**Алексей Шипунов. Наглядная статистика. Используем R!**

Для статистической обработки данных и их последующего визуального представления большинство пользователей применяют Excel (см., например, [Левин. Статистика для менеджеров с использованием Microsoft Excel](http://baguzin.ru/wp/?p=5285)). Однако, если у вас нестандартные задачи, как в смысле обработки данных, так и их представления, вас может заинтересовать статистическая среда R. Освоив R, вы сможете автоматизировать свою работу, запускать статистическую обработку прямо из текста документа, получать оригинальные графики высокого качества и сохранить их в переносимых форматах, легко повторить анализ при изменении исходных данных.

Алексей Шипунов и др. Наглядная статистика. Используем R! – М.: ДМК Пресс, 2014. – 298 с.



Купить цифровую книгу в [ЛитРес](http://www.litres.ru/e-baldin/naglyadnaya-statistika-ispolzuem-r-6571649/?lfrom=13042861), бумажную книгу в [издательстве](http://dmkpress.com/catalog/computer/statistics/978-5-94074-828-1/), [Ozon](http://www.ozon.ru/context/detail/id/7952180/?partner=baguzin) или [Лабиринте](http://www.labirint.ru/books/427978/?p=13320)

Приведенные в книге примеры можно найти на [веб-странице](http://ashipunov.info/shipunov/software/r/) авторов. Там же находятся разные полезные ссылки и те файлы данных, которые не поставляются вместе с программой. Все десятичные дроби представлены в виде чисел с разделителем-точкой (типа 10.4), а не запятой (типа 10,4). Это сделано потому, что программа R по умолчанию «понимает» только первый вариант дробей. Примеры начинаются со значка «больше» >. Если пример не умещается на одной строке, все последующие его строки начинаются со знака «плюс» (не набирайте эти знаки, когда будете выполнять примеры!). Если в книге идет речь о загрузке файлов данных, то предполагается, что все они находятся в поддиректории data в текущей директории.

В главе 1 рассказывается о самых общих понятиях анализа данных: генеральной совокупности, выборке, рандомизации при отборе, двойном слепом эксперименте, корреляции.

### Глава 2. Как обрабатывать данные

К неспециализированным программам можно отнести Excel. Поскольку статистическая обработки никогда не была для него приоритетной, его возможности в этой области относительно ограничены (хотя для начинающих и их вполне достаточно). Специализированные статистические программы можно подразделить на оконно-кнопочные системы и статистические среды. Первые не особенно отличаются от электронных таблиц, однако снабжены значительно большим арсеналом доступных статистических приемов. Кроме того, у них традиционно мощная графическая часть. В России популярна система STATISTICA. Серьезным, преимуществом STATISTICA является наличие переведенной на русский язык системы помощи, свободно доступной в Интернете. Также издано немало книг. Также используются программы STADIA и PAST. В отличие от двух предыдущих программ, PAST распространяется бесплатно.

*Статистические среды.* Эта группа программ использует в основном интерфейс командной строки. Одна из наиболее продвинутых систем этого плана — это SAS (к сожалению, очень дорогая).

 Это коммерческая, очень мощная система, обладающая развитой системой помощи и имеющая долгую историю развития. Создавалась она для научной и экономической обработки данных и до сих пор является одним из лидеров в этом направлении. Написано множество книг, описывающих работу с SAS и некоторые ее алгоритмы. Вместе с тем система сохраняет множество рудиментов 70х годов, и пользоваться ей поначалу не очень легко даже человеку, знакомому с командной строкой и программированием. А стоимость самой системы просто запредельная — многие тысячи долларов!

R — это среда для статистических расчетов. Количество книг, написанных про R, за последние годы выросло в несколько раз, а количество пакетов уже приближается к трем с половиной тысячам! У R два главных преимущества: неимоверная гибкость и свободный код. У R есть и недостатки. Главный — трудность обучения программе, второй — относительная медлительность.

Для того чтобы установить R под Windows, скачайте пакет с [CRAN](https://cran.gis-lab.info/) (70Mb). На рабочем столе появится пиктограмма, при клике на нее запускается R консоль (рис. 1). С R можно также работать через сторонние графические интерфейсы, например, [R Commander](http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/) или [RStudio](https://www.rstudio.com/).



Рис. 1. R консоль

**Первые шаги.** В R любая команда имеет аргумент в круглых скобках. Если аргумент не указан, скобки все равно нужны. Если ввести функцию без скобок получите определение функции. Для более развернутой информации по функции введите help().[[1]](#footnote-1) При выходе из R в папку С:\Пользователи\[Имя пользователя]\Документы запишутся два файла: бинарный .RData и текстовый .Rhistory. Первый содержит все объекты, созданные вами за время сессии. Второй — полную историю введенных команд. Когда вы работаете в R, предыдущую команду легко вызвать, нажав стрелку «вверх». Если вы сохраните файл .Rhistory, ваши команды и объекты будут доступны и в следующей сессии.

**Как загружать данные.** Данные можно набрать прямо в R, используя функцию c(), которая объединяет аргументы в один вектор

> a <- c(1,2,3,4,5)

> a

[1] 1 2 3 4 5

Можно воспользоваться встроенной в R подпрограммой — электронной таблицей наподобие сильно упрощенного Excel, для этого надо набрать команду data.entry(b). В появившейся таблице можно редактировать данные «на месте» (рис. 2). Однако лучше научиться загружать в R файлы, созданные при помощи других программ, например, Excel.



Рис. 2. Редактор данных

Для того чтобы R «усвоил» данные, надо убедиться, что текущая папка в R и та папка, откуда будут загружаться данные, совпадают. Введите getwd() и программа вернет путь к текущей папке:

[1] "C:/Users/Сергей/Documents"

Поменять рабочую папку можно командой setwd() (рис. 3). Важно: указывайте два слэша.



Рис. 3. Смена рабочей папки

Все, что написано на строчке после символа «#» — это комментарий. Комментарии R пропускает, не читая). Например,

> save(x, file="x.rd") # Сохранить объект "x"

Для загрузки данных используйте команду read.table("mydata.txt", sep=";", head=TRUE). Аргументы указывают, что разделителем является точка с запятой, а у столбцов есть имена (рис. 4).



Рис. 4. Загрузка данных командой read.table

Если в качестве десятичного разделителя чисел используется запятая (по умолчанию, точка), укажите явным образом аргумент dec: read.table("mydata3.txt", dec=",", sep=";", h=T).

Для чтения бинарных данных, выводимых пакетами MiniTab, S, SAS, SPSS, Stata, Systat, а также формата DBF, загрузите предустановленный пакет командой library(foreign). Посмотреть, какие функцию включены в эту библиотеку можно командой help(package=foreign). Для загрузки данных из MS Excel наиболее привлекательным представляется обмен с R через буфер. Скопируйте в буфер ячейки Excel, и загрузите их в R командой read.table("clipboard").

R может загружать изображения, например, с помощью пакета pixmap, карты в формате ArcInfo и др. (пакеты maps, maptools). Чтобы загрузить эти пакеты, их нужно скачать из репозитория. Для этого пройдите по меню R консоли: *Пакеты* –> *Установить пакет(ы)*, выберите репозиторий, а затем в открывшемся окне *Packages* один или несколько пакетов для установки (рис. 5).



Рис. 5. Установка пакета из репозитория

**Как сохранять результаты.** Для сохранения таблиц данных в виде текста, который затем можно открыть в офисных приложениях, служит команда write.table(). Например,

> write.table(trees, file="trees.csv", row.names=FALSE, sep=";", quote=FALSE)

В текущую папку будет записан файл trees.csv, созданный из встроенной в R таблицы данных trees. «Встроенная таблица» означает, что эти данные доступны в R безо всякой загрузки, напрямую. Кстати говоря, узнать, какие таблицы уже встроены, можно командой data().

R можно использовать как **калькулятор**.

**Графики.** В базовом наборе есть несколько десятков типов графиков, еще больше в рекомендуемом пакете lattice, и более 1000 — в пакетах с CRAN. Начнем с простейшего примера (рис. 6):

> plot(1:20, main="Заголовок")

> legend("topleft", pch=1, legend="Мои любимые точки")

plot() — основная графическая команда, причем команда «умная»; она распознает тип объекта, который подлежит рисованию, и строит соответствующий график. Например, в приведенном примере 1:20 — это последовательность чисел от 1 до 20, то есть вектор, а для «одиночного» вектора предусмотрен график, где по оси абсцисс — индексы (то есть номера каждого элемента вектора по порядку), а по оси ординат — сами эти элементы.



Рис. 6. Пример графика с заголовком и легендой

Если в аргументе команды будет что-то другое, будет построен иной график. Например,

> plot(cars)

> title(main="Автомобили двадцатых годов")

cars — это встроенная в R таблица данных, которая использована здесь по прямому назначению, для демонстрации возможностей программы (прочитать, что такое cars, можно, вызвав справку – ?cars). Здесь данные — не вектор, а таблица из двух колонок — скорость и тормозной путь. Функция plot() автоматически нарисовала диаграмму рассеивания, где по оси x откладывается значение одной переменной, а по оси у — другой, и присвоила осям имена этих колонок (рис. 7).



Рис. 7. Пример диаграммы рассеивания на основе встроенных данных

**Графические устройства.** Когда вы вводите команду plot(), R открывает так называемое экранное графическое устройство и начинает вывод на него. Когда вы сохраняете график в файле, R откроет другое графическое устройство. Например,

> png(file="1-20.png", bg="transparent")

> plot(1:20)

> dev.off()

Команда png() открывает одноименное графическое устройство, и задает прозрачный фон (удобно для веб-дизайна). У экранных устройств такого параметра нет. Команда dev.off(), устройство закрывает, и на диске появляется файл 1-20.png. png() — растровое устройств для записи файлов. R поддерживает и векторные форматы, например, PDF. Чтобы побороть специфические для русскоязычного пользователя трудности со шрифтами, необходимо указать шрифт по умолчанию. Затем нужно закрыть графическое устройство и встроить в полученный файл шрифты. В противном случае кириллица может не отобразиться!

> pdf("1-20.pdf", family="NimbusSan")

> plot(1:20, main="Заголовок")

> dev.off()

> embedFonts("1-20.pdf")

Важно отметить, что шрифт «NimbusSan» и возможность встраивания шрифтов командой embedFonts() обеспечивается взаимодействием R со сторонней программой Ghostscript, в поставку которой входят шрифты, содержащие русские буквы (я скачал программу с сайта [freesoft.ru](http://freesoft.ru/ghostscript/download)). Поскольку я работаю под Windows, то R должен «знать» путь к программе gswin64c.exe (другими словами, нужно, чтобы путь к этой программе был записан в системную переменную PATH), иначе встраивание шрифтов не сработает. В Windows10 пройдите по меню *Пуск* –> *Система* –> *Дополнительные параметры системы*. Кликните на кнопку *Переменные среды*. В окне *Системные переменные* выберите *Path*, и добавьте путь к программе gswin64c.exe.

**Графические опции.** Если, например, вам нужно нарисовать два графика один над другим на одном рисунке, надо изменить исходные опции — разделить пространство рисунка на две части (рис. 8).



Рис. 8. Две гистограммы на одном графике

В какой-то момент я понял, что мне удобнее писать код в файле с расширением R (в блокноте), и запускать его из консоли; меню *Файл* –> *Загрузить код R*. Ключевая команда здесь — par(). В первой строчке изменяется один из ее параметров, mfrow, который регулирует, сколько изображений и как будет размещено на «листе». Значение mfrow по умолчанию — c(1,1), то есть один график по вертикали и один по горизонтали. Чтобы не печатать каждый раз команду par() со всеми ее аргументами, мы «запомнили» старое значение в объекте old.par, а в конце вернули состояние к запомненному. Команда hist() строит гистограмму.

**Интерактивная графика** позволяет поместить объект (скажем, подпись) в нужное место. Например, можно добавлять подпись в указанную мышкой область графика (пока вы еще не начали проверять — после того как введена вторая команда, надо щелкнуть левой кнопкой мыши на точке, которую хотите подписать, а затем щелкнуть в любом месте графика правой кнопкой мыши и в появившемся меню выбрать *Остановить*):

> plot(1:20)

> text(locator(), "Моя любимая точка", pos=4)

### Глава 3. Типы данных

Температура и расстояние изменяются плавно и непрерывно. Любые два показателя температуры или расстояния представляют собой интервал, куда «умещается» бесконечное множество других показателей. Поэтому они называются **интервальными данными**. А вот число людей тоже можно упорядочить, но не всегда существует промежуточное значение, потому что люди на части не делятся. Это другой тип интервальных данных, не непрерывный, а дискретный.

Статистические тесты делятся на две большие группы: параметрические и непараметрические. Чтобы данные считались параметрическими, должны одновременно выполняться три условия:

1. распределение данных близко к нормальному;
2. выборка — большая (обычно не менее 30 наблюдений);
3. данные — интервальные непрерывные.

Если хотя бы одно из этих условий не выполняется, данные считаются непараметрическими и обрабатываются непараметрическими тестами. В R интервальные данные представляют в виде числовых векторов. Чаще всего один вектор — это одна выборка. Например, данные о росте семи сотрудников компании можно представить в виде вектора х:

> x <- c(174, 162, 188, 192, 165, 168, 172)

x — это имя объекта R, «<-» — функция присвоения, c() — функция создания вектора. Собственно, R и работает в основном с объектами и функциями. У объекта может быть своя структура:

> str(x)

num [1:7] 174 162 188 192 165 168 172

То есть x — числовой вектор (в R нет скаляров, «одиночные» объекты трактуются как векторы из одного элемента). Вот так можно проверить, вектор ли перед нами:

> is.vector(x)

[1] TRUE

Названия объектов подчиняются следующим правилам: (а) используйте только латинские буквы, цифры и точку (имена объектов не должны начинаться с точки или цифры); (б) R чувствителен к регистру, X и x — это разные имена; (в) не давать объектам имена, уже занятые функциями и ключевыми словами.

Для создания векторов очень полезен оператор «:», обозначающий интервал, а также функции создания последовательностей seq() и повторения rep().

**Шкальные данные** (например, школьная оценка).К шкальным данным применимы очень многие из тех методов, которые используются для обработки интервальных непрерывных данных. Однако к числовым результатам обработки надо подходить с осторожностью, всегда помнить об условности значений шкалы. По умолчанию R будет распознавать шкальные данные как обычный числовой вектор. Если же стоит задача создать шкальные данные из интервальных, то можно воспользоваться функцией cut(). Для анализа шкальных данных всегда используются непараметрические методы.

**Номинальные данные** (или категориальные), в отличие от шкальных, нельзя упорядочивать. Обычные численные методы для номинальных данных неприменимы. Особый случай как номинальных, так и шкальных данных — бинарные данные. Бинарные данные можно представить и в виде «логического вектора», то есть набора значений TRUE или FALSE. Самая главная польза от бинарных данных — в том, что в них можно перекодировать практически все остальные типы данных. В R можно создать текстовый (character) вектор:

> sex <- c("male", "female", "male", "male", "female", "male", "male")

> is.character(sex)

[1] TRUE

> is.vector(sex)

[1] TRUE

> str(sex)

chr [1:7] "male" "female" "male" "male" "female" "male" ...

На первых порах пользователь R не всегда понимает, с каким типом объекта (вектором, таблицей, списком и т.п.) он имеет дело. Разрешить сомнения помогает str().

Предположим теперь, что sex — это описание пола сотрудников небольшой организации. Вот как R выводит содержимое этого вектора:

> sex

[1] "male" "female" "male" "male" "female" "male" "male"

В квадратных скобках выводится номер элемента вектора. Вот как его можно использовать (квадратные скобки — это тоже команда, можно это проверить, набрав помощь ?"["):

> sex[1]

[1] "male"

Объект-ориентированные, команды R кое-что понимают про объект sex, например команда table():

> table(sex)

sex

female male 2 5

А вот команде plot() сначала нужно сообщить, что этот вектор надо рассматривать как фактор (то есть номинальный тип данных):

> sex.f <- factor(sex)

> sex.f

[1] male female male male female male male

Levels: female male

Теперь команда plot() уже «понимает», что ей надо делать — строить столбчатую диаграмму:

> plot(sex.f)

Это произошло потому, что перед нами специальный тип объекта, предназначенный для категориальных данных,— фактор с двумя уровнями (градациями) (levels):

> is.factor(sex.f)

[1] TRUE

> is.character(sex.f)

[1] FALSE

> str(sex.f)

Factor w/ 2 levels "female", "male": 2 1 2 2 1 2 2

Некоторые свойства факторов. Во-первых, подмножество фактора — это фактор с тем же количеством уровней, даже если их в подмножестве не осталось:

> sex.f[6:7]

[1] male male Levels: female male

Во-вторых, факторы можно легко преобразовать в числовые значения:

> as.numeric(sex.f)

[1] 2 1 2 2 1 2 2

Зачем это нужно, становится понятным, если рассмотреть вот такой пример. Положим, кроме роста, у нас есть еще и данные по весу сотрудников:

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

И мы хотим построить такой график, на котором были бы видны одновременно рост, вес и пол. Вот как это можно сделать (рис. 9; pch и col — эти параметры предназначены для определения соответственно типа значков и их цвета на графике):

> plot(x, w, pch=as.numeric(sex.f), col=as.numeric(sex.f))

> legend("topleft", pch=1:2, col=1:2, legend=levels(sex.f))



Рис. 9. График, показывающий одновременно три переменные

В-третьих, факторы можно упорядочивать. Введем четвертую переменную — размер маек для тех же самых семерых сотрудников:

> m <- c("L", "S", "XL", "XXL", "S", "M", "L")

> m.f <- factor(m)

> m.f

[1] L S XL XXL S M L

Levels: L M S XL XXL

Как видим, уровни расположены просто по алфавиту, а нам надо, чтобы S (small) шел первым. Кроме того, надо как-то сообщить R, что перед нами — шкальные данные. Делается это так:

> m.o <- ordered(m.f, levels=c("S", "M", "L", "XL", "XXL"))

> m.o

[1] L S XL XXL S M L

Levels: S < M < L < XL < XXL

Теперь R «знает», какой размер больше. Это может сыграть критическую роль — например, при вычислениях коэффициентов корреляции. Работая с факторами, нужно помнить и об одной опасности. Если возникла необходимость перевести фактор в числа, то вместо значений вектора мы получим числа, соответствующие уровням фактора! Чтобы этого не случилось, надо сначала преобразовать фактор, состоящий из значений-чисел, в текстовый вектор, а уже потом — в числовой:

> a <- factor(3:5)

> a

[1] 345 Levels: 345

> as.numeric(a) # Неправильно!

[1] 123

> as.numeric(as.character(a)) # Правильно!

[1] 3 4 5

Когда файл данных загружается при помощи команды read.table(), то все столбцы, где есть хотя бы одно нечисло, будут преобразованы в факторы. Если хочется этого избежать (для того, например, чтобы не столкнуться с вышеописанной проблемой), то нужно задать дополнительный параметр: read.table(..., as.is=TRUE).

**Доли, счет и ранги: вторичные данные.** Наибольшее применение вторичные данные находят при обработке шкальных и в особенности номинальных данных, которые нельзя обрабатывать «в лоб». Для визуализации процентов в Excel используют графики-блины и столбчатые диаграммы. Однако, многочисленные эксперименты доказали, что читаются такие графики гораздо хуже остальных. В R есть отличная альтернатива – точечные графики (рис. 10).



Рис. 10. Точечный график результатов гадания на ромашках (проценты исходов)

Если счет и доли получают обычно из номинальных данных, то отношения и ранги «добывают» из данных количественных. Отношения особенно полезны в тех случаях, когда изучаемые явления или вещи имеют очень разные абсолютные характеристики. Например, вес людей довольно трудно использовать в медицине напрямую, а вот соотношение между ростом и весом очень помогает в диагностике ожирения. Чтобы получить ранги, надо упорядочить данные по возрастанию и заменить каждое значение на номер его места в полученном ряду. Ранги особенно широко применяются при анализе шкальных и непараметрических интервальных данных.

**Пропущенные данные.** В R пропущенные данные принято обозначать двумя большими буквами латинского алфавита *NA*. Предположим, что у нас имеется результат опроса семи сотрудников. Их спрашивали, сколько в среднем часов они спят, при этом некоторые не ответили:

> h <- c(8, 10, NA, NA, 8, NA, 8)

> h

[1] 8 10 NA NA 8 NA 8

Чтобы высчитать среднее от «непропущенной» части вектора, можно поступить одним из двух способов:

> mean(h, na.rm=TRUE)

[1] 8.5

> mean(na.omit(h))

[1] 8.5

Первый способ разрешает функции mean() принимать пропущенные данные, а второй делает из вектора h временный вектор без пропущенных данных (они просто выкидываются из вектора). В R существует пакет, предоставляющий графический интерфейс для «борьбы» с пропущенными данными, MissingDataGUI (как это делается в Excel, см. [Отражение пропущенных данных на графиках Excel](http://baguzin.ru/wp/?p=1690)).

**Основные принципы преобразования данных** в R (данные находятся в векторе data):

* Логарифмическое: log(data + 1). Поскольку преобразование «боится» нулей в данных, рекомендуется прибавлять единицу. В графических командах R есть специальный аргумент log="ocь", где вместо слова ось надо подставить x или у, и тогда соответствующая ось графика отобразится в логарифмическом масштабе.
* Квадратного корня: sqrt(data). Похоже по действию на логарифмическое. «Боится» отрицательных значений.
* Обратное: 1/(data + 1). Эффективно для стабилизации дисперсии. «Боится» нулей.
* Квадратное: data~2. Если распределение скошено влево, может дать нормальное распределение. Линеаризует зависимости и выравнивает дисперсии.
* Логит: log(p/(1 – p)). Чаще всего применяется к пропорциям. Линеаризует так называемую сигмовидную кривую. Кроме логит-преобразования, для пропорций часто используют и арксинус-преобразование, asin(sqrt(p))

**Матрицы** в R могут быть разной размерности (2-, 3- и более мерные). Матриц как таковых в R нет. Чтобы создать матрицу выполните следующие команды:

> mb <- 1:4

> attr(mb, "dim") <- c(2,2)

> mb

[,1] [,2]

[1,] 1 3

[2,] 2 4

Мы присваиваем атрибуту dim вектора mb значение в c(2,2), то есть 2 строки и 2 столбца.

**Списки.** Рассмотрим пример

> l < list("R", 1:3, TRUE, NA, list("r", 4))

> l

[[1]]

[1] "R"

[[2]]

[1] 1 2 3

[[3]]

[1] TRUE

[[4]]

[1] NA

[[5]]

[[5]][[1]]

[1] "г"

[[5]][[2]]

[1] 4

Список — это своего рода ассорти. Вектор (и, естественно, матрица) может состоять из элементов одного и того же типа, а вот список — из чего угодно, в том числе и из других списков. Элементы вектора выбираются при помощи функции — квадратной скобки:

> h[3]

[1] 8.5

Элементы матрицы выбираются так же, только используются несколько аргументов (для двумерных матриц это номер строки и номер столбца — именно в такой последовательности):

> ma[2, 1]

[1] 3

А вот элементы списка выбираются тремя различными методами. Во-первых, можно использовать квадратные скобки (возвращаемый объект – тоже список):

> l[1]

[[1]]

[1] "R"

Во-вторых, можно использовать двойные квадратные скобки (возвращаемый объект будет того типа, какого он был бы до объединения в список):

> l[[1]]

[1] "R"

В-третьих, можно использовать имена элементов списка. Но для этого сначала надо их создать:

> names(l) <- c("first", "second", "third", "fourth", "fifth")

> l$first

[1] "R"

Для выбора по имени употребляется знак доллара, а полученный объект будет таким же, как при использовании двойной квадратной скобки.

**Таблицы данных** — это гибридный тип представления, одномерный список из векторов одинаковой длины. Таким образом, каждая таблица данных — это список колонок, причем внутри одной колонки все данные должны быть одного типа, а вот сами колонки могут быть разного типа (рис. 11).



Рис. 11. Таблица данных

С помощью логического вектора можно отобрать из таблицы только данные, относящиеся к женщинам (рис. 12). После того как «отработала» селекция

> d$sex=="female"

[1] FALSE TRUE FALSE FALSE TRUE FALSE FALSE

в таблице данных остались только те строки, которые соответствуют TRUE, то есть строки 2 и 5. Знак «==», а также знаки «&», «|» и «!» используются для замены соответственно «равен?», «и», «или» и «не».



Рис. 12. Отбор данных из таблицы с помощью логического вектора

Для сортировки используется команда order() (рис. 13).



Рис. 13. Сортировка таблицы

На этом, собственно, завершается введение в R, и авторы переходят к использованию R в статистическом анализе: центральная тенденция и разброс, статистический тест, корреляция, регрессия, кластерный анализ, временные ряды, статистическая разведка. Книга также содержит полезные приложения с перечнем основных функций и операторов. На мой взгляд, для начинающих, то, что надо!

1. Определения некоторых функций на русском языке можно найти в [блоге](http://r-analytics.blogspot.ru/p/blog-page_06.html#.V0F005GLSUk) Сергея Мастицкого. [↑](#footnote-ref-1)