Описательные статистики и выборка данных

Лабораторная работа №1.

Выборка данных

Для полноценной работы с данными, в особенности с основным форматом хранения данных data.frame, необходимо уметь извлекать и группировать данные, наборы данных.

Для тренировок в базовом пакете имеются встроенные наборы данных, например:

- mtcars - информация о характеристиках автомобилей;
- iris - информация о параметрах различных сортов цветка iris;
- swiss - информация о демографических характеристиках различных районов Швейцарии;
  и прочие…..

Полезные функции для работы с данными

1) apply(A, 1, mean) - применение функции к строкам или столбцам матрицы или таблицы (основное назначение - работа с матрциами)
2) sapply(df, mean) - применение функций к элементам списка или столбцам дата фрейма.
3) lapply(df, mean) - применение функций к элементам списка или столбцам дата фрейма. Результат - список.
4) unlist() - перевод из списка в вектор
5) tapply(df, index, function) - применение функции, сгруппировав данные по “факторам” в аргументе index.

6) aggregate(df, index, fun) - аналог tapply, рекомендуется его применение, поскольку игнорируются пропущенные значения. Группирующие фактору должны быть объединены в список list.

7) subset(df,subset,select) - выбрать данные из дата фрейма по условию. В аргументе subset указываются условия для строк, в аргументе select условия для столбцов.

8) df[order(df$age),] - сортировка строк датафрейма по возрасту по возрастанию.

df[desc(order(df$age)),] - по убыванию.

9) y~x - Способ записи группировки данных. Работает с функциями типа aggregate(), графическими функциями boxplot() и прочими:

aggregate(y~x, mean) - расчет среднего значения переменной y, сгруппированной по переменной x.

Важно: x должен быть фактором.

aggregate(y~x+z, mean) - расчет среднего значения переменной y, сгруппированной по переменной x и z.

Справка по функциям

Дополнительную справочную информацию по используемым функциям можно получить, пройдя по ссылке (и найти требуемую функцию с помощью встроенного поисковика браузера):

https://r-analytics.blogspot.com/p/blog-page_06.html
https://aakinshin.net/ru/posts/r-functions/

а также во встроенной справке к функциям. Команды

?имя_функции

help("имя_функции")

Работа с dataframe

Dataframe - основной тип данных, с которым мы будем работать в R. Обычно результаты нашего исследования сохраняются в dataframe следующим образом: каждая строка это - наблюдение, каждый столбец - это переменная. Давайте посмотрим на встроенные в R данные под названием iris.

```
# выполните эту команду чтобы данные отобразились в глобальном окружении
data(iris)

# начнем с общей информации о данных
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# посмотрим на первые пять строк наших данных
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Функция str() вернет вам описание данных: число наблюдение, список переменных и их тип. Мы видим, что в данных 150 наблюдений (цветков Ириса), у каждого из них измерена длина и ширина лепестка и чашелистика, а также указано к какому виду относится каждый цветок. Команда head() позволяет взглянуть на сами данные.

Индексация dataframe

```r
# каждое наблюдение имеет номер строки и столбца, в котором оно находится
iris[1, 4] # такая команда напечатает наблюдение в первой строке в четвертом столбце

# к переменной в данных можно обратиться по ее имени при помощи знака $
iris$Sepal.Length

# или при помощи номера столбца
iris[, 1] # тоже самое, что и iris$Sepal.Length, обратите внимание что мы не указал никакого индекса на месте строк, это озн
ачает, что мы отбираем все строки, но только первый столбец

# чтобы обратиться сразу к нескольким переменным можно использовать вектор имен или индексов
iris[, c(1, 3)]
iris[, c("Sepal.Length", "Petal.Length")] # тоже самое

# чтобы отобрать все переменные кроме некоторой, используйте отрицательную индексацию
iris[, -1] # все кроме первой
iris[, -c(1, 3)] # все кроме первой и третьей
```

Работа с факторами

В заданиях этой недели вы не раз столкнетесь с факторами. Это специальный тип данных в R для хранения номинативных переменных.

```
# В наших данных переменная Species - это фактора с тремя градациями (levels).
str(iris$Species)
```

```
##  Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
# Давайте посмотрим на частоту встречаемости каждой из градации при помощи функции table()
table(iris$Species)
```

```
##
##     setosa versicolor  virginica
##         50         50         50
```

Часто возникает путаница между levels и labels фактора в R. Давайте рассмотрим пример.

```
# Создадим вектор из целых чисел, где каждое число - это один из авторов этого курса. Предположим, это данные о том, сколько уроков создал каждый из авторов. Видим, что каждый из трех авторов сделал по три урока.
v <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
str(v)
```

```
##  num [1:9] 1 1 1 2 2 2 3 3 3
```

```
# Давайте сделаем из вектора фактор при помощи функции factor()
f1 <- factor(v)
str(f1)
```

```
##  Factor w/ 3 levels "1","2","3": 1 1 1 2 2 2 3 3 3
```

```
# Теперь у нашего фактора три уровня Levels: 1 2 3. R считает, что переменная f1 - это фактор, который может принимать только три градации 1, 2 или 3. По умолчанию R создает фактор с числом уровней равным числу уникальных элементов в векторе.

# Для удобства работы с фактором мы можем дать имена нашим градациям, например, Ivan, Tolik, Polina при помощи аргумента labels.
f1 <- factor(v, labels = c("Ivan", "Tolik", "Polina"))
str(f1)
```

```
##  Factor w/ 3 levels "Ivan","Tolik",..: 1 1 1 2 2 2 3 3 3
```

```
table(f1)
```

```
## f1
##   Ivan  Tolik Polina
##      3      3      3
```

```
# Но на практике может возникнуть следующая ситуация, в нашей выборке будут представлены не все возможные градации фактора.
Тогда мы можем указать, какие градации возможны при помощи аргумента levels.
f1 <- factor(v, levels = c(1, 2, 3, 4))
str(f1)
```

```
##  Factor w/ 4 levels "1","2","3","4": 1 1 1 2 2 2 3 3 3
```

```
# таким образом наш фактор теоретически может принимать 4 градации, но в выборке представлены только три. Мы также теперь мо
жем назвать каждую градацию.
f1 <- factor(v, levels = c(1, 2, 3, 4), labels = c("Ivan", "Tolik", "Polina", "Misha"))
table(f1)
```

```
## f1
##   Ivan  Tolik Polina  Misha
##      3      3      3      0
```

```
# теперь R понимает, что Misha есть среди авторов курса, но просто не встречается в этой выборке.
```

Мораль: когда создаете фактор в R иногда важно сразу указать, какие градации он может принимать.

Задание 1

Для данных *mtcars* в базовом пакете R создайте функцию, которая рассчитывает и возвращает среднее значение времени разгона (qsec) для автомобилей, число цилиндров (cyl) у которых не равняется 3 и показатель количества миль на галлон топлива (mpg) больше 20.

**Вы можете вызвать справку по встроенным в пакеты наборам данных с подробным их описанием.

Задание 2

При помощи функции aggregate рассчитайте стандартное отклонение, среднее значение, медиану переменной hp (лошадиные силы) и переменной disp (вместимости двигателя) у машин с автоматической и ручной коробкой передач.

Полученные результаты (результаты выполнения функции aggregate) сохраните в переменную descriptions_stat.

Задание 3

Воспользуемся встроенными данными airquality. В новую переменную сохраните новый набор (выборку) данных (функция subset), оставив наблюдения только для месяцев 7, 8 и 9.

При помощи функции aggregate рассчитайте количество непропущенных наблюдений по переменной Ozone в 7, 8 и 9 месяце. Для определения количества наблюдений используйте функцию length(). (Функция aggregate по умолчанию игнорирует пропущенные данные, производя расчеты).

Результат выполнения функции aggregate сохраните в переменную result.

Подсказки:

1. Не забудьте сделать subset, чтобы отобрать наблюдения только по нужным месяцам, вам может пригодиться следующая конструкция:

```
> x <- 5
> x %in% c(3, 4, 5)
[1] TRUE
```

2. Для подсчета числа непропущенных наблюдений воспользуйтесь записью с помощью формулы, при которой пропущенные значения не учитываются:

aggregate(y ~ x + z , data, FUN)

Задание 4

Для данных *mtcars* в базовом пакете R с помощью функции table() (изучите справку по ней) получите информацию:

1. Количество автомобилей с 4,6, и 8 цилиндрами.
2. Как сгруппированы автомобили по количеству цилиндров и типу коробки передач.
3. Как сгруппированы автомобили по количеству цилиндров, типу коробки передач и количеству передач в КПП.
4. Полученные таблицы из пунктов 1-3 сохраните в текстовые документы с расширением .txt либо .csv.