

ЛАБОРАТОРНАЯ РАБОТА 1. ОСНОВЫ РАБОТЫ С R. ОБРАБОТКА СТАТИСТИЧЕСКИХ ДАННЫХ

Цель работы

Ознакомиться с интерфейсом RStudio, научиться работать в режиме консоли и путем написания скриптов, а также подключать внешние пакеты, изучить основные методы обработки статистических данных.

Задание

1. Загрузить данные для своего варианта в переменную-вектор.
2. Получить справочную информацию по своим данным, посмотреть их содержимое.
3. Проверить, есть ли среди данных пропуски.
4. Создать новую переменную-вектор, в которой будут 1, если значение в исходном векторе больше среднего, и -1, если значение переменной меньше среднего, и 0, если значение равно среднему.
5. Вывести описательную статистику.
6. Построить графики абсолютных частот и плотности распределения.

Указания к выполнению работы

Консольный режим

После запуска RStudio пользователь попадает в консольный режим работы (рис. 1). Любая команда, написанная пользователем, будет сразу выполнена R по нажатию Enter.

Рассмотрим основные выражения в R: числа, строки и логические переменные.

Можно использовать R как калькулятор, например:

```
> 1 + 1
[1] 2
> 6 * 7
[1] 42
> sqrt(16)
[1] 4
```

Результат сразу появится в консоли.

Строки печатаются в кавычках: двойных или одинарных:

```
> "Hello world!"
[1] "Hello world!"
```

```
> 'Hello world!'
[1] 'Hello world!'
```

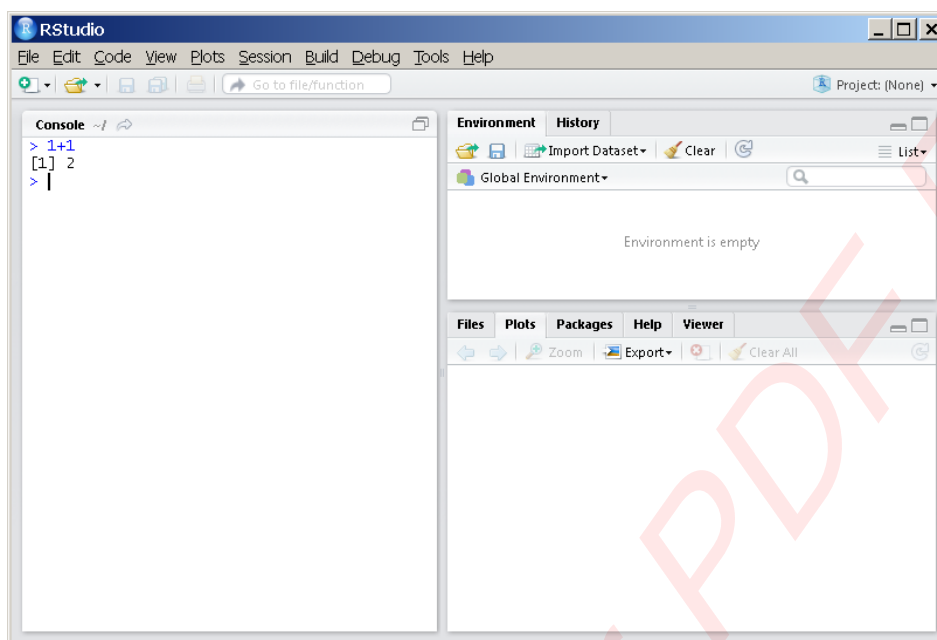


Рис. 1. Консольный режим в RStudio

Логические выражения возвращают TRUE или FALSE:

```
> 3 < 4
[1] TRUE
```

Чтобы сравнить два выражения, используется двойной знак равенства:

```
> 2 + 2 == 5
[1] FALSE
```

Как и в других языках программирования, можно сохранять значения в переменную. Сохраним 42 в переменную x:

```
> x <- 42
```

И в обратную сторону:

```
> 5 -> x
```

Можно распечатать значение переменной в любое время, просто набрав ее имя в консоли. Попробуем напечатать текущее значение x:

```
> x
[1] 42
```

Можно так же повторно назначить любое значение переменной в любое время.

R чувствителен к регистру: переменные x и X — это разные переменные:

```
> X
[1] 5
```

Чтобы вызвать функцию, нужно обратиться к ней по имени, указав в скобках нужные аргументы. Например, функция суммы:

```
> sum(1, 3, 5)
[1] 9
```

Получить помощь по функции можно командой `help(functionname)` или `?functionname`.

В правом нижнем углу на вкладке Help появится справка (рис. 2):

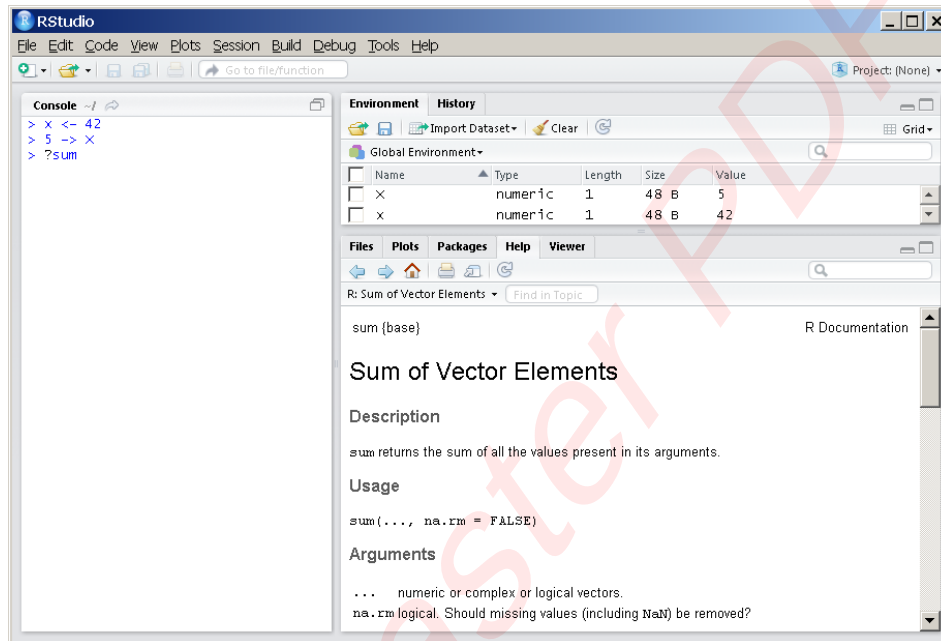


Рис. 2. Справка по функции `sum`

Зададим вектор с помощью функции `c` (сокр. от англ. **C**ombine):

```
> y <- c(-3, 2, NA, 5)
> y
[1] -3 2 NA 5
```

NA — это пропущенное наблюдение (от англ. **N**ot **A**vailable). Его не следует путать с NaN (**N**ot a **N**umber — «не число», неопределенность):

```
> 0/0
[1] NaN
```

Попробуем просуммировать элементы вектора `y`:

```
> sum(y)
[1] NA
```

Необязательным аргументом функции `sum` является `na.rm` (сокр. от англ. **R**emove **N**A), по умолчанию равный `FALSE`.

Если указать для него значение «истина», то функция суммы будет складывать все элементы вектора, исключая пропущенные:

```
> sum(y, na.rm = TRUE)
[1] 4
```

Последовательность чисел можно задать двумя способами: `start:end` либо функцией `seq()`:

```
> 5:9
[1] 5 6 7 8 9
> seq(5,9)
[1] 5 6 7 8 9
> seq(10,50, by = 10)
[1] 10 20 30 40 50
```

Обращаться к элементам вектора можно, используя квадратные скобки:

```
> sentence <- c('mack', 'the', 'knife')
> sentence[3]
[1] "knife"
> sentence[c(1,3)]
[1] "mack" "knife"
```

Либо можно задать элементам вектора имена:

```
> ranks <- 1:3
> names(ranks) <- c("first", "second", "third")
> ranks
  first second  third
    1      2      3
> ranks["first"]
first
  1
```

Написание скриптов

В R удобнее писать не по одной команде, а сразу целый набор команд и потом запускать их все на выполнение. Для этого нужны скрипты.

Чтобы создать скрипт, следует выбрать `File -> New File -> R Script`. Откроется новая область, в которой можно писать команды. Комментарии, которые не будут выполнять R, пишутся со знака `#` (рис. 3).

По нажатию `Enter` команды в скрипте выполняться не будут, а будет лишь осуществлен переход на новую строку.

Чтобы выполнить команду в режиме скрипта, следует поставить курсор на нужную строку и нажать `Ctrl+Enter`. Если команда занимает более одной строки, то необходимо ставить знак `+` в конце каждой строки. Команда выполняется построчно, в каждой строке требуется нажимать `Ctrl+Enter`, либо выделить всю команду целиком и нажать `Ctrl+Enter` один раз.

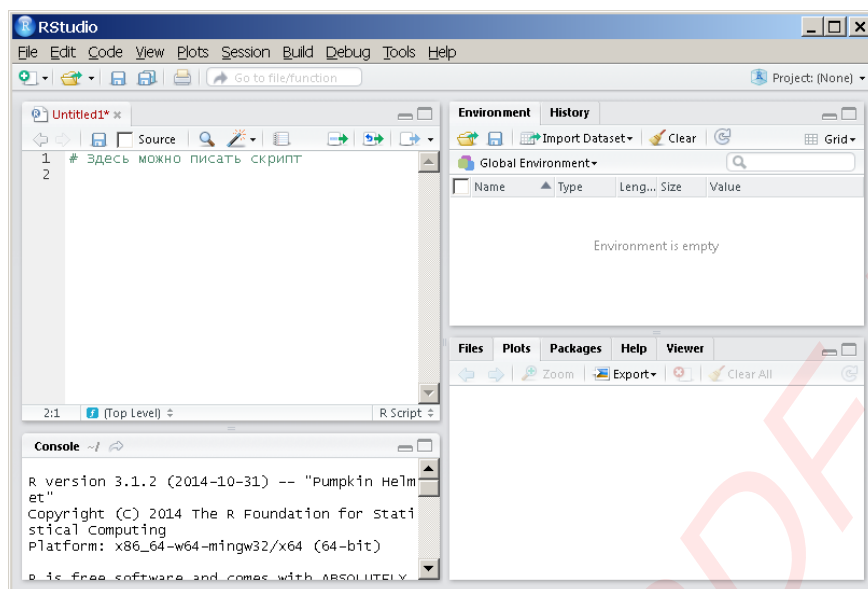


Рис. 3. Создание нового скрипта

R может подсказывать, какие команды доступны, если начать вводить первые символы и нажать либо Tab, либо Ctrl+Enter.

В основном в R работают с наборами данных. Такая структура носит в R название `data.frame` и представляет собой таблицу, в которой каждый столбец — это некоторая переменная, а каждая строка — это одно наблюдение.

Создадим в режиме скрипта `data.frame`. Пусть имеются наблюдения за ростом и весом некоторых людей. Зададим два вектора:

```
rost <- c(160, 175, 155, 190, NA)
ves <- c(NA, 70, 48, 85, 60)
```

И объединим их в набор данных, который поместим в переменную `df`, а затем выведем на экран:

```
df <- data.frame(rost, ves)
df
```

В консоли получим следующую таблицу:

```
rost ves
1 160 NA
2 175 70
3 155 48
4 190 85
5 NA 60
```

Обращаться к конкретным наблюдениям `df` можно, используя квадратные скобки:

```
> df[3,1]
[1] 155
```

Обращаться к переменным можно, используя знак `$` или указывая столбец с пропуском номера строки:

```
> df$rost
[1] 160 175 155 190 NA
```

или

```
> df[,1]
[1] 160 175 155 190 NA
```

Обращаться к наблюдениям можно, указывая конкретную строку и пропуская номер столбца:

```
> df[4,]
  rost ves
4  190  85
```

Основные описательные статистики (среднее, стандартное отклонение и медиану) можно получить с помощью функций `mean`, `sd` и `median`:

```
mean(df$rost, na.rm = T)
[1] 170
sd(df$rost, na.rm = T)
[1] 15.81139
median(df$rost, na.rm = T)
[1] 167.5
```

Можно сохранить скрипт, нажав `File -> Save`. При первом сохранении R предложит выбрать кодировку. Рекомендуется указать UTF-8 (рис. 4), чтобы русские буквы (например, в комментариях) отображались корректно. Затем необходимо выбрать директорию и задать имя файла, который будет сохранен с расширением *.R.

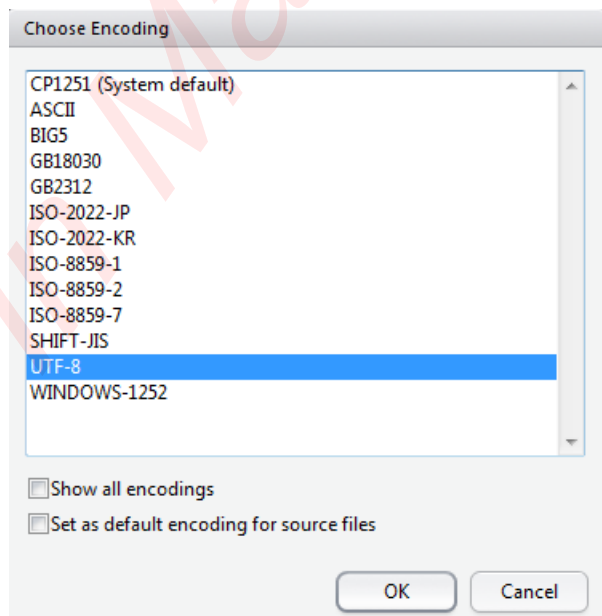


Рис. 4. Выбор кодировки при сохранении

Установка пакетов

В R существует базовый набор пакетов (библиотек), содержащих самые необходимые функции [2]. Для реализации задач эконометрики требуются дополнительные пакеты.

Для работы с внешними пакетами необходимо выполнить два действия — установку и подключение нужного пакета. Установка необходимо выполнить один раз, а подключать — в каждой рабочей сессии.

Для установки пакетов существует функция `install.packages`. Также автоматически будут доустановлены связанные пакеты.

Для подключения установленного пакета следует воспользоваться функцией `library` [1].

Например, установим следующие пакеты:

`psych` — содержит функции для расчета описательных статистик;

`dplyr` — содержит функции для работы с `data.frame`;

`ggplot2` — самый мощный пакет для построения красивых графиков, диаграмм, карт и т. д.

```
install.packages(c("psych", "dplyr", "ggplot2"))
```

Прямым сообщением об ошибке установки является только слово `Error`, появляющееся в консоли. Все остальные сообщения `Warning` являются просто предупреждениями о чем-либо.

Для того чтобы определить, какие пакеты нужны для работы, можно воспользоваться поиском в сети Интернет, задав вопрос на английском языке. Например, чтобы найти, в каком пакете находится алгоритм Левенберга–Марквардта для расчета нелинейного МНК, можно набрать в поиске «levenberg-marquardt algorithm in r» и первой же ссылкой будет пакет `minpack.lm` в R.

Для выполнения данной работы понадобится подключить следующие пакеты:

```
library("psych") # описательные статистики
library("lmtest") # тестирование гипотез в линейных моделях
library("ggplot2") # графики
library("dplyr") # манипуляции с данными
library("MASS") # подгонка распределений
```

Получим, например, описание набора данных по автомобилям `cars` командой:

```
help(cars)
```

Результат выполнения команды (в правом нижнем углу на вкладке Help) показан на рисунке 5.

В этом наборе данных 50 наблюдений и две переменных (скорость, миль/час и длина тормозного пути в футах).

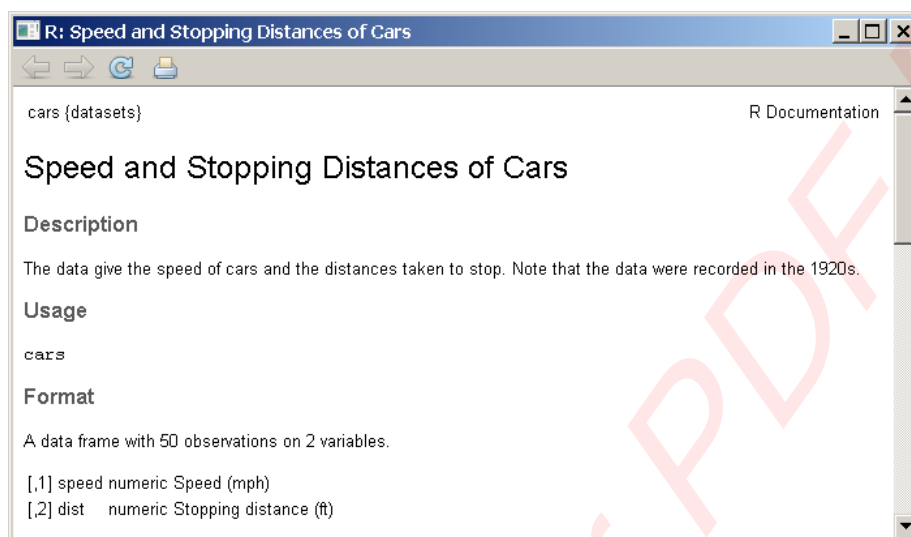



Рис. 5. Справка по набору данных cars

Поместим в переменную `d` встроенный в R набор данных по автомобилям¹:

```
d <- cars # этот набор данных находится в базовом пакете datasets
```

Теперь `d` имеет тип данных `data.frame` (набор данных), в чем можно удостовериться, посмотрев в правом верхнем углу окна таблицу среды Environment (рис. 6).

Для этого должен быть выбран режим Grid. С помощью кнопки  можно просмотреть содержимое набора данных в виде таблицы.

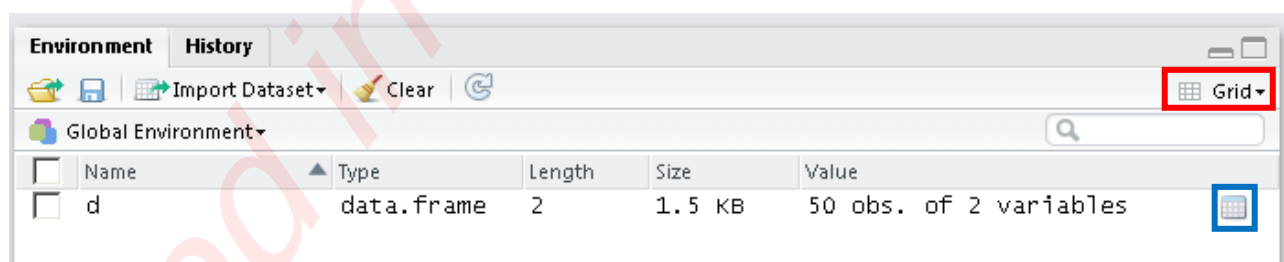


Рис. 6. Описание набора данных `d` в таблице среды Environment

Следующей командой можно посмотреть на этот набор данных, в результате чего будут перечислены все переменные и типы данных:

```
glimpse(d) # функция из пакета dplyr
```

¹ В дальнейшем можно работать как с переменной `d`, так и непосредственно с `cars`, но в последнем случае есть риск испортить исходные данные.

Результат выполнения команды появится в консоли:

```
> d <- cars
> glimpse(d)
Observations: 50
Variables: 2
$ speed (dbl) 4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13, ...
$ dist (dbl) 2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 24, 28, ...
```

Переменные `speed` и `dist` имеют тип данных `dbl` (`double`) и содержат по 50 наблюдений. Для других типов данных используются следующие сокращения: `chr` (`character/string`), `int` (`integer`), `fctr` (`factor`), `tims` (`time`), `lg1` (`logical`).

Посмотрим на первые шесть наблюдений набора данных `d`:

```
> head(d) # функция из базового пакета utils
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

и последние шесть наблюдений:

```
> tail(d) # функция из базового пакета utils
  speed dist
45    23   54
46    24   70
47    24   92
48    24   93
49    24  120
50    25   85
```

Получим таблицу с описательными статистиками: среднее, мода, медиана, стандартное отклонение, минимум/максимум, асимметрия, эксцесс и т. д.:

```
> describe(d) # функция из пакета psych
  vars n mean  sd median trimmed  mad min max range skew kurtosis
speed  1 50 15.40 5.29   15  15.47  5.93  4 25  21 -0.11  -0.67
dist   2 50 42.98 25.77   36  40.88 23.72  2 120 118  0.76   0.12
se
speed 0.75
dist  3.64
```

Построим гистограмму абсолютных частот для переменной `dist` (длины тормозного пути).

Воспользуемся функцией `qplot`, задав источник данных `d` (аргумент `data`), переменную для построения графика (`dist`), подпи-

шем оси (параметры функции `xlab` и `ylab`) и название графика (параметр `main`):

```
# функция из пакета ggplot2  
qplot(data=d, dist, xlab="Длина тормозного пути (футы)", ylab="Число автомобилей", main="Данные по автомобилям 1920х")
```

Результат выполнения данной функции показан на рисунке 7.

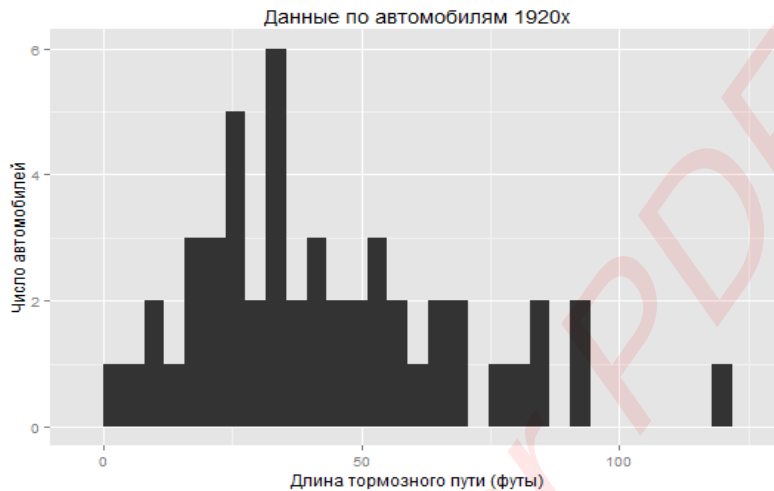


Рис. 7. Гистограмма абсолютных частот для переменной `dist`

Можно построить также гистограмму плотности распределения (рис. 8):

```
# функция из базового пакета graphics  
hist(d$dist, probability = TRUE, col="grey")
```

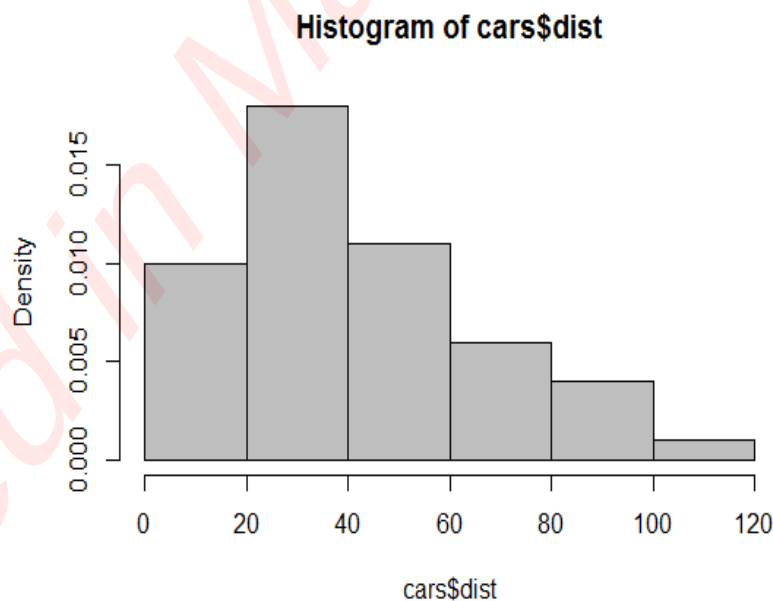
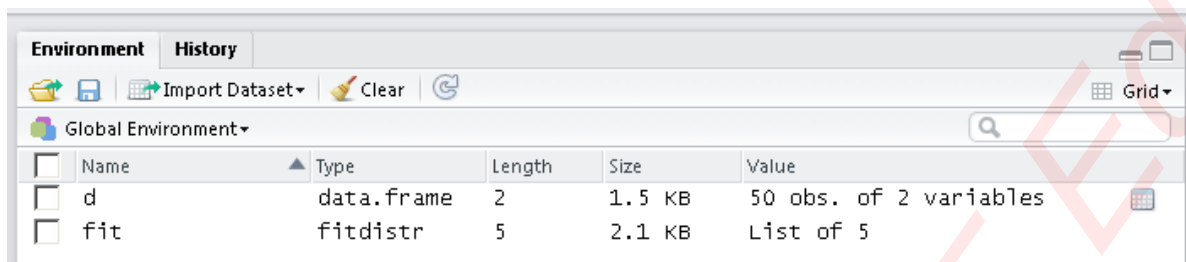


Рис. 8. Гистограмма плотности распределения для переменной `dist`

Подгоним плотность распределения Вейбулла, поместив результат (оценки параметров распределения) в переменную `fit`:

```
fit <- fitdistr(d$dist, "weibull") # функция из пакета MASS
```

Переменная `fit` теперь представляет собой список (List) из 5 элементов, что можно увидеть в Environment (рис. 9).



Name	Type	Length	Size	Value
d	data.frame	2	1.5 KB	50 obs. of 2 variables
fit	fitdistr	5	2.1 KB	List of 5

Рис. 9. Описание переменной `fit` в таблице среды Environment

Доступ к элементам списка можно получить через значок доллара `$`.

Оценки двух параметров распределения Вейбулла были рассчитаны методом максимального правдоподобия.

Посмотрим их, обратившись к элементу списка `fit`:

```
> fit$estimate
  shape    scale
1.72371 48.15234
```

Покажем на том же графике теоретическую плотность распределения Вейбулла (рис. 10):

```
xvals <- seq(0, 120, .20) # значения по оси абсцисс от 0 до 120 с шагом 0,2
lines(xvals, dweibull(xvals, shape=fit$estimate[1],
scale=fit$estimate[2]))
```

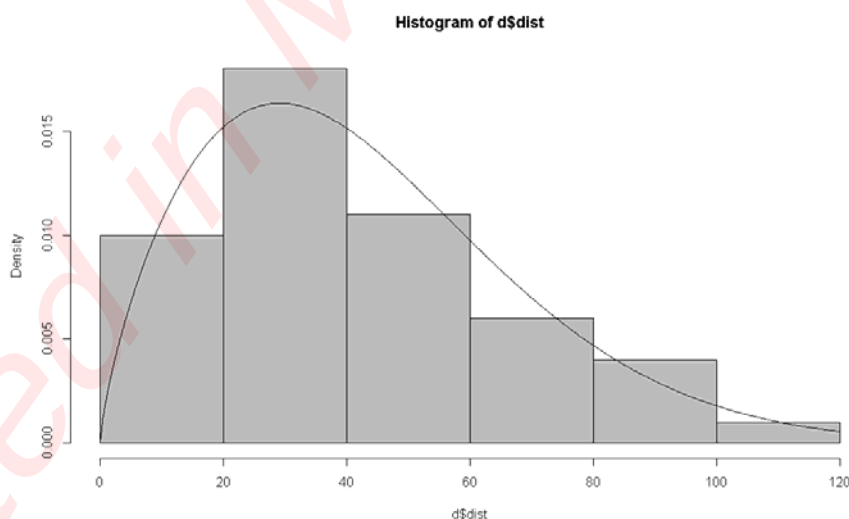


Рис. 10. Гистограмма распределения переменной `dist`

Первый аргумент функции `lines` — это значения по оси абсцисс, на основе которых будет построен график.

Далее указывается функция плотности `dweibull`. Для нее нужно указать значения аргумента для расчета и значения двух параметров распределения: коэффициент формы (`shape`) и масштаба (`scale`).

Контрольные вопросы

1. Как найти среднее выборочное значение и стандартное отклонение?
2. Как проверить наличие пропусков в исходных данных?
3. Каким образом функция `qplot` разбивает выборку на интервалы?