MAD – Data Analysis & Biostatistics in R

Review of `dplyr` Verbs

James R. Hunter, Ph.D.

DIPA, EPM, UNIFESP

18 September 2020

Section 1

Mutate

Section 2

`mutate()` Function - How We Modify (and Add) Variables

Basics of `mutate()`

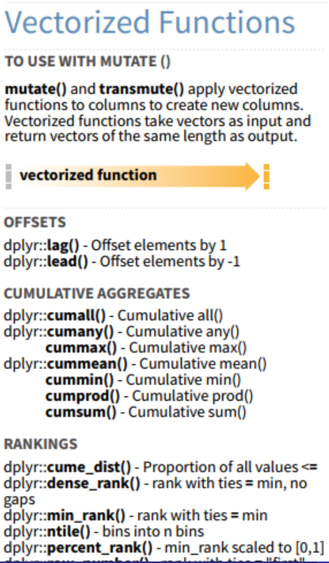
- `dplyr::mutate()`
  - 1st argument: data frame or tibble to be modified
  - 2nd argument: modification in form of assignment
    - **Here** assignment uses "=" not "<-"

`mutate()` Assignment

- Variable name on left side
- If variable name does not exist in tibble, it will be added
- If existing variable, overwrite current value
  - Do this in a new tibble

Functions for Assignment

- Wide variety



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]

Steps to Accomplish Mutation (`dt_collect`)

1. Establish the name of the revised data frame then
2. Assign to it the data from the old version then
3. Transform the date to a Date class

Code to Accomplish This

Load soro as Example

```r
soro <- readRDS("C:/Users/james/OneDrive/Documents/MAD/MAD-Infecto-2020/einstein_so
str(soro)
```

```
## 'data.frame':    200 obs. of  10 variables:
##  $ pacid     : chr  "b6d668e4f818f7b3643ed593b8fb902bf9d2501e" "a090625661c06e9c
##  $ dt_collect: chr  "28/05/2020" "11/05/2020" "16/06/2020" "10/06/2020" ...
##  $ analysis  : chr  "IgM, COVID19" "IgG, COVID19" "IgG, COVID19" "COVID IgM Inte
##  $ result    : chr  "0.74" "0.03" "0.02" "Não reagente" ...
##  $ unit      : chr  "AU/ml" "AU/ml" "AU/ml" "NULL" ...
##  $ reference : chr  "<=0.90" "<=0.90" "<=0.90" "" ...
##  $ sex       : Factor w/ 2 levels "female","male": 2 1 1 2 1 1 1 2 1 2 ...
##  $ birth_yr  : num  1989 1975 1997 2006 1983 ...
##  $ uf        : Factor w/ 25 levels "AC","AL","AM",..: 24 9 24 24 24 24 24 24 24
##  $ city      : Factor w/ 21 levels "BARUERI","CAMPINAS",..: 19 NA 19 19 19 19 19
```

```r
library(tidyverse)
library(lubridate)
soro_b <- soro %>%  # steps 1 and 2; note use of Pipe
  mutate(dt_collect = dmy(dt_collect)) # step 3

glimpse(soro_b$dt_collect)
```

```
##  Date[1:200], format: "2020-05-28" "2020-05-11" "2020-06-16" "2020-06-10" "2020-
```

Remember

- `analysis` had 2 ways of referring to each of 2 antibodies
- We want to reduce variable to values “IgG” and “IgM” only

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

```
##
## COVID IgG Interp COVID IgM Interp      IgG, COVID19      IgM, COVID19
##               36               45                60                59
```

`mutate()` with `ifelse()`

- All the values include the antibody name we want
  - "IgG" or "IgM"
- We can search for "IgG"
  - If case has it, put that value in `analysis`
    - ⋆ If not, put other
- Use `ifelse()` to make the selection
- Because only two values, transform `analysis` into factor
- Do the search with `stringr::str_detect(var, pattern)`
  - `var`: variable to be searched
  - `pattern`: pattern to detect
  - `str_detect(analysis, "IgG")`

Code for Mutation

```r
soro_b <- soro %>%
  mutate(analysis = ifelse(str_detect(analysis, "IgG"), "IgG", "IgM")) %>%
  mutate(analysis = factor(analysis))

glimpse(soro_b$analysis)
```

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

2nd Approach for analysis with forcats

- Use functions from forcats to manipulate analysis
- forcats: functions to manipulate factors
- Start by transforming analysis to a factor data type
- Call factor()

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

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

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

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

- Values now: 1, 2, 3
- Levels: a, b, c

Apply This to analysis

- What we will do with analysis is manipulate levels

```
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
##               36               45               60               59
```

`mutate()` Applied to `fct_collapse()`

- `forcats::fct_collapse()`: reduce number of levels based on existing values
- **Don't forget the Cheat Sheet: "Factors with forcats::"**
- Since we will have 2 final levels ("IgG" or "IgM")
  - Need to define each separately

Code for This

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

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

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

```
## [1] "IgG" "IgM"
```

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

```
## # A tibble: 2 x 2
##   f         n
##   <fct> <int>
## 1 IgG      96
## 2 IgM     104
```

Even More Compact Form to Get Same Result

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

Section 3

`select()`: 2nd Major `dplyr` Verb

Remove `reference` with `dplyr::select()`

- `reference` has really one value: "<=0.90"

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

```
##
##                      <=0.90 Não Reagente
##            75           114           11
```

`select()` in theory

- Works on columns (variables)
- If we want to include columns in an operation
  - Positively `select()` them in arguments

Simple select() Example

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

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

```r
a %>% select(y) #just show the selected variable
```

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

Remove a Variables with `select(-var)`

```
a
```

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

```
a %>% select(-x)
```

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

Remove `reference` with `dplyr::select()`

```r
soro_b <- soro %>%
  select(-reference)
glimpse(soro_b)
```

```
## Rows: 200
## Columns: 9
## $ pacid      <chr> "b6d668e4f818f7b3643ed593b8fb902bf9d2501e", "a090625661c...
## $ dt_collect <chr> "28/05/2020", "11/05/2020", "16/06/2020", "10/06/2020", ...
## $ analysis   <chr> "IgM, COVID19", "IgG, COVID19", "IgG, COVID19", "COVID I...
## $ result     <chr> "0.74", "0.03", "0.02", "Não reagente", "0.47", "0.90", ...
## $ unit       <chr> "AU/ml", "AU/ml", "AU/ml", "NULL", "AU/ml", "AU/ml", "NU...
## $ sex        <fct> male, female, female, male, female, female, female, male...
## $ birth_yr   <dbl> 1989, 1975, 1997, 2006, 1983, 1963, 1988, 1971, 1968, 19...
## $ uf         <fct> SP, GO, SP, SP, SP, SP, SP, SP, SP, SP, SP, SP, SP, SP, ...
## $ city       <fct> SAO PAULO, NA, SAO PAULO, SAO PAULO, SAO PAULO, SAO PAUL...
```

Section 4

`filter()`: 3rd Major `dplyr` Verb

Load babynames_peq

```r
babynames_peq <- read_csv(here::here("babynames_peq.csv"),
                          col_types = "nccnd")
glimpse(babynames_peq)
```

```
## Rows: 1,182,546
## Columns: 5
## $ year <dbl> 1973, 1973, 1973, 1973, 1973, 1973, 1973, 1973, 1973, 1973, 19...
## $ sex  <chr> "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F", "F...
## $ name <chr> "Jennifer", "Amy", "Michelle", "Kimberly", "Lisa", "Melissa", ...
## $ n    <dbl> 62451, 26964, 26931, 23532, 22669, 22480, 20896, 19350, 17261,...
## $ prop <dbl> 0.04018635, 0.01735096, 0.01732973, 0.01514252, 0.01458719, 0....
```

Garrett 1980

```r
# filter(babynames_peq, name == "Garrett", year == 1980)
babynames_peq %>%
  filter(name == "Garrett", year == 1980)
```

```
## # A tibble: 2 x 5
##    year sex   name          n       prop
##   <dbl> <chr> <chr>     <dbl>      <dbl>
## 1  1980 F     Garrett      10 0.00000562
## 2  1980 M     Garrett    1288 0.000694
```

multiple names

```
babynames_peq %>%
  filter(name %in% c("Acura", "Lexus", "Yugo")) %>%
  tail()
```

```
## # A tibble: 6 x 5
##     year sex   name      n       prop
##    <dbl> <chr> <chr> <dbl>      <dbl>
## 1   2013 M     Lexus     6 0.00000298
## 2   2014 F     Lexus    73 0.0000374
## 3   2014 M     Yugo      5 0.00000245
## 4   2015 F     Lexus    70 0.0000360
## 5   2016 F     Lexus    38 0.0000197
## 6   2017 F     Lexus    35 0.0000187
```

arrange() - 1

```r
arrange(babynames_peq, n, prop) %>%  # utiliza n e prop para determinar
  head()
```

```
## # A tibble: 6 x 5
##     year sex   name            n       prop
##    <dbl> <chr> <chr>       <dbl>      <dbl>
## 1  2007 M     Aaban           5 0.00000226
## 2  2007 M     Aareon          5 0.00000226
## 3  2007 M     Aaris           5 0.00000226
## 4  2007 M     Abd             5 0.00000226
## 5  2007 M     Abdulazeez      5 0.00000226
## 6  2007 M     Abdulhadi       5 0.00000226
```

arrange() - 2 - Descending

```
arrange(babynames_peq, desc(n)) %>%  # utiliza só n
  head()
```

```
## # A tibble: 6 x 5
##    year sex   name         n   prop
##   <dbl> <chr> <chr>    <dbl>  <dbl>
## 1  1981 M     Michael 68765 0.0369
## 2  1980 M     Michael 68693 0.0370
## 3  1975 M     Michael 68454 0.0422
## 4  1982 M     Michael 68228 0.0362
## 5  1983 M     Michael 67995 0.0365
## 6  1973 M     Michael 67846 0.0420
```

summarize()

```r
babynames_peq %>% summarise(total = sum(n), max = max(n))
```

```
## # A tibble: 1 x 2
##       total   max
##       <dbl> <dbl>
## 1 162511845 68765
```

Khaleesi

```
babynames_peq %>%
  filter(name == "Khaleesi") %>%
  summarise(total = sum(n), first = min(year))
```

```
## # A tibble: 1 x 2
##   total first
##   <dbl> <dbl>
## 1  1964  2011
```

n()

```r
babynames_peq %>% summarise(n = n(), nname = n_distinct(name))
```

```
## # A tibble: 1 x 2
##         n nname
##     <int> <int>
## 1 1182546 85727
```