R-программирование

http://www.r-project.org/

http://cran.r-project.org/

Hung Chen

перевод и адаптация Козелецкий Андрей

R, S and S-plus

- S: интерактивная среда для анализа данных, разработываемая Bell Laboratories с 1976
  - 1988 - S2: RA Becker, JM Chambers, A Wilks
  - 1992 - S3: JM Chambers, TJ Hastie
  - 1998 - S4: JM Chambers
- Экслюзивно лицензированная *AT&T/Lucent* для *Insightful Corporation*, Seattle WA. под именем: “S-plus”.
- Реализовано на C, Fortran.
- R: изначально писалась Ross Ihaka и Robert Gentleman в департаменте статистики штата Оакленд, Новая Зеландия в течении 1990-х.
- С 1997: международной команда“R-core” из 15 человек с синхронизацей через общий CVS архив.

Introduction

- R это “GNU S” — язык и среда для манипуляции с данными, вычислением и графическим отображением.
  - набор вычислительных операторов для работы с массивами в матрицах специального вида,
  - большой связный интегрированный набор промежуточных инструментов для интерактивного анализа данных,
  - графические средства анализа данных и их отображения на экране, сохранения в файле или печати на принтере
  - хорошо проработанный язык программирования, который включает в себя условные операторы, операторы циклов, пользовательские функции, в том числ и рекурсивные и средства ввода-вывода.
- Ядро R является интерпретируемым компьютернымязыком.
  - Он позвозяет ветвления и циклы, а так же модульное программирование с использованием функций.
  - Большинство доступных пользователю в R функций написаны в R, вплоть до небольшого набора внутренних примитовов..
  - И есть возможность подключать процедуры, написанные на C, C++ или FORTRAN для производительности и создания новых примитивов.

Что R может и чего не может

- ✓ обработка данных и их хранение: числа и текст
- ✓ матричная алгебра
- ✓ хэш-таблицы и регулярные выражения
- ✓ высокоуровневый анализ и статистические функции
- ✓ классы (“ОО”)
- ✓ работа с графикой
- ✓ программирование: циклы, ветвления, подпрограммы

- × не является БД, но может к ним подключаться
- × не имеет графического интерфейса, но может использовать Java, TclTk
- × интерпретируемый язык очень медленный, но можно использовать вызовы функций из C/C++
- × нет встроенных таблиц, но можно использовать Excel
- × нет профессиональной / коммерческой поддержки

Анализ данных и их представление

- R изначально содержит множество статистических функций.
  - линейные и обобщенные линейные модели
  - мдели нелинейной регрессии
  - анализ данных во времени
  - классические параметрические и непараметрические тесты
  - кластеризация
  - сглаживание
- R так же имееи набор функций, которые предоставляют гибкую графическую среду для создания различного вида представлений данных.

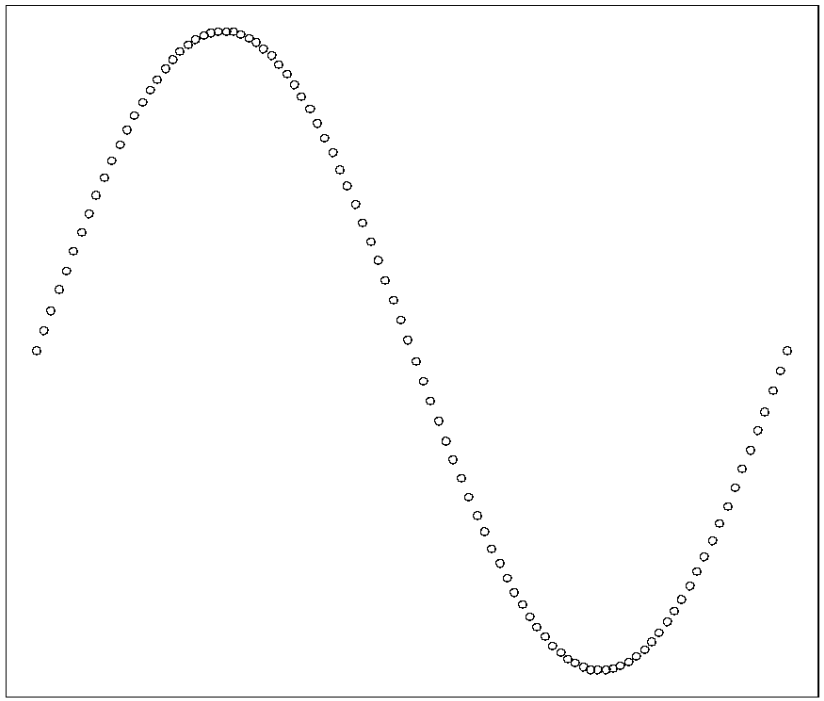
Пример R как калькулятора

```
> log2(32)
[1] 5

> sqrt(2)
[1] 1.414214

> seq(0, 5, length=6)
[1] 0 1 2 3 4 5

> plot(sin(seq(0, 2*pi, length=100)))
```



Типы переменных

примитивные (или: атомарные) типы данных в R это:

- числовые (integer, double, complex)
- символьные
- логический
- функциональные

из всего этого можно собирать вектора, массивы и списки.

Примеры работы с переменными

```
> a = 49
> sqrt(a)
[1] 7
```

числовые

```
> a = "The dog ate my homework"
> sub("dog","cat",a)
[1] "The cat ate my homework“
```

символы
строки

```
> a = (1+1==3)
> a
[1] FALSE
```

логические

Объекты

- Объект это нобор атомарных переменных или других объектов, собранных вместе
- Примеры:
  - данные о пациентах(жалобы, диагнозы, обследования)
  - генетические данные (последовательности, ID, описания)

Определения:

- class (класс): “абстрактное” определение класса
- object (объект): конкретный экземляр
- method (метод): другое обозначение для ‘функции’
- slot (слот): некоторый компонент объекта

вектора, матрицы, ...

- вектор: упорядоченная коллекция элементов одного типа

```
> a = c(1,2,3)
> a*2
[1] 2 4 6
```

- матрица: прямоугольная таблица элементов одного типа

```
> seq1 <- seq(1:6)
> mat1 <- matrix(seq1, 2, byrow = T)
> mat1
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

…, массивы, …

- массивы: 3-,4-,..размерные матрицы

```
three_d_array <- array(
  1:24, dim = c(4, 3, 2),
  dimnames = list(
    c("one", "two", "three", "four"),
    c("ein", "zwei", "drei"),
    c("un", "deux")  )  )
```

```
## , , un
##
##       ein zwei drei
## one     1    5    9
## two     2    6   10
## three   3    7   11
## four    4    8   12
##
## , , deux
##
##       ein zwei drei
## one    13   17   21
## two    14   18   22
## three  15   19   23
## four   16   20   24
```

…, списки, …

- список: упорядоченный набор данных любого типа

```
> doe = list(name="john",age=28,married=F)
> doe$name
[1] "john“
> doe$age
[1] 28
```

- Обчно, доступ к элеменам вектора получают по их индексу, а к элементам списка по имени, но оба хранилища поддерживают тот и другой способ индексации.

…, дата-фрэймы, ...

дата-фрэйм: представляет обычную таблицу с данными, полученными исследователями - аналог таблиц из Excel.

Это прямоугольная таблица со строками и колонками, данные в каждой колонке своего типа, но в разных колонках могут разные типы данных.

Пример:

```
> a
       localisation   tumorsize    progress
XX348     proximal         6.3       FALSE
XX234       distal         8.0        TRUE
XX987     proximal        10.0       FALSE
```

..., факторы, ...

Символьная строка может содержать произволный текст. Но иногда полезно пользоваться ограниченным словарём с малым набором допустимых слов. Фактор это именно такая переменная, а её ограниченный набор значний называют уровнями.

```
> a
[1] Kolon(Rektum)      Magen                Magen
[4] Magen              Magen                Retroperitoneal
[7] Magen              Magen(retrogastral)  Magen
Levels:  Kolon(Rektum)  Magen   Magen(retrogastral) Retroperitoneal
> class(a)
[1] "factor"
> as.character(a)
[1] "Kolon(Rektum)"   "Magen"              "Magen"
[4] "Magen"           "Magen"              "Retroperitoneal"
[7] "Magen"           "Magen(retrogastral)"  "Magen"
> as.integer(a)
[1] 1 2 2 2 2 4 2 3 2
> as.integer(as.character(a))
[1] NA NA NA NA NA NA NA NA NA NA NA NA
Warning message:  NAs introduced by coercion
```

…, подмножества

это отдельные элементы векторов, матриц, массивов или дата-фреймов полученные через “[ ]” с указанием их индексов или имён

```
> a
       localisation  tumorsize  progress
XX348     proximal        6.3         0
XX234       distal        8.0         1
XX987     proximal       10.0         0
> a[3, 2]
[1] 10
> a["XX987", "tumorsize"]
[1] 10
> a["XX987",]
       localisation  tumorsize  progress
XX987     proximal         10         0
```

Повторяемое исполнение

- Циклы for, repeat and while
  - for (*name* in *expr1*) *expr2*

    где **name** это переменная цикла. expr1 - векторное выражение, и **expr2** это выражение или набор выражений. **expr2** будет многократно исполнятся для кажбого **name** из **expr1**.
- Другие цикличные конструкции включают в себя:
  - repeat *expr statement*
  - while (*condition*) *expr statement*.
  - выражение **break** можно использовать для прерывания цикла, в том числе и аварийно.
  - выражение **next** используется для пропуска текущей итерации цикла и переходу к *следующей* итерации.

Циклы. Примеры.

```
for(i in 1:10) {
   print(i*i)
}

i=1
while(i<=10) {
   print(i*i)
   i=i+sqrt(i)
}
```

Ветвления

if (*logical expression*) {
  statements
} **else** {
  alternative statements
}

ветка else не обязательна

lapply, sapply, apply

- Если одинаковые или похожие задачи необходимо выполнить множество раз для всех элментов списка или всех колоном массива можно воспользоваться указанными функциями.
  - Это легче и быстрее чем циклы “for”
- lapply(li, *function*)
  - Для каждого элемента из списка **li** будет выполнена функция ***function***.
  - Результат будет тоже в виде списка.

```
> li = list("klaus","martin","georg")
> lapply(li, toupper)
> [[1]]
> [1] "KLAUS"
> [[2]]
> [1] "MARTIN"
> [[3]]
> [1] "GEORG"
```

apply

apply(*arr*, *margin*, *fct*)

Применить функцию **fct** ко всем измерениям массива **arr**, выравненных по **margin**, и вернуть вектор или массив соответсвующего размера.

```
> x
     [,1] [,2] [,3]
[1,]    5    7    0
[2,]    7    9    8
[3,]    4    6    7
[4,]    6    3    5
> apply(x, 1, sum)
[1] 12 24 17 14
> apply(x, 2, sum)
[1] 22 25 20
```

lapply, sapply, apply

sapply(*li*, *fct*)

Похоже на **apply**, но пробует упростить результат пытаясь преобразовать его в вектор или массив соответсвующего размера

```
> li = list("klaus","martin","georg")
> sapply(li, toupper)
[1] "KLAUS"  "MARTIN" "GEORG"

> fct = function(x) { return(c(x, x*x, x*x*x)) }
> sapply(1:5, fct)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
[2,]    1    4    9   16   25
[3,]    1    8   27   64  125
```

функции и операторы

Функции оперируют данными и возвращают результат
“Вход”: аргументы функции (от нуля и более)
“Выход”: результат (только один)

Пример:

```
add = function(a,b)
{ result = a+b
  return(result) }
```

Операторы:

Сокращенное написание для быстрого использования функций с двумя аргументами.
Примеры: + - * / ! & | %%

Чтение данных из файлов

- функция read.table()
  - Для чтения содержимого дата-фрейма внешний файл должен быть в специальной форме.
  - Первая линия файла должна содержать наименования колонок.
  - Каждая строка должна содержать первым элементом номер или наименование строки.

	Price	Floor	Area	Rooms	Age	Cent.heat
01	52.00	111.0	830	5	6.2	no
02	54.75	128.0	710	5	7.5	no
03	57.50	101.0	1000	5	4.2	no
04	57.50	131.0	690	6	8.8	yes

...

- функция scan()
  - inp<- scan("input.dat", list("",0,0))
  - Для разделения элементов данных в три вектора используется присваивание вроде:

```
label <- inp[[1]]; x <- inp[[2]]; y <- inp[[3]]
```

  - inp <- scan("input.dat", list(id="", x=0, y=0)); inp$id; inp$x; inp$y

Пример чтения данных из файла

- HousePrice <- read.table("houses.data", header=TRUE)

Price	Floor	Area	Rooms	Age	Cent.heat
52.00	111.0	830	5	6.2	no
54.75	128.0	710	5	7.5	no
57.50	101.0	1000	5	4.2	no
57.50	131.0	690	6	8.8	no
59.75	93.0	900	5	1.9	yes
...					

Сохранение данных

- Каждый объект в R может быть сохранен и восстановлен из фала командами “save” и “load”.
- Для этого используется XDR (external data representation) стандарт от Sun Microsystems и других, и портируемое между MS-Windows, Unix, Mac, ...

```
> save(x, file=“x.Rdata”)
> load(“x.Rdata”)
```