Data frames. The base-R way.

Silvie Cinková

2025-08-11

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
```

2 Quick orientation

```
str(gapminder)
```

```
tibble [1,704 x 6] (S3: tbl_df/tbl/data.frame)
 $ country  : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ continent: Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ year     : int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
 $ lifeExp  : num [1:1704] 28.8 30.3 32 34 36.1 ...
 $ pop      : int [1:1704] 8425333 9240934 10267083 11537966 13079460 14880372 12881816 1386
 $ gdpPercap: num [1:1704] 779 821 853 836 740 ...
```

```
summary(gapminder)
```

```
        country        continent        year         lifeExp
 Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
 Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
 Algeria    :  12   Asia    :396   Median :1980   Median :60.71
 Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
 Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
 Australia  :  12                  Max.   :2007   Max.   :82.60
```

```
 (Other)    :1632
      pop              gdpPercap
 Min.   :6.001e+04   Min.   :   241.2
 1st Qu.:2.794e+06   1st Qu.:  1202.1
 Median :7.024e+06   Median :  3531.8
 Mean   :2.960e+07   Mean   :  7215.3
 3rd Qu.:1.959e+07   3rd Qu.:  9325.5
 Max.   :1.319e+09   Max.   :113523.1
```

```
    nrow(gapminder)
```

```
[1] 1704
```

```
    ncol(gapminder)
```

```
[1] 6
```

```
    colnames(gapminder)
```

```
[1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
    str(colnames(gapminder))
```

```
 chr [1:6] "country" "continent" "year" "lifeExp" "pop" "gdpPercap"
```

3 Subset rows and columns

- a single step, unlike `dplyr`

like a vector, but 2 positions:

- `[rows_vector, columns vector ]`
- this always gives you a data frame, not vectors

4 Subset rows and columns

```
    colnames(gapminder)
```

```
[1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
    gapminder[c(1:2, 13), c(1:3)]
```

```
# A tibble: 3 x 3
  country     continent  year
  <fct>       <fct>     <int>
1 Afghanistan Asia       1952
2 Afghanistan Asia       1957
3 Albania     Europe     1952
```

5 Either col names or positions

```
    gapminder[, c(1, "year")]
```

```
Error in `gapminder[, c(1, "year")]`:
! Can't subset columns that don't exist.
x Column `1` doesn't exist.
```

This is possible in `dplyr`, but not in base R:

```
    select(gapminder, c(1, "year", pop)) %>%
      slice(1)
```

```
# A tibble: 1 x 3
  country      year     pop
  <fct>       <int>   <int>
1 Afghanistan  1952 8425333
```

In `dplyr::select` you can freely combine position indices with column names with or without quotes. This is impossible in base R, where the column vector is a regular vector: when you combine numbers and strings, it interprets the numbers as strings and hence tries to select columns that are named with numbers (e.g. "3"), while you refer to the third column without mentioning its name. A column name without quotes is considered a name of a variable (i.e. not a data frame column), so R starts looking for one outside the data frame.

- combination of column names and position indices only in `dplyr`.

6 Subset only rows (all columns)

- mind the comma!!!
- one position without comma ≈ **columns** vector!

```
  gapminder[c(1, 175), ] %>%
    slice(1)
```

```
# A tibble: 1 x 6
  country     continent  year lifeExp     pop gdpPercap
  <fct>       <fct>     <int>   <dbl>   <int>     <dbl>
1 Afghanistan Asia       1952    28.8 8425333      779.
```

7 Subset only columns (all rows)

- preceded by the comma or without

```
  #gapminder[c(1,3)] # equivalent
  gapminder[,c(1,3)] %>%
    slice(1)
```

```
# A tibble: 1 x 2
  country      year
  <fct>       <int>
1 Afghanistan  1952
```

8 `dplyr::pull` in base R

- access a column as a vector

```
  gapminder$country %>% str()
```

```
 Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
gapminder$year %>% str()
```

```
 int [1:1704] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
```

9 Filter rows by tests on column values

```
gapminder[gapminder$year > 2002,] %>% slice(1:2)
```

```
# A tibble: 2 x 6
  country     continent  year lifeExp      pop gdpPercap
  <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
1 Afghanistan Asia       2007    43.8 31889923      975.
2 Albania     Europe     2007    76.4  3600523     5937.
```

```
# gapminder["year" > 2002] # same, mind quotes!
```

10 Renaming columns

```
colnames(gapminder)[c(4, 6)]
```

```
[1] "lifeExp"   "gdpPercap"
```

```
colnames(gapminder)[c(4, 6)] <- toupper(colnames(gapminder)[c(4, 6)])
colnames(gapminder)
```

```
[1] "country"   "continent" "year"      "LIFEEXP"   "pop"       "GDPPERCAP"
```

Column names of a data frame are a character vector. You can overwrite elements inside a vector by subsetting the vector to expose them and then you assign to these positions the new values. You can think of it as of a *Find* and *Replace* procedure on a vector. This really changes the positions inside the vector.

```
a <- c("apple", "banana")
a[1] <- "CARROT"
a
```

```
[1] "CARROT" "banana"
```

11 Restore the original `gapminder`

```
gapminder <- gapminder::gapminder
```

Otherwise we would stick with its version with renamed columns.

12 The `subset` function

```
subset(gapminder, subset = year < 1957, select = c(year, country, pop))
```

```
# A tibble: 142 x 3
    year country          pop
   <int> <fct>          <int>
 1  1952 Afghanistan  8425333
 2  1952 Albania      1282697
 3  1952 Algeria      9279525
 4  1952 Angola       4232095
 5  1952 Argentina   17876956
 6  1952 Australia    8691212
 7  1952 Austria      6927772
 8  1952 Bahrain       120447
 9  1952 Bangladesh  46886859
10  1952 Belgium      8730405
# i 132 more rows
```

This is a regular base-R function. Note that you must access the column names without quotes. You can use both the arguments `subset` (rows) and `select` (columns), or either.