Введение в R

ЧАСТЬ 1 Вы ещё анализируете данные по старинке, методом внимательного вглядывания в набор точек на графике, изредка подгоняя их к прямой с помощью многочисленных «мышекликов»? Пусть это делает за вас компьютер и R – это именно тот язык, который поможет доходчиво разъяснить ему вашу проблему.

Наши эксперты

Шипунов Алексей
Биолог, преподаватель, сотрудник американского университета Айдахо, использующий R для обработки информации во всех своих проектах, начиная с 2001 года. Русский переводчик R.

Евгений Балдин
Физик, преподаватель, научный сотрудник ИЯФ им. Будкера, профессионально занимается обработкой экспериментальных данных в области физики высоких энергий. Давний сторонник свободного ПО, впечатлённый мощью R.

Прежде всего, R – язык программирования для статистической обработки данных и работы с графикой, и в то же время – это свободная программная среда с открытым исходным кодом, развиваемая в рамках проекта GNU. R можно найти в любом дистрибутиве, не ставящем своей целью уместиться на одну дискетку. Например, в Debian GNU/Linux базовый пакет носит имя *r-base*, а подключаемые модули проще всего искать по акрониму «cran».

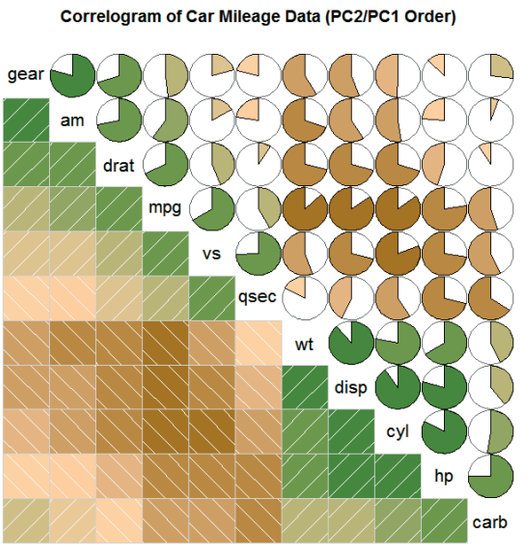
R применяется везде, где нужна работа с данными. Это не только статистика в узком смысле слова, но и первичный анализ (графики, таблицы сопряжённости), и продвинутое математическое моделирование. R без особых проблем может использоваться и там, где сейчас принято использовать коммерческие программы анализа уровня *MatLab*/*Octave*. С другой стороны, вполне естественно, что основная вычислительная мощь R лучше всего его проявляется при статистическом анализе: от вычисления средних величин до вейвлет-преобразований временных рядов.

География использования R очень разнообразна. Трудно найти американский или западноевропейский университет, где бы не работали бы с R. Очень многие серьезные компании (например, Boeing) используют R в своей деятельности. R для статистиков – это действительно глобально.

Немного истории

R возник как свободный аналог среды S-PLUS, которая, в свою очередь, является коммерческой реализацией языка расчётов S.

Язык S – довольно старая разработка (почти как *TeX*). Он возник ещё в 1976 году в компании Bell Labs, и был назван, естественно, «по мотивам» языка C. Первая реализация S была написана на FORTRAN и работала под управлением операционной системы GCOS. В 1980 г. она



была перенесена в UNIX, и с этого момента S стал распространяться, в основном, в научной среде. Начиная с третьей версии (1988 г.), коммерческая реализация S называется S-PLUS. Последняя в настоящее время продвигается компанией Insightful, и доступна под Windows и различными версиями UNIX, естественно, за плату, причем весьма и весьма немаленькую (для UNIX, например, S-PLUS предлагается за $6500). Собственно говоря, именно высокая цена и сдерживала широкое распространение этого во многих отношениях замечательного продукта. Тут-то и начинается история R.

В августе 1993 г. двое молодых новозеландских ученых анонсировали свою новую разработку, которую они назвали R. По замыслу создателей, Роберта Джентльмена [Robert Gentleman] и Росса Ихака [Ross Ihaka], она должна была стать новой реализацией языка S, отличающейся от S-PLUS некоторыми деталями, например, обращением с глобальными и локальными переменными, а также работой с памятью. Фактически, они создали не полный аналог S-PLUS, а новую «ветку» на «дереве S». Многие вещи, которые отличают R от S-PLUS, связаны с влиянием языка Scheme [функциональный язык программирования, один из наиболее популярных диалектов языка Lisp, – *прим. авт.*].

Сначала проект развивался довольно медленно, но когда в нём появилось достаточно возможностей, в том числе уникальная по лёгкости система написания дополнений или пакетов, всё большее количество людей стало переходить с S-PLUS на R. Когда же, наконец, были устранены свойственные первым версиям проблемы с памятью, то среди пользователей R стали появляться и любители других статистических пакетов (прежде всего тех, которые имеют интерфейс командной строки: *SAS*, *Stata*, *SYSTAT*). Количество книг, написанных про R, за

последние годы выросло в несколько раз, а количество пакетов уже приближается к полутора тысячам.

Идея центральной системы хранения и распространения пакетов, CRAN или Comprehensive R Archive Network (http://cran.r-project.org/), была заимствована из TeX-сообщества (CTAN или Comprehensive TeX Archive Network; аналогичной схемой пользуется и Perl-сообщество: CPAN или Comprehensive Perl Archive Network). Все три упомянутых проекта объединяет одно: стабильная база и множество дополнений. В отличие от добавления новой функциональности в монолитную программу, качественный пакет может сравнительно легко написать один человек за вполне обозримый промежуток времени.

Как скачать и установить R

Поскольку R – свободное ПО, его можно скачать и установить совершенно бесплатно. Конкретный способ, конечно, будет зависеть от установленной у вас ОС. Как уже говорилось выше, R входит в репозитории большинства распространенных дистрибутивов Linux, единственное, что нужно учесть – обновление пакетов часто отстает от выхода официальных версий (релиз-цикл R длится примерно три месяца, на момент написания статьи актуальной была версия 2.6.0). Версия R нумеруется тремя числами: первые два – это главная версия, которая обновляется два раза в год. С каждой главной версией в R привносятся изменения, причём часто – довольно значительные. Как правило, это множество новых команд, улучшенные алгоритмы выполнения старых, и, разумеется, исправления ошибок. К недостаткам смены версии можно отнести возможные проблемы с обратной совместимостью. Естественно, разработчики стараются свести такие изменения к минимуму. С другой стороны, написанные на R программы пяти-семилетней давности, как правило, работают без проблем. В общем и целом, мораль такова: обновляйте R смело, но при этом всегда читайте список изменений.

На каждую главную версию выходит, как правило, две минорных (нулевая и первая). Первая минорная версия обычно ничего нового, кроме исправления ошибок, не привносит. Таким образом, если Вы хотите всегда иметь самую свежую версию, то репозиторий пакетов, особенно в случае стабильных дистрибутивов, не годится. В этом случае надо будет скачивать R из CRAN (http://cran.r-project.org/). У этого сайта довольно много зеркал, так что можно выбрать подходящее.

Процедура компиляции R из исходного кода вполне стандартная: *./configure*, *make* и *make install* от имени суперпользователя. В общем, если процесс компиляции как таковой вас не пугает, собрать R из исходных текстов не составит труда.

Есть одна, важная для всех операционных систем особенность: R (в отличие от того же S-PLUS) держит все свои вычисления в оперативной памяти, поэтому, если в процессе работы, скажем, выключится электропитание, то результаты сессии, не записанные явным образом в файл, пропадут. Эта особенность, к сожалению, не позволяет R работать с действительно большими объёмами данных (порядка сотен тысяч и более записей), отдавая их на откуп гораздо менее удобной системе анализа ROOT (http://root.cern.ch), про которую мы уже говорили в LXF83.

Запуск

Опять-таки, каждая операционная система имеет свои особенности работы. Но, в целом, можно сказать, что и в Linux, и в Mac OS X, и в Windows существует так называемый «терминальный» способ запуска (для Mac и Windows имеется и штатный GUI с некоторыми дополнительными возможностями).

Терминальный способ прост: достаточно набрать в командной строке:

```
=> R

R version 2.4.0 Patched (2006-11-25 r39997)
Copyright (C) 2006 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R -- это свободное ПО, и оно поставляется безо всяких
гарантий. Вы вольны распространять его при соблюдении
некоторых условий. Введите 'license()' для получения
более подробной информации.

R -- это проект, в котором сотрудничает множество
разработчиков. Введите 'contributors()' для получения
дополнительной информации и 'citation()' для ознакомления
с правилами упоминания R и его пакетов в публикациях.

Введите 'demo()' для запуска демонстрационных программ,
'help()' -- для получения справки, 'help.start()' --
для доступа к справке через браузер. Введите 'q()', чтобы
выйти из R.

>
```

и появится приглашение в виде символа >. Теперь можно приступать к работе.

Если терминал запущен вне графической среды, то все изображения будут скидываться в один многостраничный PostScript-файл – **Rplots.ps**. В Mac OS X это будет происходить, даже если X11 запущен, так что полноценно использовать R под Mac можно только в GUI-варианте. Терминальный запуск под Windows таких ограничений не имеет.

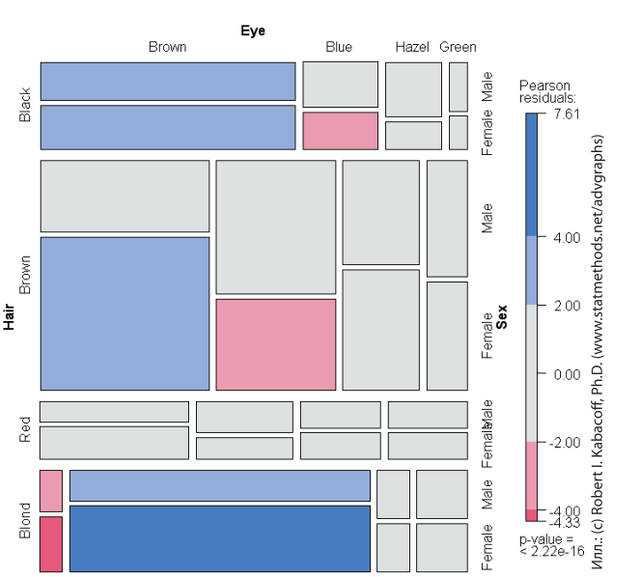
На будущее договоримся, что под «сессией R» мы будем иметь в виду терминальный запуск под X11 в GNU/Linux и GUI-запуск в Windows и Mac. GUI под эти операционные системы построены так, что они всё равно запускают терминал-подобное окно. Общение с R возможно только в режиме диалога «команда-ответ». Полноценного GUI с R не поставляется: хотя предпринимаются многочисленные попытки создать такую систему, но пока что все они далеки от завершения. Возможно, это и к лучшему, так как система из меню-окошек-опций не способна заменить полноценный интерфейс командной строки, особенно в случае таких сложных систем, как R. Интересно, что S-PLUS имеет очень приличный GUI, но если открыть любой учебник по этой системе, то можно заметить, что автор настоятельно рекомендует пользоваться командной строкой.

Первые шаги

Перед тем как начать работать, надо понять, как выйти. Для этого достаточно ввести одну команду и ответить на один вопрос:

```
> q()
> Save workspace image? [y/n/c]: n
```

Уже такой простой пример демонстрирует, что любая команда в R – это функция, которой можно передать аргумент. Даже если



аргумент не указан, то скобки всё равно надо ввести. Если этого не сделать, то вместо выхода из R на экран будет выведено определение функции:

```
> q
 function (save = “default”, status = 0, runLast = TRUE)
.Internal(quit(save, status, runLast))
<environment: namespace:base>
```

Чтобы узнать, как правильно вызывать функцию, следует воспользоваться встроенной справкой. Есть два пути. Первый – вызвать команду справки:

```
> help(q)
```

или

```
> ?q
```

Текст справки будет выведен в основном окне программы. Если внимательно прочитать его, то станет ясно, что выйти из R можно, и не отвечая на дополнительный вопрос, если ввести:

```
> q(“no”)
```

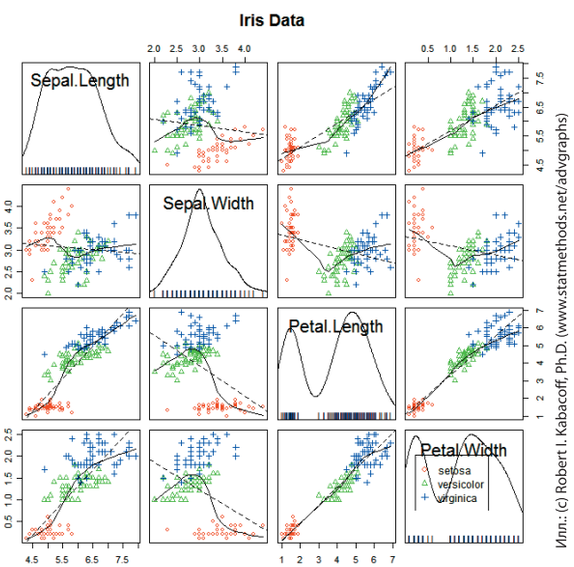
Зачем же нужен этот вопрос? Или, другими словами, что будет, если ответить положительно? В этом случае в рабочую папку R (ту, из которой он вызван), запишутся два файла: бинарный **.RData** и текстовый **.Rhistory**. Первый содержит все объекты, созданные за время сессии. Второй – полную историю введённых команд. R работает с историей команд стандартным образом: доступ к предыдущей команде осуществляется через клавишу-стрелку «вверх».

Если при выходе сохранить файл **.Rhistory**, то команды из этой сессии будут доступны и в следующей, при условии, что R будет вызван из того же самого каталога. И наоборот, если случайно сохранить рабочую среду (эти два файла), то при следующем старте они загрузятся автоматически. Иногда такое поведение R становится причиной различных недоумений, так что необходимо быть внимательным.

Итак, как войти и как выйти, уже понятно. Осталось сказать ещё немного про помощь. Её можно вызвать несколькими различными способами. Во-первых, посредством команд ? или help(), как описано выше. Во-вторых, можно вызвать команду help.start(). В этом случае откроется окно браузера, в котором будет демонстрироваться так называемая HTML-помощь. Основное её преимущество перед обычной текстовой в том, что разделы соединены гиперссылками (в Mac OS X такая помощь вызывается обычными командами). В-третьих, вместе с R устанавливается несколько руководств в формате PDF. Их можно найти в папке, содержащей документацию.

Наконец, часто бывает нужна «обратная» помощь – вы знаете, что вы хотите, но не знаете, как это сделать (какую команду вызвать). В этом случае могут помочь две команды: help.search() и apropos(). Вот как их надо вызывать:

```
> help.search(“vector”)
```



Результатом выполнения будет список команд с кратким описанием их действия. Команда apropos() выдает простой список команд, содержащих строку, которая была указана в кавычках. Кстати, обратите внимание – кавычки следует использовать обязательно. Их можно использовать и в обычных командах помощи, например, help(“q”) или ?“q”, причём иногда эти команды без кавычек просто не работают (например, нельзя получить справку по символу плюса, введя ?+, надо использовать ?“+”). Если ничего не помогает, и найти нужную функцию не удается, то приходится обращаться за справкой в Интернет или в список рассылки R-help.

Калькулятор-переросток

Так назвал один из вводных разделов своей книги «Introduction to R» один из создателей системы, Питер Далгаард [Peter Dalgaard]. Это значит, что R можно использовать, в том числе, и как обычный калькулятор. Например:

```
> 3+2
[1] 5
> 16+3/5-11*8^2
[1] -687.4
> ((((16+3)/5)-11)*8)^2
[1] 3317.76
```

Для знакомых с интерактивными языками программирования типа Python здесь нет ничего необычного. Единственная любопытная деталь – это единичка в квадратных скобках. Она означает номер элемента вектора. Отсюда сразу два логических вывода:

1. R трактует результат любой операции с числами как вектор единичной длины. Скаляров в R, вообще говоря, нет;
2. Элементы векторов нумеруются с единицы, а не с нуля, как принято во многих языках программирования.

Для читателей, менее знакомых с программированием, отметим, что порядок арифметических действий в R стандартный, знакомый со школьной математики. Скобки (раскрывающиеся изнутри наружу) позволяют этот порядок изменять:

```
> # Первый пример
> 3/7
[1] 0.4285714
> 3/7-0.4285714
[1] 2.857143e-08
> # Второй пример
> sqrt(2)*sqrt(2)
[1] 2
> (sqrt(2)*sqrt(2))-2
[1] 4.440892e-16
```

Эти примеры посложнее; кроме того, в них есть подводные камни. Почему разность **3/7-0.4285714** не равна нулю, должно быть понятно, всем, знающим арифметику. При выводе на консоль, R неявно использует функцию print(), которая округляет бесконечную периодическую дробь **0,428571**. Второй пример интереснее. Произведение двух квадратных корней из двойки должно давать **2**, что и происходит. Однако если вычесть из результата **2**, получится какое-то очень маленькое число (**$4{,}440892 \cdot 10^{-16}$**). Это происходит потому, что вычисления выполняются на компьютере, который только притворяется, что работает с дробями, в то время как на самом деле оперирует только с целыми числами. Любая компьютерная система расчётов работает подобным образом, и с этим можно только смириться. Ещё один момент: приведённые примеры показывают, как можно пользоваться символом комментария (#).

Скажем ещё немного о работе с аргументами на примере команды round() (округлить). Она принимает два аргумента: число, которое нужно округлить, и значение digits, сообщающее, до какого знака округлять. Система аргументов работает разумно, так что все равно, как именно написать команду:

```
> round(1.5, digits=0)
[1] 2
> round(1.5, d=0)
```

```
[1] 2
> round(d=0, 1.5)
[1] 2
> round(1.5, 0)
[1] 2
> round(1.5,)
[1] 2
> round(1.5)
[1] 2
```

Так происходит благодаря тому, что есть значения аргументов по умолчанию. Например, в данном случае значение по умолчанию для аргумента digits – **0**. Об этом говорит и результат вывода команды args():

```
> args(round)
> function (x, digits = 0)
```

Можно также заметить, что некоторые аргументы имеют имена. Значит, аргументы можно задавать не по порядку, а по именам. Имена можно сокращать вплоть до одной буквы, но только если нет разных аргументов, которые от такого сокращения станут неразличимыми. Можно также перечислять аргументы по порядку, через запятую (не забудьте, что для десятичных дробей используется точка!), тогда имена можно опускать.

Скрипты

Просто открыть сессию R и вводить в окно программы команды, одну за другой – это лишь один из возможных способов работы. Гораздо более продуктивный метод, который является заодно и серьёзнейшим преимуществом R – это создание скриптов, иначе говоря – программ, которые потом загружаются в R и интерпретируются им. С самого начала работы следует создавать скрипты, даже для таких задач, которые кажутся пустяковыми – в будущем это значительно сэкономит ваше бесценное время. Создание скриптов по любому поводу и даже без особого повода – одна из основ культуры работы в R.

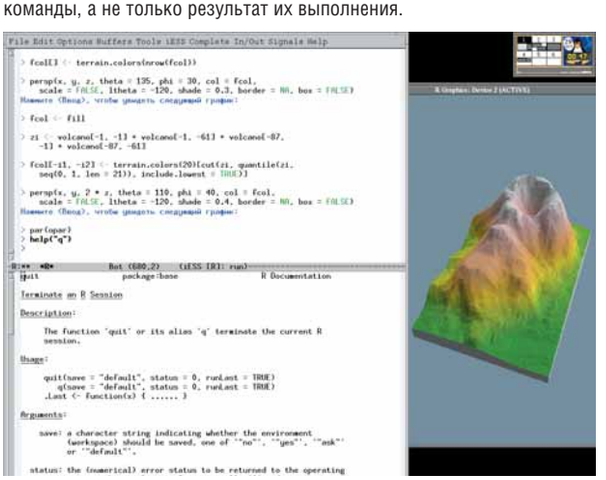
Создать скрипт очень просто. Допустим, после открытия сессии была введена необходимая для получения искомого результата последовательность команд. Чтобы сделать эту работу воспроизводимой, надо просто сохранить историю. Лучше всего это делать с помощью команды:

```
> savehistory(file="myscript.r")
```

После этого в текущем каталоге будет создан файл **myscript.r**, который легко отредактировать в любимом текстовом редакторе, а потом загрузить обратно в R командой:

```
> source("myscript.r", echo=TRUE)
```

Опция echo добавлена для того, чтобы можно было видеть сами команды, а не только результат их выполнения.



Илл.: (c) Robert I. Kabacoff, Ph.D. (www.statmethods.net/advgraphs)

Есть ещё более эффективный способ работы: вы открываете скрипт в текстовом редакторе, а потом посылаете отдельные его строки прямо в R. Есть несколько редакторов, которые умеют так делать. Во-первых, это *Emacs* с установленным пакетом *ESS* (*Emacs Speaks Statistics*). Прелесть этой системы в том, что R запускается прямо в одном из окон редактора. Во-вторых, штатные R GUI под Windows и под Mac также имеют встроенные редакторы скриптов. Правда, Windows-редактор не подсвечивает синтаксис, и вообще довольно неудобен. Вместо него тем пользователям, которых пугает перспектива освоения *Emacs*, можно порекомендовать специализированный редактор *Tinn-R* (www.sciviews.org/Tinn-R/).

Скрипт R можно выполнить, и не запуская интерактивную сессию. Для этого используются специальные опции командной строки. Например, можно поступить вот так:

```
=> R --no-save < myscript.r > out
```

Опция --no-save предписывает R не сохранять результаты сессии в файлах **.RData/.Rhistory** (фактически, отвечает «no» на упомянутый в начале статьи заключительный вопрос).

Пакеты

Ещё одно важное преимущество R – наличие для него многочисленных расширений или пакетов буквально на все случаи жизни.

Несколько пакетов присутствуют сразу после установки R на компьютер – это так называемые базовые пакеты, без которых система просто не работает (скажем, пакет, который так и называется base, или пакет *grDevices*, который управляет выводом графиков), а также «рекомендованные» пакеты (пакет для специализированного кластерного анализа *cluster*, пакет для анализа нелинейных моделей *nlme* и другие).

Кроме того, можно поставить любой из почти полутора тысяч (!) доступных на CRAN пакетов. При наличии доступа в Интернет, это можно сделать прямо из R командой install.packages() (в Mac и Windows есть соответствующие пункты меню). Если соединение с сетью похуже, то можно скачать исходные тексты (под GNU/Linux) или скомпилированные пакеты (под Mac или Windows) и установить их прямо с диска. Поскольку многие пакеты написаны на FORTRAN или C, перед использованием их необходимо скомпилировать. Для этого существует специальная форма вызова R:

```
=> R CMD INSTALL package.tar.gz
```

Естественно, это всё следует делать, если R устанавливается не из стандартного репозитория дистрибутива GNU/Linux. В противном случае можно поискать пакеты по сочетанию "cran". В Debian GNU/Linux Etch таких ровно 85 – это, конечно, не 1500 пакетов с CRAN, но скорее всего, там уже есть многое из того, что вам необходимо.

Пакет готов к работе сразу после установки – нужно только инициализировать его перед употреблением. Для этого служит команда library().

Полезные ссылки

В заключение хотелось бы представить список самых полезных, на наш взгляд, сетевых ресурсов по R:

- http://www.r-project.org/ – сайт проекта
- http://cran.r-project.org/ – CRAN
- https://stat.ethz.ch/pipermail/r-help/ – список рассылки R-help
- http://finzi.psych.upenn.edu/nmz.html – поиск в материалах по R
- http://www.statmethods.net/index.html – хороший справочный ресурс
- http://zoonek2.free.fr/UNIX/48_R/all.html – ещё один справочный ресурс
- http://pj.freefaculty.org/R/Rtips.html} – советы по использованию R

Пожалуй, для первого раза достаточно. LXF

Через месяц Мы узнаем, как подготавливать данные и строить по ним графики.