

Raport pracy projektowej nr 2

Mariola Bartosik

Wstęp

W raporcie przedstawię wyniki swoich badań prowadzonych na ramkach danych, dostępnych na stronie internetowej <http://www.gagolewski.com/resources/data/>.

Korzystałam z danych przedstawionych poniżej.

```
options(stringsAsFactors=FALSE)
Tags <- read.csv("~/R/travel_stackexchange_com/Tags.csv")
Badges <- read.csv("~/R/travel_stackexchange_com/Badges.csv")
Comments <- read.csv("~/R/travel_stackexchange_com/Comments.csv")
Posts <- read.csv("~/R/travel_stackexchange_com/Posts.csv")
Users <- read.csv("~/R/travel_stackexchange_com/Users.csv")
Votes <- read.csv("~/R/travel_stackexchange_com/Votes.csv")
PostLinks <- read.csv("~/R/travel_stackexchange_com/PostLinks.csv")
```

Każde zadanie wykonane jest na 4 sposoby, za pomocą

- funkcji bazowych R,
- funkcji z biblioteki sqldf,
- funkcji z biblioteki dplyr,
- funkcji z biblioteki data.table.

Dodatkowo porównałam czasy wykonania napisanych przeze mnie funkcji przy użyciu jednego wywołania `microbenchmark::microbenchmark()`.

W większości zadań okazało się, że funkcje korzystające z biblioteki sqldf, osiągały najgorsze wyniki w badaniach, natomiast głównie to funkcje opierające się na funkcjach bazowych języka R były najszybsze.

W raporcie sprawdziłam także równoważność moich funkcji.

Zadanie 1

Zadanie nr 1 bazowało na ramce danych Posts. Wybrałam z kolumny PostTypeId wiersze, z wartościami równymi 1, z kolumny FavoriteCount wiersze z wartościami większymi równymi 25 oraz z kolumny ViewCount wiersze z wartościami większymi równymi 10000. Na samym końcu wyodrębniłam kolumny o nazwach: "Title, Score, ViewCount, FavoriteCount".

Wyniki w 4 przypadkach były identyczne. Uzyskałam ramkę z czterema kolumnami i dziewiętnastoma wierszami.

```
head(df_dplyr_1(), 2)

##                                     Title
## 1 When traveling to a country with a different currency, how should you take your money?
## 2                                     How can I do a "broad" search for flights?
##   Score ViewCount FavoriteCount
## 1   136    16838             35
## 2    95    33554             49

head(df_table_1(), 2)

##                                     Title
## 1: When traveling to a country with a different currency, how should you take your money?
## 2:                                     How can I do a "broad" search for flights?
##   Score ViewCount FavoriteCount
## 1:   136    16838             35
## 2:    95    33554             49
```

Wyniki sprawdzenia równoważności:

```
dplyr::all_equal(df_sql_1(), df_table_1())

## [1] TRUE

dplyr::all_equal(df_sql_1(), df_base_1())

## [1] TRUE
```

Porównałam czasy wykonania napisanych przeze mnie funkcji w tym zadaniu przy użyciu wywołania `microbenchmark::microbenchmark()`.

```
library(microbenchmark)
microbenchmark::microbenchmark( sqldf = df_sql_1(),
                                base = df_base_1(),
                                dplyr = df_dplyr_1(),
                                data.table = df_table_1() )

## Unit: milliseconds
##      expr      min       lq      mean     median        uq      max neval
##  sqldf 789.1144 1001.7060 1073.5050 1067.5414 1126.2142 1488.886   100
##   base 623.7783  755.3680  820.2490  805.7341  863.6592 1240.473   100
##  dplyr 647.2965  749.5027  819.5022  809.4673  862.6063 1136.825   100
## data.table 585.8920 767.2097  838.9345  828.5897  884.1617 1186.904   100
```

Zadanie 2

Zadanie nr 2 polegało na wybraniu odpowiednich wartości z kolumny OwnerUserId (różnych od -1) z ramki Tags. Korzystałam dwukrotnie z inner join.

Złączenie tego typu zachowuje tylko wiersze występujące w obydwu złączonych zbiorach.

Złączeniu uległy kolumny WikiPostId oraz Posts z ramek Tags i Posts, a także kolumny AccountId i OwnerUserId z ramek Tags i Users. Posortowałam malejąco kolumnę powstałą po zliczeniu oraz wybrałam kolumny TagName, Count, OwnerUserId, Age, Location, DisplayName.

```
head(df_base_2(), 2)

##      TagName Count OwnerUserId Age      Location DisplayName
## 711  canada   802         101  34    Mumbai, India      hitec
## 740  europe   681         583  35 Philadelphia, PA  Adam Tuttle

head(df_dplyr_2(), 2)

##      TagName Count OwnerUserId Age      Location DisplayName
## 1  canada   802         101  34    Mumbai, India      hitec
## 2  europe   681         583  35 Philadelphia, PA  Adam Tuttle

head(df_table_2(), 2)

##      TagName Count OwnerUserId Age      Location DisplayName
## 1:  canada   802         101  34    Mumbai, India      hitec
## 2:  europe   681         583  35 Philadelphia, PA  Adam Tuttle
```

Wyniki sprawdzenia równoważności:

```
dplyr::all_equal(df_sql_2(), df_table_2())

## [1] TRUE

dplyr::all_equal(df_sql_2(), df_dplyr_2())

## [1] TRUE
```

Porównałam czasy wykonania napisanych przeze mnie funkcji w tym zadaniu przy użyciu wywołania microbenchmark::microbenchmark(). Oto wyniki:

```
library(microbenchmark)
microbenchmark::microbenchmark( sqldf = df_sql_2(),
                                base = df_base_2(),
                                dplyr = df_dplyr_2(),
                                data.table = df_table_2() )

## Unit: milliseconds
##      expr      min       lq      mean   median       uq      max neval
##    sqldf 773.5730 836.3103 1132.4710 884.2067 1542.2149 2042.699   100
##      base 531.4901 584.0236  800.0055 626.5875 1102.2856 1572.257   100
##      dplyr 553.0021 600.0194  785.6512 638.7535  911.1704 1664.025   100
## data.table 572.3691 619.2486  754.9702 634.0702  792.4934 1497.143   100
```

Zadanie 3

Zadanie nr 3 polegało na pogrupowaniu względem RelatedPostId. Zliczyłam wartości w RelatedPostId i zapisałam kolumnę z wartościami zliczenia jako NumLinks. Poza tym należało zmieścić nazwę RelatedPostId na PostId, a także zapisać wynik w postaci ramki danych o nazwie RelatedTab. Kolejnym krokiem było złączenie z ramek danych RelatedTab i Posts kolumn PostId i Id zachowując wiersze występujące w obydwu kolumnach. Następnie wybrałam wartości w kolumnie PostTypeId równe 1. Posortowałam malejąco kolumnę NumLinks oraz wybrałam poszczególne kolumny takie jak TagName, Count, OwnerUserId, Age, Location, DisplayName.

```
head(df_base_3(), 2)

##                                     Title
## 2262 Is there a way to find out if I need a transit visa for a layover in the UK?
## 2165 Do I need a visa to transit (or layover) in the Schengen area?
##      NumLinks
## 2262      594
## 2165      585

head(df_sql_3(), 2)

##                                     Title
## 1 Is there a way to find out if I need a transit visa for a layover in the UK?
## 2 Do I need a visa to transit (or layover) in the Schengen area?
##      NumLinks
## 1      594
## 2      585
```

Sprawdziłam równoważność funkcji:

```
dplyr::all_equal(df_sql_3(), df_table_3())

## [1] TRUE

dplyr::all_equal(df_base_3(), df_dplyr_3())

## [1] TRUE
```

Porównałam czasy wykonania napisanych przeze mnie funkcji w tym zadaniu przy użyciu wywołania microbenchmark::microbenchmark().

```
library(microbenchmark)
microbenchmark::microbenchmark( sqldf = df_sql_3(),
                                base = df_base_3(),
                                dplyr = df_dplyr_3(),
                                data.table = df_table_3() )

## Unit: milliseconds
##      expr      min       lq      mean   median       uq      max neval
##  sqldf 532.8894 583.27290 635.50298 612.1686 651.80420 1122.2581   100
##   base  72.4688  75.66865  87.81367  78.5910  87.21975  232.4850   100
##  dplyr 394.0849 439.94155 493.26690 456.6885 496.18045 1740.6197   100
## data.table 387.1394 430.76040 480.87313 465.0478 492.15400  920.6607   100
```

Zadanie 4

Zadanie nr 4 na początku polegało na znalezieniu odpowiednich wartości Name w ramce Badges. Takich, dla których dane w kolumnie Class mają wartość równą 1. Pogrupowałam ramkę względem Name. Wybrałam wartości po zliczeniu większe od dwóch i mniejsze od 10. Potem wybrałam tylko te wartości Name z Users, które dostaliśmy po wyselekcjowaniu z Badges. Także, należało wybrać dane, dla których wartość z kolumny Class jest równa 1. Zapisalam to jako ValuableBadges. Złączyłam tak jak wyżej z ramek Users i Badges odpowiednio kolumny UserId i Id. Na samym końcu wybrałam odpowiednio unikalne wartości z kolumn Id, DisplayName, Reputation, Age, Location.

```
head(df_base_4(), 2)

##      Id DisplayName Reputation Age                Location
## 1   19      VMAtm      18556  33      Tampa, FL, United States
## 2  101    Mark Mayo     121667  37 Sydney, New South Wales, Australia

head(df_sql_4(), 2)

##      Id      DisplayName Reputation Age                Location
## 1  108  Ankur Banerjee     31273  27                London, UK
## 2   19           VMAtm     18556  33 Tampa, FL, United States
```

Sprawdziłam równoważność funkcji:

```
dplyr::all_equal(df_sql_3(), df_table_3())

## [1] TRUE

dplyr::all_equal(df_base_3(), df_dplyr_3())

## [1] TRUE
```

Porównałam czasy wykonania napisanych przeze mnie funkcji w tym zadaniu przy użyciu wywołania `microbenchmark::microbenchmark()`.

```
library(microbenchmark)
microbenchmark( sqldf = df_sql_4(),
                 base = df_base_4(),
                 dplyr = df_dplyr_4(),
                 data.table = df_table_4() )

## Unit: milliseconds
##      expr      min       lq      mean     median        uq      max neval
##    sqldf 584.5225 606.0258 629.1017 632.5200 637.4883 775.8139   100
##      base 380.0939 419.9666 426.2338 426.2851 429.7008 564.5414   100
##     dplyr 386.9023 394.8282 425.2795 429.9413 435.6966 599.1039   100
## data.table 394.6829 437.0849 449.4452 440.7993 448.4833 613.3886   100
```

Zadanie 5

Zadanie nr 5 polegało na wybraniu z ramki Votes kolumny PostId, gdzie wartości w kolumnie VoteTypeId są równe 2. Pogrupowałam PostId. Zliczyłam wartości i zapisałam jako UpVotes. Powstała nowa ramka, którą zapisałam jako UpVotesTab. Analogicznie postępowałam przy tworzeniu DownVotesTab, gdzie zliczone wartości zapisałam jako DownVotes, a wcześniej z kolumny VoteTypeId wybrałam wartości równe 3. Złączyłam za pomocą left join kolumny PostId z obu ramek. Złączenie tego typu wybiera każdy wiersz z pierwszego zbioru, dołączając do niego wszystkie pasujące wiersze z drugiego zbioru występujące w lewym zbiorze. Wybrałam kolumny PostId, UpVotes, DownVotes, gdzie kolumnie DownVotes wszystkie wartości nieokreślone zostały zamienione zerem.

```
head(df_base_5(), 5)

##   PostId UpVotes DownVotes
## 1      1      10         2
## 2      2      32         0
## 3      3      13         1
## 4      4       9         1
## 5      5      14         0
```

```
head(df_table_5(), 5)

##   PostId UpVotes DownVotes
## 1:      1      10         2
## 2:      2      32         0
## 3:      3      13         1
## 4:      7       6         3
## 5:      6      82         0
```

Sprawdziłam równoważność funkcji:

```
dplyr::all_equal(df_sql_5(), df_table_5())

## [1] TRUE

dplyr::all_equal(df_dplyr_5(), df_sql_5())

## [1] TRUE
```

Porównałam czasy wykonania napisanych przeze mnie funkcji w tym zadaniu przy użyciu wywołania `microbenchmark::microbenchmark()`.

```
library(microbenchmark)
microbenchmark::microbenchmark( sqldf = df_sql_5(),
                                base = df_base_5(),
                                dplyr = df_dplyr_5(),
                                data.table = df_table_5() )

## Unit: milliseconds
##      expr      min       lq      mean    median      uq      max neval
##   sqldf 1463.7833 1546.116 1672.799 1619.176 1739.454 2376.650   100
##    base 2273.7034 2372.642 2583.329 2471.681 2595.483 3555.116   100
##   dplyr  971.9892 1050.551 1178.005 1103.924 1233.640 1958.576   100
## data.table 923.7632 1027.637 1166.026 1129.489 1199.678 1922.170   100
```

Zadanie 6

Zadanie nr 6 było podobne do zadania 5, różnica, polegała na kolejności w złączeniach. Przez co dostałam dwie różne ramki danych. Następnie należało połączyć je i wybrać kolumny PostId i kolumnę, która powstała po odjęciu UpVotes od DownVotes.

```
head(df_table_6(), 5)
```

```
##      PostId Votes
## 1:         1     8
## 2:         2    32
## 3:         3    12
## 4:         7     3
## 5:         6    82
```

```
head(df_sql_6(), 5)
```

```
##      PostId Votes
## 1         1     8
## 2         2    32
## 3         3    12
## 4         4     8
## 5         5    14
```

Sprawdziłam równoważność funkcji:

```
dplyr::all_equal(df_sql_6(), df_table_6())
```

```
## [1] TRUE
```

```
dplyr::all_equal(df_sql_6(), df_dplyr_6())
```

```
## [1] TRUE
```

Porównałam czasy wykonania napisanych przeze mnie funkcji w tym zadaniu przy użyciu wywołania `microbenchmark::microbenchmark()`.

```
library(microbenchmark)
```

```
microbenchmark::microbenchmark( sqldf = df_sql_6(),
                                base = df_base_6(),
                                dplyr = df_dplyr_6(),
                                data.table = df_table_6() )
```

```
## Unit: milliseconds
```

```
##      expr      min       lq      mean   median      uq      max neval
##  sqldf 1733.0968 1847.097 2044.225 1997.467 2164.507 2870.248   100
##   base 2347.7436 2484.151 2673.107 2584.318 2702.466 3533.173   100
##  dplyr 1028.6701 1121.303 1280.202 1229.491 1339.755 2018.457   100
## data.table 923.0636 1063.471 1231.055 1184.399 1257.564 2009.144   100
```