

Introduction to R (Programming)

Thomas Stibor

GSI
Helmholtzzentrum für Schwerionenforschung GmbH
t.stibor@gsi.de

21th September 2020 - 25th September 2020

Literature



Figure: Programmieren mit R, Uwe Ligges, Springer-Verlag.

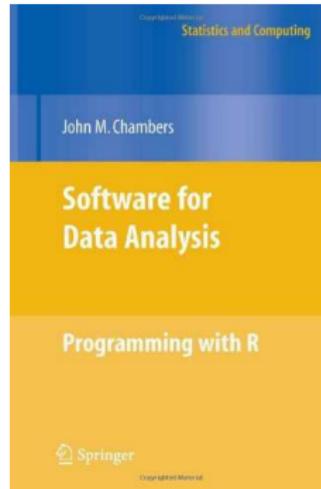


Figure: Software for Data Analysis: Programming with R, John M. Chambers, Springer-Verlag.

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

History

Ross Ihaka and Robert Gentleman. R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5:299-314, 1996 (<http://www.jstor.org/stable/1390807>).

- R is a language for developing statistical software (open source (GPL)).
- R implements a version of the S language (designed at Bell Laboratories).
- R is a command line interpreter language similar to Perl or Python.
- Large number of packages (CRAN) for a wide range of applications exist (Machine Learning, Genetics, ...)
- Available on large number of platforms: Linux, Mac OS X, Windows, FreeBSD, ...

R command line interpreter

```
stibor@herkules:~>R
```

```
R version 2.13.0 (2011-04-13)
```

```
Copyright (C) 2011 The R Foundation for Statistical Computing
```

```
ISBN 3-900051-07-0
```

```
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
```

```
You are welcome to redistribute it under certain conditions.
```

```
Type 'license()' or 'licence()' for distribution details.
```

```
Natural language support but running in an English locale
```

```
.
```

```
.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or
```

```
'help.start()' for an HTML browser interface to help.
```

```
Type 'q()' to quit R.
```

```
> all.equal((1+1/1000)^1000,exp(1))
```

```
[1] "Mean relative difference: 0.0004997918"
```

Run R scripts in batch mode: R CMD BATCH file.R or R --no-save < file.R

Basic Operations, Functions and Values

operator, function, value	description
<code>^</code> or <code>**</code>	power
<code>*</code> , <code>/</code>	multiplication, division
<code>max()</code> , <code>min()</code>	maxima, minima
<code>abs()</code>	absolute value
<code>sqrt()</code>	square root
<code>round()</code> , <code>floor()</code> , <code>ceiling()</code>	rounds the values
<code>sum()</code> , <code>product()</code>	sum, product
<code>log()</code> , <code>log10()</code> , <code>log2()</code>	natural logarithm, base 10 log., base 2
<code>exp()</code>	exponential function
<code>sin()</code> , <code>cos()</code> , <code>tan()</code>	trigonometric functions
<code>pi</code>	number π
<code>Inf</code> , <code>-Inf</code>	infinity
<code>NaN</code>	not a number
<code>NA</code>	not available
<code>NULL</code>	null object

Lists, Vectors, Matrices

An R list is an object consisting of an ordered collection of objects known as its components.

```
X <- list(spot.name="Hookipa",
           n.windydays = 320, pi*c(1:10))
names(X)
unlist(X)
sum(unlist(X[3]))
X$spot.name <- "River_Gorge"
is.list(X)
# use a list to store different type of
# objects (matrices, vectors, scalars, etc.)
list.container <- list(mat = matrix(c(1,2,3,4), nrow=2),
                         vec = c(5,6,7,8), exp(1))
```

Lists, Vectors, Matrices

```
vec <- c(1:100)
# sum numbers 1,2,...,100
s <- 0
for(i in 1:length(vec))
  s <- s + vec[i]
# smarter and faster
sum(vec)

s <- 0
for(i in 1:length(vec))
  s <- s + sqrt(log(vec[i]))

sum((log(vec))^(1/2))

# dot product
x <- c(1:10); y <- c(11:20);
as.real(x %*% y)
```

Lists, Vectors, Matrices

```
S <- 1e03
n <- m <- matrix(runif(S*S,min=-10,max=0),nrow=S,ncol=S)
# avoid using for loops
system.time({
for(r in 1:nrow(m))
  for(c in 1:ncol(m))
    m[r,c] <- m[r,c]^2
})
# use apply, mapply, ....
system.time({n <- apply(n, 2, function(x) x^2)})

identical(m,n)
```

Lists, Vectors, Matrices

```
m <- matrix(rnorm(20), nrow=4, ncol=5)
# singular-value decomposition
svd.mat <- svd(m)
# reconstruct matrix m
n <- svd.mat$u %*% diag(svd.mat$d) %*% t(svd.mat$v)
all.equal(m, n)

# compute eigenvalues and eigenvectors
eigen(m %*% t(m))

# solve linear eq. system
A <- matrix(c(3,5,1,
              2,7,3,
              4,4,2), byrow=T, nrow=3)
y <- c(3,4,5)
s <- solve(A,y)
s <- as.matrix(s)
A %*% s == y
```

Functions

```
# recursive function
n.choose.k <- function(n,k) {
  if (k == 1)
    return(n)
  if (n == k)
    return(1)
  return( n.choose.k(n-1,k) + n.choose.k(n-1,k-1) )
}

# create 10 random tuples (x1,x2) \in {1,2,\dots,15}
# and verify whether n.choose.k is correctly computed
i <- 1;
while(i <= 20) {
  x <- sort(sample(size=2,x=c(1:15)), decreasing = TRUE)
  print(choose(x[1],x[2]) == n.choose.k(x[1],x[2]))
  i <- i + 1;
}
```

(Some) Distributions in R

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
uniform	unif	min, max
Wilcoxon	wilcox	m, n

Approximate π Example

```
N.vec <- c(5,50,100,500,1000,10000);

pdf("pi_estm.pdf", width = 10.0, height = 10.0);
par(mfrow=c(3,2));

for(N in N.vec) {

  x.y.rand <- matrix(runif(2*N, min=-1,max=1), nrow=N, ncol=2);

  bool.in.out <- ((x.y.rand[,1]^2 + x.y.rand[,2]^2) < 1);
  in.circle <- sum(bool.in.out);

  est.pi <- 4*in.circle/N;
  sprintf("Estimated value of Pi is %f\n", est.pi);

  text.res <- paste("Estimated value of Pi = ", format(est.pi, digits = 5),
                     ", error = ", format(abs(pi-est.pi), digits = 8), sep = "");
  text.N <- paste("Number of points", N, sep = "");

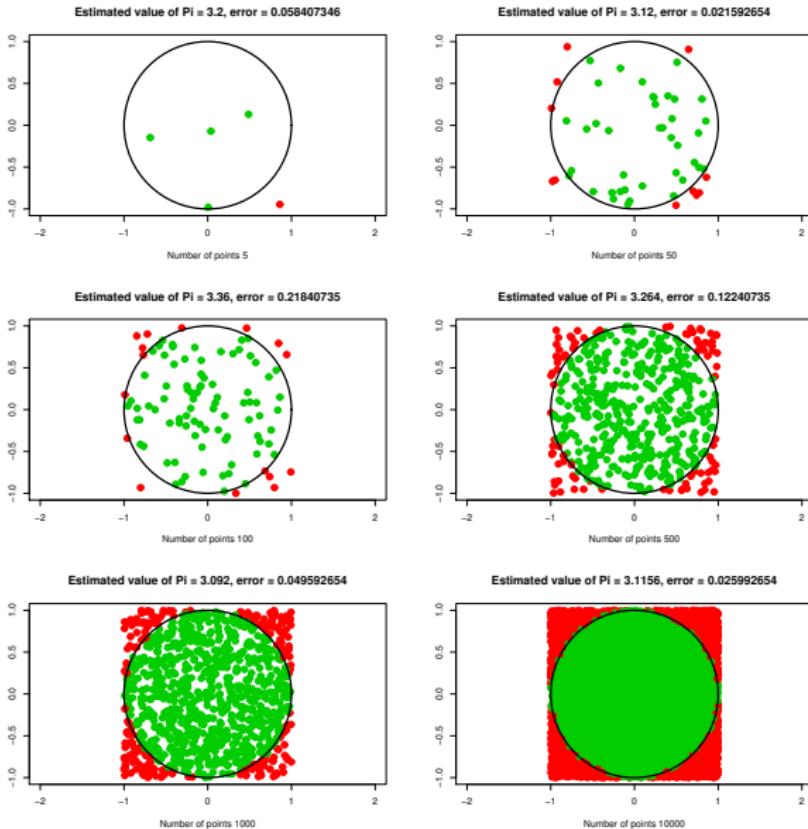
  plot(x.y.rand, xlim=c(-1,1), ylim=c(-1,1),
        xlab=text.N, ylab="", main=text.res,
        pch=19, cex=1.5, col=bool.in.out+2, asp=1);

  symbols(x=0, y=0, circles=1, inches=FALSE,
           xlim=c(-1,1), ylim=c(-1,1), add=TRUE, lwd=2);

}

dev.off();
```

Approximate π Example (cont.)



Markov Chain Example

Obtain a sample from a Markov chain specified by initial state vector and transition matrix.

```
States <- LETTERS[1:3] # three states A,B,C
init.pi <- c(0.6, 0.1, 0.3)
names(init.pi) = States

A <- matrix(c(0.2, 0.5, 0.3,
              0.6, 0.2, 0.2,
              0.1, 0.1, 0.8), byrow=T, nrow=3)
dimnames(A) = list(from = States, to = States)

l <- 10; # sample has l states
states <- c()
states <- c(states, sample(States, 1, prob = init.pi))
for (i in 2:l) {
  state = sample(States, 1, prob = A[states[i - 1], ])
  states = c(states, state)
}
```

CRAN Packages

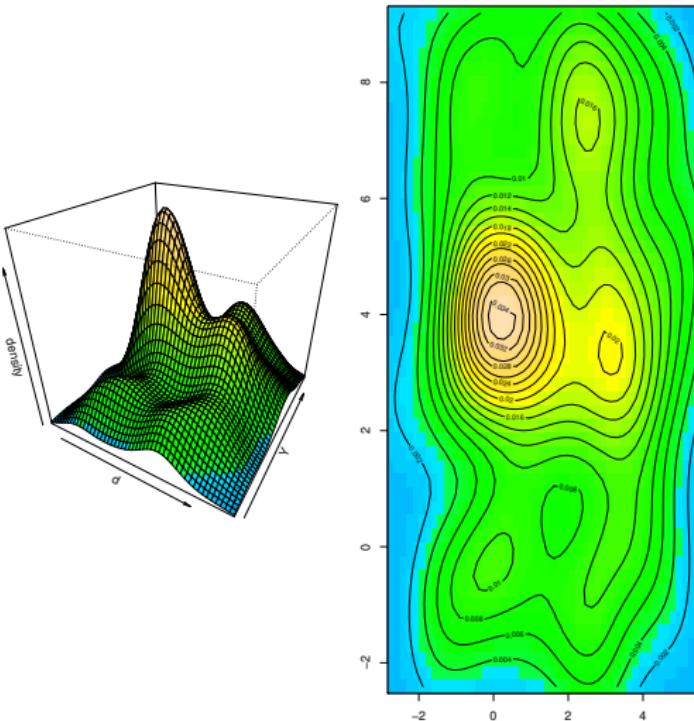
Suppose you want to sample from a Bernoulli distribution.

```
> rbern
```

will not work. However, there exists (for sure) a CRAN package (<http://cran.r-project.org/web/packages/>).

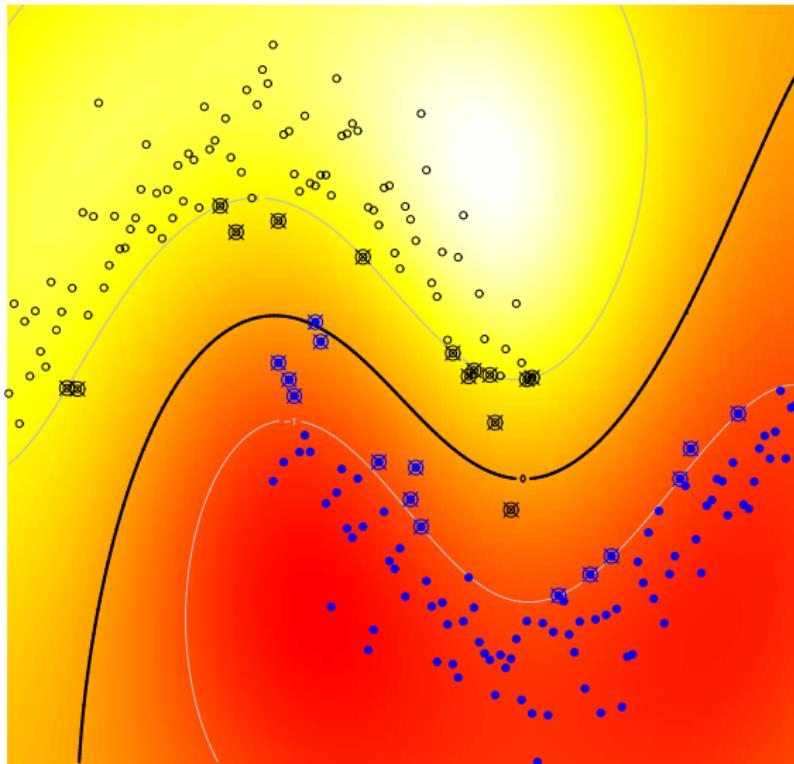
```
> install.packages("Rlab")
* DONE (Rlab)
> library(Rlab)
> rbern(n=10, prob=1/3)
[1] 0 1 1 0 0 1 0 0 0 1
```

Graphics (Kernel Density Estimator)



Taken from <http://addictedito.r.free.fr/graphiques/>

Visualize SVM Decision Boundary



Environments for R

I personally prefer Emacs and the ESS packages

The screenshot shows an Emacs interface with two main panes. The left pane displays an R script named `markov.chain.R`. The right pane shows the output of running this script, which includes system monitoring data from `top` and a table of process IDs (PID) and their details.

```
File Edit Options Buffers Tools Imenu S ESS Help
1 States <- LETTERS[1:3] # three states A,B,C
2 init.pi <- c(0.6, 0.1, 0.3)
3 names(init.pi) = States
4
5 A <- matrix(c(0.2, 0.5, 0.3,
6             0.6, 0.2, 0.2,
7             0.1, 0.1, 0.8), byrow=T, nrow=3)
8 dimnames(A) = list(from = States, to = States)
9
10 l <- 10; # sample has l states
11 states <- c()
12 states <- c(states, sample(States, 1, prob = init.pi))
13 for (i in 2:l) {
14   state = sample(States, 1, prob = A[states[i - 1], ])
15   states = c(states, state)
16 }
```

```
U:*** markov.chain.R All (13,0) (ESS[S] [R] Rox)----5:17PM-----
top - 17:17:19 up 13 min, 2 users, load average: 0.05 0.10 0.11
Tasks: 189 total, 1 running, 188 sleeping, 0 stopped, 0 zombie
CPU(s): 0.7us, 0.3sy, 0.6ni, 99.0id, 0.0wa, 0.0hi, 0.0si, 0.0st
Mem: 3925780k total, 853136k used, 3072624k free, 75152k buffers
Swap: 976584k total, 0k used, 976584k free, 272780k cached
U:*** *terminal* Bot (155,8) (Term: char run)----5:17PM-----
```

The right pane also contains a help buffer for the `sample` function, showing its documentation and arguments.

```
> states <- c(states, sample(States, 1, prob = init.pi))
+ for (i in 2:l) {
+   state = sample(States, 1, prob = A[states[i - 1], ])
+   states = c(states, state)
+ }
+ states
[1]: At "C" "C" "C" "C" "C" "C" "C" "C" "C"
> help(sample)
> U:-- *R* Bot (111,2) (iESS [R] run)----5:17PM-----
Sample package:base R Documentation
Random Samples and Permutations
Description:
'sample' takes a sample of the specified size from the elements of
'x' using either with or without replacement.

Usage:
sample(x, size, replace = FALSE, prob = NULL)
sample.int(n, size = n, replace = FALSE, prob = NULL)

Arguments:
x: Either a vector of one or more elements from which to choose,
or a positive integer. See 'Details'.
n: a positive number, the number of items to choose from. See
'Details.'
```

Good and easy to learn IDE is <http://www.rstudio.org/>.

Summary

R is a very powerful language for data analysis and graphics.

There are tons of features in R, I haven't considered in this brief overview.
Go and explore the power and richness of R.



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