PDU 2020/2021

Praca domowa nr 2 (max. = 35 p.)

Maksymalna ocena: 35 p.

Termin oddania pracy: 11.05.2021, godz. 10:00

Do przesłania przy użyciu platformy `Moodle` **jedno archiwum `.zip`**[1] o nazwie typu `Nazwisko_Imie_NrAlbumu_pd2.zip`. W archiwum znajdować się powinien jeden katalog, `Nazwisko_Imie_NrAlbumu_pd2`, dopiero w którym umieszczone zostaną następujące pliki:

- `Nazwisko_Imie_NrAlbumu_pd2.R` - plik `R`-a zawierający kod potrzebny do rozwiązania zadań (tzn. definicje potrzebnych funkcji, zob. Informacje ogólne).
- `Nazwisko_Imie_NrAlbumu_pd2.Rmd` oraz jego skompilowana wersja `Nazwisko_Imie_NrAlbumu_pd2.html` - raport w `Markdown/knitr`
  - W raporcie należy wykonać polecenie `R`:

```
source("Nazwisko_Imie_NrAlbumu_pd2.R")
```

  - Wynikowy raport (`.html`) powinien zawierać sprawdzenie równoważności wyników uzyskanych przy pomocy każdej metody, pomiar i ocenę czasu wykonania oraz interpretację zapytań (zob. Informacje Ogólne).
  - Raport powinien także zawierać krótki wstęp, w tym wczytanie danych, oraz podsumowanie.

1 Zbiory danych

Będziemy pracować na uproszczonym zrzucie zanonimizowanych danych z serwisu https://travel.stackexchange.com/ (na marginesie: pełen zbiór danych dostępny jest pod adresem https://archive.org/details/stackexchange), który składa się z następujących ramek danych:

- Badges.csv.gz
- Comments.csv.gz
- PostLinks.csv.gz
- Posts.csv.gz
- Tags.csv.gz
- Users.csv.gz
- Votes.csv.gz

Przed przystąpieniem do rozwiązywania zadań zapoznaj się z ww. serwisem oraz znaczeniem poszczególnych kolumn w ww. zbiorach danych, zob. http://www.gagolewski.com/resources/data/travel_stackexchange_com/readme.txt .

Przykładowe wywołanie – ładowanie zbioru `Tags`:

[1] A więc nie: .rar, .7z itp.

```r
# dopilnujmy by w ramce nie było kolumn typu factor
if ( options()$stringsAsFactors )
    options(stringsAsFactors=FALSE) # dla R w wersji < 4.0
# ww. pliki znajdują się w katalogu travel_stackexchange_com
Tags <- read.csv("travel_stackexchange_com/Tags.csv.gz")
head(Tags)
```

Uwaga:

1. **Nazwy ramek danych po wczytaniu zbiorów powinny wyglądać następująco:** `Badges`, `Comments`, `Tags`, `Posts`, `Users`, `Votes`, `PostLinks`.
2. **W przypadku, gdy tworzą Państwo ramki danych o typie `data.table` powinny mieć one nazwy:** `BadgesDT`, `CommentsDT`, `TagsDT`, `PostsDT`, `UsersDT`, `VotesDT`, `PostLinksDT`.

2 Informacje ogólne

Rozwiąż poniższe zadania przy użyciu wywołań funkcji bazowych oraz tych, które udostępniają pakiety `dplyr` oraz `data.table` – nauczysz się ich samodzielnie; ich dokumentację łatwo odnajdziesz w internecie. Każdemu z 5 poleceń SQL powinny odpowiadać cztery równoważne sposoby ich implementacji w R, kolejno:

1. `sqldf::sqldf()` - rozwiązanie referencyjne;
2. tylko funkcje bazowe;
3. `dplyr`;
4. `data.table`.

Każde z zadań powinno być rozwiązane za pomocą funkcji:

1. `df_sql_i(df1, df2, ...)` ,
2. `df_base_i()` ,
3. `df_dplyr_i()` ,
4. `df_table_i()` ,

gdzie `i` to numer zadania, a `df1`, `df2`, ... potrzebne ramki danych (np. `df_sql_1(Users, Posts)`)

W raporcie:

1. Koniecznie upewnij się, że zwracane wyniki są ze sobą tożsame (z dokładnością do permutacji wierszy wynikowych ramek danych).
2. Ponadto w każdym przypadku porównaj czasy wykonania napisanych przez Ciebie wyrażeń przy użyciu jednego wywołania `microbenchmark::microbenchmark()`, np.:

```r
microbenchmark::microbenchmark(
   sqldf = df_sql_i(df1, df2, ...),
   base = df_base_i(df1, df2, ...),
   dplyr = df_dplyr_i(df1, df2, ...),
   data.table = df_table(df1, df2, ...)
)
```

3. Ponadto w każdym przypadku należy podać słowną („dla laika”) interpretację każdego zapytania.

Łączna ocena każdego z 5. zadań to 6 pkt (za poszczególne komponenty umieszczone w pliku `.R` oraz raporcie: rozwiązanie i sprawdzenie równoważności wyników, pomiar i ocena czasu wykonania, opis słowny zapytań) oraz 5 pkt za ogólną postać raportu (np. komentarze, wstęp, podsumowanie).

3 Zadania do rozwiązania

```sql
--- 1)
SELECT TagName, Count
FROM Tags
ORDER BY Count DESC
LIMIT 10
```

```sql
--- 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
```

```sql
--- 3)
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
```

```sql
--- 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
```

```sql
--- 5)
SELECT
    Questions.Id,
    Questions.Title,
    BestAnswers.MaxScore,
    Posts.Score AS AcceptedScore,
    BestAnswers.MaxScore-Posts.Score AS Difference
FROM (
        SELECT Id, ParentId, MAX(Score) AS MaxScore
        FROM Posts
        WHERE PostTypeId==2
        GROUP BY ParentId
    ) AS BestAnswers
JOIN (
        SELECT * FROM Posts
        WHERE PostTypeId==1
    ) AS Questions
    ON Questions.Id=BestAnswers.ParentId
JOIN Posts ON Questions.AcceptedAnswerId=Posts.Id
ORDER BY Difference DESC
LIMIT 10
```