Computations with `dplyr::mutate`

Silvie Cinková

2025-08-07

Table of contents

1 Libraries and data

```
library(gapminder)
library(dplyr)
```

```
Attaching package: 'dplyr'
```

```
The following objects are masked from 'package:stats':

    filter, lag
```

```
The following objects are masked from 'package:base':

    intersect, setdiff, setequal, union
```

```
library(magrittr)
americas_2002 <- gapminder %>% filter(continent == "Americas", year == 2002)
gapminder_2002 <- gapminder %>% filter(year == 2002) %>% select(!c(year))
```

2 `mutate`

- works in two contexts:
  - adds a new column
  - edits an already existing column
- can add/edit several columns at once
- uses its dedicated helper functions
  - `across()`, `c_across()`
- one more grouping function: `rowwise()`

3 Adding a new column with `mutate`

```
americas_2002 %>%
  mutate(GDP_millions = pop * gdpPercap/1000000 ) %>%
  slice(1:10)
```

```
# A tibble: 10 x 7
   country            continent  year lifeExp      pop gdpPercap GDP_millions
   <fct>              <fct>     <int>   <dbl>    <int>     <dbl>        <dbl>
 1 Argentina          Americas   2002    74.3 38331121     8798.      337223.
 2 Bolivia            Americas   2002    63.9  8445134     3413.       28825.
 3 Brazil             Americas   2002    71.0 179914212     8131.     1462921.
 4 Canada             Americas   2002    79.8 31902268    33329.     1063270.
 5 Chile              Americas   2002    77.9 15497046    10779.      167039.
 6 Colombia           Americas   2002    71.7 41008227     5755.      236013.
 7 Costa Rica         Americas   2002    78.1  3834934     7723.       29619.
 8 Cuba               Americas   2002    77.2 11226999     6341.       71186.
 9 Dominican Republic Americas   2002    70.8  8650322     4564.       39478.
10 Ecuador            Americas   2002    74.2 12921234     5773.       74595.
```

4 Adding a new column

```
americas_2002 %>%
  mutate(richest = max(gdpPercap),
         perc_richest = round(gdpPercap/richest, 2))
```

```
# A tibble: 25 x 8
   country          continent  year lifeExp    pop gdpPercap richest perc_richest
   <fct>            <fct>     <int>   <dbl>  <int>     <dbl>   <dbl>        <dbl>
 1 Argentina        Americas   2002    74.3 3.83e7     8798.  39097.         0.23
 2 Bolivia          Americas   2002    63.9 8.45e6     3413.  39097.         0.09
 3 Brazil           Americas   2002    71.0 1.80e8     8131.  39097.         0.21
 4 Canada           Americas   2002    79.8 3.19e7    33329.  39097.         0.85
 5 Chile            Americas   2002    77.9 1.55e7    10779.  39097.         0.28
 6 Colombia         Americas   2002    71.7 4.10e7     5755.  39097.         0.15
 7 Costa Rica       Americas   2002    78.1 3.83e6     7723.  39097.         0.2
 8 Cuba             Americas   2002    77.2 1.12e7     6341.  39097.         0.16
 9 Dominican Repu~  Americas   2002    70.8 8.65e6     4564.  39097.         0.12
10 Ecuador          Americas   2002    74.2 1.29e7     5773.  39097.         0.15
```

```
# i 15 more rows
```

Notice how you can concatenate new columns in one `mutate` action and that the second uses the result of the first within the same command. But I am not sure whether this is always granted.

5 Adding a new column computed on groups

```
gapminder_2002 %>%
  group_by(continent) %>%
  mutate(richest = max(gdpPercap),
         perc_richest = round(gdpPercap/richest, 2)) %>%
  ungroup()
```

```
# A tibble: 142 x 7
   country     continent lifeExp       pop gdpPercap richest perc_richest
   <fct>       <fct>       <dbl>     <int>     <dbl>   <dbl>        <dbl>
 1 Afghanistan Asia         42.1  25268405      727.  36023.         0.02
 2 Albania     Europe       75.7   3508512     4604.  44684.         0.1
 3 Algeria     Africa       71.0  31287142     5288.  12522.         0.42
 4 Angola      Africa       41.0  10866106     2773.  12522.         0.22
 5 Argentina   Americas     74.3  38331121     8798.  39097.         0.23
 6 Australia   Oceania      80.4  19546792    30688.  30688.         1
 7 Austria     Europe       79.0   8148312    32418.  44684.         0.73
 8 Bahrain     Asia         74.8    656397    23404.  36023.         0.65
 9 Bangladesh  Asia         62.0 135656790     1136.  36023.         0.03
10 Belgium     Europe       78.3  10311970    30486.  44684.         0.68
# i 132 more rows
```

Here we do the same operation as with Americas, but this time for each continent. The `richest` column has identical values for countries on one continent.

Note the difference in the behavior of `max()` and division (`/`). The `max` function always looks at the entire group whereas division operates on each row independently.

6 Adding a value on a condition: `if_else`

```
gapminder_2002 %>%
  mutate(is_Asia = if_else(
    condition = continent == "Asia",
    true = "Asian country",
    false = "Not in Asia"))
```

```
# A tibble: 142 x 6
   country     continent lifeExp      pop gdpPercap is_Asia
   <fct>       <fct>       <dbl>    <int>     <dbl> <chr>
 1 Afghanistan Asia         42.1 25268405      727. Asian country
 2 Albania     Europe       75.7  3508512     4604. Not in Asia
 3 Algeria     Africa       71.0 31287142     5288. Not in Asia
 4 Angola      Africa       41.0 10866106     2773. Not in Asia
 5 Argentina   Americas     74.3 38331121     8798. Not in Asia
 6 Australia   Oceania      80.4 19546792    30688. Not in Asia
 7 Austria     Europe       79.0  8148312    32418. Not in Asia
 8 Bahrain     Asia         74.8   656397    23404. Asian country
 9 Bangladesh  Asia         62.0 135656790     1136. Asian country
10 Belgium     Europe       78.3 10311970    30486. Not in Asia
# i 132 more rows
```

7 Adding a value on a condition: `case_when`

```
gapminder %>% filter(year == 2002) %>%
  mutate(is_Asia = case_when(
    continent == "Asia" ~  "Asian country",
    continent == "Africa" ~ "African country",
    continent == "Europe" ~ "European country",
    continent == "Americas" ~ "American country",
    continent == "Oceania" ~ "Oceanian country",
    .default = NA
    ))
```

```
# A tibble: 142 x 7
   country     continent  year lifeExp      pop gdpPercap is_Asia
   <fct>       <fct>     <int>   <dbl>    <int>     <dbl> <chr>
 1 Afghanistan Asia       2002    42.1 25268405      727. Asian country
 2 Albania     Europe     2002    75.7  3508512     4604. European country
```

```
 3 Algeria     Africa     2002    71.0  31287142     5288. African country
 4 Angola      Africa     2002    41.0  10866106     2773. African country
 5 Argentina   Americas   2002    74.3  38331121     8798. American country
 6 Australia   Oceania    2002    80.4  19546792    30688. Oceanian country
 7 Austria     Europe     2002    79.0   8148312    32418. European country
 8 Bahrain     Asia       2002    74.8    656397    23404. Asian country
 9 Bangladesh  Asia       2002    62.0 135656790     1136. Asian country
10 Belgium     Europe     2002    78.3  10311970    30486. European country
# i 132 more rows
```

Note the specific syntax of `case_when`!

8 `mutate` an existing column with `across`

- convert pop to millions and round to 3 decimal points

```
gapminder_2002 %>%
  mutate(across(pop, ~ round(.x/1000000, 3)))
```

```
# A tibble: 142 x 5
   country     continent lifeExp     pop gdpPercap
   <fct>       <fct>       <dbl>   <dbl>     <dbl>
 1 Afghanistan Asia         42.1  25.3        727.
 2 Albania     Europe       75.7   3.51      4604.
 3 Algeria     Africa       71.0  31.3       5288.
 4 Angola      Africa       41.0  10.9       2773.
 5 Argentina   Americas     74.3  38.3       8798.
 6 Australia   Oceania      80.4  19.5      30688.
 7 Austria     Europe       79.0   8.15     32418.
 8 Bahrain     Asia         74.8   0.656    23404.
 9 Bangladesh  Asia         62.0 136.        1136.
10 Belgium     Europe       78.3  10.3      30486.
# i 132 more rows
```

`across` lists the columns on which you want to perform the same transformation. Note its peculiar syntax: it is inside `mutate` and incorporates both the enumeration of the columns to transform and the transformation, which is written in the formula notation with tilde and `.x` for the first argument.

9 `mutate` an existing column on a selection of columns

```
gapminder_2002 %>% select(!c(continent)) %>%
  mutate(across(
    where(~ is.numeric(.x)),
          ~ round(.x, digits = 0)
    )    )
```

```
# A tibble: 142 x 4
   country     lifeExp       pop gdpPercap
   <fct>         <dbl>     <dbl>     <dbl>
 1 Afghanistan      42  25268405       727
 2 Albania          76   3508512      4604
 3 Algeria          71  31287142      5288
 4 Angola           41  10866106      2773
 5 Argentina        74  38331121      8798
 6 Australia        80  19546792     30688
 7 Austria          79   8148312     32418
 8 Bahrain          75    656397     23404
 9 Bangladesh       62 135656790      1136
10 Belgium          78  10311970     30486
# i 132 more rows
```

`across` works well with all column-selecting helper functions, such as `where` or `starts_with`.

10 Compute summary stats across columns

- `sum` , `mean` and many other functions collapse all values in a column.
- Override with `rowwise() %>% mutate(c_across(….)`

```
iris %>% slice(1:3) %>% rowwise() %>%
  mutate(sum_leafmeasure = sum(c_across(where(~ is.numeric(.x))))) %>%
  ↪ ungroup()
```

```
# A tibble: 3 x 6
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species sum_leafmeasure
         <dbl>       <dbl>        <dbl>       <dbl> <fct>             <dbl>
1          5.1         3.5          1.4         0.2 setosa             10.2
2          4.9         3            1.4         0.2 setosa              9.5
3          4.7         3.2          1.3         0.2 setosa              9.4
```

11 Without `rowwise` and `c_across`

```
iris %>% slice(1:3) %>%
  mutate(sum_leafmeasure = sum(Sepal.Length, Sepal.Width, Petal.Length,
  ↪  Petal.Width))
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species sum_leafmeasure
1          5.1         3.5          1.4         0.2  setosa            29.1
2          4.9         3.0          1.4         0.2  setosa            29.1
3          4.7         3.2          1.3         0.2  setosa            29.1
```

12 The same with `mean`

```
iris %>% slice(1:3) %>% rowwise() %>%
  mutate(mean_leafmeasure = mean(c_across(!c(Species)))) %>%   ungroup()
```

```
# A tibble: 3 x 6
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species mean_leafmeasure
         <dbl>       <dbl>        <dbl>       <dbl> <fct>              <dbl>
1          5.1         3.5          1.4         0.2 setosa              2.55
2          4.9         3            1.4         0.2 setosa              2.38
3          4.7         3.2          1.3         0.2 setosa              2.35
```

13 Common stat. functions requiring `rowwise`

- `sum()` – sums values across columns
- `mean()` – computes average across columns
- `sd()` – standard deviation
- `var()` – variance
- `min()` / `max()` – minimum / maximum
- `median()` – median value
- `IQR()` – interquartile range