# Sauchanka_Lizaveta_317026_pd2

Lizaveta Sauchanka

09 05 2021

# Spis treści



# Rozdział I

## Wprowadzenie

Praca Domowa 2 z przedmiotu PDU polegała na rozwiązaniu pieciu (w moim przypadku czterech) poleceń SQL w czterech równoważnych sposobach i ich implementacji w R:

- rozwiązanie referencyjne;
- rozwiązanie w funkcjach bazowych R;
- rozwiązanie w funkcjach z pakietu **dplyr**;
- rozwiązanie w funkcjach z pakietu **data.table**.

Pracowałam na uproszczonym zrzucie zanonimizowanych danych z serwisu https://travel.stackexchange.com/, a mianowicie na trzech zbiorach danych (ramkach danych) **Tags**, **Posts**, **Users**, **Comments**.

W tym raporcie będę sprawdzała równowaznosci wyników (Rozdział II), mierzyła i oceniała czas wykonania funkcji (Rozdział 3).

# Rozdział II

## Interpretacja zapytań oraz sprawzenie wyników równowaznosci

### Zadanie 1

Teraz przejdziemy do treści Zadania 1. Mamy zaimplementować następujące polecenie SQL:

```
## SELECT TagName, Count
## FROM Tags
## ORDER BY Count DESC
## LIMIT 10
```

Innymi słowy, mamy wypisać kolumny *TagName* oraz *Count* z ramki danych **Tags**, posortować malejąco względem kolumny *Count* i wypisać pierwsze 10 rzędów.

Więc zobaczmy jak ma wyglądać ta ramka danych:

```
df_sql_1(Tags)
```

```
##                TagName Count
## 1                visas  5271
## 2                  usa  2858
## 3            air-travel  2830
## 4                   uk  2114
## 5             schengen  2094
## 6  customs-and-immigration  1798
## 7              transit  1204
## 8               trains  1031
## 9            passports   954
## 10       indian-citizens   916
```

Teraz przejdziemy do implementacji Zadania 1 w **funkcjach bazowych** R, w **funkcjach z pakietu dplyr** oraz w **funkcjach z pakietu data.table**:

```
df_base_1(Tags)
```

```
##                 TagName Count
## 1                 visas  5271
## 2                   usa  2858
## 3           air-travel  2830
## 4                    uk  2114
## 5             schengen  2094
## 6  customs-and-immigration  1798
## 7               transit  1204
## 8                trains  1031
## 9             passports   954
## 10      indian-citizens   916
```

```
df_dplyr_1(Tags)
```

```
##                 TagName Count
## 1                 visas  5271
## 2                   usa  2858
## 3           air-travel  2830
## 4                    uk  2114
## 5             schengen  2094
## 6  customs-and-immigration  1798
## 7               transit  1204
## 8                trains  1031
## 9             passports   954
## 10      indian-citizens   916
```

```
df_table_1(Tags)
```

```
##                 TagName Count
## 1                 visas  5271
## 2                   usa  2858
## 3           air-travel  2830
## 4                    uk  2114
## 5             schengen  2094
## 6  customs-and-immigration  1798
## 7               transit  1204
## 8                trains  1031
## 9             passports   954
## 10      indian-citizens   916
```

Sprawdzimy czy te 3 ramki są identyczne za pomocą funkcji *all_equal()* z pakietu **dplyr** oraz *compare()* z pakietu **compare**:

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

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

```
dplyr::all_equal(df_sql_1(Tags), df_dplyr_1(Tags))
```

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

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

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

```
compare::compare(df_sql_1(Tags), df_base_1(Tags))
```

```
## TRUE
```

```
compare::compare(df_sql_1(Tags), df_dplyr_1(Tags))
```

```
## TRUE
```

```
compare::compare(df_sql_1(Tags), df_table_1(Tags))
```

```
## TRUE
```

Podsumowując, wszystkie wyniki implementacji Zadania 1 (funkcje bazowe, funkcje z pakietu **dplyr**, funkcje z pakietu **data.table**) są identyczne.

# Zadanie 2

Zobaczmy treść Zadania 2:

```
## SELECT Users.DisplayName, Users.Age, Users.Location,
##                   AVG(Posts.Score) as PostsMeanScore,
##                   MAX(Posts.CreationDate) AS LastPostCreationDate
##      FROM Posts
##      JOIN Users ON Users.AccountId=Posts.OwnerUserId
##      WHERE OwnerUserId != -1
##      GROUP BY OwnerUserId
##      ORDER BY PostsMeanScore DESC
##      LIMIT 10
```

Mamy wypisać kolumny *DisplayName*, *Age* oraz *Location* z ramki danych **Users**, policzyć średnie wartości kolumny *Score* z ramki dannych **Posts** i wypisać te wyniki jako kolumna *PostsMeanScore*, policzyć maksymalne wartości kolumny *CreationDate* z ramki dannych **Posts** i wypisać te wyniki jako kolumna *LastPostCreationDate*. Dlatego grupujemy odnośnie kolumny *OwnerUserId* z ramki danych **Posts**.

Dalej łączymy wybrane kolumny z ramki **Users** z wybranymi kolumnami z ramki danych **Posts**. Kluczem ramki **Users** jest kolumna *AccountId*, natomiast kluczem ramki danych **Posts** jest kolumna *OwnerUserId*.

Wybieramy wszystkie rzędy, w których wartość kolumny *OwnerUserId* nie równa się -1, sortujemy malejąco względem kolumny **PostsMeanScore** i wypisujemy pierwsze 10 rzędów.

Zobaczmy jak ta ramka danych wyglada:

```
df_sql_2(Posts, Users)
```

```
##    DisplayName Age                  Location PostsMeanScore
## 1         Oded  44      London, United Kingdom           52.0
## 2         Rook  NA                                     50.0
## 3     JPhi1618  NA Dallas, Texas, United States           45.0
## 4        csmba  42           San Francisco, CA           43.0
## 5     Petrogad  32                                     41.0
## 6         Josh  NA                   Australia           33.0
## 7  Dan Esparza  43   Atlanta, GA, United States           30.0
## 8        James  NA                                     29.5
## 9  Brad Rhoads  NA                  Nampa, ID           29.0
## 10       Dexter  34      London, United Kingdom           28.5
##       LastPostCreationDate
## 1  2011-11-03T14:40:36.870
## 2  2016-05-27T03:17:41.753
## 3  2015-09-26T17:43:19.090
## 4  2012-01-15T14:27:11.070
## 5  2016-04-16T19:44:21.020
## 6  2011-12-23T18:16:21.877
## 7  2014-01-26T21:12:59.673
## 8  2017-05-11T00:13:37.897
## 9  2016-04-11T11:10:14.240
## 10 2015-05-06T00:23:36.353
```

Przejdziemy do implementacji Zadania 2 w **funkcjach bazowych** R, w **funkcjach z pakietu dplyr** oraz w **funkcjach z pakietu data.table**:

```
df_base_2(Posts, Users)
```

```
##    PostsMeanScore    LastPostCreationDate DisplayName Age
## 1            52.0 2011-11-03T14:40:36.870        Oded  44
## 2            50.0 2016-05-27T03:17:41.753        Rook  NA
## 3            45.0 2015-09-26T17:43:19.090    JPhi1618  NA
## 4            43.0 2012-01-15T14:27:11.070       csmba  42
## 5            41.0 2016-04-16T19:44:21.020    Petrogad  32
## 6            33.0 2011-12-23T18:16:21.877        Josh  NA
## 7            30.0 2014-01-26T21:12:59.673 Dan Esparza  43
## 8            29.5 2017-05-11T00:13:37.897       James  NA
## 9            29.0 2016-04-11T11:10:14.240 Brad Rhoads  NA
## 10           28.5 2015-05-06T00:23:36.353      Dexter  34
##                      Location
## 1        London, United Kingdom
## 2
## 3  Dallas, Texas, United States
## 4             San Francisco, CA
## 5
## 6                     Australia
## 7    Atlanta, GA, United States
## 8
## 9                     Nampa, ID
## 10       London, United Kingdom
```

```
df_dplyr_2(Posts, Users)
```

```
## # A tibble: 10 x 5
##    PostsMeanScore LastPostCreationDate  DisplayName   Age Location
##             <dbl> <chr>                 <chr>       <int> <chr>
##  1           52   2011-11-03T14:40:36.~ Oded           44 "London, United Kingd~
##  2           50   2016-05-27T03:17:41.~ Rook           NA ""
##  3           45   2015-09-26T17:43:19.~ JPhi1618       NA "Dallas, Texas, Unite~
##  4           43   2012-01-15T14:27:11.~ csmba          42 "San Francisco, CA"
##  5           41   2016-04-16T19:44:21.~ Petrogad       32 ""
##  6           33   2011-12-23T18:16:21.~ Josh           NA "Australia"
##  7           30   2014-01-26T21:12:59.~ Dan Esparza    43 "Atlanta, GA, United ~
##  8           29.5 2017-05-11T00:13:37.~ James          NA ""
##  9           29   2016-04-11T11:10:14.~ Brad Rhoads    NA "Nampa, ID"
## 10           28.5 2015-05-06T00:23:36.~ Dexter         34 "London, United Kingd~
```

```
df_table_2(Posts, Users)
```

```
##     PostsMeanScore    LastPostCreationDate DisplayName Age
##  1:           52.0 2011-11-03T14:40:36.870        Oded  44
##  2:           50.0 2016-05-27T03:17:41.753        Rook  NA
##  3:           45.0 2015-09-26T17:43:19.090    JPhi1618  NA
##  4:           43.0 2012-01-15T14:27:11.070       csmba  42
##  5:           41.0 2016-04-16T19:44:21.020    Petrogad  32
##  6:           33.0 2011-12-23T18:16:21.877        Josh  NA
##  7:           30.0 2014-01-26T21:12:59.673 Dan Esparza  43
##  8:           29.5 2017-05-11T00:13:37.897       James  NA
##  9:           29.0 2016-04-11T11:10:14.240 Brad Rhoads  NA
## 10:           28.5 2015-05-06T00:23:36.353      Dexter  34
##                      Location
##  1:       London, United Kingdom
##  2:
##  3: Dallas, Texas, United States
##  4:            San Francisco, CA
##  5:
##  6:                    Australia
##  7:   Atlanta, GA, United States
##  8:
##  9:                    Nampa, ID
## 10:       London, United Kingdom
```

i sprawdzimy czy te 3 ramki są identyczne analogicznie do Zadania 1:

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

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

```
dplyr::all_equal(df_sql_2(Posts, Users), df_dplyr_2(Posts, Users))
```

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

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

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

```
compare::compare(df_sql_2(Posts, Users), df_base_2(Posts, Users),  # ignorujemy kolejność wypisywania kolumn
                 ignoreColOrder = T)
```

```
## TRUE
##   reordered columns
```

```
compare::compare(df_sql_2(Posts, Users), df_dplyr_2(Posts, Users), # ignorujemy kolejność wypisywania kolumn
                 ignoreColOrder = T)
```

```
## TRUE
##   reordered columns
```

```
compare::compare(df_sql_2(Posts, Users), df_table_2(Posts, Users), # ignorujemy kolejność wypisywania
                 ignoreOrder = T,                                  # kolumn, upuszczamy atrybuty
                 ignoreAttrs = T,
                 ignoreColOrder = T)
```

```
## TRUE
##   reordered columns
##   dropped attributes
```

Wszystkie ramki danych są identyczne.

# Zadanie 3

Treść Zadania 3 brzmi następująco:

```
## SELECT DisplayName, QuestionsNumber, AnswersNumber
##       FROM
##         (
##           SELECT COUNT(*) as AnswersNumber, Users.DisplayName, Users.Id
##           FROM Users
##           JOIN Posts ON Users.Id = Posts.OwnerUserId
##           WHERE Posts.PostTypeId = 1
##           GROUP BY Users.Id
##         )  AS Tab1
##       JOIN
##         (
##           SELECT COUNT(*) as QuestionsNumber, Users.Id
##           FROM Users
##           JOIN Posts ON Users.Id = Posts.OwnerUserId
##           WHERE Posts.PostTypeId = 2
##           GROUP BY Users.Id
##         ) AS Tab2
##       ON Tab1.Id = Tab2.Id
##       WHERE QuestionsNumber < AnswersNumber
##       ORDER BY AnswersNumber DESC
```

Aby otrzymać wynikową ramkę danych, musimy najpierw stworzyć pomocniczą tabelkę **Tab1**, zatem stworzyć pomocniczą tabelkę **Tab2**, dalej połaczyć tabelki **Tab1** i **Tab2** według kolumn *Id*, posortować tak, aby wartości kolumny *QuestionsNumber* były mniejsze od wartości kolumny *AnswersNumber* i pososrtować malejąco według kolumny *AnswersNumber*.

Przejdziemy teraz do tworzenia pomocniczej tabelki **Tab1**. Aby stworzyć **Tab1** mamy wypisać kolumny *DisplayName* i *Id* z ramki danych **Users**, policzyć wystąpienia w odpowiedziach nr Id i zapisać to jako kolumna *AnswersNumber*. Dalej łaczymy ramkę danych **Posts** i wybrane kolumny z ramki danych **Users**. Kluczami ramek są kolumny *OwnerUserId* z **Posts** i *Id* z **Users**. Ostatnim krokiem jest grupowanie odnośnie kolumny *Id*.

Ramke danych **Tab2** tworzymy analogicznie (za wykluczeniem łączenia: wypisujemy kolumnę *Id* z ramki danych **Users** i liczymy wystąpienia w odpowiedziach nr Id i zapisujemy wynik jako kolumna *QuestionsNumber*).

Teraz zobaczmy jak wygląda wynikowa ramka danych (wypiszemy tylko pierwsze 10 rzędów):

```
##                 DisplayName QuestionsNumber AnswersNumber
## 1                hippietrail            257           441
## 2         RoflcoptrException            206           238
## 3              Andrew Grimm             49           212
## 4                       nsn             90           174
## 5                      Ivan             71           122
## 6                  Blaszard             44            80
## 7                 Adrien Be             62            79
## 8                   neubert             31            73
## 9                   shirish              5            67
## 10 Canada - Area 51 Proposal              1            61
```

Przejdziemy do implementacji Zadania 1 w **funkcjach bazowych** R, w **funkcjach z pakietu dplyr** oraz w **funkcjach z pakietu data.table**:

```
head(df_base_3(Users, Posts), 10)
```

```
##                 DisplayName AnswersNumber QuestionsNumber
## 1                hippietrail           441            257
## 2         RoflcoptrException           238            206
## 3              Andrew Grimm           212             49
## 4                       nsn           174             90
## 5                      Ivan           122             71
## 6                  Blaszard            80             44
## 7                 Adrien Be            79             62
## 8                   neubert            73             31
## 9                   shirish            67              5
## 10 Canada - Area 51 Proposal            61              1
```

```
head(df_dplyr_3(Users, Posts), 10)
```

```
## # A tibble: 10 x 3
##    DisplayName              AnswersNumber QuestionsNumber
##    <chr>                            <int>           <int>
##  1 hippietrail                        441             257
##  2 RoflcoptrException                 238             206
##  3 Andrew Grimm                       212              49
##  4 nsn                                174              90
##  5 Ivan                               122              71
##  6 Blaszard                            80              44
##  7 Adrien Be                           79              62
##  8 neubert                             73              31
##  9 shirish                             67               5
## 10 Canada - Area 51 Proposal           61               1
```

```
head(df_table_3(Users, Posts), 10)
```

```
##                 DisplayName AnswersNumber QuestionsNumber
##  1:             hippietrail           441            257
##  2:      RoflcoptrException           238            206
##  3:           Andrew Grimm           212             49
##  4:                    nsn           174             90
##  5:                   Ivan           122             71
##  6:               Blaszard            80             44
##  7:              Adrien Be            79             62
##  8:                neubert            73             31
##  9:                shirish            67              5
## 10: Canada - Area 51 Proposal           61              1
```

Sprawdzimy, czy te 3 ramki są identycne:

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

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

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

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

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

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

```
compare::compare(df_sql_3(Users, Posts), df_base_3(Users, Posts),  # ignorujemy kolejność wypisywania kolumn
            ignoreColOrder = T)
```

```
## TRUE
##   reordered columns
```

```
compare::compare(df_sql_3(Users, Posts), df_dplyr_3(Users, Posts), # ignorujemy kolejność wypisywania kolumn
            ignoreColOrder = T)
```

```
## TRUE
##   reordered columns
```

```
compare::compare(df_sql_3(Users, Posts) ,df_table_3(Users, Posts), # ignorujemy kolejność wypisywania
            ignoreColOrder = T,                    # kolumn, upuszczamy atrybuty
            ignoreOrder = T,
            ignoreAttrs = T)
```

```
## TRUE
##   reordered columns
##   dropped attributes
```

Więc wszystkie ramki danych są identyczne.

# Zadanie 4

Zobaczmy trześć Zadania 4:

```
## SELECT
##          Posts.Title, Posts.CommentCount,
##          CmtTotScr.CommentsTotalScore,
##          Posts.ViewCount
##      FROM (
##          SELECT
##              PostID,
##              UserID,
##              SUM(Score) AS CommentsTotalScore
##          FROM Comments
##          GROUP BY PostID, UserID
##      ) AS CmtTotScr
##      JOIN Posts ON Posts.ID=CmtTotScr.PostID
##      WHERE Posts.PostTypeId=1
##      ORDER BY CmtTotScr.CommentsTotalScore DESC
##      LIMIT 10
```

Zgodnie z treścią zadania, mamy najpierw stworzyć pomocniczą ramkę danych **CmtTotScr**. Następnie wybrane kolumny *Title*, *CommentCount* i *ViewCount* z **Posts** połączyć z wybranymi kolumnami *CommentsTotalScore* z ramki danych **CmtTotScr** (kluczem ramki **Posts** jest kolumna *Id*, natomiast kluczem ramki **CmtTotScr** jest kolumna *PostId*), posortować tak, aby w wynikowej ramce danych wartości kolumny *PostTypeId* z **Posts** były równe 1, posortować malejąco względem kolumny *CommentsTotalScore* i wypisać pierwsze 10 rzędów.

Tworzymy pomocniczą ramkę danych w następujący sposób: wybieramy kolumny *PostID* i *UserID* z ramki *Comments* oraz sumujemy i grupujemy wartości z *Score* z **Comments** odnośnie kolumn *PostID* i *UserID*.

Więc zobaczmy tą ramkę danych:

```
df_sql_4(Comments, Posts)
```

```
##                                                                Title
## 1  Boss is asking for passport, but it has a stamp in it I don't want him to see. What to do?
## 2                      Why don't airlines have backup planes just in case of an emergency?
## 3            OK we're all adults here, so really, how on earth should I use a squat toilet?
## 4       How to cross a road by foot in a country that drives on the "other" side of the road
## 5                                  Where can I change my clothes at the airport?
## 6  Boss is asking for passport, but it has a stamp in it I don't want him to see. What to do?
## 7                              How to avoid toddlers on a long-distance plane flight?
## 8     Job interview in London requires me to wire money to the travel agent. Is this a scam?
## 9                                  What to do without underwear on a 4 day trip?
## 10                OK, we are all adults here, so what is a bidet for and how do I use it?
##    CommentCount CommentsTotalScore ViewCount
## 1            24                207     54982
## 2            26                172     14516
## 3            27                155     73808
## 4            25                140      5240
## 5            16                128     12020
## 6            24                121     54982
## 7            19                120     24955
## 8            23                116     14827
## 9            13                110     11713
## 10           28                109     52265
```

Teraz wypiszemy ramki dannych, które są stworzone za pomocą **funkcji bazowych**, **funkcji z pakietu dplyr** i **funkcji z pakietu data.table**

```
df_base_4(Comments, Posts)
```

```
##    CommentsTotalScore
## 1                 207
## 2                 172
## 3                 155
## 4                 140
## 5                 128
## 6                 121
## 7                 120
## 8                 116
## 9                 110
## 10                109
##                                                                Title
## 1  Boss is asking for passport, but it has a stamp in it I don't want him to see. What to do?
## 2                      Why don't airlines have backup planes just in case of an emergency?
## 3            OK we're all adults here, so really, how on earth should I use a squat toilet?
## 4       How to cross a road by foot in a country that drives on the "other" side of the road
## 5                                  Where can I change my clothes at the airport?
## 6  Boss is asking for passport, but it has a stamp in it I don't want him to see. What to do?
## 7                              How to avoid toddlers on a long-distance plane flight?
## 8     Job interview in London requires me to wire money to the travel agent. Is this a scam?
## 9                                  What to do without underwear on a 4 day trip?
## 10                OK, we are all adults here, so what is a bidet for and how do I use it?
##    CommentCount ViewCount
## 1            24     54982
## 2            26     14516
## 3            27     73808
## 4            25      5240
## 5            16     12020
## 6            24     54982
## 7            19     24955
## 8            23     14827
## 9            13     11713
## 10           28     52265
```

```
df_dplyr_4(Comments, Posts)
```

```
## # A tibble: 10 x 4
##    CommentsTotalSco~ Title                          CommentCount ViewCount
##                <int> <chr>                                 <int>     <int>
##  1               207 "Boss is asking for passport, but i~         24     54982
##  2               172 "Why don't airlines have backup pla~         26     14516
##  3               155 "OK we're all adults here, so reall~         27     73808
##  4               140 "How to cross a road by foot in a c~         25      5240
##  5               128 "Where can I change my clothes at t~         16     12020
##  6               121 "Boss is asking for passport, but i~         24     54982
##  7               120 "How to avoid toddlers on a long-di~         19     24955
##  8               116 "Job interview in London requires m~         23     14827
##  9               110 "What to do without underwear on a ~         13     11713
## 10               109 "OK, we are all adults here, so wha~         28     52265
```

```
df_table_4(Comments, Posts)
```

```
##     CommentsTotalScore
##  1:                207
##  2:                172
##  3:                155
##  4:                140
##  5:                128
##  6:                121
##  7:                120
##  8:                116
##  9:                110
## 10:                109
##                                                                                                  Title
##  1: Boss is asking for passport, but it has a stamp in it I don't want him to see. What to do?
##  2:                                                  Why don't airlines have backup planes just in case of an emergency?
##  3:                                        OK we're all adults here, so really, how on earth should I use a squat toilet?
##  4:                                    How to cross a road by foot in a country that drives on the "other" side of the road
##  5:                                                                    Where can I change my clothes at the airport?
##  6: Boss is asking for passport, but it has a stamp in it I don't want him to see. What to do?
##  7:                                                                 How to avoid toddlers on a long-distance plane flight?
##  8:                                  Job interview in London requires me to wire money to the travel agent. Is this a scam?
##  9:                                                                    What to do without underwear on a 4 day trip?
## 10:                                                 OK, we are all adults here, so what is a bidet for and how do I use it?
##     CommentCount ViewCount
##  1:           24     54982
##  2:           26     14516
##  3:           27     73808
##  4:           25      5240
##  5:           16     12020
##  6:           24     54982
##  7:           19     24955
##  8:           23     14827
##  9:           13     11713
## 10:           28     52265
```

i sprawdzimy czy te 3 funkcje są równoważne:

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

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

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

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

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

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

```
compare::compare(df_base_4(Comments, Posts), df_sql_4(Comments, Posts),  # ignorujemy kolejność wypisywania kolumn
                 ignoreColOrder = T)
```

```
## TRUE
##   reordered columns
```

```
compare::compare(df_dplyr_4(Comments, Posts), df_sql_4(Comments, Posts), # ignorujemy kolejność wypisywania kolumn,
                 ignoreOrder = T,                                        # upuszczamy atrybuty
                 ignoreAttrs = T,
                 ignoreColOrder = T)
```

```
## TRUE
##   reordered columns
##   dropped attributes
```

```
compare::compare(df_sql_4(Comments, Posts), df_table_4(Comments, Posts), # ignorujemy kolejność wypisywania kolumn,
                 ignoreColOrder = T,                                     # upuszczamy atrybuty
                 ignoreOrder = T,
                 ignoreAttrs = T)
```

```
## TRUE
##   reordered columns
##   dropped attributes
```

Otrzymujemy we wszystkich przypadkach wartość TRUE, więc ramki danych we wszystkich 4 (sqldf, base, dplyr, data.table) implementacjach są identycne.

# Rozdział III

## Pomiar i ocena czasu wykonania funkcji

### Zadanie 1

Skoro wyniki implementacji Zadania 1 we wszystkich czterech przypadkach są identyczne, możemy zmierzyć i ocenić czas wykonania tych funkcji. Wykorzystując funkcję *microbenchmark()* z pakietu **microbenchmark**, otrzymujemy:

```
## Unit: microseconds
##       expr    min       lq      mean   median       uq      max neval
##      sqldf 9221.9 10063.40 12037.081 10534.65 12408.15 36990.7   100
##       base  392.9   449.10   761.890   486.15   525.10 22367.5   100
##      dplyr 3208.1  3588.05  4073.013  3834.70  4378.95  6233.9   100
## data.table  233.7   579.80   715.018   759.30   861.65  1612.0   100
```

Zobaczmy, że rozwiązanie w *data.table* prawie jest takie same według prędkości jak rozwiązanie w *funkcjach bazowych*. Wolniejsza jest funkcja *dplyr* (rozwiązanie w funkcjach z pakietu dplyr) i najwolniejszą jest funkcja *sqldf* (rozwiązanie referencyjne).

Ale moim zdaniem najwygodniejszymi są rozwiązania *dplyr* i *data.table*. Kod napisany tymi funkcjami jest przejrzysty i w miarę krótki.

### Zadanie 2

Otrzymaliśmy, że wszystkie ramki danych są identyczne, więc zmierzymy i policzmy czas wykonania tych funkcji za pomocą funkcji *microbenchmark*:

```
## Unit: milliseconds
##       expr      min       lq      mean   median       uq      max neval
##      sqldf 232.9361 241.0871 249.45866 247.0362 254.16830 288.6899   100
##       base 535.3753 566.7699 585.55266 577.4452 595.21075 716.4216   100
##      dplyr 148.8767 162.5872 178.03778 177.4623 183.85740 289.0448   100
## data.table  14.7201  17.9561  23.45956  19.6342  22.86275 145.6875   100
```

*data.table* jest najszybszą funkcją. Ona liczy **38** razy szybczej niż funkcja *base*!

Drugą według szybkości jest funkcja *dplyr*, trzecią według szybkości jest funkcja *sqldf*, i najwolniejsza funkcja to *base*.

Moin zdaniem, najlepszymi funkcjami według przejrzystości i długości kodu są *data.table*, *dplyr* i *sqldf*. Mimo to, że w tym przypadku *sqldf* nie jest funkcją szybką, kod tej funkcji jest czytelny i krótki.

Zobaczymy wykres pomiaru czasu wykonania funkcji:

### Zadanie 3

Na podstawie sprawzenia wyników równowaznosci Zadania 3 możemy teraz zmierzyć i ocenić czas wykonania tych funkcji za pomocą funkcji *microbenchmark*:

```
## Unit: milliseconds
##       expr      min       lq      mean    median        uq       max neval
##      sqldf 268.8452  282.5904  302.99467  292.5748  309.80610  542.3479   100
##       base 500.6504  549.3775  587.11736  569.5918  615.53435  847.5881   100
##      dplyr 969.4177 1079.1246 1168.08938 1148.3963 1215.81265 1860.4573   100
## data.table  38.5428   43.3746   59.98207   61.7141   70.26485  208.2238   100
```

W tym przypadku funkcja *data.table* jest najszybsza i z tego możemy wywnioskować, że ona jest najleprzym rozwiązaniem.

Podobnie jak i *data.table*, *sqldf* będzie drugiej według prędkości. *sqldf* ma łatwo pisany kod (składnia tego kodu jest prosta) i czytelny.

Rozwiązanie w funkcjach bazowych *base* jest jednym z najwolniejszych rozwiązań, ale najwolniejszym rozwiązaniem jest, niestety, *dplyr*. Moim zdaniem, prędkość wykonywania zapytania Zadania 3 zależała od kodu. Tworzyłam dużo kopii tabel oraz niektóre funkcje bazowe nie udało mo się zamienić funkcjami z pakietu **dplyr**.

## Zadanie 4

Na podstawie sprawzenia wyników równowaznosci Zadania możemy teraz zmierzyć i ocenić czas wykonania tych funkcji za pomocą funkcji *microbenchmark*:

```
## Unit: milliseconds
##       expr       min        lq       mean    median        uq       max neval
##      sqldf  361.1776  373.4933  383.53028  376.7432  388.7875  433.9699   100
##       base 3047.1992 3183.7811 3259.08425 3223.7831 3323.1332 3971.2123   100
##      dplyr 1080.1035 1151.7917 1183.82562 1171.0269 1207.0951 1347.6940   100
## data.table   31.8022   45.6350   71.06114   66.5513   76.4789  235.8667   100
```

Podobnie jak i w Zadaniu 2 i Zadaniu 3 funkcja *data.table* jest najszybszą. W porównaniu do funkcji *base*, ona przyspiesza wykonanie zapytania prawie o **90** razy!

W tym przypadku jednoznacznie funkcja data.table jest najszybszym rozwiązaniem.

# Rozdział IV

# Podsumowanie

Mówiąc o Zadaniu 1, nie jesteśmy w stanie ocenić prędkość, zalety lub wady wszystkich 4 funkcji, bo polecenie jest któtkie, zapytanie jest proste i niektóre funkcje jesteśmy w stanie skrócić i przyspeszyć na różne sposoby. Więc na podstawie moich sprawdzeń, dla krótkich funkcji najleprzymi rozwiązaniami są *data.table* oraz *funkcje bazowe R*.

Mówiąc o Zadaniach 2-4, *data.table* jest najlepszym rozwiązaniem (biorąc pod uwagę tylko prędkość wykonania). *data.table* jest również funkcja z dobrze czytelnym i łatwo pisanym kodem.

Na drugim miejscu według moich sprawdzeń wyników i oceny czasu dla sładanych zapytań jest funkcja *dplyr*. To jest jedna z najszybszych funkcji. Ale moim zdaniem, w przypadku Zadania 3 napisany przeze mnie kod nie jest dobrzy, z tego wynika mała prędkośc wykonywania zapytania.

Dla dużych zapytań funkcja sqldf jest wygodna (mówiąc o czytelności i napisaniu kodu) oraz dość szybka, ale dla niewielkich ramek danych jest za wolna.

Rozwiązanie w funkcjach bazowych dla dużych ramek danych jest najgorsza. Kod napisany za pomocą funkcji bazowych jest trudno czytelny i najwolniejszym.