Pakiet DPLYR

pakiet R służący do przekształcania i podsumowywania danych tabelarycznych.

Tutoriale i dokumentacja:

http://genomicsclass.github.io/book/pages/dplyr_tutorial.html
https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html
https://www.rdocumentation.org/packages/dplyr/versions/0.5.0

filter   - wybieranie obserwacji na podstawie wartości.
arrange - zmiana kolejności
select   - wybieranie zmiennych na podstawie nazw
mutate   - tworzenie nowych zmiennych przekształcając stare.
summarize – podsumowanie wielu wartości (zwijanie).
group_by – zmienia zakres działania funkcji tak aby zamiast na całym zbiorze działały na grupach.

Zainstalujmy i dodajmy pakiet DPLYR: (użyjemy tidyverse)

```
install.packages("tidyverse")
```

Do testów użyjemy danych z pakietu lotów z nowego jorku w 2013 roku:

```
library(nycflights13)
```

Zadanie 0:

a)Połączyć się z serwerem MySQL,
b)Przetestować polecenie select dla tabeli flights.
c) Pobrać tabelę z serwera bazy danych mysql jako tibble.
d)skopiować tabelę z R do bazy SQLITE

Wybierzmy przykładowo dane z drugiego grudnia:

```
drugiGrudnia <- filter(flights,month=="12",day==2)
> drugiGrudnia
# A tibble: 1,004 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013    12     2       12           2250     82.0      103           2356     67.0  B6        1816 N316JB
 2  2013    12     2      452            500    - 8.00     626            651    -25.0  US        1895 N554UW
 3  2013    12     2      515            515      0        742            808    -26.0  UA        1452 N69806
 4  2013    12     2      538            540    - 2.00     833            850    -17.0  AA        2243 N5EBAA
 5  2013    12     2      544            550    - 6.00    1006           1027    -21.0  B6         939 N529JB
 6  2013    12     2      552            600    - 8.00     642            658    -16.0  US        1909 N948UW
 7  2013    12     2      553            600    - 7.00     651            659    - 8.00 US        2167 N749US
 8  2013    12     2      554            600    - 6.00     719            717      2.00 EV        5716 N829AS
 9  2013    12     2      554            545      9.00     825            835    -10.0  UA        1500 N37263
10  2013    12     2      555            600    - 5.00     838            849    -11.0  B6         353 N775JB
# ... with 994 more rows, and 7 more variables: origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
```

Możemy stworzyć zmienna factor z nazwami miesięcy :

```
flights$month <- factor(flights$month,
levels=c(1,2,3,4,5,6,7,8,9,10,11,12),labels=c("January","February","March","April","May","June","July","August","September","October","November","December"))
```

Zobaczmy przefiltrowane dane (te z 2 grudnia):

```
> drugiGrudnia <- filter(flights,month=="December",day==2)
> drugiGrudnia
# A tibble: 1,004 x 19
   year month      day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
  <int> <fct>    <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013 December     2       12           2250      82.0      103           2356      67.0  B6        1816 N316JB
 2  2013 December     2      452            500     - 8.00     626            651     -25.0  US        1895 N554UW
 3  2013 December     2      515            515       0        742            808     -26.0  UA        1452 N69806
 4  2013 December     2      538            540     - 2.00     833            850     -17.0  AA        2243 N5EBAA
 5  2013 December     2      544            550     - 6.00    1006           1027     -21.0  B6         939 N529JB
 6  2013 December     2      552            600     - 8.00     642            658     -16.0  US        1909 N948UW
 7  2013 December     2      553            600     - 7.00     651            659     - 8.00 US        2167 N749US
 8  2013 December     2      554            600     - 6.00     719            717       2.00 EV        5716 N829AS
 9  2013 December     2      554            545       9.00     825            835     -10.0  UA        1500 N37263
10  2013 December     2      555            600     - 5.00     838            849     -11.0  B6         353 N775JB
# ... with 994 more rows, and 7 more variables: origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>,
#   minute <dbl>, time_hour <dttm>
>
```

Filtrując korzystamy z operatorów <.> , <=, >=, !=, ==
Można też korzystać z operatorów logicznych i: &, lub: | i nie: !

Jak można wyświetlić dane lotów z okresu wakacyjnego (lipiec-wrzesień)?

To samo można osiągnąć wykorzystując %in%:

```
daneZwakacji <- filter(flights,flights$month %in% c("July","August","September"))
```

Zadania1:

Ćwiczenia1
1.Znajdź wszystkie loty, które:
a)Były opóźnione podczas przylotu co najmniej o dwie godziny.
b)Leciały do Houston (IAH lub HOU).
c)Były obsługiwane przez linie United, American lub Delta.
d)Odlatywały latem (w lipcu, sierpniu i wrześniu)
e)Przyleciały z ponad dwugodzinnym opóźnieniem, ale nie odleciały opóźnione.
f)Były opóźnione o co najmniej godzinę, ale zrekompensowały opóźnienie o ponad 30 minut
podczas lotu.
g)Wyruszyły między północą a 6 rano (włącznie).

Zamiast wybierać wiersze możemy też ustalić ich kolejność za pomocą filter():

```
 arrange(flights,month,day)
# A tibble: 336,776 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1     1      517            515      2.00      830            819      11.0  UA        1545 N14228
 2  2013     1     1      533            529      4.00      850            830      20.0  UA        1714 N24211
 3  2013     1     1      542            540      2.00      923            850      33.0  AA        1141 N619AA
```

```
 4  2013     1     1      544           545     -1.00     1004          1022    -18.0  B6        725 N804JB
 5  2013     1     1      554           600     -6.00      812           837    -25.0  DL        461 N668DN
 6  2013     1     1      554           558     -4.00      740           728     12.0  UA       1696 N39463
 7  2013     1     1      555           600     -5.00      913           854     19.0  B6        507 N516JB
 8  2013     1     1      557           600     -3.00      709           723    -14.0  EV       5708 N829AS
 9  2013     1     1      557           600     -3.00      838           846    - 8.00 B6         79 N593JB
10  2013     1     1      558           600     -2.00      753           745     8.00 AA        301 N3ALAA
# ... with 336,766 more rows, and 7 more variables: origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Możemy też sortować malejąco:

```
arrange(flights,month,desc(day))
# A tibble: 336,776 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1  2013     1    31        1           2100       181      124           2225       179    WN        530 N550WN
 2  2013     1    31        4           2359         5.00     455            444        11.0  B6        739 N599JB
 3  2013     1    31        7           2359         8.00     453            437        16.0  B6        727 N505JB
 4  2013     1    31       12           2250        82.0      132              7        85.0  B6         30 N178JB
 5  2013     1    31       26           2154       152        328             50       158    B6        515 N663JB
 6  2013     1    31       34           2159       155        135           2315       140    EV       4162 N24128
 7  2013     1    31       37           2249       108        132           2357        95.0  B6         22 N239JB
 8  2013     1    31       54           2250       124        152           2359       113    B6        608 N281JB
 9  2013     1    31      453            500     -  7.00      651            648         3.00 US       1117 N702UW
10  2013     1    31      522            525     -  3.00      820            820         0    UA       1018 N36207
# ... with 336,766 more rows, and 7 more variables: origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

Ćwiczenia2

1.Jak za pomocą funkcji arrange() posortować wszystkie brakujące wartości, tak aby znalazły się na początku? (Wskazówka: użyj funkcji is.na()).
2.Posortuj dane flights, aby znaleźć najbardziej opóźnione loty. Znajdź te, które odleciały najwcześniej.
3.Posortuj dane flights, aby znaleźć najszybsze loty.
4.Które loty trwały najdłużej? Które najkrócej?

Funkcja select() może zostać wykorzystana do uzyskania podzbioru na podstawie operacji na nazwach zmiennych np.

```
select(flights,month,day,dep_time,dep_delay,arr_time,arr_delay,tailnum)
```

Możemy wybrać wartości od-do danej kolumny włącznie:

```
select(flights, month:tailnum)
# A tibble: 336,776 x 11
   month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum
  <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>
 1     1     1      517            515      2.00      830            819        11.0  UA       1545 N14228
 2     1     1      533            529      4.00      850            830        20.0  UA       1714 N24211
 3     1     1      542            540      2.00      923            850        33.0  AA       1141 N619AA
 4     1     1      544            545     -1.00     1004           1022       -18.0  B6        725 N804JB
 5     1     1      554            600     -6.00      812            837       -25.0  DL        461 N668DN
 6     1     1      554            558     -4.00      740            728        12.0  UA       1696 N39463
 7     1     1      555            600     -5.00      913            854        19.0  B6        507 N516JB
 8     1     1      557            600     -3.00      709            723       -14.0  EV       5708 N829AS
 9     1     1      557            600     -3.00      838            846       - 8.00 B6         79 N593JB
10     1     1      558            600     -2.00      753            745        8.00 AA        301 N3ALAA
# ... with 336,766 more rows
```

```
>
```

Albo z wyłączeniem:

```
select(flights, -(year:day))
# A tibble: 336,776 x 16
   dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier flight tailnum origin dest  air_time
      <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>    <int> <chr>   <chr>  <chr>    <dbl>
 1      517            515      2.00      830            819     11.0  UA        1545 N14228  EWR    IAH      227
 2      533            529      4.00      850            830     20.0  UA        1714 N24211  LGA    IAH      227
 3      542            540      2.00      923            850     33.0  AA        1141 N619AA  JFK    MIA      160
 4      544            545     -1.00     1004           1022    -18.0  B6         725 N804JB  JFK    BQN      183
 5      554            600     -6.00      812            837    -25.0  DL         461 N668DN  LGA    ATL      116
 6      554            558     -4.00      740            728     12.0  UA        1696 N39463  EWR    ORD      150
 7      555            600     -5.00      913            854     19.0  B6         507 N516JB  EWR    FLL      158
 8      557            600     -3.00      709            723    -14.0  EV        5708 N829AS  LGA    IAD       53.0
 9      557            600     -3.00      838            846    - 8.00 B6          79 N593JB  JFK    MCO      140
10      558            600     -2.00      753            745     8.00  AA         301 N3ALAA  LGA    ORD      138
# ... with 336,766 more rows, and 4 more variables: distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
>
```

Funkcje pomocnicze funkcji select():
starts_with("abc") wybiera nazwy rozpoczynające się od „abc”.
ends_with("xyz") wybiera nazwy kończące się na „xyz”.
contains("abc") wybiera nazwy zawierające „abc”.
matches("”) wybiera zmienne pasujące do wyrażenia regularnego
num_range("x", 1:3) pasuje do x1, x2 i x3.

Możemy też użyć select() z funkcją pomocniczą everything():

```
select(flights, dep_time,arr_time,everything())
```

Ćwiczenia3
1.Wybierz wartości zmiennych dep_time, dep_delay, arr_time i
arr_delay ze zbioru danych flights.
2.Co się stanie, gdy w wywołaniu select() kilkakrotnie wpiszesz nazwę tej samej zmiennej?
3.Do czego służy funkcja one_of()? Dlaczego może się okazać przydatna razem z tym wektorem?
vars <- c(
"year", "month", "day", "dep_delay", "arr_delay")

Funkcja mutate() pozwala nam na dodawanie nowych zwmiennych (kolumn) do naszej ramki danych.
Nowe kolumny dodawane są na koniec.

Zobaczmy przykład dla tabeli z mniejszą ilością kolumn:

```
> flights_small <- select(flights,ends_with("delay"),distance,air_time)
> flights_small
# A tibble: 336,776 x 4
   dep_delay arr_delay distance air_time
       <dbl>     <dbl>    <dbl>    <dbl>
 1      2.00      11.0     1400      227
 2      4.00      20.0     1416      227
 3      2.00      33.0     1089      160
 4     -1.00     -18.0     1576      183
 5     -6.00     -25.0      762      116
```

```
 6    -4.00    12.0      719   150
 7    -5.00    19.0     1065   158
 8    -3.00   -14.0      229    53.0
 9    -3.00   - 8.00     944   140
10    -2.00     8.00     733   138
# ... with 336,766 more rows

mutate(flights_small,gain = arr_delay - dep_delay,speed=distance/air_time*60)
# A tibble: 336,776 x 6
   dep_delay arr_delay distance air_time   gain speed
       <dbl>     <dbl>    <dbl>    <dbl>  <dbl> <dbl>
 1      2.00     11.0     1400    227     9.00   370
 2      4.00     20.0     1416    227    16.0    374
 3      2.00     33.0     1089    160    31.0    408
 4     -1.00    -18.0     1576    183   -17.0    517
 5     -6.00    -25.0      762    116   -19.0    394
 6     -4.00     12.0      719    150    16.0    288
 7     -5.00     19.0     1065    158    24.0    404
 8     -3.00    -14.0      229     53.0 -11.0    259
 9     -3.00    - 8.00     944    140   - 5.00   405
10     -2.00     8.00      733    138    10.0    319
# ... with 336,766 more rows
```

Operatory arytmetyczne +, -, *, /, ^
Arytmetyka modulo(%/% i %%)
%/% (dzielenie całkowite) i %% (reszta), gdzie x == y * (x %/% y) + (x %% y).

Ćwiczenia4
1.na podstawie zmiennej dep_time oblicz hour i minute.
2.Przekształć dep_time i sched_dep_time, wyznaczając
liczbę minut, jaka upłynęła od północy.

Funkcja summarize() zawija dane do jednego wiersza.

```
> speedData <- transmute(flights_small,gain = arr_delay - dep_delay,speed=distance/air_time*60)
> summarize(speedData,speadmean = mean(speed,na.rm=TRUE))
# A tibble: 1 x 1
  speadmean
      <dbl>
1       394
>
```

Możemy robić podsumowanie dla grupy zamiast dla całego zestawu danych, korzystając z group_by().
Pogrupujmy nasze dane ze względu na miesiąc i obliczmy średnie opóźnienie lotu w każdym miesiącu:

```
by_month <- group_by(flights,month)
> summarize(by_month, meanMonthlyDelay=mean(dep_delay,na.rm = TRUE))
# A tibble: 12 x 2
   month meanMonthlyDelay
   <int>           <dbl>
 1     1            10.0
 2     2            10.8
 3     3            13.2
 4     4            13.9
 5     5            13.0
```

```
 6    6         20.8
 7    7         21.7
 8    8         12.6
 9    9          6.72
10   10          6.24
11   11          5.44
```

Potok zamiast przypiswanie zmiennej:

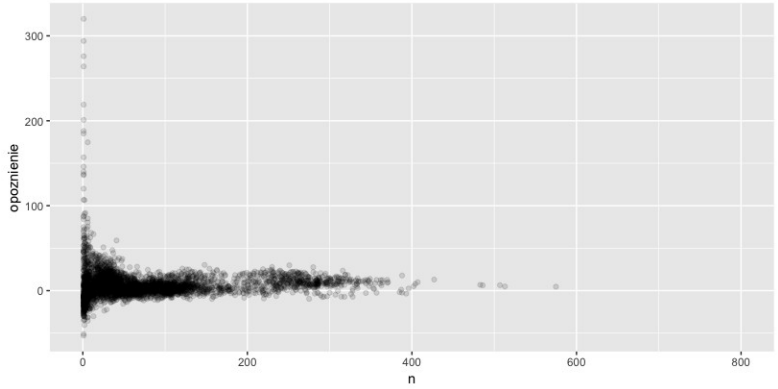
Podsumowanie liczby lotów,odległości i opóźnienia (średnie)

```
> by_destination <- group_by(flights,dest)
> View(by_destination)
> delay <-
summarize(by_destination,count=n(),dist=mean(distance,na.rm=TRUE),delay=mean(arr_delay,na.rm=TRUE))
> View(delay)
> delays<- flights %>% group_by(dest)%>%summarize(count =
n(),dist=mean(distance,na.rm=TRUE),delay=mean(arr_delay,na.rm=TRUE))
> View(delays)
```

```
> opoznione <- myFlightsTibble %>% group_by(tailnum) %>% summarize(opoznienie =
mean(arr_delay,na.rm=TRUE),n=n())
> opoznione
# A tibble: 4,044 x 3
   tailnum opoznienie     n
   <chr>        <dbl> <int>
 1 D942DN      31.5       4
 2 N0EGMQ       9.98    371
 3 N10156      12.7     153
 4 N102UW       2.94     48
 5 N103US      -6.93     46
 6 N104UW       1.80     47
 7 N10575      20.7     289
 8 N105UW      -0.267    45
 9 N107US      -5.73     41
10 N108UW      -1.25     60
```

Przy małej liczbie lotów opóźnienia bardzo się różnią, co możemy zobaczyć na wykresie:

```
ggplot(data=opoznione,mapping=aes(x=n,y=opoznienie))+geom_point(alpha=1/8)+xlim(0,800)
```



Średnia jest sumą elementów podzieloną przez ich liczbę; mediana jest wartością, powyżej i poniżej której znajduje się 50% elementów próbki
Błąd średniokwadratowy lub odchylenie standardowe, czyli w skrócie sd (ang. standard deviation), jest standardową miarą rozkładu.

```
myFlightsTibble %>% group_by(dest) %>%summarise(distance_sd=sd(distance,na.rm=TRUE))%>
%arrange(desc(distance_sd))
# A tibble: 105 x 2
   dest  distance_sd
   <chr>       <dbl>
 1 EGE         10.5
 2 SAN         10.3
 3 SFO         10.2
 4 HNL         10.0
 5 SEA          9.98
 6 LAS          9.91
 7 PDX          9.88
 8 PHX          9.86
 9 LAX          9.66
10 IND          9.46
# … with 95 more rows
```

Kwantyle są generalizacją mediany. Przykładowo instrukcja quantile(x, 0.25) znajdzie wartość zmiennej x, która jest większa niż 25% wartości i mniejsza niż pozostałe 75% wartości

min/max:

Kiedy odlatują pierwsze i ostatnie loty każdego dnia?

```
 myFlightsTibble %>% group_by(year,month,day) %>% summarize(first =
min(dep_time,na.rm=TRUE),last=max(dep_time,na.rm=TRUE))
# A tibble: 365 x 5
# Groups:   year, month [?]
    year month   day first  last
   <int> <int> <int> <dbl> <dbl>
 1  2013     1     1   517  2356
 2  2013     1     2    42  2354
 3  2013     1     3    32  2349
 4  2013     1     4    25  2358
 5  2013     1     5    14  2357
 6  2013     1     6    16  2355
 7  2013     1     7    49  2359
 8  2013     1     8   454  2351
 9  2013     1     9     2  2252
10  2013     1    10     3  2320
# … with 355 more rows
```

Korzystaliśmy już z funkcji n(), która nie przyjmuje argumentów i zwraca rozmiar bieżącej grupy. Aby obliczyć liczbę niebrakujących wartości, korzystamy z instrukcji sum(!is.na(x)). Aby sprawdzić liczbę unikatowych wartości, możemy skorzystać z funkcji n_distinct(x).

```
myFlightsTibble %>% group_by(dest) %>%summarize(carriers = n_distinct(carrier,na.rm=TRUE))%>
%arrange(desc(carriers))
# A tibble: 105 x 2
   dest  carriers
   <chr>    <int>
 1 ATL          7
 2 BOS          7
 3 CLT          7
 4 ORD          7
 5 TPA          7
 6 AUS          6
 7 DCA          6
 8 DTW          6
 9 IAD          6
10 MSP          6
```

suma mil, którą przeleciał każdy samolot:

count(dest)

```
> flights %>%count(tailnum,wt = distance )
# A tibble: 4,044 x 2
   tailnum      n
   <chr>    <dbl>
 1 D942DN    3418
 2 N0EGMQ  250866
 3 N10156  115966
 4 N102UW   25722
 5 N103US   24619
 6 N104UW   25157
 7 N10575  150194
 8 N105UW   23618
 9 N107US   21677
10 N108UW   32070
# … with 4,034 more rows
```

Liczebności i proporcje wartości logicznych:
np.: sum(x > 1), mean(x == 1)
Ile samolotów wyleciało przed 5 rano?
Jaka jest proporcja lotów opóźnionych o ponad jedną godzinę?

Ćwiczenia5
1.Rozważ następujące scenariusze i znajdz loty które:
Lot jest o 15 minut za wcześnie przez 50% czasu, i o 15 minut za późno przez 50% czasu.
Lot jest zawsze opóźniony o 10 minut.
Lot jest o 30 minut za wcześnie przez 50% czasu i jest opóźniony o 30 minut przez 50% czasu.
Przez 99% czasu lot jest zgodny z harmonogramem. Przez 1% czasu jest opóźniony o 2 godziny.
Co jest ważniejsze: opóźnienie przylotu czy odlotu?

Dla ramki bez anulowanych lotów:

```
not_cancelled <- flights %>%
filter(!is.na(dep_delay), !is.na(arr_delay))
```

2.Dla każdego samolotu oblicz liczbę lotów przed pierwszym opóźnieniem większym niż jedna godzina.
3.Do czego służy argument sort funkcji count()? Kiedy można z niego skorzystać?
4.Ile było lotów każdego dnia?
5.Ile było lotów każdego miesiąca?