У. Н. Венэбльз
Д. М. Смит
и рабочая группа
разработки R

Введение в R версия 3.5.2 (2018-12-20)

Заметки по R: среда программирования для анализа данных и графики

У. Н. Венэбльз

Введение в R версия 3.5.2 (2018-12-20). Заметки по R: среда программирования для анализа данных и графики

«Издательские решения»

Венэбльз У. Н.

Введение в R версия 3.5.2 (2018-12-20). Заметки по R:
среда программирования для анализа данных и графики /
У. Н. Венэбльз — «Издательские решения»,

ISBN 978-5-44-966153-1

R — язык программирования для статистической обработки данных и работы с графикой, а также свободная программная среда вычислений с открытым исходным кодом в рамках проекта GNU. R — язык и среда поддерживаются и развиваются организацией R Foundation, сотрудники которой написали данную книгу.R широко используется как статистическое программное обеспечение для анализа данных и фактически стал стандартом для статистических программ.

ISBN 978-5-44-966153-1

Содержание

Введение в R версия 3.5.2 (2018-12-20) Заметки по R: среда программирования для анализа данных и графики

У. Н. Венэбльз
Д. М. Смит

Переводчик Александр Александрович Фоменко

ISBN 978-5-4496-6153-1
Создано в интеллектуальной издательской системе Ridero

Предисловие

Данное введение в ***R*** получено из исходного набора примечаний, описывающих среду ***S*** и *SPlu*s, написанных в 1990—2 Биллом Венэбльзом и Дэвидом М. Смитом в университете Аделаиды. Сделано много небольших изменений для отражения различий между программами ***R*** и ***S,*** и развернули часть материала.

Выражаем искреннюю благодарность Биллу Венэбльзу (и Дэвиду Смиту), гарантировавших разрешение распространения этой модифицированной версии заметок, поддержав, таким образом, ***R*** от пути назад.

Комментарии и исправления всегда приветствуются. Пожалуйста, адресуйте корреспонденцию на электронную почту ***R****-core@R -project.org*.

Предложения читателю

Большинство новичков ***R*** начнет с вводного сеанса в Приложении А. Он познакомит со стилем сеансов ***R*** и, что еще более важно, даст некоторое впечатление о том, что фактически происходит.

Многие пользователи придут в ***R,*** главным образом, из-за его средств графики. Смотри Главу 12 [Графика], которую можно прочесть в почти любое время и не следует ожидать усвоения всех предыдущих разделов.

О переводе

Данная книга является переводом документации, доступной на английской языке в составе дистрибуции ***R***. После установки оригинал перевода доступен по адресу *\каталог R \doc\manual\R-intro*. Если данный файл перевода переименовать в *R-intro* и заменить оригинальный файл на данный, то из справки по ***R*** будет доступен данный перевод.

Перевод выполнен полностью за некоторыми отличиями:

– в частично исключены тексты, относящиеся к иным ОС, кроме Windows;

– исключены справочные приложения, в которых были собраны ссылки на функции и термины в английском тексте;

– расширен раздел по пакетам за счет описания пакетов, применяемым в эконометрике.

Переводчик будет благодарен за выявленные ошибки и неточности.

1. Введение и предварительные замечания

1.1. Среда R

R представляет собой набор программных средств для манипулирования данными, вычисления и графического отображения. Кроме этого возможно:

– эффективная обработка и хранение данных,

– набор операторов для вычислений на массивах, особенно матрицах,

– цельная, непротиворечивая, комплексная коллекция утилит для анализа данных,

– графические средства для анализа данных и отображения или непосредственно на компьютере или при выводе на печать, и

– хорошо разработанный, простой и эффективный язык программирования (называемый «***S***»), который включает условные выражения, циклы, определяемые пользователем рекурсивные функции и средства ввода и вывода. Действительно, большинство поддерживаемых системой функций сами написаны на языке ***S***.

Термин «окружение/среда» предназначен, чтобы характеризовать ее как полностью запланированную и последовательную систему, а не постепенно возникшего конгломерата весьма специфических и негибких инструментов, как часто имеет место с другим программным обеспечением анализа данных.

R является средством разработки методов интерактивного анализа данных. Она была разработана быстро и была расширена большим количеством пакетов. Однако, большинство программ, написанных в ***R***, принципиально являются программами-однодневками, написанными для конкретного случая анализа данных.

1.2. Связанное программное обеспечение и документация

R можно рассмотреть как реализацию языка ***S***, который разработан в Bell Laboratories Риком Беккером, Джоном Чемберсом и Алланом Уилксом, и который собственно лежит в основе систем *S-Plus*.

Эволюция основ языка ***S*** характеризуется четырьмя книгами Джона Чемберса с соавторами. Для ***R*** основой является «Новый Язык S: Среда программирования для анализа данных и графики», написанной Ричардом А. Беккером, Джоном М. Чемберсом и Алланом Р. Уилксом. Новые функции ***S,*** опубликованные 1991, даны в «Статистических моделях в S», отредактированном Джоном М. Чемберсом и Тревором Дж. Хэсти. Формальные методы и классы пакета методов основаны на описанных в «Программировании с данными» Джоном М. Чемберсом. См. Приложение F [Ссылки], для точной ссылки.

Сейчас есть много книг, которые описывают использование ***R*** для анализа данных и статистики, и документация для S/S-Plus может, как правило, использоваться с ***R,*** если помнить различия между реализациями ***S***.

1.3. R и статистика

Наше введение в среду ***R*** не упоминает статистику, но много людей используют ***R*** в качестве системы статистики. Будем думать о ней как о среде, в пределах которой были реализованы много классических и современных статистических методов. Некоторые из них встроены в основу среды ***R,*** но многие предоставлены как пакеты. В составе ***R*** существует около 25 пакетов (названных «стандартными» и «рекомендуемыми» пакетами), и еще больше доступно через семейство сайтов CRAN (через *http://CRAN.R -project.org)* и из других источников. Более подробную информацию о пакетах рассмотрим позже (см. Главу 13 [Пакеты]).

Большинство классических статистик и многое из последних методик доступно для использования в ***R,*** но пользователи должны быть готовы к небольшим усилиям для поиска нужного.

Есть важное различие в философии между ***S*** (и, следовательно, ***R***) и другими основными статистическими системами. В ***S*** статистический анализ обычно делается как ряд шагов с промежуточными результатами, сохраненными в объектах. Таким образом, тогда как SAS и SPSS дадут обильные результаты регрессионного или дискриминантного анализа, ***R*** выведет минимум результатов и сохранит их в подогнанном объекте для последующего использования функциями ***R.***

1.4. R и система Window

Самый удобный способ пользоваться ***R*** – это использовать графическую рабочую станцию с окнами. Это руководство нацелено на пользователей, у которых есть это средство. В особенности мы будем иногда обращаться к использованию ***R*** в Windows, хотя обширный объем того, что сказано, обычно применим к любой реализации среды ***R.***

Большинство пользователей, время от времени, непосредственно сталкивается с операционной системой на своем компьютере. В этом руководстве, главным образом, обсуждается взаимодействие с операционной системой на машинах UNIX. Если ***R*** исполняется под Windows или Mac OS, то будет необходимо внести некоторые небольшие корректировки.

Установка рабочей станции, чтобы в полной мере воспользоваться настраиваемыми функциями ***R,*** является простой, хотя и несколько утомительной процедурой и здесь рассматриваться не будет. При трудностях пользователям следует найти местного опытного специалиста.

1.5. Использование R в интерактивном режиме

При использовании программы ***R*** она выдает запрос ожидания входных команд. Запрос по умолчанию **'>»**, который на UNIX совпадает с запросом оболочки, и таким образом, может казаться, что ничего не происходит. Однако, как увидим, при желании легко изменить на другой запрос ***R.*** Предположим, что запрос оболочки UNIX – «$».

В использовании ***R*** под UNIX предложенная процедура для первого случая следующая:

– Создать отдельный подкаталог, скажем '*work*' для файлов с данными, на которых будет использоваться ***R***. Он будет рабочим каталогом всякий раз при использовании ***R*** для этой определенной задачи.

```
$ mkdir work
$ cd work
```

– Начать программу ***R*** командой

```
$ R
```

– Здесь можно давать команды

– Для завершения программы ***R*** введите:

```
q ()
```

В этом этапе Вас спросят, хотите ли Вы сохранить данные своего сеанса ***R.*** На некоторых системах это будет сделано с помощью диалогового окна, а на других Вы получите текстовый запрос, на который Вы можете ответить «да», «нет» или «отмена» (достаточно будет ввести первую букву) для сохранения данных перед выходом, выйти без сохранения, или вернуться в сеанс ***R.*** Сохраненные данные будут доступны в будущем сеансе ***R***.

Дальнейшие сеансы ***R*** требуют меньше действий.

– Сделайте '*work*' рабочим каталогом и запустите программу как прежде:

```
$ cd work
$ R
```

– Используйте программу ***R,*** которая завершится командой *q ()* в конце сеанса.

Для использования ***R*** под Windows процедура в основном такая же. Создайте папку как рабочий каталог, и установите его в поле «*Start In*' ярлыка ***R.*** Затем запустите ***R,*** дважды щелкая по иконке.

1.6. Первый сеанс

Читателям, желающим испытать ***R*** на компьютере, прежде чем приступить настоятельно советуем проработать вводный сеанс, данный в Приложении А [Сеанс выборки].

1.7. Получение справки по функциям и средствам

У ***R*** есть встроенное справочное средство. Для получения дополнительной информации о любой определенной именованной функции, например *solve*, напишите команду:

> *help (solve)*

Альтернатива:

>*? solve*

Для средств, указанных специальными символами, параметр должен быть включен в двойные или одинарные кавычки, делая его «символьной строкой»: это также необходимо для нескольких слов с синтаксическим значением, включая *if, for* и *function*.

> *help (» [[»)*

Может использоваться любая форма символа кавычки для исключения других кавычек, как в строке, «It's important». Наше соглашение состоит в предпочтительности использования символа двойной кавычки.

На большинстве установок ***R*** справка доступна в формате HTML, достаточно выполнить:

> *help.start ()*

что запустит Веб-браузер, предоставляющий возможность использовать гиперссылки в справке. Ссылки *«Search Engine and Keywords'* на странице, загруженной *help.start (),* особенно полезны, так как содержат высокоуровневый список понятий, используемый при поиске доступных функций. Это может оказаться отличным способом быстро решить проблемы и понять широкий спектр возможностей ***R.***

Команда *help.search* (альтернативно??) позволяет искать справку различными способами.

Например,

>*?? solve*

Попробуйте *?help.search* для деталей и большего количества примеров.

Пример на теме справки обычно можно выполнить:

> *example (topic)*

У версий ***R*** для Windows есть другие дополнительные системы справочной информации. Используйте:

>*?help*

для получения дальнейшей информации.

1.8. Команды R, учет регистра и т. д.

Технически ***R*** является языком выражений с очень простым синтаксисом. Он учитывает регистр, как большинство других программ UNIX, таким образом, ***A*** и ***a*** являются различными символами и ссылаются на разные переменные. Набор символов, которые могут использоваться для имен ***R,*** зависит от операционной системы и страны, в которой ***R*** исполняется (технически говоря, от используемой локализации – *locale*). Обычно разрешены все алфавитно-цифровые символы плюс».» и «_», с ограничением, что имя должно начинаться с».» или буквы, и если начинается с».», то второй символ не может быть цифрой. Имена в фактически неограниченны.

Для портативного кода ***R*** (включая использовуемый в пакетах ***R***) должна использоваться только *A—Za—z0—9.*

Простые команды состоят из выражений (*expression*), либо присвоений (*assignments*). Если выражение вводится как команда, то оно вычисляется, выводится (пока специально не сделано невидимым) и значение теряется. Присвоение также вычисляет выражение и передает значение переменной, но результат автоматически не выводится.

Команды разделены либо точкой с запятой (»;»), либо новой строкой. Простые команды могут группироваться в одно составное выражение фигурными скобками (» {» и»}»). Комментарии могут быть помещены практически где угодно, начинаясь со знака «решетки» («#»), при этом все до конца строки является комментарием.

Если команда не полна в конце строки, то ***R*** даст особое приглашение, по умолчанию:

```
+
```

на второй и последующих строках и продолжит читать ввод, пока команда синтаксически не полна. Этот запрос может быть изменен пользователем. Мы, как правило, будем опускать приглашение продолжения ввода и обозначим продолжение простым отступом.

Командные строки, вводимые на консоли, ограничены в размере доя 4095 байт (не символов).

1.9. Повтор и коррекция предыдущих команд

R реализует механизм для повторного вызова и выполнения предыдущих команд. Вертикальные клавиши со стрелками на клавиатуре могут использоваться для прокрутки вперед и назад по истории команд. Как только команда локализована таким способом, курсор может быть перемещен в пределах команды, используя горизонтальные клавиши со стрелками, и символы могут быть удалены клавишей DEL или добавлены другими клавишами. Более подробная информация предусмотрена далее: см. Приложение C [Редактор командной строки].

Кроме того, редактор текста Emacs предоставляет более полный механизм поддержки (через ESS – Emacs Speaks Statistic) для интерактивной работы с ***R.*** Смотри раздел «***R*** and Emacs» в The ***R*** statistical system FAQ.

1.10. Выполнение команд из файла или перенаправление вывода в файл

Если команды были сохранены во внешнем файле, скажем *«command. R'* в рабочем каталоге '*work*', то они могут быть выполнены в любое время в сеансе ***R*** командой:

> *source («commands. r»)*

Для Windows *Source* также доступен в меню File. Функция *sink:*

> *sink("record.lis»)*

отклонит весь последующий вывод консоли во внешний файл *'record.lis'*. Команда

> *sink ()*

восстановит вывод в консоли еще раз.

1.11. Сохранение данных и удаление объектов

Сущности, которые ***R*** создает и манипулирует, известны как объекты *(object)*. Они могут быть переменными, массивами чисел, символьными строками, функциями или более общими структурами, построенных из таких компонентов.

Во время сеанса ***R*** объекты создаются и хранятся по имени (обсуждается в следующей секции). Команда ***R:***

```
> objects ()
```

(также как ***ls ())*** может использоваться для вывода на экран имен (в основном) объектов, которые в настоящий момент хранятся в пределах ***R.*** Набор объектов, сохраненных в настоящий момент, называют рабочей областью *(workspace)*.

Для удаления объектов доступна команда *rm*:

```
> rm (x, y, z, ink, junk, temp, foo, bar)
```

Все объекты, создаваемые во время сеанса ***R,*** могут храниться постоянно в файле для использования в будущем сеансе ***R.*** В конце каждого сеанса ***R*** предоставляется возможность сохранить все в имеющиеся в данный момент объекты. Если подтвердить необходимость этого, то объекты записываются в файл, называемый». *RData'* в текущем каталоге, а строки команд, использованных в сеансе, сохраняются в файл». *Rhistory*'.

При последующем запуске ***R*** рабочая область загружается из этого файла. Одновременно загружается присоединенная история команд.

Рекомендуется использовать отдельные рабочие каталоги для анализов, проводимых с ***R.*** Очень распространено использовать для анализа объекты с именами ***x*** и ***y***. Подобные имена часто значимы в контексте отдельного анализа, но может быть довольно трудно решить то, чем они отличаются, если несколько анализов было выполнено в одном и том же каталоге.

2. Простые манипуляции; числа и векторы

2.1. Вектора и присваивания

R оперирует именованными структурами данных (*data structures)*. Простейшая такая структура – это числовой вектор, который является отдельным объектом, состоящим из упорядоченного набора чисел. Чтобы создать вектор с именем ***x***, скажем, состоящий из пяти чисел, а именно, 10.4, 5.6, 3.1, 6.4 и 21.7, используют команду ***R:***

> x <– c (10.4, 5.6, 3.1, 6.4, 21.7)

Это – оператор присваивания, использующий функцию ***c (),*** которая в этом контексте может взять произвольное число аргументов вектора и значением которой является вектор, полученный путем объединения аргументов конец с концом.

С другими параметрами кроме типов векторов, таких как параметры режима списка, действие *c ()* довольно отличается. Посмотрите Раздел 6.2.1 [Конкатенация списков].

Отдельное число, входящее в выражении, трактуется как вектор единичной длины.

Учтите, что оператор присваивания (» <-»), который состоит этих двух символов ‘ <’ («меньше чем») и ‘-’ («минус»), выполняется однонаправленно и «указывает» на объект, получающий значение выражения. В большинстве случаев можно использовать оператор ‘=’ в качестве альтернативы.

Также можно сделать присвоение, используя функцию *assign ()*. Эквивалентный выше приведенному способ присвоения выглядит как:

> assign («x», c (10.4, 5.6, 3.1, 6.4, 21.7))

Обычный оператор <– может считаться синтаксическим сокращением для него.

Присвоения могут также быть сделаны в другом направлении, используя очевидное изменение в операторе присваивания. Таким образом, то же самое присвоение могло быть сделано, используя:

> c (10.4, 5.6, 3.1, 6.4, 21.7) -> x

Если выражение используется в качестве полной команды, значение печатается и *теряется*. Итак, если бы нам пришлось использовать команду:

> 1/x

то обратные величины пяти значений были бы напечатаны в терминале (а значение ***x***, конечно, не изменилось).

В действительности все еще доступно *.Last.value* до выпполнения любого другого оператора.

Дальнейшее присваивание:

> y <-c (x, 0, x)

создаст вектор y с 11 элементами, состоящими из двух копий ***x*** и нулем между ними.

2.2. Векторная арифметика

Векторы могут использоваться в арифметических выражениях, и в этом случае операции выполняются поэлементно. Векторы, используемы в одном выражении, не обязательно должны иметь одинаковую длину. Если длины отличаются, то результат выражения – это вектор с длиной самого длинного вектора, который находится в выражении. Короткие векторы в выражении используются повторно столько раз, сколько это необходимо (возможно не целое число раз), до тех пор, пока они не совпадут с длиной самого длинного вектора. В частности константа просто повторяется. Так, учитывая предыдущие присваивания, команда:

```
> v <-2*x + y +1
```

создаст новый вектор ***v*** длины 11, составленный путем сложения элемент за элементом *2*x* повторенного 2.2 раза, ***y*** повторяется только раз, и 1 повторяется 11 раз.

Элементарными арифметическими операторами являются +, -, *, / и ^ для возведения в степень. Дополнительно присутствуют все простые арифметические функции. *log, exp, sin, cos, tan, sqrt* и так далее, все они имеют свое обычное значение. *max* и *min* выбирают самые большие и наименьшие элементы вектора соответственно. *range* является функцией, значение которой – вектор длины два, а именно, *c (min (x), max (x)). length (x)* является числом элементов в ***x***, *sum (x)* дает сумму всех элементов ***x,*** и *prod (x)* их произведение.

Двумя статистическими функциями являются *mean (x),* которая вычисляет среднее выборки, что соответствует *sum (x) /length (x),* и *var (x),* которая дает

```
sum ((x-mean (x)) ^2) / (length (x) -1)
```

или дисперсию выборки. Если параметром *var ()* является *n-на-p* матрица, значение – матрица ковариации выборки *p-на-p*, полученная путем интерпретации строк как ***p*** независимых векторов-выборок.

sort (x) возвращает вектор того же размера как ***x*** с элементами, расположенными в возрастающем порядке; однако доступны и другие более гибкие средства сортировки (см. *order ()*, или *sort.list (),* которые производят перестановку при сортировке).

Заметим, что *max* и *min* выбирают самое большое и наименьшее значение в их аргументах, даже если им дают несколько векторов. Параллельные функции максимума и минимума *pmax* и *pmin* возвращают вектор (длины, равной их самому длинному аргументу), который содержит в каждом своем элементе наибольшее (наименьшее) значение на этой позиции во всех входных векторах.

В большинстве случаев пользователю не важно, являются ли «числа» в числовых векторах целыми, реальными или даже комплексными. Внутренние расчеты осуществляются в реальных числах двойной точности, или комплексных числах двойной точности, если входные данные являются комплексными.

Для работы с комплексными числами надо явно предоставить мнимую часть. Таким образом:

```
sqrt (-17)
```

даст NaN и предупреждение, но

```
sqrt (-17+0i)
```

сделает вычисления как комплексных чисел.

2.3. Генерация регулярных последовательностей

У ***R*** есть много средств для генерации используемых последовательностей обычных чисел. Например, *1:30* является вектором *c (1, 2…, 29, 30).* У оператора двоеточия есть высокий приоритет в пределах выражения, таким образом, например *2*1:15* является вектором *c (2, 4…, 28, 30*). Введите *n <-10* и сравните последовательности *1:n-1* и *1: (n-1).*

Выражение *30:1* может использоваться для создания обратной последовательности.

Функция *seq ()* является более общим средством для генерации последовательности. У нее имеется пять параметров, только некоторые из которых могут специфицироваться в любом вызове. Первые два параметра, если дано, специфицируют начало и конец последовательности, и если только эти два параметра, то результат аналогичен оператору двоеточия. Например, *seq (2,10)* дает такой же вектор как 2:10.

Аргументы для *seq ()* и ко многим другим функциям ***R,*** могут также быть даны в именованной форме, когда порядок, в котором они появляются, не важен. Первые два аргумента можно назвать *from=value* и *to=value*; таким образом, *seq (1,30), seq (from=1, to=30)* и *seq (to=30, from=1)* являются одинаковыми с 1:30. Следующие два аргумента для *seq ()* можно назвать *by=value* и *length=value*, которые специфицируют размер шага и длину для последовательности соответственно. Если ни один из них не дан, то по умолчанию предполагается *by=1.*

Например:

> seq (-5, 5, by=.2) -> s3

генерирует вектор *c (-5.0, -4.8, -4.6, …, 4.6, 4.8, 5.0).* Подобно этому:
> s4 <– seq (length=51, from=-5, by=.2)

генерируется аналогичный вектор.

Пятый аргумент можно назвать *along=vector*, который используется как единственный аргумент и создает последовательность *1, 2…, length (вектор)*, или пустую последовательность, если вектор пуст (такое тоже может быть).

Соответствующая функция *rep ()*, которую можно использоваться для тиражирования объекта различными сложными способами. Самая простая форма:

> s5 <– rep (x, times=5)

которая поместит пять копий *x* от начала до конца в *s5.* Другая полезная версия

> s6 <– rep (x, each=5)

которая повторит каждый элемент ***x*** пять раз перед пересылкой в следующую.

2.4. Логические векторы

Так же как числовые векторы, ***R*** позволяет манипулирование логическими величинами. У элементов логического вектора могут быть значение TRUE, FALSE, и NA (для «не доступно», см. ниже). Первые два часто сокращаются как ***T*** и ***F***, соответственно. Заметим, однако, что ***T*** и ***F*** – только переменные, которые установлены в TRUE и FALSE по умолчанию, но не зарезервированные слова и, следовательно, могут быть перезаписаны пользователем. Следовательно, следует всегда использовать TRUE и FALSE.

Логические векторы генерируются условиями. Например:

> temp <– x> 13

устанавливает *temp* как вектор одинаковой длины как ***x*** со значением FALSE, соответствующих тем элементам ***x***, где условие не соблюдается, и TRUE, где имеет место.

Логическими операторами являются *<, <=,>,> =, ==*

Конец ознакомительного фрагмента.