hw01

БорисихинЛев_РИ-411055

2024-10-17

Загрузим данные в датафрейм с использованием функции `read.table()`

```
data.df <- read.table("https://people.math.umass.edu/~anna/Stat597AFall2016/rnf6080.dat")
cat("Число строк: ", ncol(data.df))
```

```
## Число строк:  27
```

```
cat("\nЧисло столбцов: ", nrow(data.df))
```

```
##
## Число столбцов:  5070
```

Получим названия колонок с помощью `colnames()`

```
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"
```

Извлечем элемент, находящийся в 5-й строке и 7-м столбце

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

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

Выведем вторую строку датафрейма

```
data.df[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

Присвоим имена столбцам датафрейма: первые три столбца — это `year`, `month` и `day`, остальные столбцы — от 0 до 23

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

Используем функции `head()` и `tail()` для просмотра первых и последних 6 строк. Столбцы с 0 по 23 отображают осадки по часам.

```
head(data.df)
```

	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(data.df)
```

	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

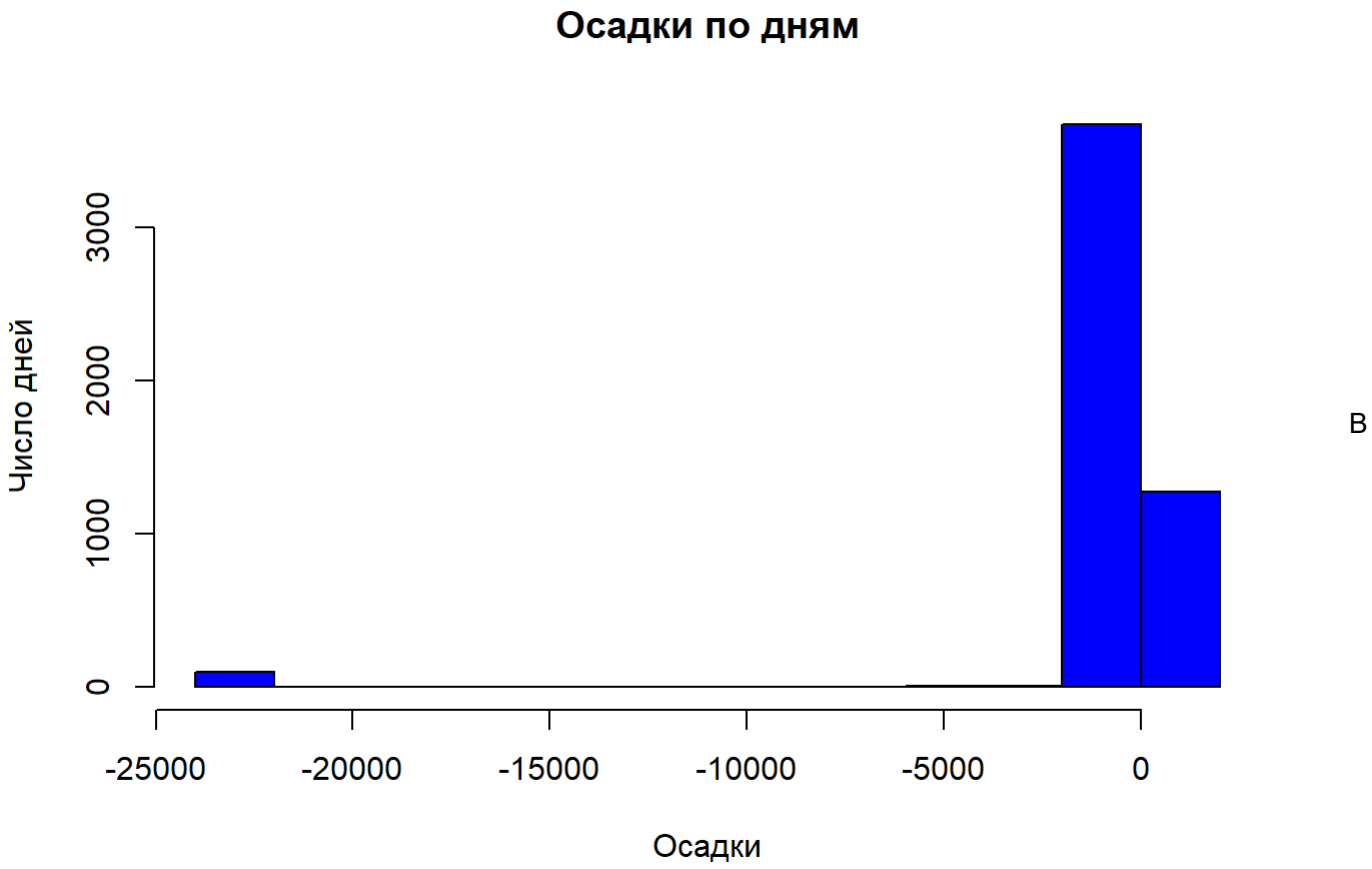
6 rows | 1-10 of 28 columns

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

	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(data.df$daily, main = "Осадки по дням", xlab = "Осадки", ylab = "Число дней", col='blu
e')
```



В

данных есть некорректные значения (-999), исправим это.

```
fixed.df <- data.df[data.df$daily > 0, ]
hist(fixed.df$daily, main = "Осадки по дням", xlab = "Осадки", ylab = "Число дней", breaks =
80, col='green')
```

Новый датафрейм исключает отрицательные значения и дни без осадков. Новая гистограмма более точна, так как построена по валидным данным.

Создадим массив строк

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

Найдем максимальное значение в массиве строк (результат зависит от кодировки и локали)

```
max(v)
```

```
## [1] "8"
```

Отсортируем массив строк (результат также зависит от кодировки и локали)

```
sort(v)
```

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

Следующая команда приведет к ошибке, так как нельзя складывать строки

```
sum(v)
```

Создадим вектор, содержащий строки и числа. В первом случае всё корректно, во втором — ошибка, так как попытка обратиться к индексу числа является некорректной.

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

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

Создадим датафрейм с колонками `z1`, `z2`, `z3`, присвоим им значения “5”, 7, 12 соответственно и сложим элементы 2 и 3 столбцов первой строки.

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

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

Создадим список с элементами `z1`, `z2`, `z3`, `z4` и сложим 2-й и 4-й элементы. В последней команде ошибка, так как `l4[2]` возвращает не значение элемента, а подсписок вида `list(z2=42)`.

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

Создадим последовательность чисел от 1 до 10000 с шагом 372 с помощью функции `seq()`

```
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
```

Создадим последовательность из 50 чисел от 1 до 10000 с использованием параметра `length.out`, который распределяет значения равномерно.

```
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
```

Разница между командами в том, что первая повторяет всю последовательность трижды, а вторая — каждый элемент трижды перед переходом к следующему.

```
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
```