Домашнее задание №1

Швецова Ксения Георгиевна

Группа РИ-411055

Работа с данными

Результат представлен схемой код-пояснение.

```
data.df <-
read.table("http://people.math.umass.edu/~anna/Stat597A
Fall2016/rnf6080.dat")
```

Эта команда загружает данные из указанного URL-адреса в датафрейм под именем data.df. Функция read.table() используется для чтения таблиц из файлов или URL-адресов. В данном случае данные представлены в виде текстового файла, содержащего числовые значения, разделённые пробелами.

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

Функция dim() возвращает размерность объекта, в данном случае датафрейма data.df. Она выводит количество строк и количество столбцов в формате (строки, столбцы).

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

Функция colnames() возвращает список имён столбцов датафрейма data.df.

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

Эта команда извлекает значение из ячейки, находящейся на пересечении пятой строки и седьмого столбца датафрейма data.df.

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

Команда выводит всю вторую строку датафрейма data.df. Символ , указывает, что нужно выбрать все столбцы данной строки.

```
names(data.df) <- c("year", "month", "day",
paste0("hour_", seq(0, 23)))
```

Здесь происходит изменение имён столбцов датафрейма. Команда назначает первым трём столбцам имена year, month и day, а остальным 24 столбцам назначаются имена hour_0, hour_1, ..., hour_23. Функция paste0() объединяет строку "hour_" с последовательностью чисел от 0 до 23, создавая соответствующие имена для каждого часа.

```
head(data.df, n = 10)
```

```
tail(data.df, n = 15)
```

Функции head() и tail() позволяют посмотреть начало и конец датафрейма соответственно. В данном случае head(data.df, n = 10) показывает первые 10 строк датафрейма, а tail(data.df, n = 15) — последние 15 строк.

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

Добавляется новая колонка daily в датафрейм data.df, в которой хранится сумма осадков за каждый день. Функция rowSums() вычисляет сумму значений каждой строки в указанных столбцах (в данном случае это столбцы с 4-го по 27-й, которые соответствуют часам суток).

```
hist(data.df$daily, main="Гистограмма осадков по дню",
xlab="Осадки", ylab="Частота")
```

Команда строит гистограмму распределения значений в колонке daily датафрейма data.df. Аргумент main задаёт заголовок графика, xlab — подпись оси X, а ylab — подпись оси Y.

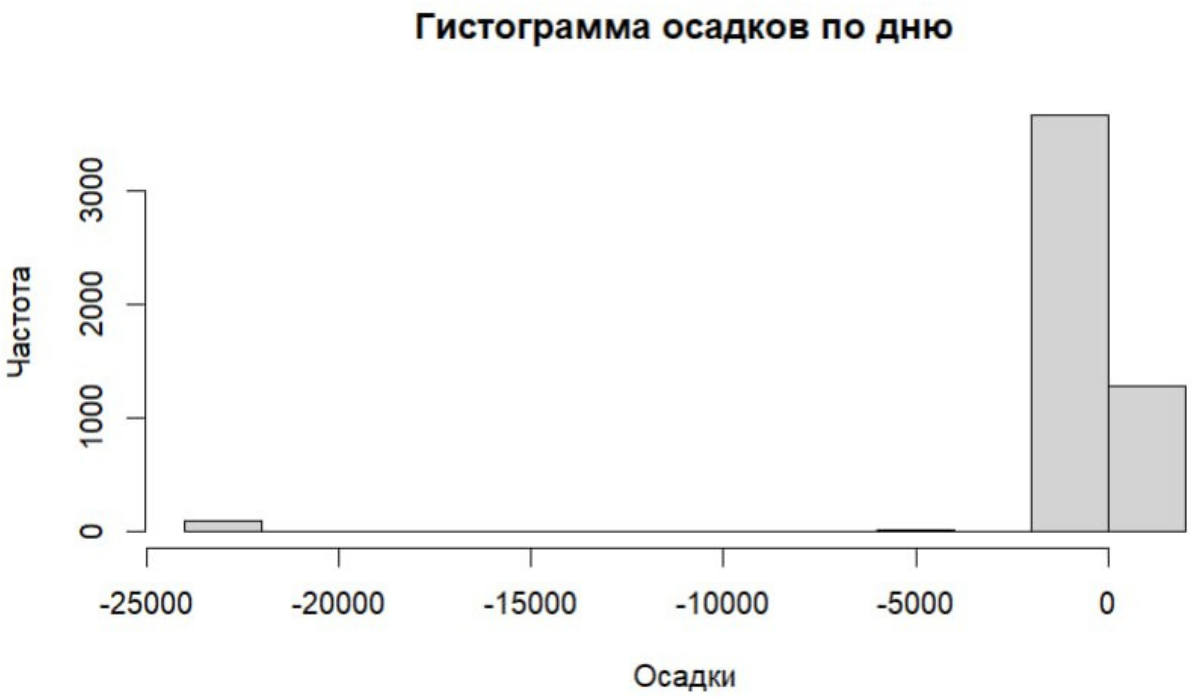

Рисунок 1 – Гистограмма распределения значений

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

Создаётся копия датафрейма data.df под новым именем fixed.df. Этот шаг необходим для внесения исправлений без изменения оригинального датафрейма.

```
fixed.df[,4:27][fixed.df[,4:27]<0] <- 0
```

Данная команда заменяет все отрицательные значения в столбцах с 4-го по 27-й на нули. Сначала выбираются все элементы этих столбцов, затем среди них находятся те, которые меньше нуля, и им присваивается значение 0.

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

Пересчитывается сумма осадков для каждого дня с учётом исправленных значений. Как и раньше, используется функция rowSums(), но теперь она применяется к скорректированным данным в датафрейме fixed.df.

```
hist(fixed.df$daily, main="Гистограмма осадков по дню",
xlab="Осадки", ylab="Частота")
```

Строится новая гистограмма на основе исправленного датафрейма fixed.df. График отображает распределение значений в колонке daily с теми же параметрами подписи осей и заголовком, что и в первой гистограмме.

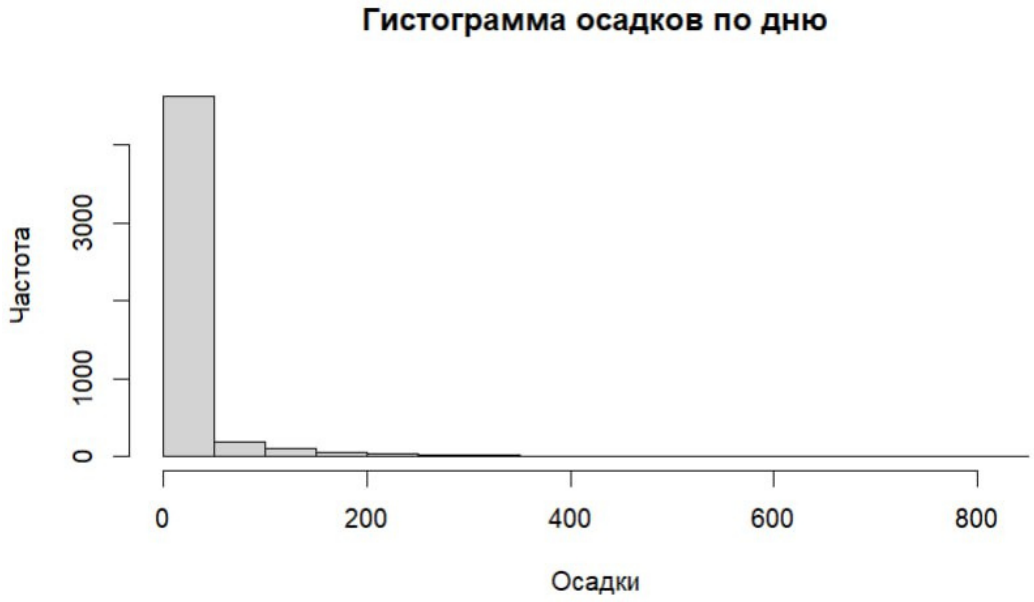

Рисунок 2 – Исправленная гистограмма

Синтаксис и типизирование

Часть 1:

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

В результате получаем вектор v с элементами типа *character*.

Функция c() объединяет указанные элементы в вектор. Поскольку все элементы заданы в кавычках, они интерпретируются как строки ("character"). Таким образом, v является вектором строковых значений.

```
max(v)
```

Результатом будет ошибка.

Функция max() пытается найти максимальное значение среди элементов вектора. Однако, поскольку элементы вектора являются строковыми значениями, сравнение между ними невозможно, поэтому возникает ошибка.

```
sort(v)
```

В итоге получится отсортированный вектор строковых значений.

Функция sort() сортирует элементы вектора в алфавитном порядке, так как элементы вектора являются строками, то сортировка осуществляется на основе буквенных значений.

```
sum(v)
```

Выведется ошибка.

Аналогично функции max(), функция sum() ожидает числовые значения для сложения. Поскольку элементы вектора v являются строками, попытка сложить их приводит к ошибке.

Часть 2

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

Здесь ошибка возникает из-за неправильного индексации. Когда ты пишешь 2[3], это эквивалентно записи 2, потому что индекс [3] не имеет смысла для скалярного значения. Правильный способ доступа к элементам вектора выглядит так: v2[2] + v2[3].

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

В этом примере ошибка связана с типом данных. Столбец z1 в датафрейме df3 содержит строковый тип данных ("5"), а столбцы z2 и z3 — числовые типы. При попытке сложить строковую и числовую величины возникает ошибка.

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

Результаты:

l4[[2]] + l4[[4]]: Верный результат.

l4[2] + l4[4]: Ошибочный результат.

l4[[2]] и l4[[4]] возвращают конкретные элементы списка (здесь это числа 42 и 126), и их можно спокойно сложить.

l4[2] и l4[4] возвращают части списка, а не отдельные элементы. Это значит, что при попытке сложить два подсписка, что вызывает ошибку. Правильным способом доступа к отдельным элементам списка является использование двойных квадратных скобок [[ ]].

Работа с функциями и операторами

Используем функцию seq:

- `seq(1, 10000, 372)`

Выход:

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

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

Выход:

```
[1]     1   203   405   607   809  1011  1213
1415  1617  1819  2021  2223  2425  2627  2829
3031  3233  3435  3637  3839  4041  4243  4445
4647  4849  5051  5253  5455  5657  5859  6061
6263  6465  6667  6869  7071  7273  7475  7677
7879  8081  8283  8485  8687  8889  9091  9293
9495  9697  9899 10001
```

Разница между rep(1:5, times=3) и rep(1:5, each=3)

rep(1:5, times = 3) повторяет весь вектор 1:5 три раза подряд.

rep(1:5, each = 3) сначала повторяет первый элемент три раза, затем второй элемент три раза и так далее.