

hw01

Kameneva Polina

2024-10-20

Работа с данными

По адресу <http://people.math.umass.edu/~anna/Stat597AFall2016/rnf6080.dat> можно получить набор данных об осадках в Канаде с 1960 по 1980 годы. Необходимо загрузить эти данные при помощи `read.table`. Воспользуйтесь справкой, чтобы изучить аргументы, которые принимает функция.

1. Загрузите данные в датафрейм, который назовите `data.df`.

```
url <- "http://people.math.umass.edu/~anna/Stat597AFall2016/rnf6080.dat"
data.df <- read.table(url, header = FALSE)
```

2. Сколько строк и столбцов в `data.df`? Если получилось не 5070 наблюдений 27 переменных, то проверяйте аргументы.

```
dim(data.df)
```

```
## [1] 5070 27
```

3. Получите имена колонок из `data.df`

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

```
## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9" "V10" "V11" "V12"
## [13] "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22" "V23" "V24"
## [25] "V25" "V26" "V27"
```

4. Найдите значение из 5 строки седьмого столбца.

```
data.df[5, 7]
```

```
## [1] 0
```

5. Напечатайте целиком 2 строку из `data.df`.

```
data.df[2, ]
```

```
## V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18 V19 V20 V21
## 2 60 4 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## V22 V23 V24 V25 V26 V27
## 2 0 0 0 0 0 0
```

6. Объясните, что делает следующая строка кода `names(data.df) <- c("year", "month", "day", seq(0,23))`. Воспользуйтесь функциями `head` и `tail`, чтобы просмотреть таблицу. Что представляют собой последние 24 колонки?

```
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
## 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

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
## 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
## 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
## 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
## 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
## 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
## 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
##       23
## 5065   0
## 5066   0
## 5067   0
## 5068   0
## 5069   0
## 5070   0
```

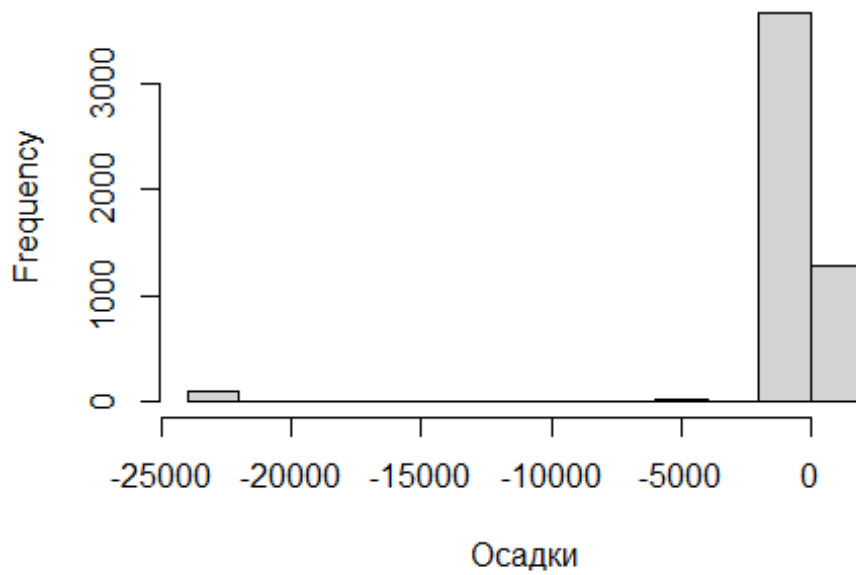
Строка кода `names(data.df) <- c("year", "month", "day", seq(0,23))` переименовала названия столбцов таблицы. Первые три показывают дату, остальные 24 - осадки в Канаде с 1960 по 1980 года по часам.

- Добавьте новую колонку с названием `daily`, в которую запишите сумму крайних правых 24 колонок. Постройте гистограмму по этой колонке. Какие выводы можно сделать?

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

hist(data.df$daily, main = "Гистограмма осадков в Канаде", xlab = "Осадки")
```

Гистограмма осадков в Канаде

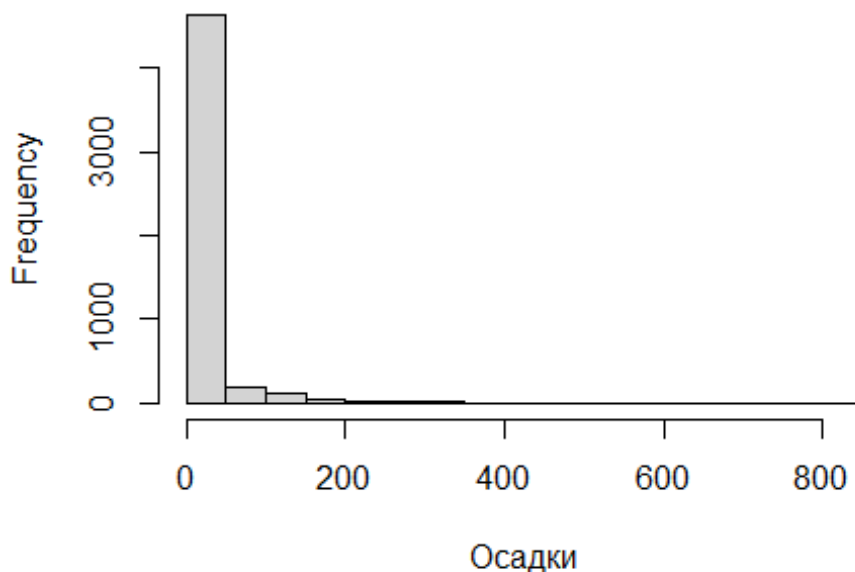


Гистограмма отображается нелогично, так как в таблице есть отрицательные осадки.

8. Создайте новый датафрейм `fixed.df` в котором исправьте замеченную ошибку.
Постройте новую гистограмму, поясните почему она более корректна.

```
fixed.df <- data.df  
fixed.df$dialy[fixed.df$dialy < 0] <- 0  
hist(fixed.df$dialy, main = "Новая гистограмма", xlab = "Осадки")
```

Новая гистограмма



Данная гистограмма наиболее корректна, так как в ней отсутствуют отрицательные значения осадков, которые были заменены на 0. По гистограмме можно сделать вывод, что большую часть времени (более 4000 дней) в Канаде не было осадков.

Синтаксис и типизирование

1. Для каждой строки кода поясните полученный результат, либо объясните почему она ошибочна.

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

## [1] "8"

sort(v)

## [1] "15" "16" "23" "4"  "42" "8"
```

В первой строке заводится массив данных, состоящий из строковых значений. `max(v)` вернул "8", так как это не число, а строка, в остальных значениях первым символом выступает 1, 2 или 4, что меньше 8. `sort(v)` вернул отсортированный список строк. `sum(v)` выдал ошибку, так как строки нельзя сложить.

2. Для следующих наборов команд поясните полученный результат, либо объясните почему они ошибочны.

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

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

## [1] 19
```

```
l4 <- list(z1="6", z2=42, z3="49", z4=126)
l4[[2]] + l4[[4]]

## [1] 168
```

v2[2] + v2[3] - одного элемента не существует. df3[1,2] + df3[1,3] сложил два элемента первой строки l4[2] + l4[4] - происходит сложение списков, а не чисел

Работа с функциями и операторами

1. Оператор двоеточие создаёт последовательность целых чисел по порядку. Этот оператор — частный случай функции seq(), которую вы использовали раньше. Изучите эту функцию, вызвав команду ?seq. Используя полученные знания выведите на экран: Числа от 1 до 10000 с инкрементом 372. Числа от 1 до 10000 длиной 50.

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

## [1] 1 373 745 1117 1489 1861 2233 2605 2977 3349 3721 4093 4465 4837
## [16] 5581 5953 6325 6697 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
## [7] 1225.3673 1429.4286 1633.4898 1837.5510 2041.6122 2245.6735
## [13] 2449.7347 2653.7959 2857.8571 3061.9184 3265.9796 3470.0408
## [19] 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
## [31] 6122.8367 6326.8980 6530.9592 6735.0204 6939.0816 7143.1429
## [37] 7347.2041 7551.2653 7755.3265 7959.3878 8163.4490 8367.5102
## [43] 8571.5714 8775.6327 8979.6939 9183.7551 9387.8163 9591.8776
## [49] 9795.9388 10000.0000
```

2. Функция rep() повторяет переданный вектор указанное число раз. Объясните разницу между rep(1:5,times=3) и 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
```

Первый вариант выводит вектор целиком, повторяя его трижды. Второй вариант выводит весь вектор, повторяя каждый из его элементов по три раза.