

hw01

Домашнее задание № 1

Работа с данными.

По адресу <http://www.stats.uwo.ca/faculty/braun/data/rnf6080.dat>

(<http://www.stats.uwo.ca/faculty/braun/data/rnf6080.dat>) можно получить набор данных об осадках в Канаде с 1960 по 1980 годы. Необходимо загрузить эти данные при помощи `read.table`. Воспользуйтесь справкой, чтобы изучить аргументы, которые принимает функция. - Загрузите данные в датафрейм, который назовите `data.df`.

```
data.df <- read.table("http://www.stats.uwo.ca/faculty/braun/data/rnf6080.dat")
```

- Сколько строк и столбцов в `data.df`? Если получилось не 5070 наблюдений 27 переменных, то проверяйте аргументы. 5070 obs. of 27 variables
- Получите имена колонок из `data.df`.

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

```
## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9" "V10" "V11"
## [12] "V12" "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22"
## [23] "V23" "V24" "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
## 2 60  4  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   V21 V22 V23 V24 V25 V26 V27
## 2   0   0   0   0   0   0   0
```

- Объясните, что делает следующая строка кода
`names(data.df) <- c("year", "month", "day", seq(0,23))`.

```
names(data.df)
```

```
## [1] "V1" "V2" "V3" "V4" "V5" "V6" "V7" "V8" "V9" "V10" "V11"
## [12] "V12" "V13" "V14" "V15" "V16" "V17" "V18" "V19" "V20" "V21" "V22"
## [23] "V23" "V24" "V25" "V26" "V27"
```

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

```
## [1] "year" "month" "day" "0" "1" "2" "3" "4"
## [9] "5" "6" "7" "8" "9" "10" "11" "12"
## [17] "13" "14" "15" "16" "17" "18" "19" "20"
## [25] "21" "22" "23"
```

меняет названия колонок - Воспользуйтесь функциями `head` и `tail`, чтобы просмотреть таблицу. Что представляют собой последние 24 колонки?

```
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
## 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
## 2   60     4   2 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
## 4   60     4   4 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
## 6   60     4   6 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   22 23
## 1   0 0
## 2   0 0
## 3   0 0
## 4   0 0
## 5   0 0
## 6   0 0
```

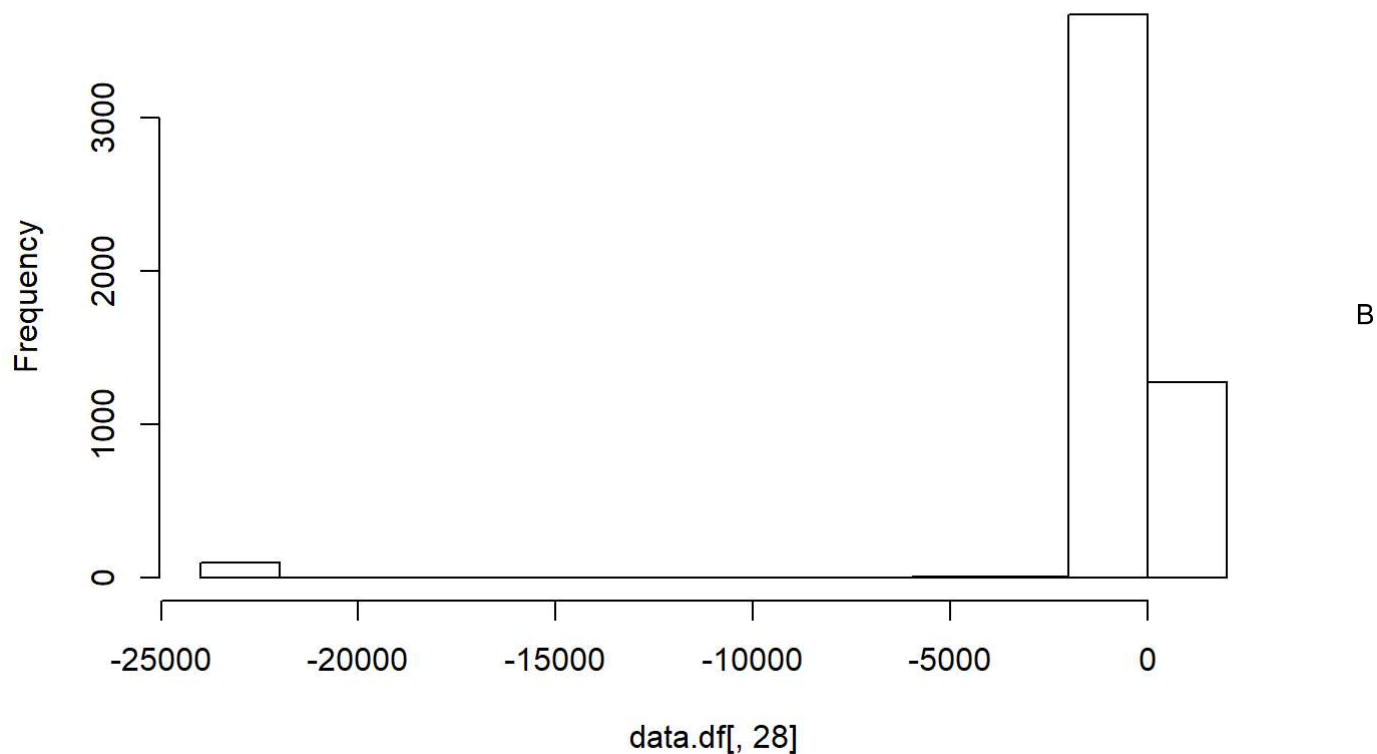
```
tail(data.df)
```

```
##   year month day 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
## 5065   80    11 25 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
## 5067   80    11 27 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
## 5069   80    11 29 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
##   21 22 23
## 5065   0 0 0
## 5066   0 0 0
## 5067   0 0 0
## 5068   0 0 0
## 5069   0 0 0
## 5070   0 0 0
```

Последние 24 колонки - осадки за каждый час - 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[, 28])
```

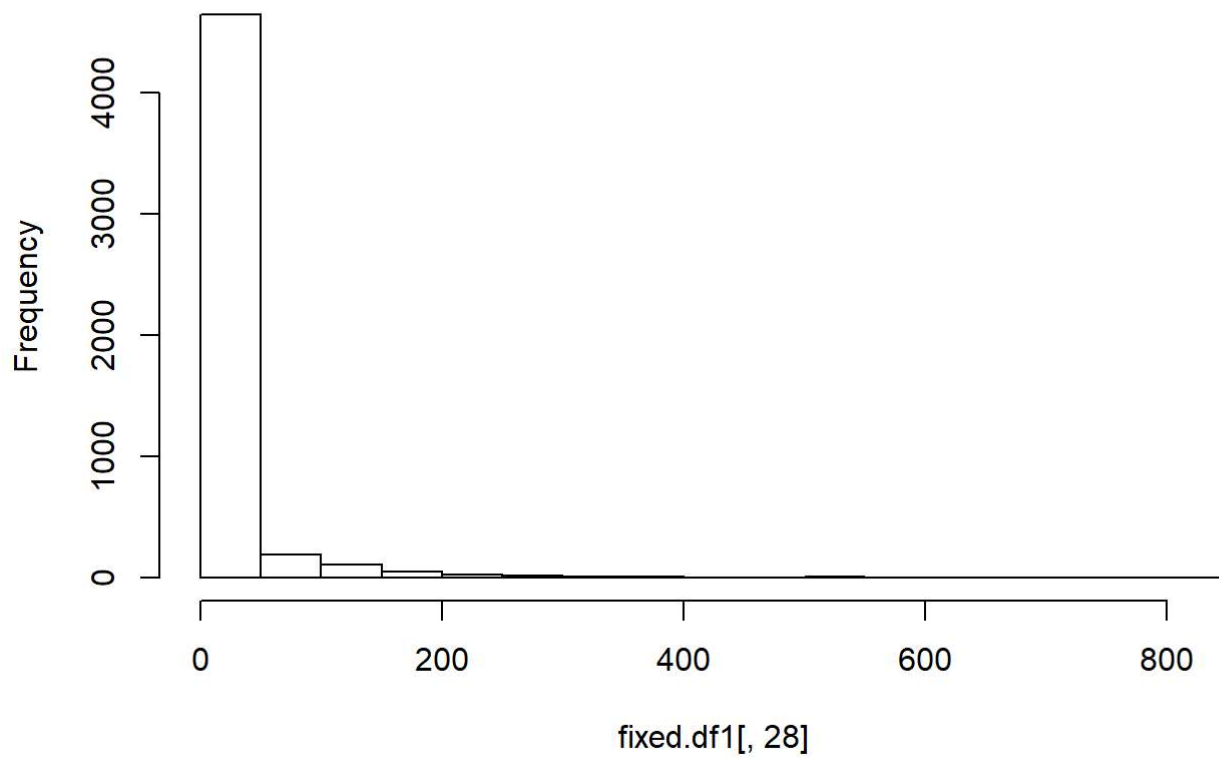
Histogram of data.df[, 28]



данных имеются отрицательные значения количества осадков. Их нужно убрать. - Создайте новый датафрейм `fixed.df` в котром исправьте замеченную ошибку. Постройте новую гистограмму, поясните почему она более корректна. Два возможных решения: обнулить то, что меньше нуля или просто удалить. Что правильнее? На любой из этих гистограмм нет некорректных значений.

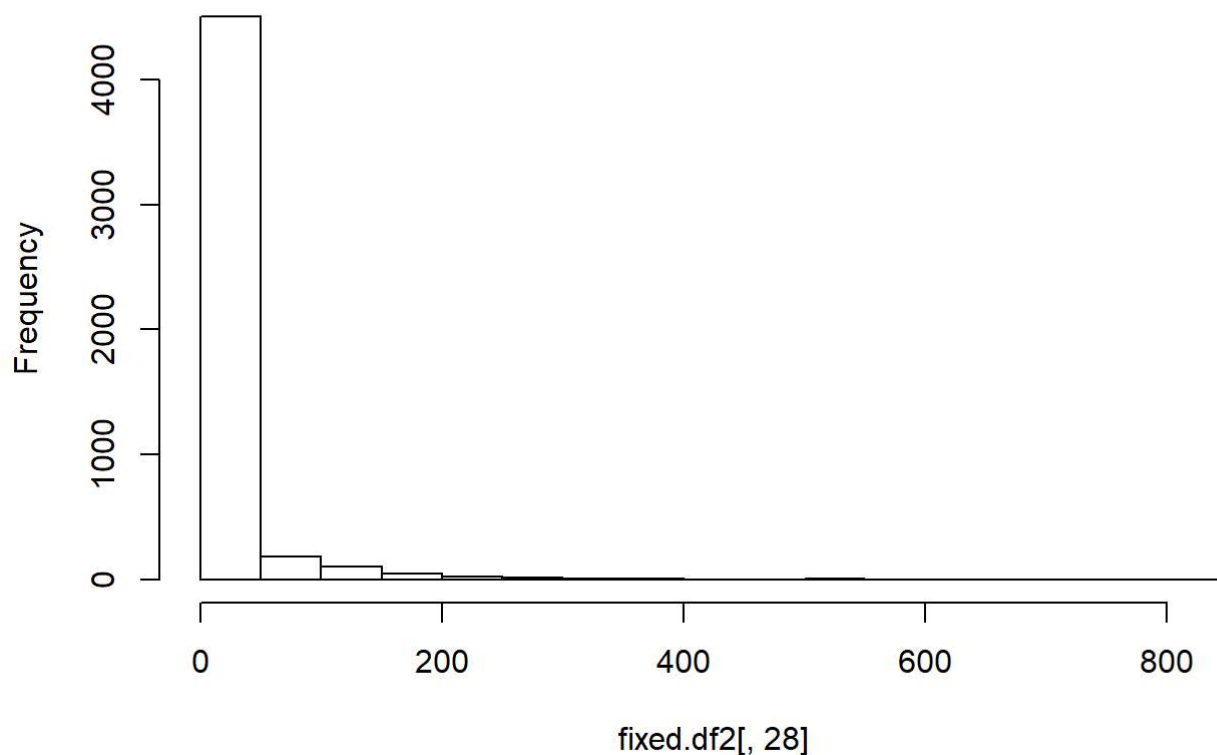
```
fixed.df1 <- data.df  
fixed.df1$daily[which(fixed.df1$daily < 0)] = 0  
hist(fixed.df1[, 28])
```

Histogram of fixed.df1[, 28]



```
fixed.df2<-subset(data.df,data.df[,"daily"]>=0);  
hist(fixed.df2[, 28])
```

Histogram of fixed.df2[, 28]



Синтаксис и типизирование

- Для каждой строки кода поясните полученный результат, либо объясните почему она ошибочна.

```
v <- c("4", "8", "15", "16", "23", "42") #символьный вектор
max(v) #сравнение символьное по первому отличающемуся элементу (максимум из 4 8 1 1 2 4 - это 8)
```

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

```
sort(v) #аналогично по первому символу строки
```

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

```
#sum(v) Error in sum(v) : invalid 'type' (character) of argument
sum(as.integer(v))
```

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

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

```
#векторы
v2 <- c("5",7,12) # элементы вектора одного типа (все стали символьными)
# v2[2] + v2[3] #Error in v2[2] + v2[3] : non-numeric argument to binary operator
as.integer(v2[2]) + as.integer(v2[3])
```

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

```
#В датафреймах и листах элементы могут быть разных типов
#датафреймы
df3 <- data.frame(z1="5",z2=7,z3=12)
df3[1,2] + df3[1,3]
```

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

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

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

```
#l4[2] + l4[4] #ссылка
l4[2]
```

```
## $z2
## [1] 42
```

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.

```
seq(1,10000,by=372);
```

```
## [1] 1 373 745 1117 1489 1861 2233 2605 2977 3349 3721 4093 4465 4837
## [15] 5209 5581 5953 6325 6697 7069 7441 7813 8185 8557 8929 9301 9673
```

- Числа от 1 до 10000 длиной 50.

```
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
```

- 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) #повторение всей последовательности
```

```
## [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
rep(1:5, each=3) #повторение поэлементно
```

```
## [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
```