

PDU 2018/2019

Praca domowa nr 1 (max. = 25 p.)

Maksymalna ocena: 25 p. (7 zadań po max. 3,5 p. oraz max. 0,5 p. za ogólną postać raportu)

Termin oddania pracy: 15.04.2019, godz. 23:59

Do przesłania na adres `zogala@rexamine.com` ze swojego konta pocztowego `*@*pw.edu.pl`:

- `Nick_Nazwisko_Imie_NrAlbumu_pd1.Rnw` (rozwiązania zadań – raport w LaTeX/knitr),
- `Nick_Nazwisko_Imie_NrAlbumu_pd1.pdf` (skompilowana do pdf wersja powyższego).

Uwaga: tytuł wiadomości to [PDU] Praca domowa nr 1.

Nick to wymyślony przez Ciebie identyfikator, który pojawi się w arkuszu ocen i zapewni Ci odpowiednią anonimowość. Zapamiętaj go, bo przysyłając kolejne prace domowe, będziesz używał(a) tego samego nicka.

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:

- http://www.gagolewski.com/resources/data/travel_stackexchange_com/Badges.csv.gz
- http://www.gagolewski.com/resources/data/travel_stackexchange_com/Comments.csv.gz
- http://www.gagolewski.com/resources/data/travel_stackexchange_com/PostLinks.csv.gz
- http://www.gagolewski.com/resources/data/travel_stackexchange_com/Posts.csv.gz
- http://www.gagolewski.com/resources/data/travel_stackexchange_com/Tags.csv.gz
- http://www.gagolewski.com/resources/data/travel_stackexchange_com/Users.csv.gz
- http://www.gagolewski.com/resources/data/travel_stackexchange_com/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`:

```
options(stringsAsFactors=FALSE)
# ww. pliki pobralismy do katalogu pd2/travel_stackexchange_com
Tags <- read.csv("pd2/travel_stackexchange_com/Tags.csv.gz")
head(Tags)
```

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 7 poleceń SQL powinny odpowiadać cztery równoważne sposoby ich implementacji w R, kolejno:

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

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)`)

Upewnij się, że zwracane wyniki są ze sobą tożsame (ewentualnie z dokładnością do permutacji wierszy wynikowych ramek danych). Ponadto w każdym przypadku porównaj czasy wykonania napisanych przez Ciebie wyrażeń przy użyciu jednego wywołania `microbenchmark::microbenchmark()` (0,5 p.), np.:

```
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, ...)  
)
```

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

Wszystkie rozwiązania umieścić w jednym (estetycznie sformatowanym) raporcie knitr/LaTeX. Za bogate komentarze do kodu, dyskusję i ewentualne rozwiązania alternatywne można otrzymać max. 0.5 p.

3 Zadania do rozwiązania

```
--- 1)
SELECT
  Users.DisplayName,
  Users.Age,
  Users.Location,
  SUM(Posts.FavoriteCount) AS FavoriteTotal,
  Posts.Title AS MostFavoriteQuestion,
  MAX(Posts.FavoriteCount) AS MostFavoriteQuestionLikes
FROM Posts
JOIN Users ON Users.Id=Posts.OwnerUserId
WHERE Posts.PostTypeId=1
GROUP BY OwnerUserId
ORDER BY FavoriteTotal DESC
LIMIT 10
```

```
--- 2)
SELECT
  Posts.ID,
  Posts.Title,
  Posts2.PositiveAnswerCount
FROM Posts
JOIN (
  SELECT
    Posts.ParentID,
    COUNT(*) AS PositiveAnswerCount
  FROM Posts
  WHERE Posts.PostTypeID=2 AND Posts.Score>0
  GROUP BY Posts.ParentID
```

```
    ) AS Posts2
    ON Posts.ID=Posts2.ParentID
ORDER BY Posts2.PositiveAnswerCount DESC
LIMIT 10
```

```

--- 3)
SELECT
    Posts.Title,
    UpVotesPerYear.Year,
    MAX(UpVotesPerYear.Count) AS Count
FROM (
    SELECT
        PostId,
        COUNT(*) AS Count,
        STRFTIME('%Y', Votes.CreationDate) AS Year
    FROM Votes
    WHERE VoteTypeId=2
    GROUP BY PostId, Year
) AS UpVotesPerYear
JOIN Posts ON Posts.Id=UpVotesPerYear.PostId
WHERE Posts.PostTypeId=1
GROUP BY Year

```

```

--- 4)
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
WHERE Difference>50
ORDER BY Difference DESC

```

```

--- 5)
SELECT
    Posts.Title,
    CmtTotScr.CommentsTotalScore
FROM (
    SELECT
        PostID,
        UserID,
        SUM(Score) AS CommentsTotalScore
    FROM Comments
    GROUP BY PostID, UserID
) AS CmtTotScr
JOIN Posts ON Posts.ID=CmtTotScr.PostID AND Posts.OwnerUserId=CmtTotScr.UserID
WHERE Posts.PostTypeId=1
ORDER BY CmtTotScr.CommentsTotalScore DESC
LIMIT 10

```

```

--- 6)
SELECT DISTINCT
    Users.Id,
    Users.DisplayName,
    Users.Reputation,
    Users.Age,
    Users.Location
FROM (
    SELECT
        Name, UserID
    FROM Badges
    WHERE Name IN (
        SELECT
            Name
        FROM Badges
        WHERE Class=1
        GROUP BY Name
        HAVING COUNT(*) BETWEEN 2 AND 10
    )
    AND Class=1
) AS ValuableBadges
JOIN Users ON ValuableBadges.UserId=Users.Id

```

```

--- 7)
SELECT
    Posts.Title,
    VotesByAge2.OldVotes
FROM Posts
JOIN (
    SELECT
        PostId,
        MAX(CASE WHEN VoteDate = 'new' THEN Total ELSE 0 END) NewVotes,
        MAX(CASE WHEN VoteDate = 'old' THEN Total ELSE 0 END) OldVotes,
        SUM(Total) AS Votes
    FROM (
        SELECT
            PostId,
            CASE STRFTIME('%Y', CreationDate)
                WHEN '2017' THEN 'new'
                WHEN '2016' THEN 'new'
                ELSE 'old'
            END VoteDate,
            COUNT(*) AS Total
        FROM Votes
        WHERE VoteTypeId=2
        GROUP BY PostId, VoteDate
    ) AS VotesByAge
    GROUP BY VotesByAge.PostId
    HAVING NewVotes=0
) AS VotesByAge2 ON VotesByAge2.PostId=Posts.ID
WHERE Posts.PostTypeId=1
ORDER BY VotesByAge2.OldVotes DESC
LIMIT 10

```