HW1

Morozov A.

19.10.2024

ЧАСТЬ 1

Загружаем данные в датафрейм для обработки.

```
data.df <- read.table("https://people.math.umass.edu/~anna/Stat597AFall2016/rnf6080.dat");

head(data.df)
```

```
> data.df <- read.table("https://people.math.umass.edu/~anna/Stat597AFall2016/rnf6080.dat");
> head(data.df)
  V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25
1 60  4  1  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
2 60  4  2  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
3 60  4  3  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
4 60  4  4  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
5 60  4  5  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
6 60  4  6  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
  V26 V27
1   0   0
2   0   0
3   0   0
4   0   0
5   0   0
```

Определим количество строк и столбцов таблицы.

```
cat("Количество столбцов -", ncol(data.df), "\n")

cat("Количество строк -", nrow(data.df), "\n")
```

```
> cat("Количество столбцов -", ncol(data.df), "\n")
Количество столбцов - 27
> cat("Количество строк -", nrow(data.df), "\n")
Количество строк - 5070
```

Выведем названия столбцов.

```
cat(colnames(data.df))
```

```
> cat(colnames(data.df))
V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27
```

Выведем значение в 7 строке в 16 столбце.

```
data.df[7, 16]
```

```
> data.df[7, 16]
[1] 0
```

Выведем 126 строку.

```
data.df[126, ]
```

```
> data.df[126, ]
    V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25
126 60  8  4  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    V26 V27
126   0   0
```

Заменим имена столбцов на более читаемые.

```
names(data.df) <- c("year", "month", "day", seq(0, 23))

head(data.df)
```

```
> names(data.df) <- c("year", "month", "day", seq(0, 23))
> head(data.df)
  year month day 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
1   60     4   1 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
2   60     4   2 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
3   60     4   3 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
4   60     4   4 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
5   60     4   5 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
6   60     4   6 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

```
tail(data.df)
```

```
> tail(data.df)
     year month day 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
5065   80    11  25 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
5066   80    11  26 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
5067   80    11  27 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
5068   80    11  28 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
5069   80    11  29 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
5070   80    11  30 0 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Добавим столбец столбец с суммой последних 24 столбцов.

```
data.df$daily <- rowSums(data.df[4:27])

data.df[20, ]
```

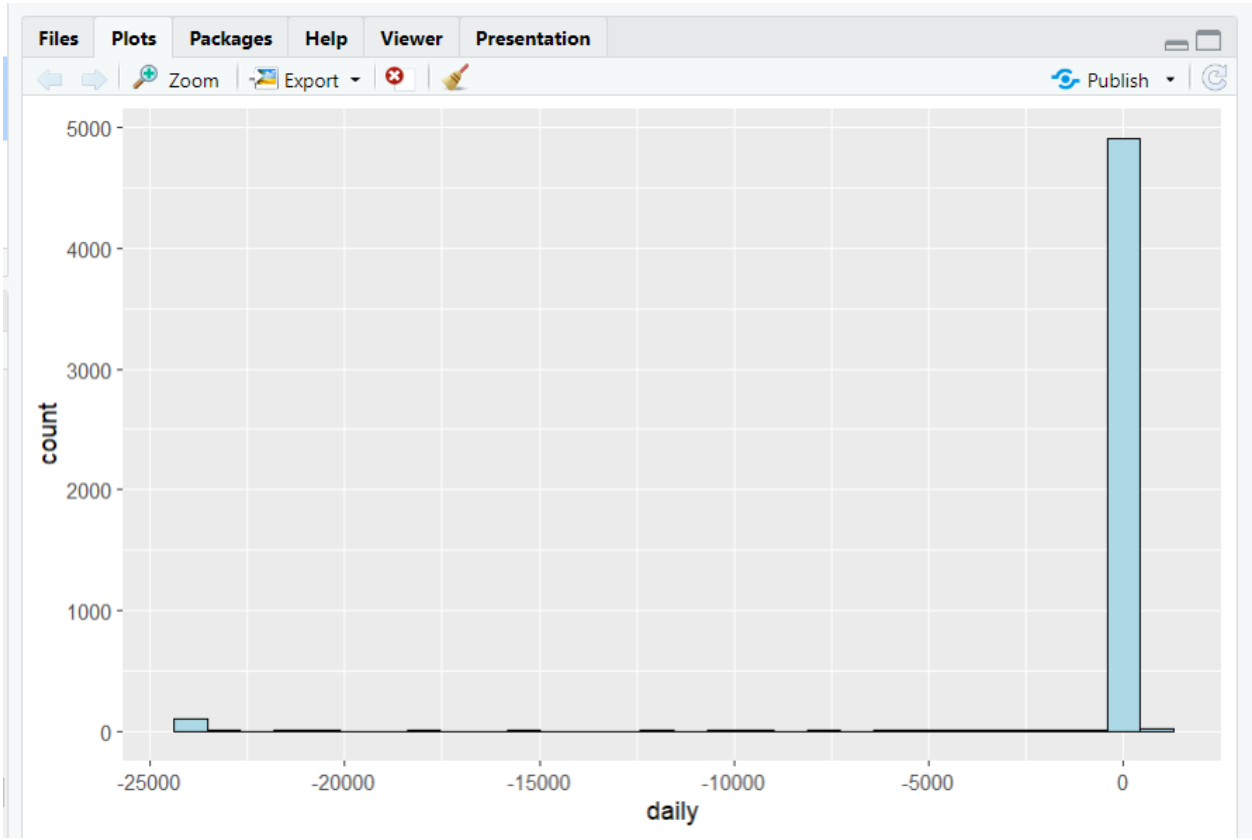
```
> data.df$daily <- rowSums(data.df[4 : 27])
> data.df[20, ]
   year month day 0 1 2 3 4 5  6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 daily
20   60     4  20 0 0 0 0 8 0 13 8 3 0  0  0  0  0  0  0  0  0  0  0  0  0  0    32
```

Построим гистограмму по добавленному столбцу.

```
library("ggplot2")

ggplot(data=data.df, aes(x=daily)) + geom_histogram(color="black", fill="lightblue")
```

```
> library("ggplot2")
> ggplot(data=data.df, aes(x=daily)) + geom_histogram(color="black", fill="lightblue")
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Посмотрим данные меньше нуля.

```
head(subset(data.df, daily < 0), n=10)
```

```
> head(subset(data.df, daily < 0), n=10)
     year month day    0    1    2    3    4    5    6    7    8    9   10   11   12   13   14
495    62     5   7    0    0    0    0    0    0    0    0    0    0 -999 -999 -999 -999    5
675    62    11   3    0    0    0    0    0    0    0    0    0    0 -999 -999   18   10   18
678    62    11   6    0    0    0    3   10   15   23    8   10    0    0    0    0    0 -999
718    63     4  16    0    0    5    0   18    0   13    3   13   18    5    8    3   10   25
1870   68     5   8    3    0    3   13    3    5   20   20   38   43   20   18   15 -999 -999
2119   69     6  12    0    0    0    0    0    0    0 -999    5    0    0    0    0    0    0
2244   69    10  15 -999    0    0    0    0    0    0    0    0    0    0    0    0    0 -999
2259   69    10  30 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999
2261   70     4   1    0    0    0    0    0    0    0    0    0 -999 -999 -999 -999 -999 -999
2262   70     4   2 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999 -999
       15    16  17   18   19   20   21   22   23  daily
495  -999  -999   0    3    0    0    0    0    0  -5986
675     8     0   0    0    0    0    0    0    0  -1944
678  -999     0   0    0    0    0    0    0    0  -1929
718    18    23  25   10   13   23   15 -999    0   -751
```

```
summary(data.df[4:28])
```

```
> summary(data.df[4:28])
       0                  1                  2                  3                  4
 Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-999.00
 1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00
 Median :   0.00   Median :   0.00   Median :   0.00   Median :   0.00   Median :   0.00
 Mean   : -20.88   Mean   : -21.16   Mean   : -21.19   Mean   : -20.83   Mean   : -20.86
 3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00
 Max.   : 325.00   Max.   : 323.00   Max.   : 239.00   Max.   : 119.00   Max.   : 198.00
       5                  6                  7                  8                  9
 Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-999.0   Min.   :-999.00
 1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.0   1st Qu.:   0.00
 Median :   0.00   Median :   0.00   Median :   0.00   Median :   0.0   Median :   0.00
 Mean   : -21.08   Mean   : -21.67   Mean   : -21.19   Mean   : -20.9   Mean   : -20.73
 3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.0   3rd Qu.:   0.00
 Max.   : 348.00   Max.   : 173.00   Max.   : 394.00   Max.   : 320.0   Max.   : 259.00
      10                 11                 12                 13                 14
 Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-999.0
 1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.0
 Median :   0.00   Median :   0.00   Median :   0.00   Median :   0.00   Median :   0.0
 Mean   : -21.18   Mean   : -20.97   Mean   : -19.89   Mean   : -20.56   Mean   : -21.4
 3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.0
 Max.   : 109.00   Max.   : 130.00   Max.   : 203.00   Max.   : 218.00   Max.   : 297.0
      15                 16                 17                 18                 19
 Min.   :-999.0   Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-999.00
 1st Qu.:   0.0   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00
 Median :   0.0   Median :   0.00   Median :   0.00   Median :   0.00   Median :   0.00
 Mean   : -20.9   Mean   : -20.83   Mean   : -19.97   Mean   : -20.18   Mean   : -19.81
 3rd Qu.:   0.0   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00
 Max.   : 183.0   Max.   : 183.00   Max.   : 345.00   Max.   : 393.00   Max.   : 208.00
      20                 21                 22                 23              daily
 Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-999.00   Min.   :-23976.0
 1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:   0.00   1st Qu.:     0.0
 Median :   0.00   Median :   0.00   Median :   0.00   Median :   0.00   Median :     0.0
 Mean   : -19.64   Mean   : -19.74   Mean   : -19.54   Mean   : -19.32   Mean   :  -494.4
 3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:   0.00   3rd Qu.:     2.0
 Max.   : 155.00   Max.   : 178.00   Max.   : 381.00   Max.   : 279.00   Max.   :   840.0
```

Сохраним оригинальную версию данных.

```
fixed.df <- data.df[1:27]
```

Заменим *-999* на *0*.

```
fixed.df[fixed.df == -999] <- 0
```

Пересчитаем новую колонку.

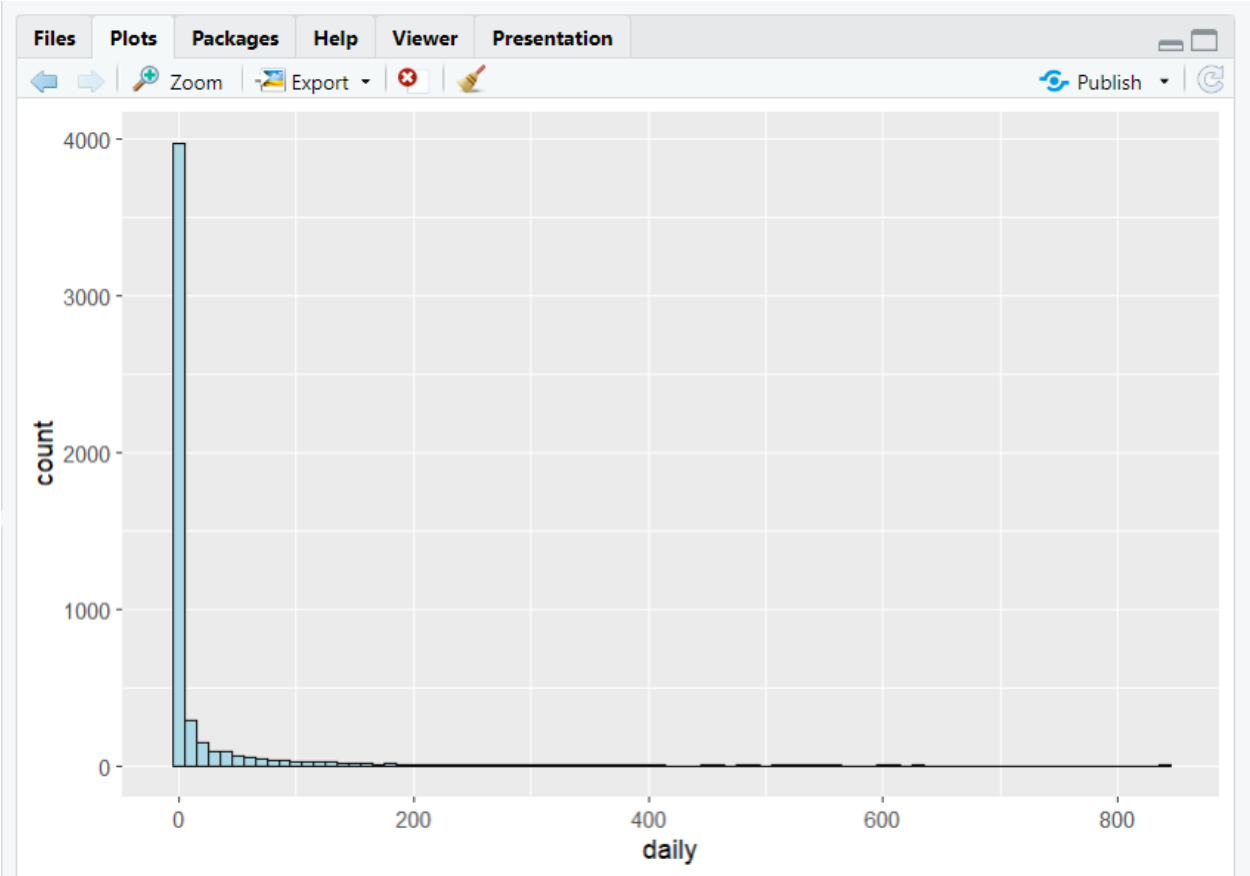
```
fixed.df$daily <- rowSums(fixed.df[4:27])
```

Построим новую гистограмму.

```
library("ggplot2")

ggplot(data=fixed.df, aes(x=daily)) + geom_histogram(binwidth=10, color="black",
fill="lightblue")
```

```
 Max.   : 155.00   Max.   : 178.00   Max.   : 381.00   Max.   : 279.00   Max.   :   840.0
> fixed.df <- data.df[1:27]
> fixed.df[fixed.df == -999] <- 0
> fixed.df$daily <- rowSums(fixed.df[4:27])
> library("ggplot2")
> ggplot(data=fixed.df, aes(x=daily)) + geom_histogram(binwidth=10, color="black", fill="lightblue")
```



ЧАСТЬ 2

Создадим вектор со строками, содержащими числа.

```
v <- c("4", "8", "15", "16", "23", "42")
```

Выведем наибольшее. Выведется «8», так как среди строк максимальное определяется по ASCII-коду символа строки и 8 последняя из присутствующих.

```
max(v)
```

```
> v <- c("4", "8", "15", "16", "23", "42")
> max(v)
[1] "8"
```

При попытке сортировки увидим соответствующий ранее сказанному результат.

```
sort(v)
```

```
> sort(v)
[1] "15" "16" "23" "4"  "42" "8"
```

При попытке вычисления суммы получим ошибку, так как вектор содержит строки, а они не подлежат использованию с оператором сложения.

```
sum(v)
```

```
> sum(v)
Ошибка в sum(v) : неправильный 'type' (character) аргумента
```

Создадим ещё вектор.

```
v2 <- c("5", 7, 12)
```

Попытаемся сложить второй элемент второго вектора и третий элемент первого вектора и получим ошибку по вышеназванной причине.

```
v2[2] + v[3]
```

```
> v2 <- c("5", 7, 12)
> v2[2] + v[3]
Ошибка в v2[2] + v[3] : нечисловой аргумент для бинарного оператора
```

Создадим датафрейм и сложим его два элемента.

```
df3 <- data.frame(z1 = "5", z2 = 7, z3 = 12)
```

```
df3[1, 2] + df3[1, 3]
```

```
> df3 <- data.frame(z1 = "5", z2 = 7, z3 = 12)
> df3[1, 2] + df3[1, 3]
[1] 19
```

Создадим список и попробуем сложить его два элемента.

```
l4 <- list(z1 = "6", z2 = 42, z3 = "49", z4 = 126)
```

Для этого нужно использовать дважды вложенные скобки.

```
l4[[2]] + l4[[4]]
```

```
l4[2] + l4[4]
```

```
> l4 <- list(z1 = "6", z2 = 42, z3 = "49", z4 = 126)
> l4[[2]] + l4[[4]]
[1] 168
> l4[2] + l4[4]
Ошибка в l4[2] + l4[4] : нечисловой аргумент для бинарного оператора
```

Так как по типу данных числовой тип мы можем извлечь только из дважды вложенного.

```
is.numeric(l4[2])
```

```
is.numeric(l4[[2]])
```

```
> is.numeric(l4[2])
[1] FALSE
> is.numeric(l4[[2]])
[1] TRUE
```

ЧАСТЬ 3

Выведем последовательность от 1 до 10000 с шагом 372.

```
seq(1, 10000, 372)
```

```
> seq(1, 10000, 372)
 [1]    1  373  745 1117 1489 1861 2233 2605 2977 3349 3721 4093 4465 4837 5209 5581 5953 6325 6697
[20] 7069 7441 7813 8185 8557 8929 9301 9673
```

Выведем 50 равноудалённых чисел от 1 до 10000.

```
seq(1, 10000, length.out = 50)
```

```
[20] 7069 7441 7813 8185 8557 8929 9301 9673
> seq(1, 10000, length.out = 50)
 [1]     1.0000   205.0612   409.1224   613.1837   817.2449  1021.3061  1225.3673  1429.4286
 [9]  1633.4898  1837.5510  2041.6122  2245.6735  2449.7347  2653.7959  2857.8571  3061.9184
[17]  3265.9796  3470.0408  3674.1020  3878.1633  4082.2245  4286.2857  4490.3469  4694.4082
[25]  4898.4694  5102.5306  5306.5918  5510.6531  5714.7143  5918.7755  6122.8367  6326.8980
[33]  6530.9592  6735.0204  6939.0816  7143.1429  7347.2041  7551.2653  7755.3265  7959.3878
[41]  8163.4490  8367.5102  8571.5714  8775.6327  8979.6939  9183.7551  9387.8163  9591.8776
[49]  9795.9388 10000.0000
>
```

Выведем повторение чисел от 1 до 5 3 раза.

```
rep(1:5, times=3)
```

Выведем числа от 1 до 5, повторяя каждое три раза.

```
rep(1:5, each=3)
```

```
> rep(1:5, times=3)
 [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
> rep(1:5, each=3)
 [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
```