hw01

Chufarov Konstantin

2024-10-19

Анализ данных: дождевые осадки

Загрузка данных

Для начала загрузим набор данных о количестве осадков в Канаде за период с 1960 по 1980 годы, используя функцию `read.table()`. Назовем датафрейм `rainfall_data`.

```
rainfall_data <- read.table("https://people.math.umass.edu/~anna/Stat597AFall2016/rnf6080.dat")
cat("Число строк в таблице: ", nrow(rainfall_data))
```

```
## Число строк в таблице:  5070
```

```
cat("\nЧисло столбцов в таблице: ", ncol(rainfall_data))
```

```
##
## Число столбцов в таблице:  27
```

Имена колонок

Получим список названий столбцов для анализа структуры данных.

```
colnames(rainfall_data)
```

```
##  [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"
```

Доступ к элементам

Получим значение из 5-й строки и 7-го столбца, чтобы убедиться, что данные загружены правильно.

```
rainfall_data[5, 7]
```

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

Также выведем на экран вторую строку таблицы.

```
rainfall_data[2, ]
```

	V1 <int>	V2 <int>	V3 <int>	V4 <int>	V5 <int>	V6 <int>	V7 <int>	V8 <int>	V9 <int>
2	60	4	2	0	0	0	0	0	0

1 row | 1-10 of 28 columns

Замена заголовков

Теперь заменим имена столбцов на более удобные для анализа.

```
names(rainfall_data) <- c("year", "month", "day", seq(0, 23))
```

Посмотрим, как выглядят первые и последние строки таблицы, используя `head()` и `tail()`.

```
head(rainfall_data)
```

	year <int>	month <int>	day <int>	0 <int>	1 <int>	2 <int>	3 <int>	4 <int>	5 <int>
1	60	4	1	0	0	0	0	0	0
2	60	4	2	0	0	0	0	0	0
3	60	4	3	0	0	0	0	0	0
4	60	4	4	0	0	0	0	0	0
5	60	4	5	0	0	0	0	0	0
6	60	4	6	0	0	0	0	0	0

6 rows | 1-10 of 28 columns

```
tail(rainfall_data)
```

	year <int>	month <int>	day <int>	0 <int>	1 <int>	2 <int>	3 <int>	4 <int>	5 <int>
5065	80	11	25	0	0	0	0	0	0
5066	80	11	26	0	0	0	0	0	0
5067	80	11	27	0	0	0	0	0	0
5068	80	11	28	0	0	0	0	0	0
5069	80	11	29	0	0	0	0	0	0
5070	80	11	30	0	0	0	0	0	0

6 rows | 1-10 of 28 columns

Добавление столбца с суточными осадками

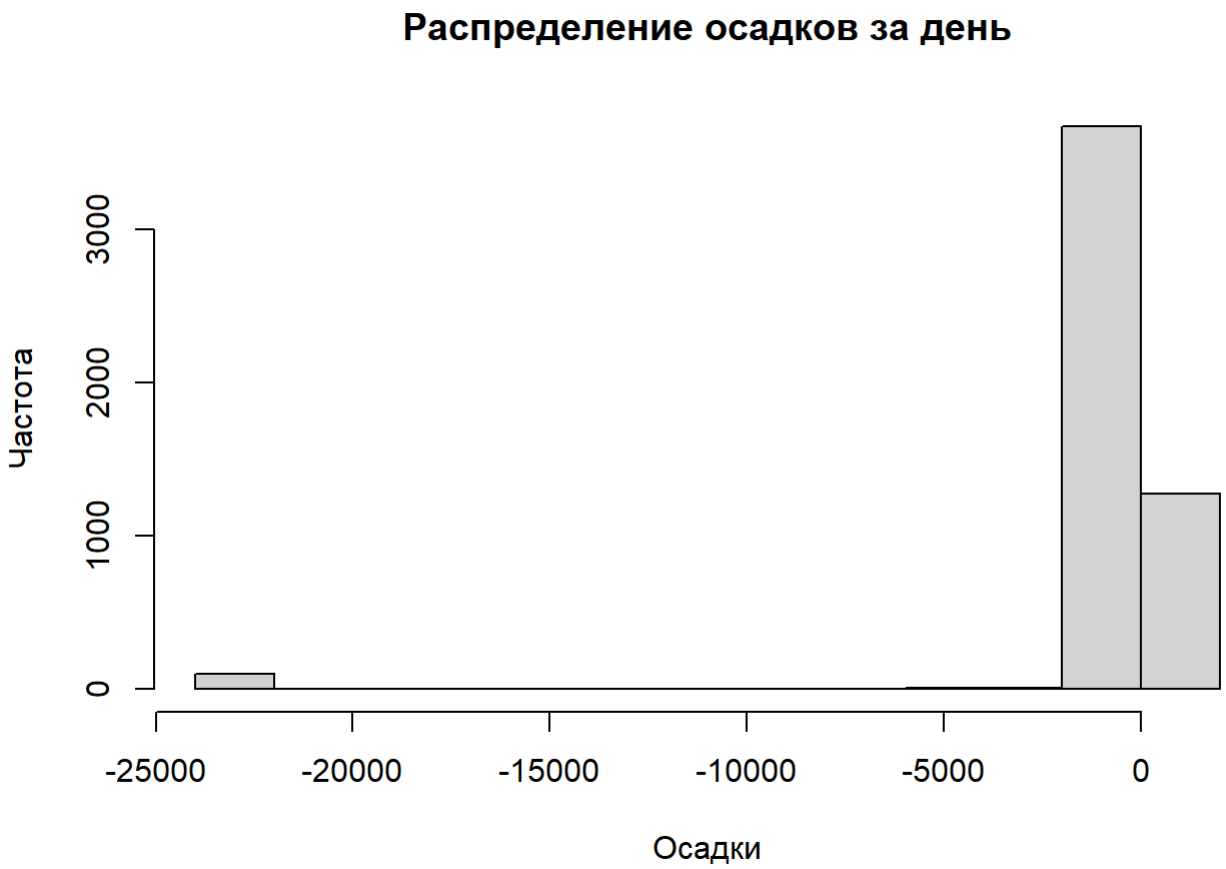
Добавим новый столбец `daily`, который будет содержать сумму значений осадков за каждый день.

```
rainfall_data$daily <- rowSums(rainfall_data[, 4:27])
head(rainfall_data)
```

	year <int>	month <int>	day <int>	0 <int>	1 <int>	2 <int>	3 <int>	4 <int>	5 <int>
1	60	4	1	0	0	0	0	0	0
2	60	4	2	0	0	0	0	0	0
3	60	4	3	0	0	0	0	0	0
4	60	4	4	0	0	0	0	0	0
5	60	4	5	0	0	0	0	0	0
6	60	4	6	0	0	0	0	0	0

6 rows | 1-10 of 29 columns

```
hist(rainfall_data$daily, main = "Распределение осадков за день", xlab = "Осадки", ylab = "Частота")
```

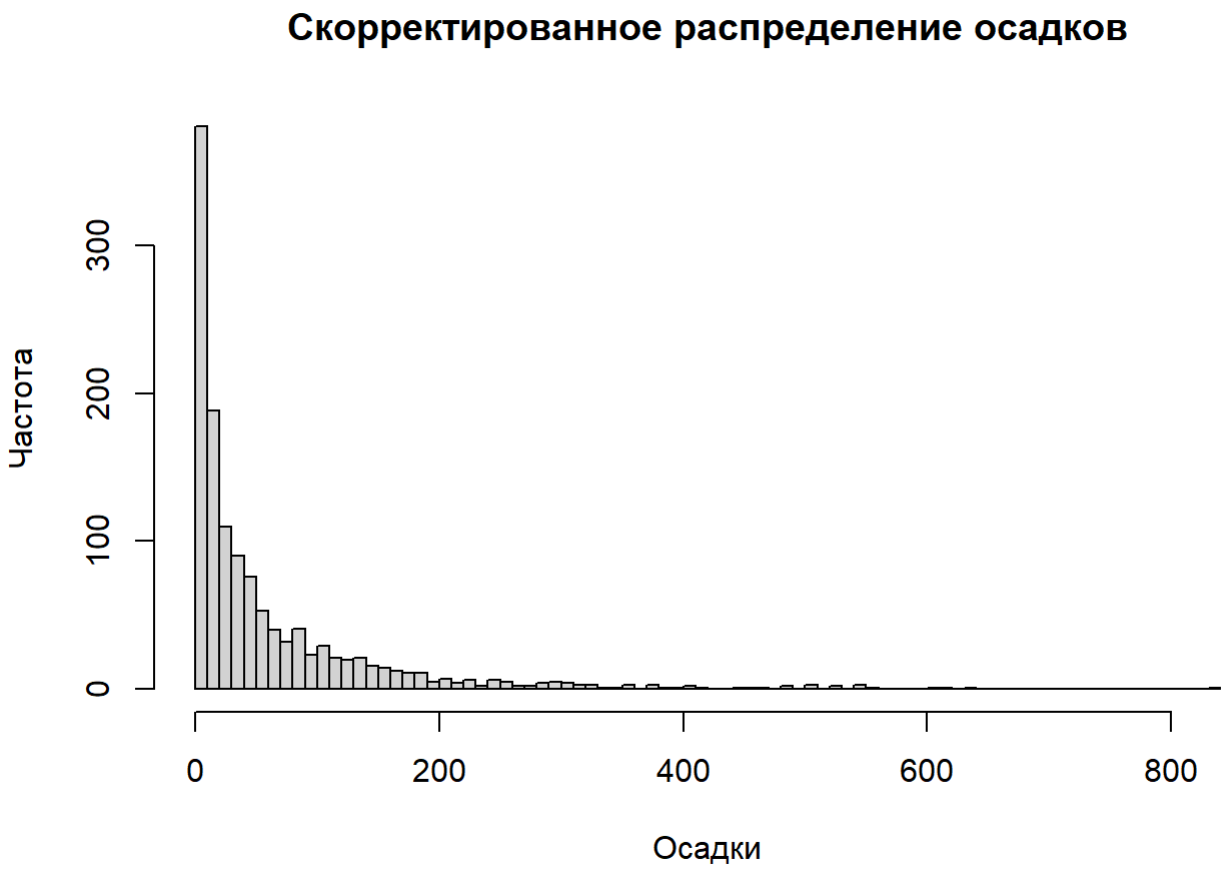


Мы видим, что в таблице есть некорректные значения, такие как -999, которые необходимо исключить.

Исправление ошибки

Создадим новый датафрейм, в котором будут удалены некорректные данные.

```
cleaned_data <- rainfall_data[rainfall_data$daily > 0, ]
hist(cleaned_data$daily, main = "Скорректированное распределение осадков", xlab = "Осадки", y
lab = "Частота", breaks = 80)
```



Работа с векторами

Пример 1: Вектор строк

Попробуем применить различные функции к строковому вектору.

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

```
## [1] 42
```

```
sort(v_numeric)
```

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

```
sum(v_numeric)
```

```
## [1] 108
```

Пример 2: Вектор с элементами разных типов

Рассмотрим смешанный тип данных. Сложение вызовет ошибку. Вектор смешанного типа автоматически преобразуется в строковый.

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

Пример 3: Работа с датафреймом

Произведём вычисления с элементами датафрейма.

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

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

Пример 4: Операции со списком

Работаем со списками:

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

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

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

Второе выражение вызовет ошибку, так как обращение l4[2] и l4[4] возвращает список, а не числовые значения.

Операторы и функции

Последовательность чисел

Создадим последовательность чисел при помощи функции `seq()`.

```
seq(1, 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
```

```
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
```

Повторение элементов вектора

Разница между функциями `rep()`.

```
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
```

Первое выражение повторяет весь вектор 1:5 три раза. Второе выражение повторяет каждый элемент вектора три раза.