

Домашнее задание № 1

Латников Вячеслав

2024-10-18

Работа с данными.

- Загрузите данные в датафрейм, который назовите `data.df`. Сколько строк и столбцов в `data.df`? Если получилось не 5070 наблюдений 27 переменных, то проверяйте аргументы.

```
data.df <- read.table("http://people.math.umass.edu/~anna/Stat597AFall12016/rnf6080.dat")
dim(data.df)
```

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

- Получите имена колонок из `data.df`.

```
names(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"
```

- Найдите значение из 5 строки седьмого столбца.

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

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

- 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  0
##  V22 V23 V24 V25 V26 V27
## 2   0   0   0   0   0   0
```

- 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 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
## 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
## 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
## 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
```

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

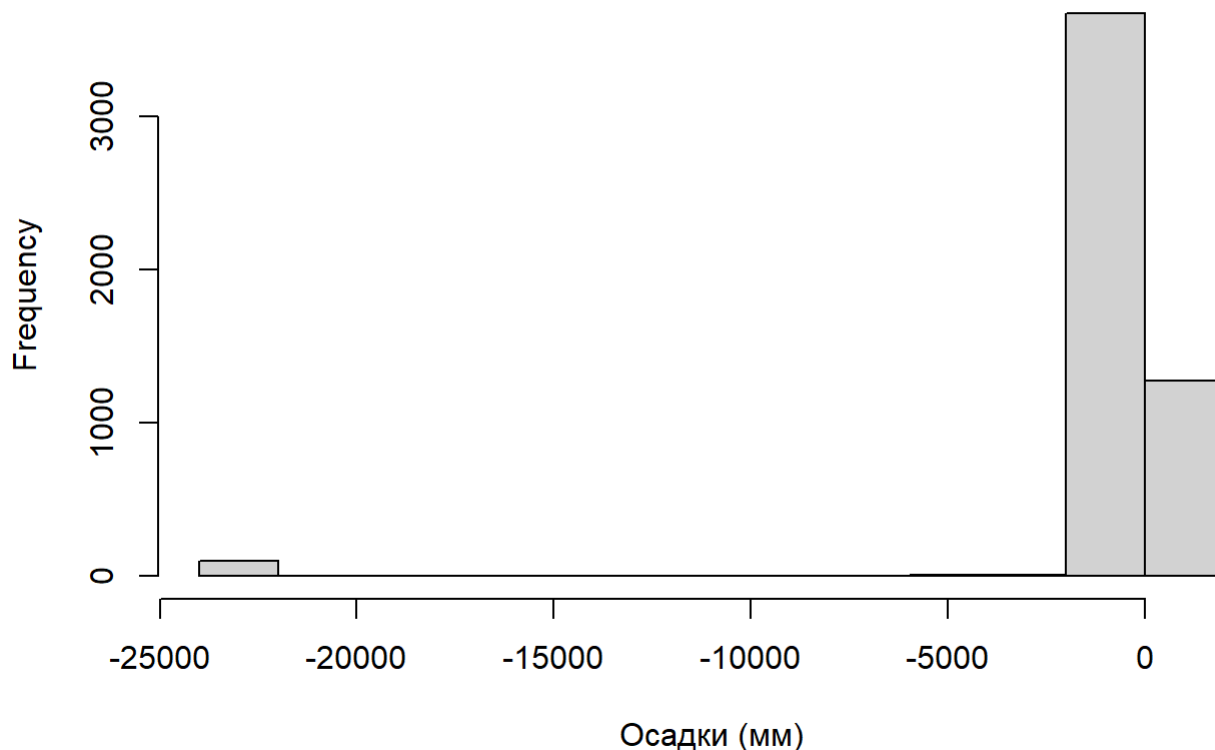
```
##      year month day 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
## 5065   80    11  25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## 5066   80    11  26 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## 5067   80    11  27 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## 5068   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
## 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
## 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
##      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))` присваивает имена колонкам. Первые три колонки называются "year" (год), "month" (месяц) и "day" (день), а оставшиеся 24 колонки представляют собой количество осадков за каждый час суток (от 0 до 23 часов).

- 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 суточных осадков", xlab="Осадки (мм)")
```

Гистограмма суточных осадков



На гистограмме видно значительное количество отрицательных значений вплоть до -25000 мм, что физически невозможно.

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

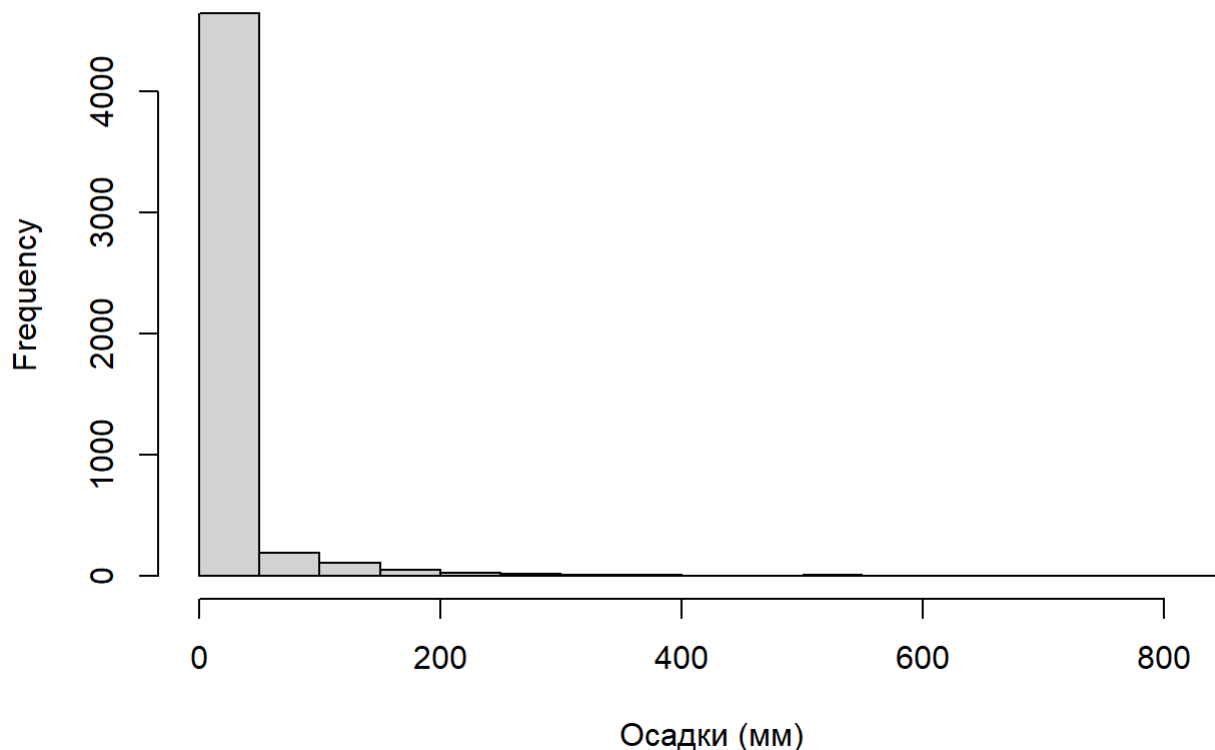
```
fixed.df <- data.df
```

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

```
fixed.df$daily[fixed.df$daily < 0] <- 0
```

```
hist(fixed.df$daily, main="Гистограмма исправленных суточных осадков", xlab="Осадки (мм)")
```

Гистограмма исправленных суточных осадков



В данном случае мы заменяем отрицательные значения осадков на 0, что даёт нам более правильную картину.

Синтаксис и типизирование

- 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"
```

Данный вектор содержит строковые данные, а не числовые. `max(v)` выведет 8, так как она больше в алфавитном порядке. По такому же принципу сработает `sort(v)`. `sum(v)` выведет ошибку, тк не сможет сложить строковые значения.

- Для следующих наборов команд поясните полученный результат, либо объясните почему они ошибочна.

```
v2 <- c("5",7,12) v2[2] + 2[3]
```

Здесь создается вектор v2, который содержит смесь типов данных: строку “5” и числа 7 и 12. Все числа будут преобразованы в строки. Корректный синтаксис для обращения к элементам вектора: v2[2] + v2[3], но нужно, чтобы они были числами.

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

```
## [1] 19
```

Здесь происходит обращение к элементам датафрейма: df3[1,2] — это значение в первой строке второго столбца (7), а df3[1,3] — значение в первой строке третьего столбца (12). Оба значения — числовые, поэтому их сложение возвращает 19.

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

```
## [1] 168
```

l4[2] + l4[4] выведет ошибку, тк здесь возвращаются списки, а не сами значения. Операция сложения между списками невозможна. l4[[2]] — это доступ к элементу списка по индексу 2 (значение 42), а l4[[4]] — элемент с индексом 4 (значение 126). Оба значения — числа, поэтому их сложение возвращает 168.

Работа с функциями и операторами

*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.

```
# Числа от 1 до 10000 с шагом 372
seq(from = 1, to = 10000, by = 372)
```

```
## [1] 1 373 745 1117 1489 1861 2233 2605 2977 3349 3721 4093 4465 4837 5209
## [16] 5581 5953 6325 6697 7069 7441 7813 8185 8557 8929 9301 9673
```

```
# Числа от 1 до 10000 длиной 50
seq(from = 1, to = 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
```

*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
```

times=3 повторяет весь вектор три раза. each=3 повторяет каждый элемент вектора три раза, прежде чем перейти к следующему элементу.