Raport pracy projektowej nr 2

12.05.2020 **MXMX**

Wstep

W raporcie przedstawie wyniki swoich badań prowadzonych na ramkach danych, dostepnych na stronie internetowej http://www.gagolewski.com/resources/data/.

Korzystałam z danych przedstawonych 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 pomoca:

- 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 niektórych zadaniach okazało sie, że funkcje korzystajace z biblioteki sqldf, osiagały najsłabsze wyniki w badaniach. Jednakże wszystko zależało od konkretnego przypadku.

W raporcie sprawdziłam także równoważność moich funkcji.

Plan każdego zadania opisanego w raporcie:

1. opis
2. wynik funkcji
3. równoważność funkcji
4. sprawdzenie czasów wykonania napisanych funkcji

Zadanie 1

W zadaniu należało z Postów wybrać Identyfikatory Typów Postów równe 1, Ulubiona Liczbe wieksza równa 25 oraz Liczbe Obserwujacych wieksza równa 10000. Na koniec wybrałam Tytuły, Wyniki, Liczby Obserujacych, Ulubione Liczby.

```
head(df_dplyr_1(), 5)
```

```
##                                                                                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?
## 3                                     Tactics to avoid getting harassed by corrupt police?
## 4                                                              How to avoid drinking vodka?
## 5                         Flight tickets: buy two weeks before even during holiday seasons?
##   Score ViewCount FavoriteCount
## 1   136     16838            35
## 2    95     33554            49
## 3   156     13220            42
## 4   149     15197            29
## 5   109     49440            36
```

Wyniki sprawdzenia równoważności:

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

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

```
dplyr::all_equal(df_sql_1(), df_dplyr_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 wywolania microbenchmark::microbenchmark(). Oto wyniki:

```
## Unit: milliseconds
##        expr      min       lq     mean   median       uq       max neval
##       sqldf 465.3121 685.0362 744.5089 745.7440 802.3308 1103.6253   100
##        base 332.7947 508.4629 552.5811 551.0104 601.8623  978.6635   100
##       dplyr 337.8434 472.4400 572.0524 572.3564 634.5258 1076.2433   100
##  data.table 341.0138 531.0666 586.9606 584.3256 638.1422 1040.9480   100
```

Okazało sie, że funkcja korzystajaca z biblioteki sqldf osiagneła najgorszy wynik. Natomiast najlepiej wypadła funkcja korzystajaca z funkcji bazowych R, podobne wyniki uzyskała funkcja korzystajaca z bibilioteki dplyr.

Zadanie 2

W tym zadaniu musiałam wybrać z Tagów Identyfikatory Użytkowników rożne od -1. Złaczyłam Identyfikatory Postów Wiki z Tagów i Posty z Postów, zachowujac tylko wartości wystepujace w obydwu złaczonych zbiorach. W ten sam sposób złaczyłam Identyfiakotory Kont z Tagów i Identyfikatory Użytkowników z Postów. Posortowałam malejaco Zliczenia oraz wybrałam Nazwy Tagów, Zliczenia, Identyfikatory Użytkownikow, Wiek, Lokalizacje, Nazwe Wyświetlana.

```
head(df_table_2(), 7)
```

```
##          TagName Count OwnerUserId Age         Location      DisplayName
## 1:        canada   802         101  34    Mumbai, India            hitec
## 2:        europe   681         583  35 Philadelphia, PA      Adam Tuttle
## 3: visa-refusals   554        1737  34     New York, NY Benjamin Pollack
## 4:     australia   411         101  34    Mumbai, India            hitec
## 5:            eu   204         583  35 Philadelphia, PA      Adam Tuttle
## 6: new-york-city   204         101  34    Mumbai, India            hitec
## 7:   south-korea   193         101  34    Mumbai, India            hitec
```

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
```

```
dplyr::all_equal(df_sql_2(), df_base_2())
```

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

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

```
## Unit: milliseconds
##        expr      min        lq     mean    median       uq        max neval
##       sqldf 740.1641 1307.0122 1388.098 1398.9004 1500.935   1733.070   100
##        base 509.0385  888.0871 2399.516  969.4074 1044.879 144367.670   100
##       dplyr 538.9831  936.5240 1012.060 1010.5085 1086.453   1349.119   100
##  data.table 547.9809  969.4334 1035.573 1026.9082 1110.826   1408.382   100
```

Okazuje sie, że i tutaj funkcja opierajaca sie na funkcjach bazowych osiagneła najlepsze wyniki,a najsłabsze funkcja korzystajaca z biblioteki sqldf. Wiec podobnie jak w zadaniu nr 1.

Zadanie 3

Zadanie nr 3 polegało na pogrupowaniu wzgledem Identyfikatora Powiazanego Postu. Zliczyłam wartosci Identyfikatora Powiazanego Postu i zapisałam jako Liczbe Linków. Poza tym należało zmiecić nazwe Identyfikatora Powiazanego na Identyfiaktor Postu, a także zapisac wynik jako Powiazana Zakładka. Kolejnym krokiem było złaczenie IdentyfikatoróW Postów i Identyfikatorów z Powiazanej Zakładki zachowujac dane wystepujace w obydwu zbiorach. Nastepnie wybrałam wartości Identyfikatorów Postów równe 1. Posortowałam malejaco Numery Linków oraz wybrałam Nazwy Tagów, Zliczenia, Identyfikatory Użytkowników, Wiek, Lokalizacje, Nazwe Wyświetlana.

```
head(df_base_3(), 3)
```

```
##                                                                        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?
## 1168        Should my first trip be to the country which issued my Schengen Visa?
##      NumLinks
## 2262      594
## 2165      585
## 1168      331
```

Sprawdziłam równoważność funkcji:

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

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

```
dplyr::all_equal(df_sql_3(), df_dplyr_3())
```

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

```
dplyr::all_equal(df_sql_3(), df_base_3())
```

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

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

```
## Unit: milliseconds
##        expr      min       lq     mean   median       uq       max neval
##       sqldf 497.9296 545.5560 712.9202 619.9620 867.3120 1188.8151   100
##        base  69.2606  71.8927 104.1230  99.3384 110.8982  315.8103   100
##       dplyr 368.3135 410.5007 546.8540 556.1381 657.5821  939.8920   100
##  data.table 356.0085 427.7073 539.9578 526.4511 644.0710  855.0583   100
```

Okazuje sie, że zdecydowanie najlepsze, kilkuktornie mniejsze wyniki czasu potrzebnego do wykonania funkcji osiagneła bazowa. Po raz kolejny funkcja korzystajaca z bilblioteki sqldf osiagneła najsłabsze wyniki.

Zadanie 4

Zadanie nr 4 na poczatku polegało na znalezieniu odpowiednich Nazw Znaczków. Takich, których Klasy sa równe 1. Pogrupowałam Znaczki wzgledem Nazw. Wybrałam wartości po zliczeniu wieksze od dwóch i mniejsze od 10. Potem wybrałam tylko te Nazwy Użytkowników, które dostaliśmy po wyselekcjowaniu z ZnaczkóW. Także, należało wybrać Klase równa 1. Zapisałam to jako Wartościowe Znaczki. Złaczyłam tak jak w zadaniu 3 Identyfikatory Użytkowników z Użytkowników i Identyfikatory z Znaczków. Na samym końcu wybrałam odpowiednio unikalne wartości z Identyfikatorów, Nazwy Wyświetlanej, Reputacji, Wieku, Lokalizacji.

```
head(df_base_4(),5)
```

```
##    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
## 4 108     Ankur Banerjee      31273  27                         London, UK
## 6 466     iHaveacomputer       8360  NA                       Down underer
## 7 693 RoflcoptrException      33300  NA
```

Sprawdziłam równoważność funkcji:

```
dplyr::all_equal(df_sql_4(), df_table_4())
```

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

```
dplyr::all_equal(df_sql_4(), df_dplyr_4())
```

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

```
dplyr::all_equal(df_sql_4(), df_base_4())
```

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

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

```
## Unit: milliseconds
##        expr      min       lq     mean    median        uq       max neval
##       sqldf 795.6408 911.2942 998.1999 1006.9873 1069.9580 1291.7264   100
##        base 530.8467 628.2658 678.5104  673.9930  714.5748  945.2171   100
##       dplyr 536.2133 603.1490 670.9736  670.1964  721.1451  954.9919   100
##  data.table 544.7278 653.4627 699.0523  698.0307  737.0682 1057.2304   100
```

W tym przypadku funkcje korzystajace z bibliotek dplyr i data.table oraz korzystajacej z funkcji bazowych R osiagneły podobne wyniki czasu, natomiast po raz kolejny funkcja korzystajaca z sqldf była wolniejsza.

Zadanie 5

Zadanie nr 5 polegało na wybraniu z Głosów, Identyfikatorów Postów, których Identyfikatory Typów Głosów sa równe 2. Pogrupowałam Identyfikatory Postów. Zliczyłam wartości i zapisałam jako Głosy na Tak. Powstał nowy zbiór, ktory zapisałam jako Zakładka z Głosami na Tak. Analogicznie postepowałam przy tworzeniu Zakładka z Głosami na Nie, gdzie zliczone wartości zapisałam jako Głosy na Nie, a wczesniej wybrałam Identyfikatory Typów Głosów równe 3. Złaczyłam Identyfikatory Postów z Zakładnki Głosów na Tak i Zakładki Głosów na Nie w taki sposób, że wybierałam wartości z pierwszego zbioru, dołaczajac wszystkie pasujace wartości z drugiego zbioru. Wybrałam Identyfikatory Postów, Głosy na Tak, Głosy na Nie, gdzie w Głosach na Nie wszystkie wartości nieokreślone zostały zamienione zerem.

```
head(df_base_5(), 4)
```

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

Sprawdziłam równoważność funkcji:

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

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

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

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

```
dplyr::all_equal(df_sql_5(), df_base_5())
```

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

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

```
## Unit: milliseconds
##        expr       min       lq     mean   median       uq        max neval
##       sqldf 1401.8130 2174.723 2396.627 2464.547 2637.935   3098.690   100
##        base 2212.8868 3329.990 3500.289 3519.749 3890.366   4563.955   100
##       dplyr  938.5556 1520.310 7663.320 1699.141 1882.832 602105.998   100
##  data.table  875.7885 1463.399 1613.740 1672.126 1816.053   2215.877   100
```

W tym przypadku jest zupełnie inaczej niż w poprzednich, ponieważ funkcja bazowa uzyskała najgorszy wynik. Co ciekawe to funkcja korzystajaca z biblioteki data.table była najszybsza. Wyniki sa zdecydowanie wieksze niż w poprzednich zadaniach.

Zadanie 6

Zadanie nr 6 było podobne do zadania 5, różnica, polegała na kolejności w złaczeniach. Przez co dostałam dwa różne zbiory. Nastepnie należało połaczyć je i wybrać Identyfikatory Postów i wartości, które powstały po odjeciu Górnych Głosów od Dolnych Głosów.

```
head(df_table_6(), 7)
```

```
##    PostId Votes
## 1:      1     8
## 2:      2    32
## 3:      3    12
## 4:      7     3
## 5:      6    82
## 6:     11    28
## 7:      9    13
```

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
```

```
dplyr::all_equal(df_sql_6(), df_base_6())
```

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

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

```
## Unit: milliseconds
##        expr       min       lq     mean   median       uq        max neval
##       sqldf 1680.3035 2534.881 3858.634 2807.882 3056.100 115541.493   100
##        base 2241.1405 3395.601 3583.092 3599.384 3907.166   4665.198   100
##       dplyr  952.4579 1599.322 1736.768 1778.829 1889.642   2263.697   100
##  data.table  909.0267 1555.121 1667.181 1698.316 1846.540   2278.837   100
```

W tym ostatnim przypadku okazało sie, że funkcja korzystajaca z funkcji bazowych R była najwolniejsza. Tutaj tak jak w zadaniu nr 5 funkcja z data.table osiagneła najlepszy wynik.