

Дмитрий Храмов



Сбор данных в Интернете

на языке R



УДК 004.738.5:004.438R
ББК 32.971.353

X89

Храмов Д. А.
X89 Сбор данных в Интернете на языке R. – М.: ДМК Пресс, 2017. – 280 с.: ил.

ISBN 978-5-97060-459-5

Всё, что регистрирует человек и созданные им машины, может считаться данными. Фиксируя новое и переводя архивы в цифровую форму, мы с каждым днём производим всё больше данных. Но гораздо чаще случается так, что данные разбросаны по всемирной сети на многочисленных страницах онлайн-магазинов, заметках в социальных сетях, логах серверов и т. п. Прежде чем начать работать с такими данными, их необходимо собрать и сохранить в пригодном для анализа виде. Решению этих вопросов и посвящена данная книга.

Основной материал книги разделён на две части. В первой части дано краткое введение в R – описание среды разработки, языка и основных пакетов-расширений. Вторая часть посвящена непосредственно сбору данных: работе с открытыми данными, извлечению данных из веб-страниц и из социальных сетей. Также рассмотрены необходимые технические вопросы: протокол HTTP, функции импорта данных различных форматов и регулярные выражения. Завершается рассказ созданием карт на основе собранных данных.

Издание предназначено специалистам по анализу данных, а также программистам, интересующимся сбором данных в Интернете.

УДК 004.738.5:004.438R
ББК 32.971.353

ISBN 978-5-97060-459-5

© Храмов Д.А., 2016
© Оформление, издание, ДМК Пресс, 2017

Содержание

Введение	11
Кто и зачем собирает данные.....	11
Почему R?	12
Как устроена эта книга	13
Обратная связь	13
ЧАСТЬ I. ПРОГРАММИРОВАНИЕ НА R	14
Глава 1. Знакомство с R	15
Установка	15
Работа в среде RGui	17
Справка	22
Глава 2. Скаляры, векторы и матрицы	24
Арифметические операции и присваивание	24
Имена	25
Простые типы данных.....	26
Числа	26
Символьный тип.....	28
Логический тип	30
Векторы	31
Векторизация и логическая индексация	36
Матрицы и массивы	39
Резюме	41
Глава 3. Списки и таблицы	42
Списки	42
Таблицы.....	45
Функции, применяемые к составным данным.....	50
apply.....	50
lapply.....	51
sapply.....	52
do.call	53
Резюме	53
Глава 4. Управление процессом вычислений	54
Циклы	54
Цикл со счётчиком	54
Цикл с предусловием	57
Условные операторы.....	58
Резюме	59

Глава 5. Базовая графика	60
Функции низкого и высокого уровней.....	60
Глобальные и локальные параметры графиков.....	65
Легенда.....	67
Комбинации графиков.....	67
Графики функций.....	69
Экспорт в файлы.....	70
Резюме и ссылки.....	70
Глава 6. Функции	72
Создание функций.....	72
Локальные и глобальные переменные. Области видимости.....	74
Диагностические сообщения.....	76
Функции в качестве аргументов.....	76
Функциональное программирование.....	78
Резюме.....	79
Глава 7. Факторы и даты	80
Категориальные данные.....	80
Дата и время.....	83
Резюме.....	86
Глава 8. Пакеты	87
Установка и загрузка.....	87
Выбор пакета.....	89
Справка и её разновидности.....	89
Как самому создать пакет R?.....	91
Пакет magrittr: конвейер операций.....	92
Глава 9. Ввод и вывод данных. Работа с файлами	94
Рабочий каталог пользователя.....	94
Запись данных в стандартное устройство вывода.....	94
Запись в текстовые файлы.....	95
Таблицы.....	95
Строки.....	97
Матрицы.....	97
Чтение из текстовых файлов.....	97
Элементы данных: scan.....	97
Строки: readLines.....	99
Таблицы.....	100

Работа с данными в бинарном формате.....	101
Управление файлами и каталогами.....	102
Взаимодействие с базами данных.....	103
DBI + SQLite.....	103
sqlf.....	103
Резюме.....	104
Ссылки к части I.....	105
ЧАСТЬ II. СБОР ДАННЫХ.....	106
Глава 10. Открытые данные.....	107
Что это такое?.....	107
Данные Всемирного банка.....	108
Где взять данные?.....	113
Резюме.....	114
Глава 11. Протокол HTTP.....	115
Основные понятия.....	115
Запрос.....	116
Ответ.....	117
Коды состояния.....	118
Передача параметров.....	119
HTTP в R.....	120
Пакет http.....	120
Пакет RCurl.....	122
Кириллица и кодирование URL.....	123
Пример: геокодирование с помощью Google Maps Geocoding.....	124
Пример: доступ к API портала открытых данных РФ.....	126
Ссылки.....	129
Глава 12. Импорт данных.....	130
Чтение файлов.....	130
Скачивание.....	131
Excel.....	132
JSON.....	133
Пример: какой из JSON-пакетов самый популярный?.....	133
Google Spreadsheets.....	136
Архивы.....	137
Завершающий штрих: проверка типа данных.....	138
Ссылки.....	139

Глава 13. Веб-скрапинг	140
Используйте структуру данных	140
Элементы HTML и CSS	143
div и span	143
Классы и идентификаторы	144
Путь к элементу	146
XPath.....	146
CSS.....	149
Как найти путь к элементу при помощи браузера	150
Проверка и упрощение пути. Консоль разработчика.....	153
Резюме	155
Лирическое отступление: построение графов	155
Ссылки	157
Поиск в Интернете	157
HTML и CSS:.....	158
XPath.....	158
Глава 14. Пакет rvest	159
Пакеты для веб-скрапинга.....	159
Получение и обработка HTML-документа.....	160
Поиск элемента	162
Разбор элемента	164
Пример: получаем ссылку и скачиваем файл	165
Таблицы.....	166
Пример: извлечение таблицы из Википедии	166
Пример: разбор страницы сериала «Светлячок»	167
Пример: извлечение данных об инвестиционных фондах	169
Работа с формами. Сессии.....	171
Пример: аутентификация на форуме	173
Функции навигации	174
Работа с кодировками	175
Заключительные замечания и ссылки	175
Глава 15. RSelenium: управляем браузером	177
Пример: перевод с помощью Yandex.Translate	179
Пример: динамически генерируемая ссылка на файл	180
Selenium и браузеры	183
Резюме и ссылки	183

Глава 16. PhantomJS и обработка динамических веб-страниц	185
Динамические страницы: описание проблемы	185
Установка	186
Запуск	186
Пример: рендеринг веб-страницы	187
Сохранение веб-страницы в файл	188
Резюме и ссылки	190
Глава 17. Facebook	192
Протокол авторизации OAuth 2.0	192
Получение маркера доступа пользователя API Graph	193
Доступ к данным с помощью gvest и jsonlite	196
Пакет Rfacebook и создание приложения	198
Глава 18. Сбор информации с помощью API ВКонтакте	204
Создание приложения	204
Регистрация приложения	204
Получение кода доступа	206
Получение данных	207
Реализация в R	208
Построение графа связей	210
Получение другой информации из сети	212
Поиск пользователя	213
Ограничения	214
Глава 19. Использование Twitter API	215
Получение доступа к Twitter API	215
Подключение к Twitter из R	215
Поиск и сохранение его результатов в базе данных	217
Фильтрация результатов поиска	218
Построение облака слов	219
Данные для анализа	220
Лексический корпус и терм-документная матрица	220
Ключевые слова и их частоты	221
Облако слов	221
Ограничения Search API	223
Streaming API	223
Ссылки	223

Глава 20. Регулярные выражения	225
Символы и метасимволы	225
Квантификаторы	227
Положение образца внутри строки	228
Операторы	229
«Жадность» и «лень» квантификаторов	230
Классы символов	232
Заключительные замечания	233
Ссылки	234
Глава 21. Создание карт на основе собранных данных	235
Интерактивные карты в leaflet	235
Переходим к созданию карты	239
Извлечение адресов и названий магазинов	240
Геокодирование	242
Отображение на карте	243
Работа с шейп-файлами	244
Ссылки	247
Ссылки к части II	249
Приложение А. Среда разработки RStudio	250
Создание скрипта	251
Автодополнение имён объектов	252
Выполнение	252
Рабочее пространство	253
История команд	254
Сохранение файлов	256
Кодировки файлов	256
Управление файлами в рабочем каталоге	257
Управление пакетами	257
Поиск и замена	258
Автоматическое создание функций	259
Комментирование	260
Переход к определению функции	260
Ссылки	261
Приложение Б. Языки поисковых запросов Google и Яндекс	262
Почему важно уметь пользоваться ЯПЗ	263
Предотвращение перегрузок сервиса	263

Приложение В. Введение в HTML и CSS	264
Веб-страница.....	264
Гиперссылки	266
Шрифт	267
Цвет.....	268
Стиль.....	268
Выравнивание	270
Рисунки.....	270
Списки	271
Маркированные.....	271
Нумерованные	271
Вложенные.....	272
Таблицы.....	272
Ссылки	273
Приложение Г. Регулярные выражения	274
Предметный указатель	276

Глава 1

Знакомство с R

В этой главе мы установим R и научимся пользоваться средой разработки RGui. По пути решим простую, но весьма распространённую задачу – построим график функции.

Установка

Рассмотрим способ установки R, который подойдёт для любого типа операционных систем, поддерживаемых пакетом: Linux, Mac или Windows. В случае Windows этот путь – единственный, для Linux и Mac проще воспользоваться менеджером пакетов соответствующей системы.

Чтобы установить R, зайдём на его официальный сайт и выберем интересующую версию программы. Сам R, документация к нему и дополнительные пакеты распространяются через сеть ftp- и веб-серверов, называемую CRAN (Comprehensive R Archive Network). Поэтому следующим шагом будет выбор одного из более чем шести десятков зеркал CRAN. После этого, указав тип операционной системы, скачиваем дистрибутив R.

Запустим инсталляционный файл и будем следовать указаниям программы-установщика. Единственный момент, который потребует вашего внимания, – выбор разрядности операционной системы (рис. 1.1).

После установки запустим программу. В Linux и Mac, по умолчанию, R работает в консоли. Кроме того, вместе с пакетом поставляется графический интерфейс. В Linux он основан на связке Tcl/Tk и запускается командой `R --gui=Tk`. В Mac встроенный графический интерфейс для R называется R.app.

Версия R для Windows поставляется с графической оболочкой RGui (рис. А.5), которую мы и рассмотрим в дальнейшем.

Заметим, что работа со всеми указанными выше графическими оболочками выглядит примерно одинаково.

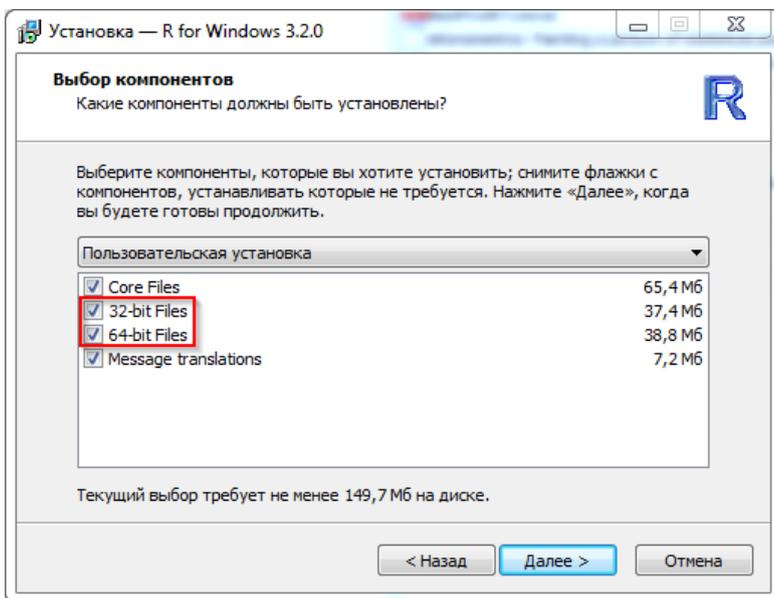


Рис. 1.1 ❖ Выбор компонентов в ходе установки R под Windows

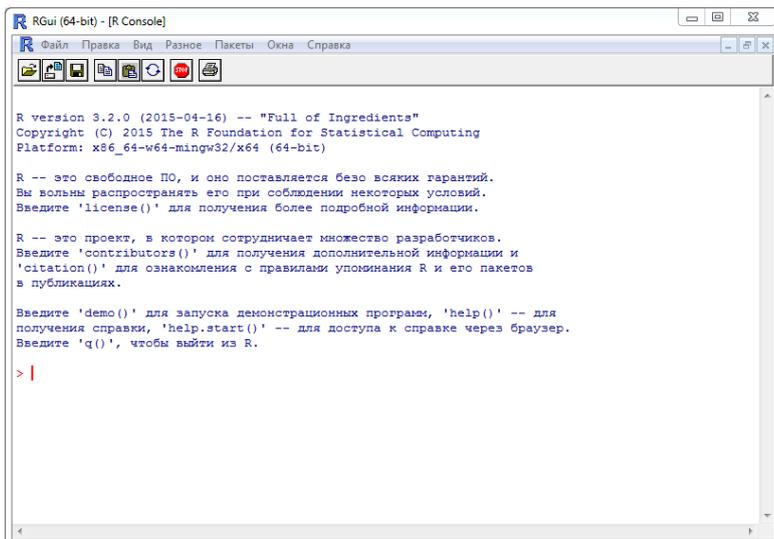


Рис. 1.2 ❖ Консоль R в графической оболочке RGui (Windows)

Работа в среде RGui

RGui – это стандартная графическая оболочка R под Windows, являющаяся простейшей средой разработки. Она быстро загружается и достаточно удобна в использовании. В RGui есть три вида окон:

- консоль;
- редактор скриптов;
- окно графического устройства.

Команды R вводятся в консоли (рис. А.5) после приглашения пользователя (значка ' $>$ ') и отправляются на выполнение нажатием *Enter*.

Управление консолью:

- дополнение команды – *Tab*;
- перемещение по истории команд – клавиши со стрелками;
- прекращение выполнения команды – *Esc*;
- переключение в другое окно – *Ctrl+Tab*;
- очистка консоли – *Ctrl+L*.

Для создания собственных программ (скриптов)¹ удобнее использовать не консоль, а редактор (рис. 1.3).

Открыть его можно в меню **Файл/Новый скрипт**. Первое окно открывается с помощью меню, последующие – так же или комбинацией клавиш *Ctrl+N*.

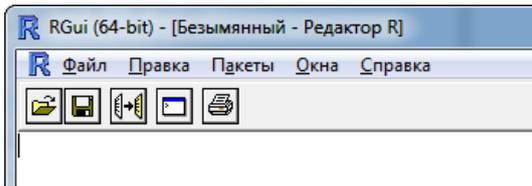


Рис. 1.3 ❖ Интерфейс редактора кода в RGui

Обратите внимание на то, как изменилась панель инструментов по сравнению с консолью (рис. А.5).

В качестве примера построим график синусоиды:

```
x <- seq(-pi, pi, .1)
y <- sin(x)
plot(x, y)
```

В первой строке формируется одномерный массив (вектор) x -координат, значения которых изменяются от $-\pi$ до π с шагом 0,1. Этот массив сохраняется в переменной x . Комбинация символов $<-$ обозначает операцию присваивания.

¹ Мы используем термины «скрипт» и «программа» как синонимы.

Во второй строке создаётся вектор y , состоящий из синусов элементов вектора x .

Наконец, в третьей строке функция `plot` строит требуемый график.

В редакторе можно набирать команды и выполнять их как по одной, так и целыми блоками, с помощью комбинации `Ctrl+R` (на панели инструментов также есть соответствующая кнопка). Например, можно выделить весь скрипт – `Ctrl+A` и отправить его на выполнение `Ctrl+R` (рис. 1.4).

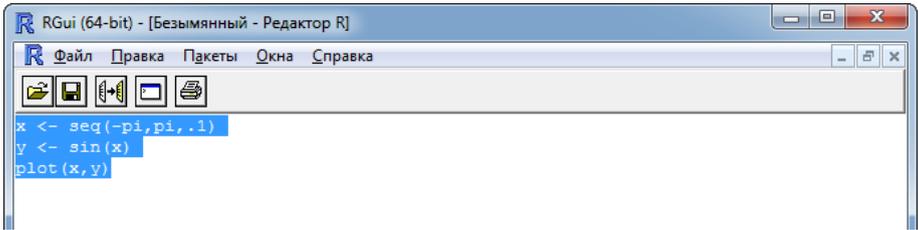


Рис. 1.4 ❖ Выделенную часть кода можно отправить на выполнение комбинацией клавиш `Ctrl+R`

В результате получим графическое окно с построенным в нём графиком синусоиды (рис. 1.5). По терминологии R окна, в которых строятся графики, и файлы, в которых эти графики сохраняются, вместе называются *графическими устройствами*.

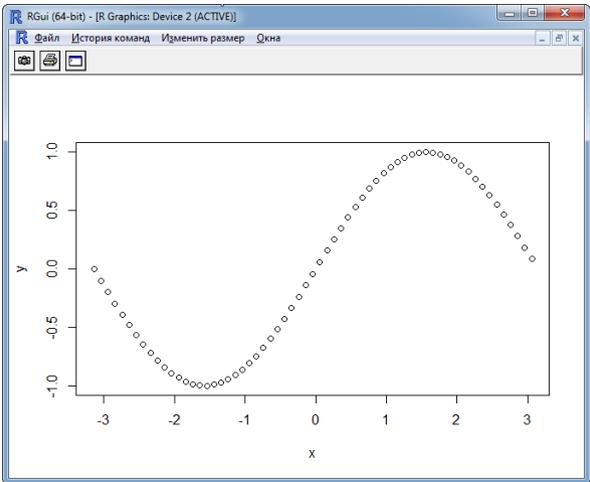


Рис. 1.5 ❖ Графическое устройство в RGui

Вернуться к консоли можно нажатием кнопки на панели инструментов (рис. 1.6) или при помощи *Ctrl+Tab*.

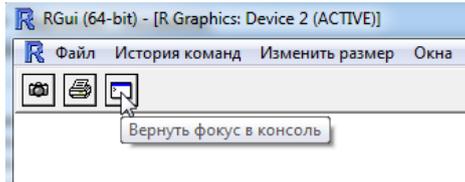


Рис. 1.6 ❖ Кнопка **Вернуть фокус в консоль** на панели инструментов графического устройства

Управление окнами при помощи меню **Окна** показано на рис. 1.7.

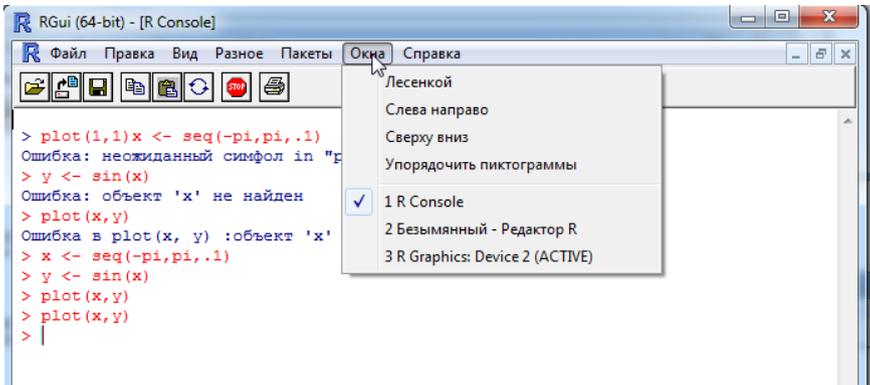


Рис. 1.7 ❖ Управление окнами с помощью меню **Окна**

Сохранить скрипт можно при помощи команд **Сохранить** или **Сохранить как...** меню **Файл** редактора или соответствующей кнопки на его панели управления.

Заметим, что скрипты R, объединённые общей темой, удобно держать в отдельном каталоге.

Выйти из R можно, вызвав в консоли `q()` или воспользовавшись меню **Файл/Выйти**.

При выходе из RGui среда предложит сохранить рабочее пространство (рис. 1.8).

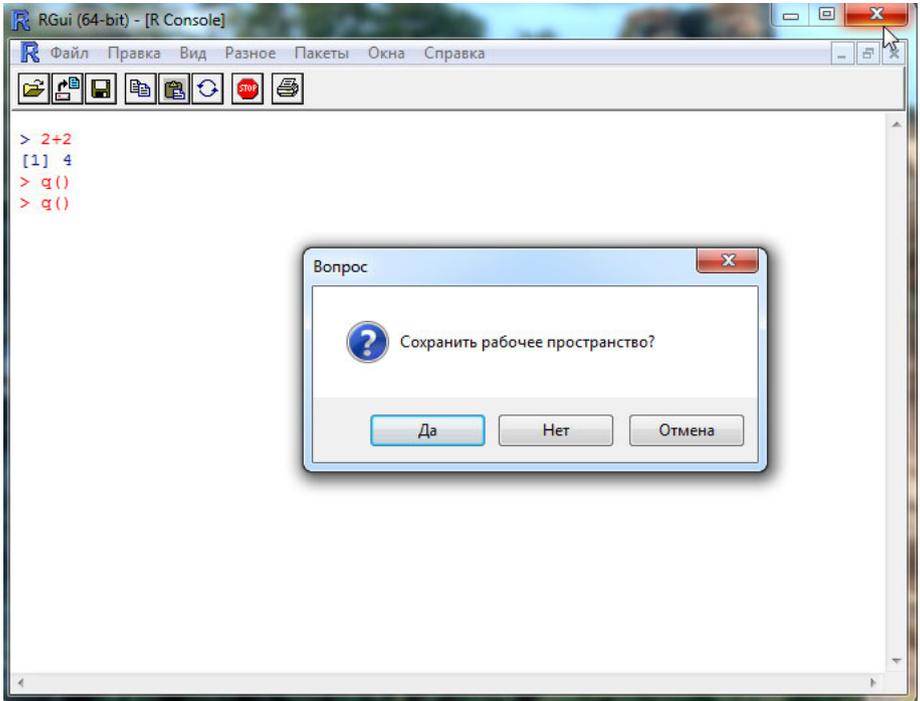


Рис. 1.8 ❖ Сохранение рабочего пространства по выходе из RGui

Рабочее пространство (workspace) – это область оперативной памяти, в которой хранятся все созданные пользователем объекты (векторы, матрицы, таблицы, списки, функции и т. п.). Сохранение рабочего пространства в файле и его последующая загрузка в новом сеансе работы с R (**Файл/Загрузить рабочее пространство...**) позволяют продолжить работу с того места, где она была прервана. При этом сохраняются значения всех переменных, вычисленные на прошлом сеансе работы.

Просмотреть список объектов в рабочем пространстве можно при помощи функции `ls`. Вызовем её в консоли

```
> ls()
```

и получим в результате

```
## [1] "x" "y"
```

Действительно, в памяти сейчас хранятся векторы координат синусоиды.

Два символа решётки (##) указывают на то, что далее идёт результат выполнения программы (в консоли они не отображаются, см. рис. 1.9). Смысл единицы ([1]) станет ясен в следующей главе.

Другие команды управления рабочим пространством:

```
# Сохранить рабочее пространство в файл .RData в текущем рабочем каталоге
save.image()
# Сохранить заданные объекты в файле
save(object_list, file="myfile.RData")
# Загрузить образ рабочего пространства
load("myfile.RData")
# Очистить рабочее пространство
rm(list=ls())
```

Символ # открывает строку комментария.

Сохранить или загрузить образ рабочего пространства можно при помощи меню **Файл**. Следует иметь в виду, что это меню, как и любое другое в RGui, просто запускает на выполнение соответствующие функции R.

Например, команда меню **Файл/Сохранить рабочее пространство...** вызывает функцию `save.image`, и результат этого вызова можно увидеть в консоли (рис. 1.9).

```
> x <- seq(-pi, pi, .1)
> y <- sin(x)
> plot(x,y)
> ls()
[1] "x" "y"
> save.image("C:\\Users\\Admin\\Documents\\myimage")
> |
```

Рис. 1.9 ❖ Команда меню **Файл/Сохранить рабочее пространство...** представляет собой вызов функции `save.image`

Клавиши со стрелками вверх и вниз позволяют перемещаться по списку выполненных ранее команд – *истории команд*. Историю команд также можно сохранить в файл на диске.

```
# Вывод истории команд
history() # выводит список 25 последних выполненных команд
history(max.show=Inf) # выводит все выполненные в сессии команды
# Сохранить историю команд
savehistory(file="myfile") # по умолчанию ".Rhistory"
# Загрузить историю команд
loadhistory(file="myfile") # по умолчанию ".Rhistory"
```

Помимо RGui, для R существуют другие, более продвинутые среды разработки, например R Commander или RStudio. Так же, как и сам R, они являются кросс-платформенными и распространяются свободно. Заметим, что хотя возможностей у этих сред гораздо больше, они дополняют возможности RGui, но не перечёркивают их. Так что навыки работы с RGui пригодятся и при работе в более совершенной среде.

Справка

Верхний пункт меню Справка (в консоли это Консоль, в редакторе скриптов, соответственно, Редактор) даёт короткую подсказку по работе с соответствующим окном. Справка для редактора показана на рис. 1.10.

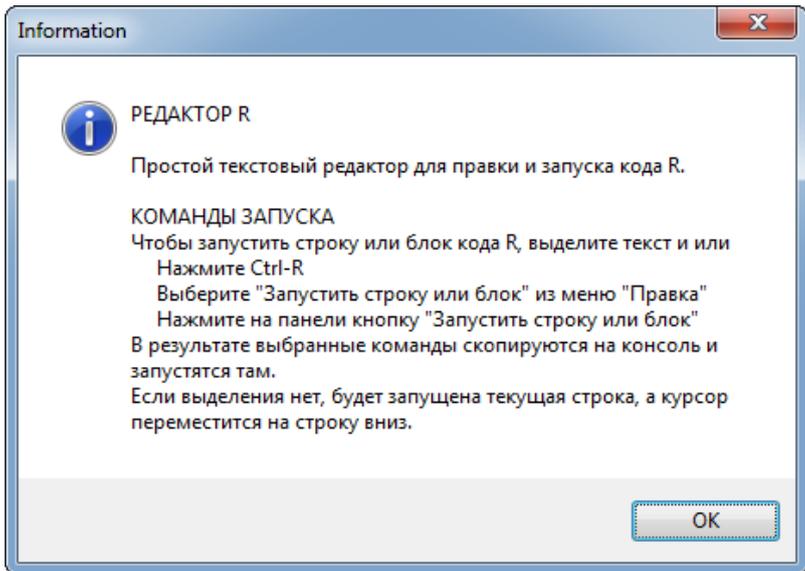
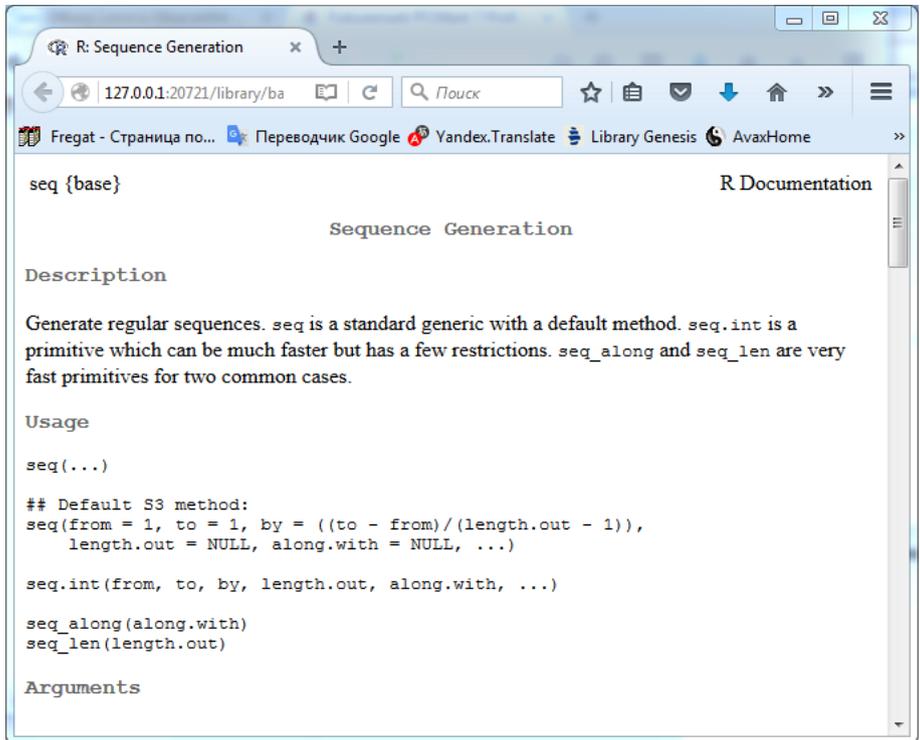


Рис. 1.10 ❖ Справка по работе в редакторе скриптов RGui

Справку по функции с именем `fun` можно получить, набрав в консоли `?fun`. Например, для получения справки по функции создания последовательностей `seq` наберём:

```
?seq
# или
help(seq)
```

В ответ на выполнение этой команды откроется браузер со справочной информацией (рис. 1.11).

Рис. 1.11 ❖ Справка по функции `seq`

Глава 2

Скаляры, векторы и матрицы

В этой главе мы научимся создавать переменные и выполнять над ними арифметические и логические операции. Познакомимся с основными типами данных: числовыми, символьными и логическими, а затем узнаем, что все они представляют собой векторы, которые в R вовсе не составной тип данных, а наоборот – простейший. Научимся создавать из векторов матрицы и в несколько строк кода вычислять интегралы.

Арифметические операции и присваивание

Арифметические операции в R выглядят так же, как и во многих других языках программирования:

```
3+2
2*(4-1)/6
3^2; sqrt(4) # ';' служит для разделения выражений в строке
```

```
## [1] 5
## [1] 1
## [1] 9
## [1] 2
```

А вот запись операции присваивания имеет особенности. Оно может обозначаться знаком равенства '=' или «стрелками», позволяющими присваивать как влево ($a \leftarrow b$), так и вправо ($b \rightarrow a$).

```
a <- 2 # присвоим значение a - традиционная форма записи для R
3 -> b # присвоим значение b
c = a+b # и это - то же присваивание
c      # напечатаем значение c
```

```
## [1] 5
```

Мы будем использовать обозначение ' \leftarrow ' как из соображений совместности со старыми версиями пакета, так и чтобы отличать присваивание от других операций, например от задания имён столбцов в таблицах – в последнем случае всегда используется '='.

Имена

При назначении имён переменных и функций придерживаются следующих правил:

1. Использовать для имён только латинские буквы, символ подчёркивания '_', цифры и точку '.'; при этом имя не должно начинаться с цифры или точки: `plot_new2` и `plot.new` – это правильно, а `.plot` и `2plot` – нет.
2. Помнить, что R чувствителен к регистру: `var`, `var` и `VAR` – это разные имена.
3. Не давать объектам имена, занятые распространёнными функциями (например, не следует создавать функцию с именем `c()`) или ключевыми словами (особенно это касается `T`, `F`, `NA`, `NaN`, `Inf`).

Одна из особенностей R состоит в том, что в именах объектов допустимо использовать точку. Её нередко применяют для разделения смысловых частей имени, подобно тому как в других языках используют символ подчёркивания (который в ранних версиях R использовать было нельзя). Так, функция `install.packages()` служит для установки новых пакетов расширений, но это именно функция, а не метод `packages()` объекта `install`.

Проверить, существует ли переменная или функция с заданным именем, можно при помощи функции `exists`:

```
exists('a')    # переменная 'a' уже определена,
## [1] TRUE

exists('d')    # ...а 'd' - ещё нет
## [1] FALSE

exists('all')  # существует функция с именем 'all'
## [1] TRUE
```

Параметр `mode` позволяет указать вид объектов, среди которых выполняется поиск. Так, переменная `a` существует, а вот функция `a` – нет:

```
exists("a", mode="function")
## [1] FALSE
```