "0.03" "0.02" NA ...
 $ unit      : chr [1:99] "AU/ml" "AU/ml" "AU/ml" NA ...
 $ reference : chr [1:99] "<=0.90" "<=0.90" "<=0.90" "" ...
 $ sex       : Factor w/ 2 levels "male","female": 1 2 2 1 2 2 2 1 2 1 ...
 $ birth_yr  : num [1:99] 1989 1975 1997 2006 1983 ...
 $ uf        : chr [1:99] "SP" "GO" "SP" "SP" ...
 $ city      : chr [1:99] "SAO PAULO" NA "SAO PAULO" "SAO PAULO" ...
 - attr(*, "spec")=
  .. cols(
  ..   pacid = col_character(),
  ..   dt_collect = col_character(),
  ..   analysis = col_character(),
  ..   result = col_character(),
  ..   unit = col_character(),
  ..   reference = col_character(),
  ..   sex = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
  ..   birth_yr = col_number(),
  ..   uf = col_character()
```

`glimpse()` Alternativo a `str()`

```
1 tibble::glimpse(soro)
```

```
Rows: 99
Columns: 10
$ pacid      <chr> "b6d668e4f818f7b3643ed593b8fb902bf9d2501e", "a090625661c06e…
$ dt_collect <chr> "28/05/2020", "11/05/2020", "16/06/2020", "10/06/2020", "30…
$ analysis   <chr> "IgM, COVID19", "IgG, COVID19", "IgG, COVID19", "COVID IgM …
$ result     <chr> "0.74", "0.03", "0.02", NA, "0.47", "0.9", NA, "30.77", "0.…
$ unit       <chr> "AU/ml", "AU/ml", "AU/ml", NA, "AU/ml", "AU/ml", NA, "AU/ml…
$ reference  <chr> "<=0.90", "<=0.90", "<=0.90", "", "<=0.90", "<=0.90", "", "…
$ sex        <fct> male, female, female, male, female, female, female, male, f…
$ birth_yr   <dbl> 1989, 1975, 1997, 2006, 1983, 1963, 1988, 1971, 1968, 1976,…
$ uf         <chr> "SP", "GO", "SP", "SP", "SP", "SP", "SP", "SP", "SP", "SP",…
$ city       <chr> "SAO PAULO", NA, "SAO PAULO", "SAO PAULO", "SAO PAULO", "SA…
```

Ver em Mais Detalhe

- `summarytools::dfSummary()`
  - Resumo curto de cada variável no conjunto
  - Presentação baseado no tipo da variável
  - Muitas opções
  - Eu deixo fora a coluna "graph"
    - `graph.col = FALSE` para omitir

```
1 library(summarytools)
2 dfSummary(soro, graph.col = FALSE)
```

Variável *pacid*

```
Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 2

-----------------------------------------------------------------------------------------
No   Variable      Stats / Values                  Freqs (% of Valid)   Valid      Missing
---- ------------ ------------------------------- -------------------- ---------- ---------
1    pacid        1. 373a2ae841153ee5f4d86c245     2 ( 2.0%)           99         0
     [character]  2. 95ecc1410a0f8abfde332e73d     2 ( 2.0%)           (100.0%)   (0.0%)
                  3. 0a67cd063da4bfade9be8e0e4     1 ( 1.0%)
                  4. 0d4c2100337f05f197256160d     1 ( 1.0%)
                  5. 0f68cd676ae5b106902b8a4b9     1 ( 1.0%)
                  6. 1226637f4eb61db0f06c6fac2     1 ( 1.0%)
                  7. 161f5d5b585c29048f523bd61     1 ( 1.0%)
                  8. 19210bb9bc58276bc7daf9fb9     1 ( 1.0%)
                  9. 1cbf74c501b4c182e8a0475fc     1 ( 1.0%)
                  10. 200dbeeda1df13240200a8996    1 ( 1.0%)
                  [ 87 others ]                   87 (87.9%)
-----------------------------------------------------------------------------------------
```

Variável *dt_collect*

Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 53

No	Variable	Stats / Values	Freqs (% of Valid)	Valid	Missing
1	dt_collect [character]	1. 05/06/2020	6 (6.1%)	99 (100.0%)	0 (0.0%)
		2. 04/06/2020	5 (5.1%)		
		3. 08/06/2020	4 (4.0%)		
		4. 10/06/2020	4 (4.0%)		
		5. 12/05/2020	4 (4.0%)		
		6. 14/05/2020	4 (4.0%)		
		7. 19/05/2020	4 (4.0%)		
		8. 21/05/2020	4 (4.0%)		
		9. 30/04/2020	4 (4.0%)		
		10. 02/06/2020	3 (3.0%)		
		[36 others]	57 (57.6%)		

Variável *analysis*

Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 95

No	Variable	Stats / Values	Freqs (% of Valid)	Valid	Missing
1	analysis [character]	1. COVID IgG Interp 2. COVID IgM Interp 3. IgG, COVID19 4. IgM, COVID19	22 (22.2%) 22 (22.2%) 29 (29.3%) 26 (26.3%)	99 (100.0%)	0 (0.0%)

Variável *result*

Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 59

No	Variable	Stats / Values	Freqs (% of Valid)	Valid	Missing
1	result [character]	1. 0.02	7 (11.7%)	60 (60.6%)	39 (39.4%)
		2. 0.54	4 (6.7%)		
		3. 0.04	3 (5.0%)		
		4. 0.06	3 (5.0%)		
		5. Reagente	3 (5.0%)		
		6. (Empty string)	2 (3.3%)		
		7. 0.03	2 (3.3%)		
		8. 0.33	2 (3.3%)		
		9. 0.36	2 (3.3%)		
		10. 0.5	2 (3.3%)		
		[29 others]	30 (50.0%)		

Variáveis *unit reference sex*

```
Data Frame Summary
soro
Dimensions: 99 x 3
Duplicates: 93

-------------------------------------------------------------------------------
No   Variable      Stats / Values      Freqs (% of Valid)   Valid      Missing
---- ------------ ------------------- -------------------- ---------- ---------
1    unit         1. AU/ml            52 (100.0%)          52         47
     [character]                                           (52.5%)    (47.5%)

2    reference    1. (Empty string)   44 (44.4%)           99         0
     [character]  2. <=0.90           52 (52.5%)           (100.0%)   (0.0%)
                  3. Não Reagente      3 ( 3.0%)

3    sex          1. male             48 (48.5%)           99         0
     [factor]     2. female           51 (51.5%)           (100.0%)   (0.0%)
-------------------------------------------------------------------------------
```

Variável *birth_yr*

```
Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 51

----------------------------------------------------------------------------------
No   Variable    Stats / Values              Freqs (% of Valid)   Valid     Missing
---- ----------- --------------------------- -------------------- --------- ---------
1    birth_yr    Mean (sd) : 1978.8 (16.3)   47 distinct values   96        3
     [numeric]   min < med < max:                                 (97.0%)   (3.0%)
                 1933 < 1979.5 < 2020
                 IQR (CV) : 21.2 (0)
----------------------------------------------------------------------------------
```

Variável *uf*

```
Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 96

----------------------------------------------------------------------------
No   Variable      Stats / Values   Freqs (% of Valid)   Valid      Missing
---- ------------ ---------------- -------------------- ---------- ---------
1    uf            1. GO             1 ( 1.0%)          99         0
     [character]   2. MS             1 ( 1.0%)          (100.0%)   (0.0%)
                   3. SP            97 (98.0%)
----------------------------------------------------------------------------
```

Variável *city*

```
Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 93

-------------------------------------------------------------------------
------
No   Variable      Stats / Values           Freqs (% of Valid)   Valid
Missing
---- ------------ ------------------------ -------------------- --------- ---
------
1    city         1. BARUERI                2 ( 2.3%)           88        11
     [character]  2. JUNDIAI                6 ( 6.8%)           (88.9%)
(11.1%)
                  3. SANTANA DE PARNAIBA    2 ( 2.3%)
                  4. SAO PAULO             77 (87.5%)
```

Munging Este Conjunto dos Dados

Tasks

- `dt_collect`: formato não padronizado, caractere
  - Transformar para Date com funções do pacote `lubridate`
- `analysis`: tem maneiras diferentes para designar o mesmo teste
  - Pode isolar o nome de anticorpo com as funções de `stringr`
- `result`: problema de `Não reagente` como 0
  - Outros valores string
  - Resolver os valores string e transformar a `numeric`
- `unit`: somente um valor
  - Retirar do conjunto: não útil para a análise
  - Utilize `janitor::remove_constant()`
- `reference`: 3 valores; qual é a utilidade da variável?
  - Pode atribuir valores úteis para os 3 valores ou retirar

Limpar os Nomes das Variáveis

Limpeza dos Nomes

- Primeiro passo de *munging* universal
- Nossos nomes já são limpos
- `janitor::clean_names()`

Exemplo da Limpeza dos Nomes

```
test_df <- as.data.frame(matrix(ncol = 6))
names(test_df) <- c("firstName", "$abc@!*", "% successful (2009)",
                    "REPEAT VALUE", "REPEAT VALUE", "")
test_df
```

```
  firstName $abc@!* % successful (2009) REPEAT VALUE REPEAT VALUE
1        NA      NA                   NA            NA           NA NA
```

```
# apply clean_names()

test_df <- janitor::clean_names(test_df)

test_df
```

```
  first_name abc percent_successful_2009 repeat_value repeat_value_2  x
1         NA  NA                      NA            NA             NA NA
```

Atribuir Nomes às Variáveis

- Pode usar `names()` para criar nomes para suas variáveis
- Nomes precisam ser num vetor com o mesmo número de itens que o número das colunas
- `names(test_df) <-` para receber o vetor

```
names(test_df) <- c("first_name", "last_name",
                    "percent_successful_2009",
                    "value_1", "value_2", "standard")
```

Munging as Variáveis

Traduzir Datas do Formato Texto a *Date*

- Formato atual de `dt_collect`
  - *String* em "dd/mm/yyyy" (08/06/2020)
  - Formato padrão brasileiro

Analisando o Formato

- Pacote `lubridate`
- Nomes das funções combinações das 1ªs letras de *day*, *month*, *year*
  - Na ordem que aparece na data
  - Em nosso caso, usaríamos `dmy()`
- Se fosse um data padrão americano (“mm/dd/yyyy”)
  - Função seria `mdy()`
- `lubridate` tem todas as possibilidades
- Todos os formatos funcionam com qualquer separador
  - Ignora os

Conversão das Datas com *lubridate*

```
1 br_text <- "28/05/2020"
2 (br_date <- dmy(br_text))
```

```
[1] "2020-05-28"
```

```
1 us_text <- "05-28-2020"
2 (us_date <- mdy(us_text))
```

```
[1] "2020-05-28"
```

Sequências das Funções – *Pipe*

Necessidade de Ligar Funções

- De forma que podemos mais tarde entender e lembrar
- Exemplo teórico (de Ismay e Kim, **ModernDive**)
  - *Data frame* `x`
  - Funções `f()`, `g()`, e `h()`
- Sequência das ações:
  - Começa com `x` então
  - Use `x` como uma entrada para a função `f()` então
  - Use o resultado da `f(x)` como uma entrada para a função `g()` então
  - Use o resultado da `g(f(x))` como uma entrada para a função `h()`
- Solução de parênteses aninhados
  - `h(g(f(x)))`
  - Fácil a entender – **NÃO**

Operador *Pipe* (|>)

- Pega o que está no lado esquerdo do operador
- Ele torna primeiro argumento da função no lado direto
- Quer dizer “e então”
- Forma alternativa “%>%” (de Tidyverse)

Exemplo Utilizando o Pipe

```
x |>
  f() |>
  g() |>
  h()
```

1. Pega `x` e então
2. Use este resultado como a entrada para próxima função `f()` e então
3. Use este resultado como a entrada para próxima função `g()` e então
4. Use este resultado como a entrada para próxima função `h()`

mutate() Function - Modificar (e Adicionar) Variáveis

Fundamentos de *mutate()*

- `dplyr::mutate()`
  - 1º argumento: *data frame* ou *tibble* a ser modificado
  - 2º argumento: modificação na forma de atribuição
    - **Aqui** atribuição usa "=" não "<-"

Atribuição em `mutate()`

- Nome de variável no lado esquerdo
- Se este nome de variável não existe no *tibble*, If variable name does not exist in tibble, ele será adicionado
- Se variável existente, substuir o valor atual
  - **VSS**: Copiar seu *tibble* primeiro para um novo objeto

Quais Funções Pode Usar no `mutate()`

Vectorized Functions

TO USE WITH MUTATE ()

mutate() and **transmute()** apply vectorized functions to columns to create new columns. Vectorized functions take vectors as input and return vectors of the same length as output.

vectorized function

OFFSETS

dplyr::**lag()** - Offset elements by 1
dplyr::**lead()** - Offset elements by -1

CUMULATIVE AGGREGATES

dplyr::**cumall()** - Cumulative all()
dplyr::**cumany()** - Cumulative any()
cummax() - Cumulative max()
dplyr::**cummean()** - Cumulative mean()
cummin() - Cumulative min()
cumprod() - Cumulative prod()
cumsum() - Cumulative sum()

RANKINGS

dplyr::**cume_dist()** - Proportion of all values <=
dplyr::**dense_rank()** - rank with ties = min, no gaps
dplyr::**min_rank()** - rank with ties = min
dplyr::**ntile()** - bins into n bins
dplyr::**percent_rank()** - min_rank scaled to [0,1]
dplyr::**row_number()** - rank with ties = "first"

MATH

+, - , *, /, ^, %/%, %% - arithmetic ops
log(), log2(), log10() - logs
<, <=, >, >=, !=, == - logical comparisons

MISC

dplyr::**between()** - x >= left & x <= right
dplyr::**case_when()** - multi-case if_else()
dplyr::**coalesce()** - first non-NA values by element across a set of vectors
dplyr::**if_else()** - element-wise if() + else()
dplyr::**na_if()** - replace specific values with NA
pmax() - element-wise max()
pmin() - element-wise min()
dplyr::**recode()** - Vectorized switch()
dplyr::**recode_factor()** - Vectorized switch() for factors

Modificar *dt_collect* com `mutate()`

1. Criar o nome da nova versão do *tibble*
2. Atribuir para ele os dados da versão antiga
3. Transformar a data para o classe `Date`

Código Que Faz Isso

```
1 soro_b <- soro |> # steps 1 and 2; note use of Pipe
2   dplyr::mutate(dt_collect = dmy(dt_collect)) # step 3
3
4 glimpse(soro_b$dt_collect)
```

```
 Date[1:99], format: "2020-05-28" "2020-05-11" "2020-06-16" "2020-06-10"
"2020-04-30" ...
```

Limpar as Categorias de `analysis`

Lembrete

- `analysis` teve 2 maneiras para referir a cada um dos 2 anticorpos
- Queremos reduzir variável para os valores “IgG” e “IgM” só

```
1 table(soro$analysis)
```

```
COVID IgG Interp COVID IgM Interp     IgG, COVID19     IgM, COVID19 
              22                22               29               26 
```

mutate() com *ifelse()*

- Todos os valores incluem o nome do anticorpo
  - "IgG" or "IgM"
- Podemos procurar dentro de *string* para *sub-string* "IgG"
  - Se caso o tem, pode atribuir esse valor para a `analysis`
    - Senão, atribuir o outro valor ("IgM")
- Use `ifelse()` para tomar esta decisão
- Porque tem um número pequeno de valores (2),
  - Transformar `analysis` em `factor`
- Fazer a pesquisa com `stringr::str_detect(var, pattern)`
  - `var`: variável a ser pesquisadoable to be searched
  - `pattern`: padrão que quer procurar
  - `str_detect(analysis, "IgG")`

Código para `mutate()`

```
1  soro_b <- soro %>%
2    mutate(analysis = ifelse(str_detect(analysis, "IgG"), "IgG", "IgM")) %>%
3    mutate(analysis = factor(analysis))
4
5  glimpse(soro_b$analysis)
```

```
Factor w/ 2 levels "IgG","IgM": 2 1 1 2 2 2 2 1 2 2 ...
```

Outra Maneira para Modificar `analysis` com `forcats`

- Use funções de `forcats` para manipular `analysis`
- `forcats`: funções que manipulam *factors*
- Começar por transformar `analysis` para o tipo de dado `factor`
- Chamar `factor()`

```
1 x <- c("a", "b", "c")
2 glimpse(x)
```

```
chr [1:3] "a" "b" "c"
```

```
1 fct_x <- factor(x)
2 glimpse(fct_x)
```

```
Factor w/ 3 levels "a","b","c": 1 2 3
```

- Valores agora: 1, 2, 3
- *Levels*: a, b, c

Aplicar a `analysis`

- Vamos manipular os níveis de `analysis`

```
soro_b <- soro |>
  mutate(analysis_f = factor(analysis))
glimpse(soro_b$analysis_f)
```

```
 Factor w/ 4 levels "COVID IgG Interp",..: 4 3 3 2 4 4 2 3 4 4 ...
```

```
levels(soro_b$analysis_f)
```

```
[1] "COVID IgG Interp" "COVID IgM Interp" "IgG, COVID19"     "IgM, COVID19"
```

```
table(soro_b$analysis_f)
```

```
COVID IgG Interp COVID IgM Interp     IgG, COVID19     IgM, COVID19
              22               22               29               26
```

fct_collapse() Aplicado a `analysis`

- `forcats::fct_collapse()`: reduzir o número de níveis baseado em valores da variável
- **Não esqueça o Cheat Sheet: "Factors with forcats"**
- Porque teremos 2 níveis finais ("IgG" ou "IgM")
  - Precisa definir cada um separadamenteNeed to define each separately

Code for This

```
soro_b <- soro |>
  mutate(analysis_f = factor(analysis)) |>
  mutate(analysis_f = fct_collapse(analysis_f,
                                   IgG = c("COVID IgG Interp", "IgG, COVID1
                                   IgM = c("COVID IgM Interp", "IgM, COVID1
glimpse(soro_b$analysis_f)
```

```
 Factor w/ 2 levels "IgG","IgM": 2 1 1 2 2 2 2 1 2 2 ...
```

```
fct_count(soro_b$analysis_f)
```

```
# A tibble: 2 × 2
  f         n
  <fct> <int>
1 IgG      51
2 IgM      48
```

Forma Mais Compacta para Obter o Mesmo Resultado

```
soro_b <- soro |>
mutate(analysis_f = fct_collapse(factor(analysis),
                                 IgG = c("COVID IgG Interp", "IgG, COVID19"
                                 IgM = c("COVID IgM Interp", "IgM, COVID19"
```

Valores Não-Númericos em `result`

Problema - Valores *String* `result`

- “Não reagente” e “Reagente”

```
Data Frame Summary
soro
Dimensions: 99 x 1
Duplicates: 59

----------------------------------------------------------------------------
No   Variable      Stats / Values      Freqs (% of Valid)   Valid     Missing
---- ------------ ------------------- -------------------- --------- ---------
1    result       1. 0.02              7 (11.7%)           60        39
     [character]  2. 0.54              4 ( 6.7%)           (60.6%)   (39.4%)
                  3. 0.04              3 ( 5.0%)
                  4. 0.06              3 ( 5.0%)
                  5. Reagente          3 ( 5.0%)
                  6. (Empty string)    2 ( 3.3%)
                  7. 0.03              2 ( 3.3%)
                  8. 0.33              2 ( 3.3%)
                  9. 0.36              2 ( 3.3%)
                  10. 0.5              2 ( 3.3%)
                  [ 29 others ]       30 (50.0%)
----------------------------------------------------------------------------
```

Base R - Estratégia

- Tratar “Não reagente” como 0
- Tratar “Reagente” e *strings* em branco como NA
- Use `for` *loop* para testar todos os casos
- Use `if...then...else` para testar os valores e fazer as trocas

```
soro_b <- soro
for(i in 1:nrow(soro_b)){
  if(soro_b$result[i] == "Nao reagente") {
    soro_b$result[i] <- 0
    } else {
      if(soro_b$result[i] %in% c("Reagente", "")){
        soro_b$result[i] <- NA
      } # end second if
    }  # end else
  }  # end of if
}  # end of loop
soro_b$result <- as.numeric(soro_b$result)
# above line is what made else test optional
```

Tidyverse: *mutate()* & *ifelse()*

- Mesma Lógica

```r
soro_b <- soro %>%
  mutate(result = as.numeric(ifelse(result == "Não reagente", 0, result)))
summarytools::dfSummary(soro_b$result, graph.col = FALSE)
```

```
Data Frame Summary
soro_b
Dimensions: 99 x 1
Duplicates: 61

----------------------------------------------------------------------
--
No   Variable    Stats / Values          Freqs (% of Valid)   Valid     Missing
---- ----------- ---------------------- -------------------- --------- -------
--
1    result      Mean (sd) : 2.8 (7)     37 distinct values   55        44
     [numeric]   min < med < max:                             (55.6%)   (44.4%)
                 0 < 0.5 < 30.8
                 IQR (CV) : 0.7 (2.5)
----------------------------------------------------------------------
--
```

New Problem with *result()*

- What is that 30.8 Value?
- Mean = 1.7
- Value is 5.29 standard deviations outside mean
- Reference value from `reference` is “<=0.90”
  - This value 30 times higher than reference
- **Outlier**
- Important Issue in statistics
- Lesson: Take careful note of range of numerical values
  - Problem to be solved during analysis phase

Retirar Variáveis Desnecessárias

Retirar `unit`

- Use `janitor::remove_constant()`
- `unit` só tem 1 valor: "AU/ml"
- Não existe variância para medir
- `remove_constant()`: retira colunas que tem só 1 valor (mais NA)

```
1 table(soro$unit, useNA = "ifany")
```

```
AU/ml  <NA>
   52    47
```

Retirar `unit` - 2

```
soro_b <-  soro %>%
  janitor::remove_constant(na.rm = TRUE)
glimpse(soro_b)
```

```
Rows: 99
Columns: 9
$ pacid      <chr> "b6d668e4f818f7b3643ed593b8fb902bf9d2501e",
"a090625661c06e…
$ dt_collect <chr> "28/05/2020", "11/05/2020", "16/06/2020", "10/06/2020",
"30…
$ analysis   <chr> "IgM, COVID19", "IgG, COVID19", "IgG, COVID19", "COVID IgM
…
$ result     <chr> "0.74", "0.03", "0.02", NA, "0.47", "0.9", NA, "30.77",
"0.…
$ reference  <chr> "<=0.90", "<=0.90", "<=0.90", "", "<=0.90", "<=0.90", "",
"…
$ sex        <fct> male, female, female, male, female, female, female, male,
f…
$ birth_yr   <dbl> 1989, 1975, 1997, 2006, 1983, 1963, 1988, 1971, 1968,
1976
```

Retirar `reference` com *dplyr::select()*

- `reference` só tem um valor útil: "<=0.90"

```
1 table(soro$reference, useNA = "ifany")
```

```
          <=0.90 Não Reagente
      44      52             3
```

`select()`: 2º Verbo Importante de `dplyr`

- Funciona com colunas (variáveis)
- Se queremos incluir colunas em uma operção
  - `select()` elas positivamente em argumentos
- Se queremos excluir colunas em uma operação
  - `select()` elas negativamente em argumentos

Exemplo Simples de `select()`

```
1 a <- tibble(x = c("a", "b", "c"),
2              y = 1:3,
3              z = c("d", "e", "f"))
4 a #show the tibble on the screen
```

```
# A tibble: 3 × 3
  x         y z
  <chr> <int> <chr>
1 a         1 d
2 b         2 e
3 c         3 f
```

```
1 a |> select(y) #just show the selected variable
```

```
# A tibble: 3 × 1
      y
  <int>
1     1
2     2
3     3
```

Retirar Variáveis com `select(-var)`

```
1 a
```

```
# A tibble: 3 × 3
  x         y z
  <chr> <int> <chr>
1 a         1 d
2 b         2 e
3 c         3 f
```

```
1 a |> select(-x)
```

```
# A tibble: 3 × 2
      y z
  <int> <chr>
1     1 d
2     2 e
3     3 f
```

Retirar `reference` com `dplyr::select()`

```
1 soro_b <- soro %>%
2   select(-reference)
3 glimpse(soro_b)
```

```
Rows: 99
Columns: 9
$ pacid      <chr> "b6d668e4f818f7b3643ed593b8fb902bf9d2501e",
"a090625661c06e…
$ dt_collect <chr> "28/05/2020", "11/05/2020", "16/06/2020", "10/06/2020",
"30…
$ analysis   <chr> "IgM, COVID19", "IgG, COVID19", "IgG, COVID19", "COVID IgM
…
$ result     <chr> "0.74", "0.03", "0.02", NA, "0.47", "0.9", NA, "30.77",
"0.…
$ unit       <chr> "AU/ml", "AU/ml", "AU/ml", NA, "AU/ml", "AU/ml", NA,
"AU/ml…
$ sex        <fct> male, female, female, male, female, female, female, male,
f…
$ birth_yr   <dbl> 1989, 1975, 1997, 2006, 1983, 1963, 1988, 1971, 1968,
1976
```

Transformar `uf` para um `factor`

```
1 soro_b <- soro %>%
2   mutate(uf = factor(uf))
3 glimpse(soro_b$uf)
```

```
Factor w/ 3 levels "GO","MS","SP": 3 1 3 3 3 3 3 3 3 3 ...
```

Combinar Todas as Ops com o *Pipe*

- Usar o *pipe*, podemos combinar todas essas operações em um comando grande

```
soro_b <- soro %>%
  mutate(dt_collect = dmy(dt_collect)) %>%
  mutate(analysis = factor(analysis)) %>%
  mutate(analysis = fct_collapse(analysis,
                                 IgG = c("COVID IgG Interp", "IgG, COVID19"
                                 IgM = c("COVID IgM Interp", "IgM, COVID19"
  mutate(result = as.numeric(ifelse(result == "Não reagente", 0, result)))
  janitor::remove_constant(na.rm = TRUE) %>%  # unit variable
  select(-reference) %>%
  mutate(uf = factor(uf))
```

Resultado Final de *Munging*

```
Rows: 99
Columns: 8
$ pacid      <chr> "b6d668e4f818f7b3643ed593b8fb902bf9d2501e",
"a090625661c06e…
$ dt_collect <date> 2020-05-28, 2020-05-11, 2020-06-16, 2020-06-10, 2020-04-
30…
$ analysis   <fct> IgM, IgG, IgG, IgM, IgM, IgM, IgM, IgG, IgM, IgM, IgM,
IgM,…
$ result     <dbl> 0.74, 0.03, 0.02, NA, 0.47, 0.90, NA, 30.77, 0.41, 0.54,
0.…
$ sex        <fct> male, female, female, male, female, female, female, male,
f…
$ birth_yr   <dbl> 1989, 1975, 1997, 2006, 1983, 1963, 1988, 1971, 1968,
1976,…
$ uf         <fct> SP, GO, SP, SP, SP, SP, SP, SP, SP, SP, SP, SP, SP, SP,
SP
```

`soro_b` Segue a Definição de *"Tidy"*?