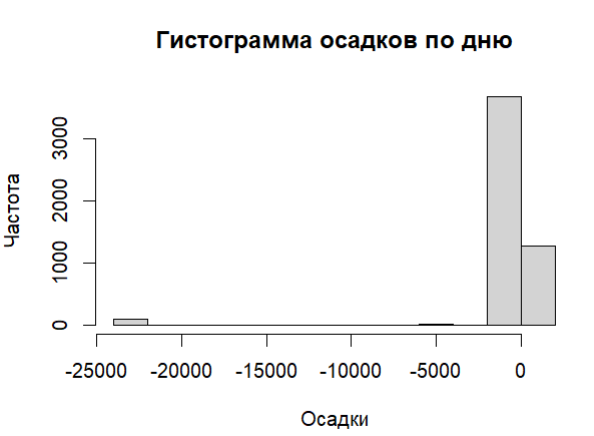
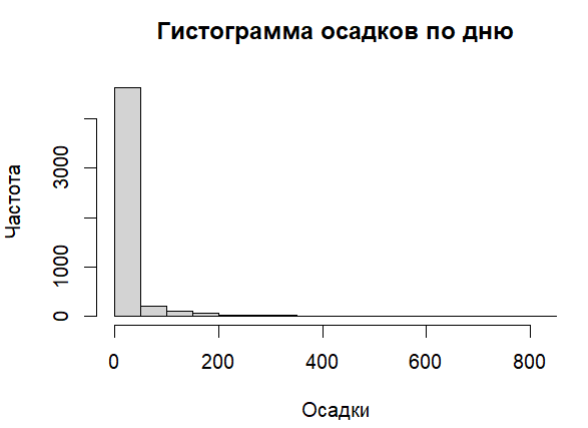
**РАБОТА С ДАННЫМИ**



Вывод: строим гистограмму, получилась некорректная гистограмма из-за значений <0



Вывод: гистограмма получилась более правильной, но все равно из-за неверного изначального заполнения нельзя сделать её точной

**СИНТАКСИС И ТИПИЗИРОВАНИЕ**

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

Переменной v присваивается вектор символов (строк), представляющих числа.

max(v)

Функция max() возвращает максимальный элемент вектора. Однако вектор v состоит из строковых значений, а не числовых. При сравнении строк функция определяет максимум по алфавитному порядку, а не по числовому. Результатом будет "8", так как она стоит выше в алфавите, чем остальные строки.

sort(v)

Функция sort() сортирует элементы вектора в алфавитном порядке (поскольку это строки). Результат будет: ["15", "16", "23", "4", "42", "8"]. Числа сортируются как строки, поэтому 4 и 8 идут после 16 и 23.

sum(v)

Функция sum() пытается сложить элементы вектора, но так как это строки, а не числа, она вызывает ошибку: invalid 'type' (character) of argument. Для работы функции sum() значения должны быть числовыми.

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

Создается вектор с элементами "5", 7, и 12. Однако первый элемент "5" — это строка.

v2[2] + 2[3]

Здесь v2[2] вернет второй элемент вектора v2, то есть число 7.

2[3] — это некорректное обращение. В R это вызывает ошибку, так как запись 2[3] не имеет смысла (число 2 не является вектором, и у него нет третьего элемента). Ошибка: Error in 2[3] : subscript out of bounds.

df3 <- data.frame(z1="5", z2=7, z3=12)

Создается датафрейм с колонками z1, z2, и z3. Первый элемент — строка "5", второй и третий — числа 7 и 12.

df3[1, 2] + df3[1, 3]

df3[1, 2] возвращает значение 7 из колонки z2, а df3[1, 3] — значение 12 из колонки z3. Складываются два числа: 7 + 12 = 19. Результат: 19.

l4 <- list(z1="6", z2=42, z3="49", z4=126)

Создается список с элементами "6", 42, "49", и 126.

l4[[2]] + l4[[4]]

l4[[2]] обращается ко второму элементу списка, который равен 42, а l4[[4]] — к четвертому элементу, равному 126. Складываются два числа: 42 + 126 = 168. Результат: 168.

l4[2] + l4[4]

l4[2] и l4[4] возвращают списки, а не отдельные значения. Складывать списки напрямую нельзя, и это вызовет ошибку: non-numeric argument to binary operator. Для доступа к элементам нужно использовать двойные квадратные скобки [[ ]].

**РАБОТА С ФУНКЦИЯМИ И ОПЕРАТОРАМИ**

Числа от 1 до 10000 с инкрементом 372:

Используем аргумент by, чтобы задать шаг последовательности.

seq(from = 1, to = 10000, by = 372)

Здесь последовательность начнется с 1 и будет увеличиваться с шагом 372 до достижения или превышения значения 10000.

Числа от 1 до 10000 длиной 50:

Используем аргумент length.out, чтобы задать количество элементов последовательности.

seq(from = 1, to = 10000, length.out = 50)

Эта команда создаст последовательность из 50 чисел, равномерно распределённых между 1 и 10000.

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.

Результат: 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5.

rep(1:5, each=3):

Повторяет каждый элемент вектора 1:5 по три раза.

Результат: 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5.