"Домашняя работа 1"

Izotov Ilya

"26 сентября 2018 г"

Работа с данными.

По адресу http://people.math.umass.edu/~anna/Stat597AFall2016/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://people.math.umass.edu/~anna/Stat597AFall2016/rnf6080.dat")
```

- Сколько строк и столбцов в data.df? Если получилось не 5070 наблюдений 27 переменных, то проверяйте аргументы.

```
length(data.df[1,]); length(data.df[,1])
```

```
## [1] 27
```

```
## [1] 5070
```

- Получите имена колонок из data.df.

```
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"
```

Как видно, стандартные имена столбцов записываются как V1, V2...

- Найдите значение из 5 строки седьмого столбца.

```
paste(data.df[5,7])
```

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

- Напечатайте целиком 2 строку из data.df

```
paste(data.df[2,])
```

```
##  [1] "60" "4"  "2"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"
## [15] "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"  "0"
```

- Объясните, что делает следующая строка кода names(data.df) <- c("year "month "day seq(0,23)). Воспользуйтесь функциями head и tail, чтобы просмотреть таблицу. Что представляют собой последние 24 колонки?

```
names(data.df) <- c("year", "month", "day", seq(0,23))
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  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  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  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  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  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  0  0
##   22 23
## 1  0  0
## 2  0  0
## 3  0  0
## 4  0  0
## 5  0  0
## 6  0  0
```

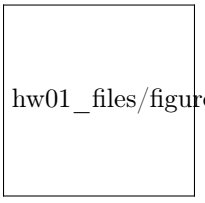
```
tail(data.df)
```

```
##      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  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  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  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  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  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  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
```

Первая строка дает колонкам названия.
1 колонка - год, 2 - месяц, 3 - день, остальные - час от 0 до 23.
Head показывает верхнюю часть таблицы, а tail - нижнюю.

- Добавьте новую колонку с названием daily, в которую запишите сумму крайних правых 24 колонок. Постройте гистограмму по этой колонке. Какие выводы можно сделать?

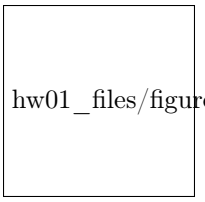
```
data.df.daily <- data.frame(data.df, daily=rowSums(data.df)-data.df$year-data.df$month-data.df$day)
hist(data.df.daily$daily)
```



В данных имеются отрицательные значения. Эти данные необходимо обнулить, т.к. таких значений быть на практике не мо

- Создайте новый датафрейм fixed.df в котром исправьте замеченную ошибку. Постройте новую гистограмму, поясните почему она более корректна.

```
fixed.df <- data.df.daily
fixed.df[fixed.df < 0] <- 0
hist(fixed.df$daily)
```



hw01_files/figure-latex/unnamed-chunk-8-1.pdf

Получившаяся гистограмма более корректна, поскольку имеет только неотрицательные значения. Ошибочные отрицатель

Синтаксис и типизирование

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

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

v <- c("4 "8 "15 "16 "23 "42") - в переменную v помещается вектор символов (char)

max(v) - поиск наибольшего символа, с которого начнется строка. В кодировке код цифры 8 стоит после цифр 0-7, поэтому максимальным элементом вектора будет выбран символ “8”

sort(v) - сортировка элементов вектора по возрастанию. Символ “8” будет последним, поскольку он максимальный. “15” - будет первым, “16” - вторым, так как оба элемента начинаются на одинаковый символ, то будет сравнение по второму символу. И так дальше.

sum(v) - выполнено не будет, так как тип вектора символьный, а не числовой.

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

```
#Набор команд 1
v2 <- c("5",7,12)
v2[2] + v2[3]

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

#Набор команд 3
l4 <- list(z1="6", z2=42, z3="49", z4=126)
l4[[2]] + l4[[4]]
l4[2] + l4[4]
```

Набор команд 1 - вторая команда не будет выполнена, поскольку при инициализации переменной v2 в нее был передан элемент типа char. Вектор может хранить в себе элементы только одного типа, поэтому остальные элементы вектора тоже стали типа char. Суммировать элементы типа char оператором + не получится.

Набор команд 2 - создает датафрейм размером 1x3. Датафрейм может иметь в своем составе разные типы элементов, поэтому в данном случае будет успешно выполнено сложение двух чисел.

Набор команд 3 - создает список элементов. Элементы так же могут быть разных типов. Первое сложение успешно выполняется, поскольку указывается конкретный порядковый номер элемента списка. Следующая команда выполнена не будет, поскольку неправильное обращение к элементу. Правильным вариантом будет l4$z2[1]+l4$z4[1].

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:

1. Числа от 1 до 10000 с инкрементом 372.

```
seq(from = 1, to = 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
```

2. Числа от 1 до 10000 длиной 50.

```
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
```

- 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)
```

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

Данный вектор будет повторяться в том порядке, в котором передан был изначально. В times указано количество повторений.

```
rep(1:5, each=3)
```

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

В данном случае каждый элемент вектора будет повторен сразу друг за другом в количестве раз, указанном в each.