История возникновения и основные принципы организации среды R

Система статистического анализа и визуализации данных R состоит из следующих основных частей:

° языка программирования высокого уровня R, позволяющего одной строкой

реализовать различные операции с объектами, векторами, матрицами, списками и т.д.;

° большого набора функций обработки данных, собранных в отдельные пакеты(package);

° развитой системой поддержки, включающей обновление компонентов среды.

Начало пути относится к 1993 г., когда двое молодых новозеландских учёных Росс Ихака (Ross Ihaka) и Роберт Джентльмен (Robert Gentleman), анонсировали свою новую разработку, которую назвали R. В скором времени возникла распределенная система хранения и распространения пакетов к R, известная под аббревиатурой "CRAN" (Comprehensive R Archive Network –

http://cran.r-project.org), основная идея организации которой – постоянное расширение, коллективное тестирование и оперативное распространение прикладных средств обработки данных.

Скачать дистрибутив системы вместе с базовым набором из 29 пакетов (54 мегабайта) можно совершенно бесплатно с основного сайта проекта http://cran.r-project.org. Процесс инсталляции системы из скачанного дистрибутива затруднений не вызывает и не требует никаких особых комментариев.

Из чего состоит оболочка Rgui и как в ней работать?

После запуска Rgui Вы увидите перед собой приложение, состоящее из трех элементов: *меню*, *панель инструментов* и *консоль*. Также в ходе работы нам понадобятся две дополнительные области: *графический модуль* и *скриптовое окно*. Мы отдельно рассмотрим значение и функциональность каждой из этих областей, т. к. данная информация необходима для понимания принципов работы в среде R. Расположение данных элементов можно увидеть на рисунке.

Меню

Как и в других программах, меню играет ключевую роль в создании, загрузке и сохранении файлов, настройке графического интерфейса и других параметров приложения. Меню располагается в верхней части Rgui, но состав его опций меняется в зависимости от активного окна. Например, когда мы работаем в графическом модуле, появляется такая опция, как "Экспорт картинок", которая отсутствует во время работы в консоли или в скриптовом окне.

Панель инструментов

Состоит из ряда графических кнопок ("Сохранить", "Копировать" и других), расположенных под областью меню. Функционально кнопки панели инструментов копируют опции меню, но отличаются от них доступностью в один клик.

Консоль и Графический модуль

Консоль - это рабочая область R, внизу которой находится командная срока. Именно туда пользователь вводит команды, и там же видит результат. Другими словами, командная строка работает по принципу "вопрос-ответ". Ответ консоли может выглядеть как описательная статистика для таблицы, результат статистического теста или моделирования. Однако, при необходимости визуализации данных результат будет находиться не в консоли, а в отдельном окне, называемым графическим модулем. Кликнув на модуль, Вы можете сохранить график как растровое (.jpeg, png, .tiff) или векторное (.svg, .eps) изображение.

Скриптовое окно

Данная область просто незаменима при написании собственных алгоритмов команд, называемых скриптами или скриптовыми программами. Я рекомендую все команды писать именно в скриптовом окне и лишь потом переводить их в консоль (для этого воспользуйтесь комбинацией клавиш *CTRL+R*). На это есть две причины. Во-первых, если Вы пользуетесь только консолью и совершили ошибку в предыдущих командах, то Вам придется перезапустить все команды заново, что достаточно проблематично. Во-вторых, записав алгоритм команд в скриптовом окне, Вы можете сохранить его как скрипт (текстовый файл с разрешением .R). Позже можно продолжить работать со скриптом, загрузив его через меню Rgui или же поделиться этим файлом с коллегами. Также в этом окне Вы можете запускать и редактировать чужие скриптовые программы

Чтобы быстро и эффективно загружать свои данные в среду R важно понимать, что такое «рабочая директория». Говоря простыми словами — это папка (директория), где расположен исполняемый файл программы. Для того, чтобы узнать рабочую папку R, следует набрать команду: *> getwd()*

Файлы, расположенные в этой папке, не нуждаются в указании «пути к файлу» при загрузке в среду R, что существенно экономит время, и уменьшает количество букв в коде. Именно поэтому важно не только знать, но и уметь изменять расположение рабочей папки.

Для удобства хранения скриптов, исходных данных и результатов расчетов стоит выделить на пользовательском компьютере специальный рабочий каталог. Весьма нежелательно использовать в названии рабочего каталога символы кириллицы.
Некоторые команды:

getwd() – узнать текущую рабочую директорию (get working directory)

setwd() – изменить рабочую директорию (set working directory)- при выполнении приведенной команды внешне ничего не произойдет, однако последующее применение команды *getwd()* покажет, что путь к рабочей папке изменился.

dir() посмотреть список файлов в текущей директории

dir.create("new") - создать папку

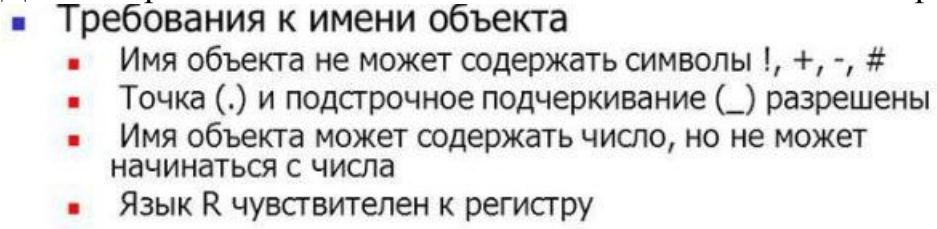
savehistory("new.txt") – сохранить историю набранных команд

***q()* – выход из R**

R – объектно-ориентированный язык: переменные, данные, матрицы, функции, результаты и т.д., хранятся в оперативной памяти компьютера в форме объектов, которые имеют имя с использованием операторов присвоения:

- Стрелка (<-)
- Знак равенства (=)

Для отображения его значения необходимо напечатать в рабочей области имя объекта.



возвращает число элементов, содержащихся в объекте x

length(x)

```
#возвращает минимальный элемент, содержащихся в объекте x
min<-min(x)
#возвращает максимальный элемент, содержащихся в объекте x
max<-max(x)
range(x)
# возвращает порядковый номер элемента объекта x с минимальным значением
which.min(x)
# меняет порядок элементов объекта x на обратный
rev(x)
# сортирует элементы объекта x по возрастанию
sort(x)
# для сортировки по убыванию используйте
rev(sort(x))
t=table(x)
cut(x, breaks)
# делит вектор x на равные интервалы (факторы); в качестве аргумента breaks
# может выступать либо необходимое число интервалов, либо вектор,
# содержащий перечень "точек разрыва"
i=cut(x, breaks=seq(4,32,by=7))
table(i)
hist(x,breaks=20,freq=F)
# bw- параметр сглаживания
lines(density(x,bw=0.4),col="red")

d1
d2<-sort(d1)
d2
hist(d2,breaks=100,col="lightblue")
```

v2<-d2[1]

v2

d2[20]

r<-(d2[20]-d2[1])/5

r

i=cut(d2, seq(from = 8, to = 26.5, by = 3.7),right=T)

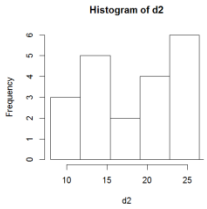
int<-table(i)

int

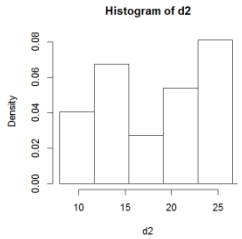
функция `hist()` автоматически выбирает количество столбцов для отображения на графике, а также создает названия осей и заголовок графика.
Прежде всего, важно обратить внимание на размер шага, используемого для разбиения данных на классы при построении гистограммы. Для более детального изучения этих свойств следует выбрать более дробное деление данных на классы (т.е. использовать меньший шаг). Сделать это позволяет аргумент `breaks` (*разломы*) функции `hist()`:

По умолчанию функция `hist()` отображает по оси ординат частоты встречаемости для каждого класса значений `X`. Такое поведение функции можно изменить, придав аргументу `freq` (от *frequency* - частота) значение `FALSE`. В этом случае ось ординат будет отражать плотность вероятности каждого класса так, что суммарна площадь под гистограммой составит 1:

hist(d2,breaks=c(27, seq(8,26.5, 3.7)), freq =T)

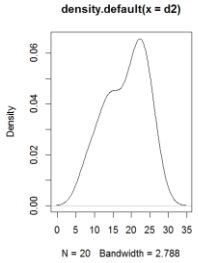


hist(d2,breaks=c(27, seq(8,26.5, 3.7)), freq =F)



Вместо гистограммы (или в добавок к ней) в некоторых случаях стоит воспользоваться *кривой плотности вероятности*. Оценка плотности вероятности выполняется при помощи функции `density()`, которую можно применить в качестве аргумента функции `plot()` для графического изображения результата

plot(density(d2))

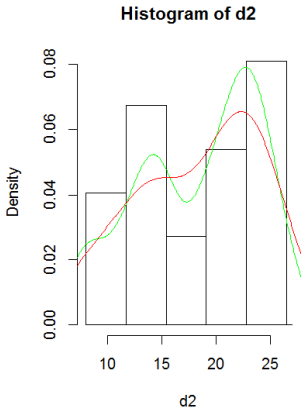


Для полноты картины гистограмму можно совместить с кривой плотности вероятности. При этом сначала необходимо построить саму гистограмму, а затем добавить к ней кривую плотности при помощи функции `lines()`

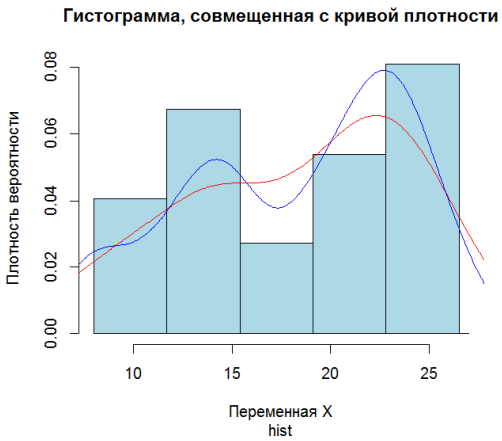
```
hist(d2,breaks=c(27, seq(8,26.5, 3.7)), freq =F)

lines(density(d2),col="red")

lines(density(d2,bw=1.8),col="green")
```




```
hist(d2,breaks=c(27, seq(8,26.5, 3.7)), freq =F,col = "lightblue",
     xlab = "Переменная X",
     ylab = "Плотность вероятности", sub="hist",
     main = "Гистограмма, совмещенная с кривой плотности")
points(x=d2, y=rep(0, length(d2)), col="red", pch=17)
lines(density(d2),col="red")
lines(density(d2,bw=1.8),col=4)
```

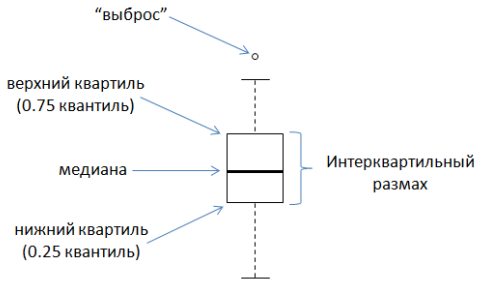


Обратите внимание на дополнительные аргументы, использованные вместе с функцией `hist()`: `xlab` и `ylab` - для создания названий осей, и `main` - для создания заголовка рисунка. Аналогично, в качестве управляющих аргументов функции `lines()` были применены аргументы `col` (для установки цвета линии) и `lwd` (для установки толщины линии). Эти же аргументы мы использовали ранее при построении графиков с помощью функции `plot()`

Диаграммы размахов, или "ящики с усами" (англ. *box-whisker plots*), получили свое название за характерный вид: точку или линию, соответствующую медиане или средней арифметической, окружает прямоугольник ("ящик"), длина которого соответствует одному из показателей разброса или точности оценки генерального параметра. Дополнительно от этого прямоугольника отходят "усы", также соответствующие по длине одному

из показателей разброса или точности. Графики этого типа очень популярны, поскольку позволяют дать очень полную статистическую характеристику анализируемой совокупности. Кроме того, диаграммы размаха можно использовать для визуальной экспресс-оценки разницы между двумя и более группами (например, между датами отбора проб, экспериментальными группами, участками пространства, и т.п.).

В R для построения диаграмм размахов служит функция `boxplot()`. Строение получаемых при помощи этой функции "ящиков с усами" представлено ниже:



boxplot(d2)

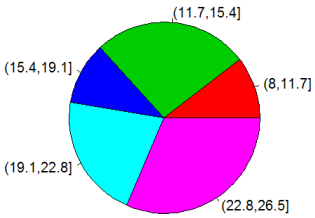
boxplot(d2,col="gray")

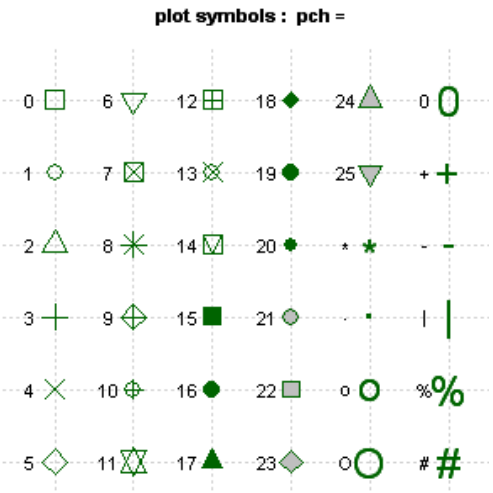
points(mean(d2),col="red",pch=18)

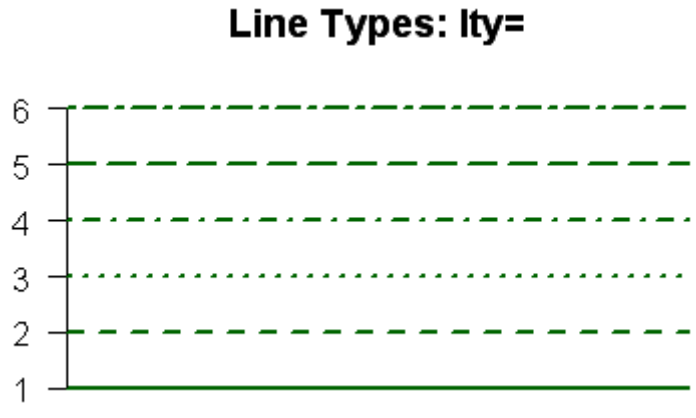
points(mean(d2), pch = 22, col = "darkgrey", lwd = 7)

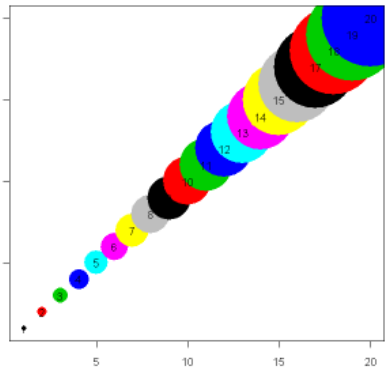
barplot(height = int) – столбчатая диаграмма

pie(int,col=2:6) – круговая диаграмма








5
10
15
20