Lesson #1

Ilya Putilin

2025-09-18

Типы данных

Непрерывные данные

В R все даннные представлют в виде векторов (даже при работе с табличными данными, к конкретным столбцам, строкам или элементам таблицы мы обращаемся как к векторам)

Создадим два вектора: x и w, в которые запишем какое-то количество чисел, в данном случае 7

Общая схема создания переменной:

```
переменная <- что-либо
```

где “<-” - оператор присвоения

```
x <- c(174, 162, 188, 192, 165, 168, 172.5)

w <- c(69, 68, 93, 87, 59, 82, 72)
```

Функция c() позволяет комбинировать отдельные элементы в вектор

Общий вид любой функции:

```
функция(аргумент1 = значение, аргумент2 = значение...)
```

Для получения справки о ЛЮБОЙ функции можно воспользоваться “?”, например: ?c, ?str, ?is.numeric

Правила для выбора названия для переменной:

1. Использовать латинские символы, цифры и точку (переменная не должна начинаться с цифры)
2. Помнить, что R чувствителен к регистру (X и x - разные символы -> разные переменные)
3. Не давать уже имеющиеся имена других переменных, функций и т.д. (только в случае, если вы хотите переписать переменную)

Также, стоит помнить, что результат выполнения любой функции можно (и нужно) записывать в в переменные, иначе результат выполнения функции будет просто выведен в консоли

Проверим, что из себя представляет переменная x с помощью функции str()

```
str(x)
```

```
##  num [1:7] 174 162 188 192 165 ...
```

Видим, что переменная x хранит в себе числовой (num) вектор длинной 7 ([1:7])

Помимо функции str() ниже будет ряд функций, которые могут быть полезны (особенно при работе с циклами, но об этом не в этом занятии)

```
ls()          # Возвращает список уже имеющихся переменных
```

```
## [1] "w" "x"
```

```
min(x)        # Возвращается минимальное значение вектора
```

```
## [1] 162
```

```
max(x)        # Возвращает максимальное значение вектора
```

```
## [1] 192
```

```
is.numeric(x)  # Проверяет, является ли переменная (вектор) числовой
```

```
## [1] TRUE
```

```
is.vector(x)    # Проверяет, является ли переменная вектором
```

```
## [1] TRUE
```

```
is.character(x) # Проверяет, является ли переменная (вектор) буквенной
```

```
## [1] FALSE
```

Помимо ручного ввода данных можно воспользоваться генерацией чисел

```
y <- 1:10  # Генерация вектора чисел от 1 до 10

y <- seq(from = 1, to = 10, by = 0.1) # Функция seq() создаст последовательность
                                      # чисел от 0 до 10 с шагом 0.1
                                      # Также, функцию можно использовать не явно
                                      # указывая параметры: seq(1, 10, 0.1)

y <- seq(100, 5, -5) # Для создания последовательности числе в порядке уменьшения
                     # параметр by должен быть отрицательным
```

Помимо простых прогрессий R позволяет генерировать числа, подчиняющиеся тому или иному закону распределения. В частности, создадим переменную x1, в которую запишем 10000 чисел, подчиняющихся закону нормального распределения

```
x1 <- rnorm(10000)
```

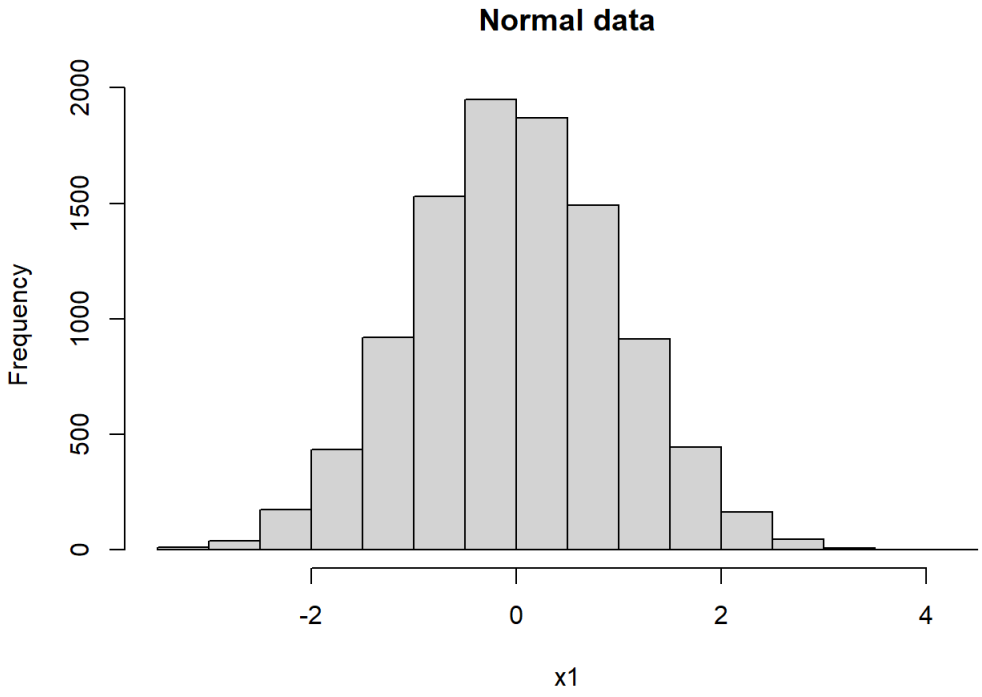
Также, создадим переменную x2, в которую запишем 10000 чисел, подчиняющихся закону равномерного распределения

```
x2 <- runif(10000)
```

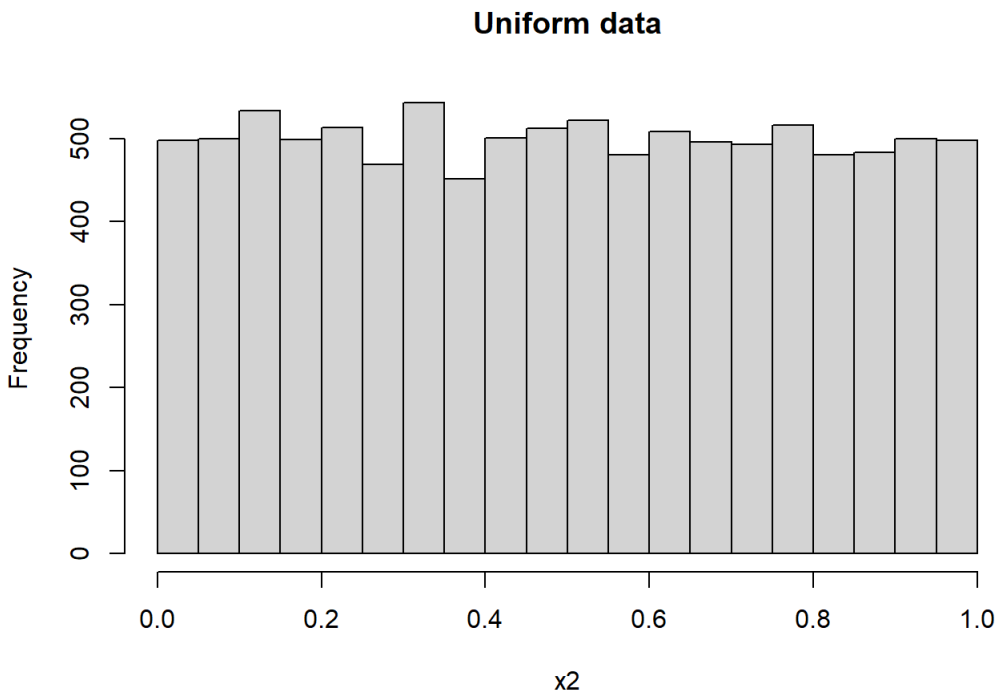
На данном этапе обучения мы не будем говорить о различных способах и критериях проверки типа распределения, однако, в рамках знакомства с базовыми функциями R и с базовой графикой R следует упомянуть функцию hist(), которая будет отрисовывать гистограмму распределения данных. Гистограмма является самым простейшим и наименее точным методом проверки типа распределения

Построим гистограммы векторов x1 и x2

```
hist(x1, main = 'Normal data')
```




```
hist(x2, main = 'Uniform data')
```



Чтобы отрисовать два графика в одном поле можно воспользоваться функцией par()

```
par(mfrow=c(1,2))    # Два столбца, одна строка
par(mfrow=c(2,1))    # Две строки, один столбец
```

После запуска функции par() нужно запустить функции hist() ещё раз Для возврата к обратным настройкам отрисовки графики выполните:

```
par(mfrow=c(1,1))
```

Шкальные данные

Перепишем вектор y, записав в него какой-то случайный набор школьных оценок

```
y <- c(2,3,4,5,3,4,5,3,3,4,2)
```

R будет считать, что он хранит в себе обычные числа (можно убедиться в этом воспользовавшись функцией str()), что неверно. Мы знаем, что этим числам предначертано быть шкалой, следовательно, числовой вектор нужно перевести в факторные:

```
y <- as.factor(y)  # также, рекомендую познакомиться с другими подобными функциями
                   # для перевода одного типа данных в другой, в частности:
                   #    as.numeric(), as.character(), as.vector() и т.д.
```

Можно видеть, что переменная y стала факторной. Содержит в себе 4 уровня: “2”, “3”, “4” и “5”. Для того, чтобы узнать это можно воспользоваться функцией str().

Также, для “вытаскивания” уровней из переменных, чтобы использовать их как параметры для других функций можно использовать функцию levels(). Чтобы получить список элементов, разделённых на уровни можно воспользоваться функцией as.numeric()

```
levels(y)
```

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

```
as.numeric(y)
```

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

Помимо использования уже имеющихся в данных шкал, можно создавать собственные, буквально разрезая, например, неприрывные данные на несколько честей. Для этого воспользуемся функцией cut()

```
xx <- cut(x, 3, labels = 1:3)  # Параметр labels использовался для того, чтобы переменная
                               # выглядела более "причёсанной". Чтобы понять, для чего это
                               # было сделано и как сделать по-другому, запустите '?cut'
                               #                                (ковычки уберите)
```

Категориальные данные

Создадим категориальную буквенную переменную длинной в 7 элементов

```
gender <- c('мужчина', "женщина",  # Буквенные переменные следует заключать в ковычки
            'мужчина', "женщина",  # иначе вместо нормального результата выполнения функции
            'мужчина', "женщина",  # c() R выдаст ошибку
            "мужчина")
```

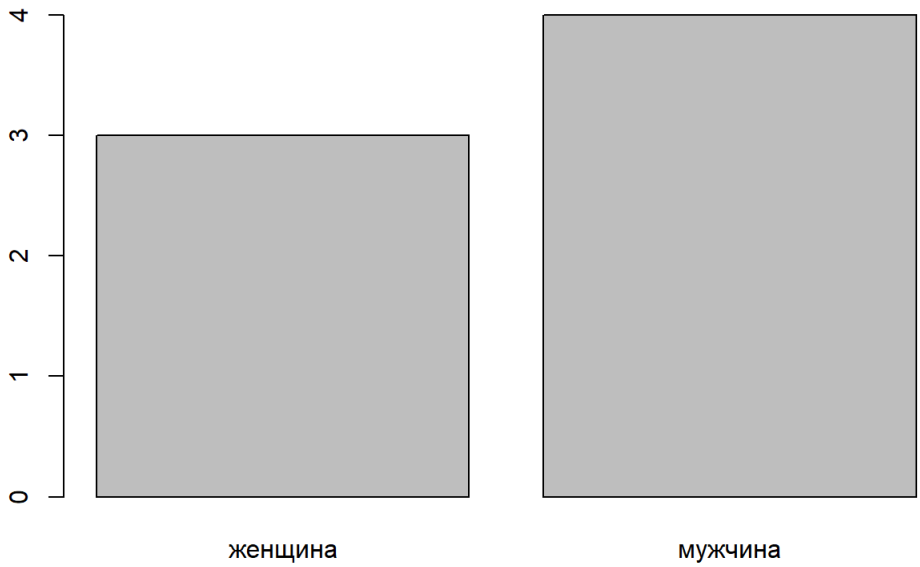
Для дальнейшего удобства любую категориальную переменную лучше сразу делать факторной

```
gender <- as.factor(gender)
```

Базовая графика

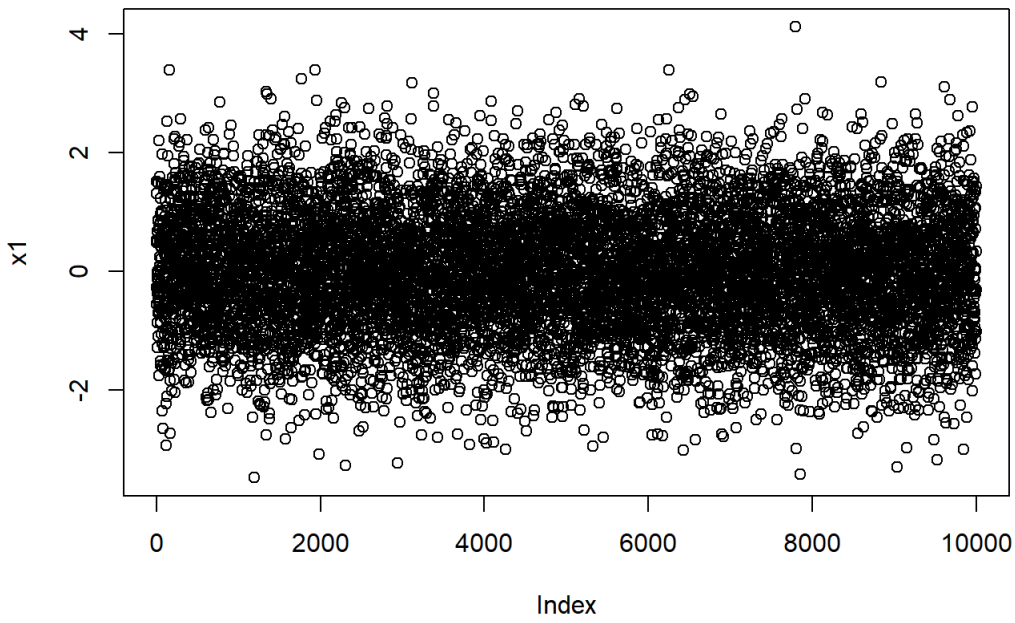
Практически всю базовую графику в R можно выполнять с помощью функции plot(). Обычно, эта функция довольно “умная” сама по себе и, в зависимости от того, какие данные вы зададите ей на отрисовку, она сама подберёт нужный тип графика. В частности, если запустить:

```
plot(gender)
```



можно видеть, что у вас автоматически отрисуется столбчатая диаграмма (barplot) по количеству “мужчин” и “женщин” в переменной gender. Или же если запустить

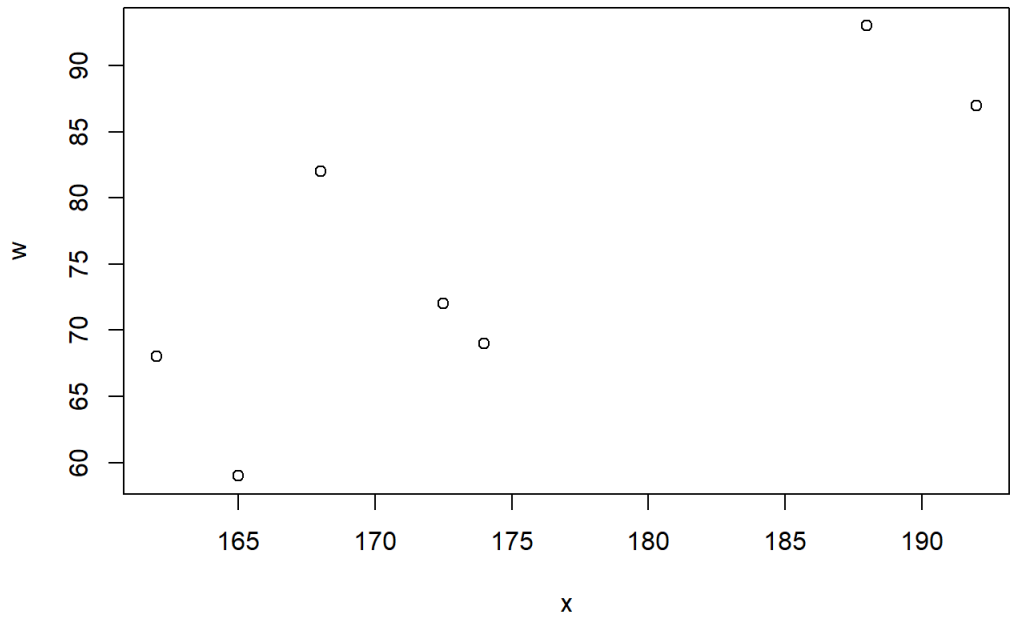
```
plot(x1)
```



то по оси X будет порядковый номер элемента в переменной x1, а по оси Y - его значение

Отрисуем график, в котором по оси X - переменная x, по оси Y - переменная w

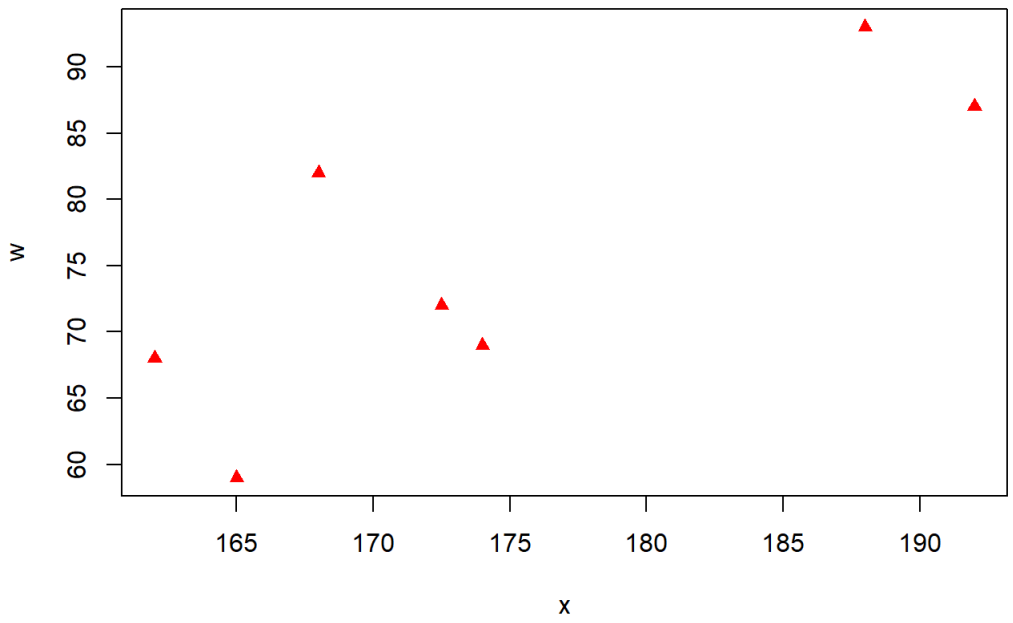
```
plot(x = x, y = w)  # Здесь некоторые параметры также можно указывать неявно: plot(x, w)
```



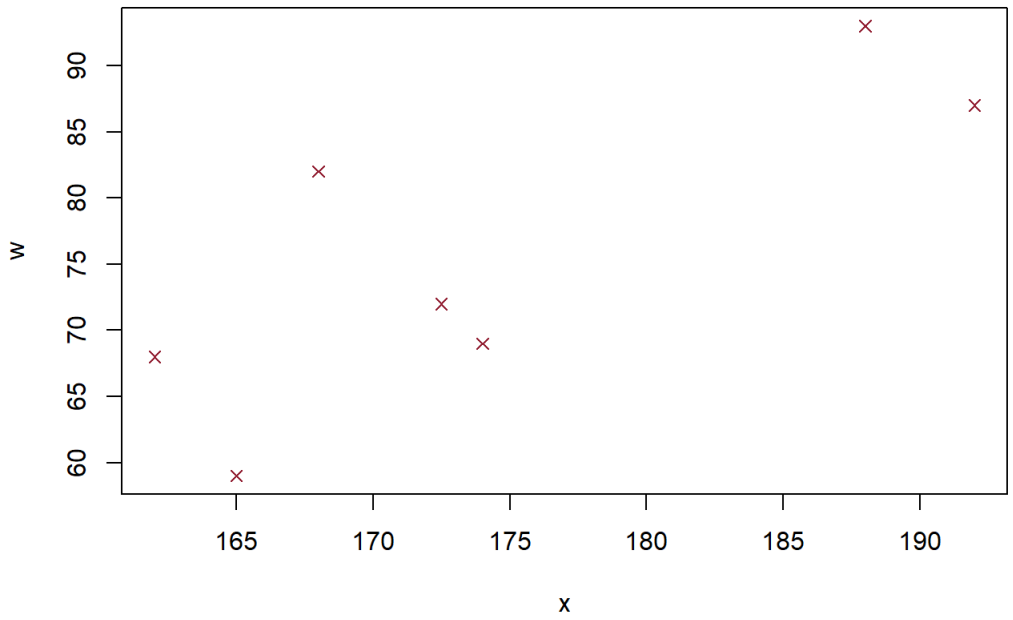
На данном точечном графике можно менять цвет точек, тип точек, названия осей и тд (см. ?plot)

Например, цвета (параметр col) можно указывать явно: английскими словами, RGB-кодом, порядковым номером (внутренняя память R хранит в себе много интересного) Тип точки меняется порядковым номером (параметр pch)

```
plot(x, w, col = 'red', pch = 17)
```

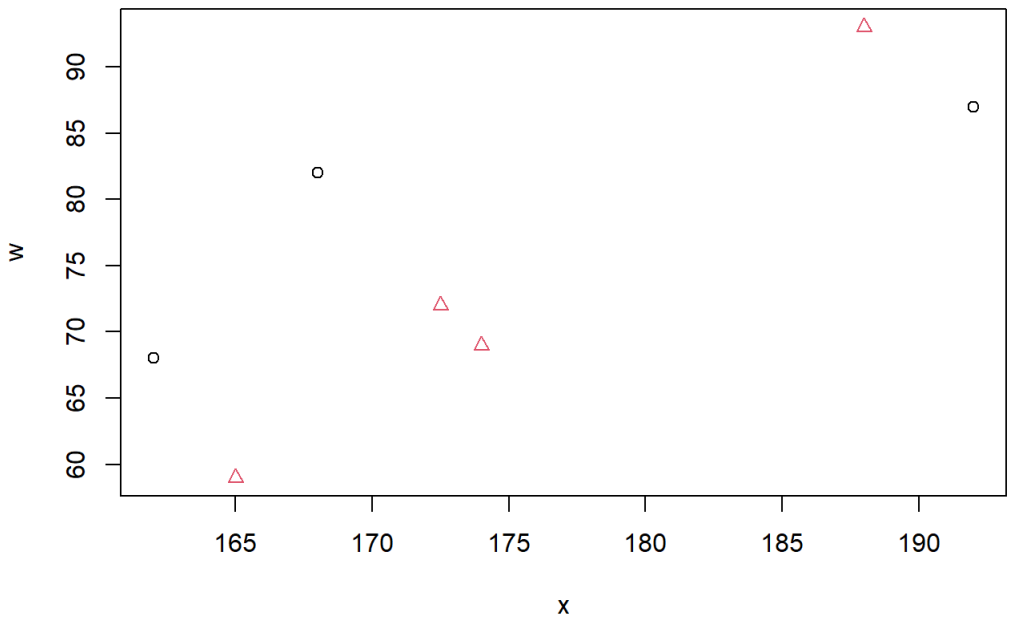



```
plot(x, w, col = '#932134', pch = 4)
```



Для того, чтобы на этом графике цветом и формой точки отобразить переменную gender можновоспользоваться функцией as.numeric() (см. выше)

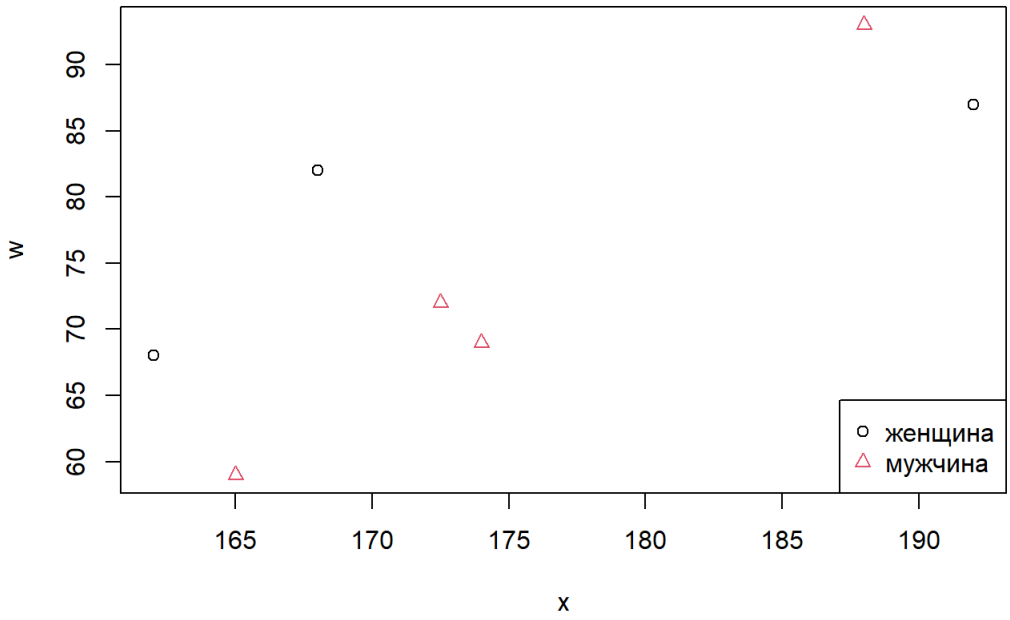
```
plot(x, w,
    col = as.numeric(gender),
    pch = as.numeric(gender))
```



В данном случае понятно, какие из получившихся точек отражают мужчин, а какие женщин, т.к. в переменной gender всего 7 элементов. Однако, если бы данных было гораздо больше, нам потребовалась бы легента - функция legend()

```
plot(x, w,
     col = as.numeric(gender),
     pch = as.numeric(gender))

legend('bottomright',          # Расположение легенды на графике (см '?legend')
       col = 1:2,              # Цвет точки на легенде. Почему 1 и 2 - см 'as.numeric(gender)'
       pch = 1:2,              # Форма точки на легенде. Почему 1 и 2 - см 'as.numeric(gender)'
       legend = levels(gender))  # В переменную legend записываем конкретные значения уровней фактора с помощью функции levels()
```



Домашнее задание

Для того, чтобы выполнить домашнее задание придётся чуть чуть заглянуть в будущее, и познакомиться с таким типом данных в R, как data.frame. Тут нет ничего страшного, просто ниже будет объяснено, как в переменную записать конкретную встроенную в R таблицу и как к конкретным столбцам обратиться, чтобы использовать их в качестве векторов.

Прежде, чем приступить к выполнению домашнего задания нужно записать встроенную (!) таблицу “mtcars” в какую-либо переменную. Назовём её df. Формально, к этой таблице можно обращаться напрямую, без записи в переменную, однако хочется на данные взглянуть своими глазами а не просто в консоли

```
df <- mtcars
```

Для того, чтобы понять что вообще представляет из себя этот набор данных, сколько столбцов и что каждый из них значит можно воспользоваться ‘?mtcars’

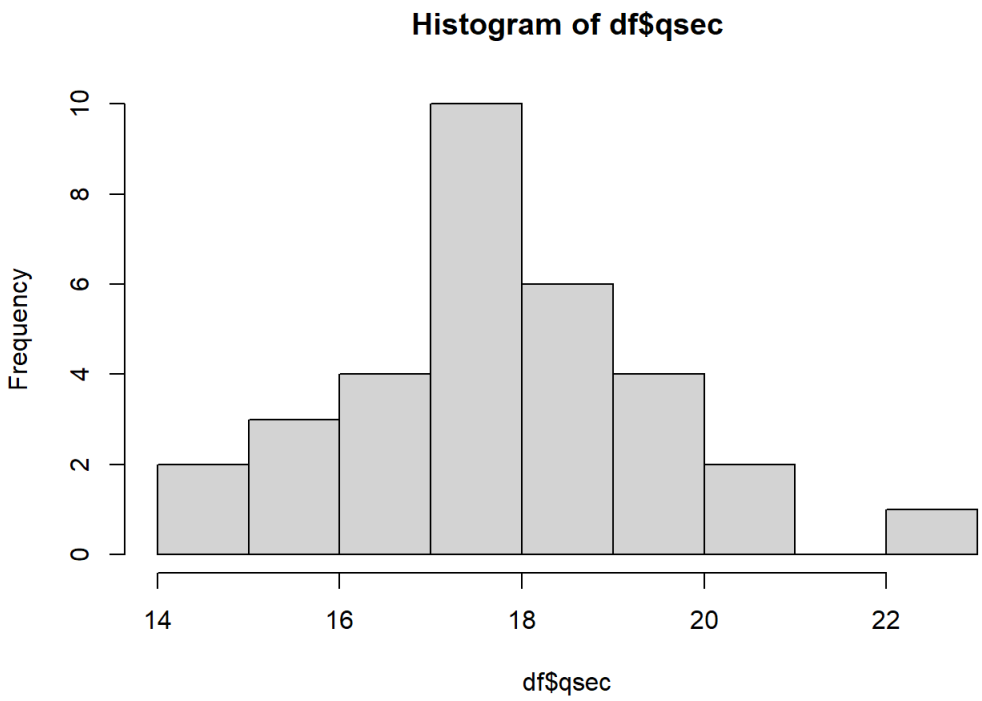
Для того, чтобы обратиться к конкретному столбцу как к вектору нужно использовать следующую нотацию:

```
переменная$столбец
```

Символ ‘$’ появляется при нажатии Shift + 4 в английской раскладке

Для примера, продемонстрирую как отрисовать гистограмму столбца qseq из этой таблицы

```
hist(df$qsec) # В сравнении с, например, hist(x1) ничего супер сложного
```



Если при работе с таблицей и с указанной выше нотацией возникают сложности, можно запустить следующие 5 строк, чтобы облегчить себе жизнь. Эти строки запишут необходимые столбцы таблицы в отдельные переменные, которые будут храниться в памяти R

```
mpg <- mtcars$mpg
disp <- mtcars$disp
hp <- mtcars$hp
cyl <- mtcars$cyl
drat <- mtcars$drat
```

Непосредственно, задание

```
#1 Построить гистограммы mpg, disp, hp
  Сделать график цветным, поиграться с nclass
  Оси назвать по-русски. Ось Y назвать "частота"
#2 Построить точечные графики
  mpg VS hp, mpg VS drat
  Цветом обозначить cyl, disp (факторный!)
  Добавить легенду
```

Hints

disp изначально представляет из себя неприрывные данные. Воспользуйтесь cut(). Информацию про то, как поменять цвет графика, названия осей и т.д. можно найти запустив строки: ?hist, ?plot