

Getting Started in *R*

Giles Hooker*

May 28, 2007

1 Overview

R is a free alternative to Splus: a nice environment for data analysis and graphical exploration. It uses the object-oriented paradigm to implement much of its functionality, and this can be exploited in programming. Here are some aspects of its usefulness:

- Data management and storage
- Manipulation of vectors and matrices
- Tools for basic statistical analysis
- Customizable functions
- Graphical display
- Online help and other references

This document provides a cursory introduction to each of these areas, enough to get the ball rolling. If you want additional introductory help, there is a web-based help that can be initialized by typing `help.start()` at the *R*-prompt.

2 Getting Started

R is freely available from the *R*project website:

www.r-project.org

where you can find source code and compiled installers for Windows, MacOS and Linux. This is also an excellent source of further documentation; see, for example, the "Search" link in the left-hand frame.

From Windows or MacOS you can work in *R* from a (rudimentary) GUI interface. In Linux there is a nice interface to emacs called ESS, which is also linked through the *R*project website (see the "Related Projects") link. Both of these have text editors and (especially useful) hot-keys for sending text from the editor to the *R*prompt.

R will save its workspace to a `.Rdata` file in its working directory. This is the directory where it was started from. When you complete an *R*session, you will be asked if you wish to

```
save workspace image?
```

Saving the workspace means writing it into a `.Rdata` file. The home directory is:

Windows: The directory of the *R*executable that started it. It may be started by double-clicking at `.Rdata` file, in which case it uses the directory of that file. You can change the home directory in the `file` menu.

Linux: The directory from which it was called. If you use Emacs/ESS, you will be prompted to verify the directory.

The home directory is important as it may be useful to know where to retrieve previous work, and in order to have the right path specification for loading in data.

*Based on an Splus tutorial by Phil Beineke, Balasubramanian Narasimhan and Victoria Stodden

3 Reading Data

A common task is reading in data from an external file. The function `read.table` is geared towards reading data into a data frame. The data file must conform in some ways. The first line should have a name for each column and every succeeding row must have a row label. Character strings containing blanks must be in quotes.

The following prototype may be helpful. Make sure you have the file `income.data` (available from the class website) in your home directory.

```
read.table('income.data')
```

`read.table` expects a tab-delineated data file. To cope with comma-delineation use `sep=' 'c'` as a second argument to `read.table`.

4 Vectors and Matrices

In using *R*, you will often find it helpful to work with matrices. Many convenient commands are built in, but it's worthwhile to be a little cautious – matrix multiplication is accomplished by `%%` not `*`, which multiplies terms element-wise. Other useful commands include `t()`, which returns the transpose of a matrix, `solve()`, which inverts one, and `diag()`, which returns a diagonal matrix.

5 Data Frames and Factors

One of the nice features about *R* is its ability to treat a data array as more than just a matrix. When data is read into the object `array` in *R* it is treated as a matrix. At this point we can add names to each attribute with a command like

```
names(array) = c('x', 'y')
```

The syntax `array$x` now refers to the `x` attribute of `array`.

An array that contains only numeric variables will be treated as containing real-valued entries. The *R* calls categorical variables “factors” and the assignment

```
array$x <- as.factor(array$x)
```

tells *R* to treat the entries in `x` as category names. Similar statements like `as.numeric` are also available.

6 Functions

Much of the power of *R* comes from the ways that people have extended it. By writing your own functions, you will further extend it for your personal use. Here's a very simple template function.

```
rolladie = function (num.sides =6, num.rolls = 1)
{
  simulated.rolls_ trunc (runif(num.rolls)*num.sides+1)
  return(simulated.rolls)
}
```

```
> rolladie()
[1] 3
> rolladie(num.sides =12)
[1] 8
```

7 Graphics

To open a graphics window, type `trellis.device()` or `motif()` or `X11()` at the command prompt. There are many, many plotting commands. For on-screen plotting, you can use `par(mfrow=c(2,2))`, which allows you to put four plots in a single window. For printing or storage, you can use:

```
> postscript('filename.ps')
```

Any plotting commands are now executed to the “filename.ps” file in your starting directory. It is best to plot these to a graphics window first and then copy your commands to the file. Be sure to close the .ps file with the command

```
> dev.off()
```

when the file is complete, or you will over-wright it with the next plot command. The commands `pdf` and `png` provide the equivalent functionality for .pdf and .png files.

8 Resources

Online help for *R* is available via the `help.start()` keys.

A useful introductory book is Venables and Ripley (3rd Edition, 1999). This deals with *Splines*, but most of the commands are identical in *R*. In addition, Chambers and Hastie (1991) is an excellent source of modelling examples.

In addition, the *R* webpages can be found at <http://www.r-project.org>.

9 A session

```
> help.start()

> x = 1:20
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
> y = rnorm(20)
> y
 [1] -1.341952160  0.331609811 -0.373547018  0.700858159  0.008217722
 [6] -1.307026818  0.215234944 -1.046575953  0.276504941 -0.050073815
[11]  1.007802834  1.441801003  1.967276064  0.456316022  2.785608606
[16]  0.348337754  0.339676019  0.999970512 -0.277285088  0.392326316
> plot(x,y)
> plot(x,y,type='l')
> plot(x,y,type='b')
> plot(x,y,type='o',pch='o')
> plot(x,y,type='o',pch='o',xlab="X",ylab="Y")
> plot(x,y,type='o',pch='o',xlab="X",ylab="Y",col=2)
> help(plot)

> x = rnorm(100)
> hist(x)

> x = seq(1,20,0.5)
> x
 [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0
[16]  8.5  9.0  9.5 10.0 10.5 11.0 11.5 12.0 12.5 13.0 13.5 14.0 14.5 15.0 15.5
[31] 16.0 16.5 17.0 17.5 18.0 18.5 19.0 19.5 20.0
> y = 1 + 2*x + rnorm(40)
Warning message:
longer object length
  is not a multiple of shorter object length in: 1 + 2 * x + rnorm(40)
> length(x)
 [1] 39
> y = 1 + 2*x + rnorm(39)
> plot(x,y)
> model = lm(y~x)
> summary(model)

Call:
lm(formula = y ~ x)

Residuals:
    Min       1Q   Median       3Q      Max
-1.88444 -0.49091 -0.05242  0.51027  1.83389

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.87644    0.29037   3.018  0.00458 **
```

```

x          2.00983    0.02437  82.457 < 2e-16 ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

Residual standard error: 0.8566 on 37 degrees of freedom
Multiple R-Squared:  0.9946, Adjusted R-squared:  0.9944
F-statistic:  6799 on 1 and 37 DF,  p-value: < 2.2e-16

> abline(model$coef)

> postscript("model.plot.ps")
> plot(x,y)
> abline(model$coef)
> dev.off()

> plot(fitted(model), resid(model))
> abline(0,0)
> lines(fitted(model), resid(model), lty=2)
> points(fitted(model), 0, pch=4)

> data = data.frame(x,y)
> data

> x = matrix( rnorm(40), 10, 4)
> x
      [,1]      [,2]      [,3]      [,4]
[1,] 0.6303337 0.15836632 -0.7264536 -0.03507108
[2,] -0.5411063 -0.26517082  1.8596637  1.11225861
[3,] 0.9334452 -1.55871295  1.6501642  0.84833266
[4,] 0.3951839 -0.49657190  1.0519572 -0.80910533
[5,] 0.8262459  1.34836739 -0.7324674 -0.10400362
[6,] -1.0999160  0.66651849  1.2068275 -0.21232881
[7,] -0.3260438  0.97123917  0.3756394  0.28085312
[8,] 0.5956828 -0.02845172 -1.2299575  0.26222653
[9,] 1.3536155  2.11253780 -1.0009799 -0.84062494
[10,] 0.9147961 -0.79626971 -0.4870342 -0.48678721

> x = data.frame(x)
> names(x)
[1] "X1" "X2" "X3" "X4"
> x$y = 3 + x1 + 2*x2 + rnorm(10)
Error: Object "x1" not found
> x$y = 3 + x$X1 + 2*x$X2 + rnorm(10)
> names(x)
[1] "X1" "X2" "X3" "X4" "y"

> x
      X1      X2      X3      X4      y
1  0.6303337 0.15836632 -0.7264536 -0.03507108 4.352411
2 -0.5411063 -0.26517082  1.8596637  1.11225861 2.670246
3  0.9334452 -1.55871295  1.6501642  0.84833266 3.457349
4  0.3951839 -0.49657190  1.0519572 -0.80910533 2.296380
5  0.8262459  1.34836739 -0.7324674 -0.10400362 7.823060
6 -1.0999160  0.66651849  1.2068275 -0.21232881 3.304207
7 -0.3260438  0.97123917  0.3756394  0.28085312 2.942363
8  0.5956828 -0.02845172 -1.2299575  0.26222653 5.022717
9  1.3536155  2.11253780 -1.0009799 -0.84062494 8.408864
10 0.9147961 -0.79626971 -0.4870342 -0.48678721 2.643683

> summary(x)

```

X1		X2		X3		X4	
Min.	:-1.0999	Min.	:-1.55871	Min.	:-1.2300	Min.	:-0.840625
1st Qu.:	-0.1457	1st Qu.:	-0.43872	1st Qu.:	-0.7310	1st Qu.:	-0.418173
Median :	0.6130	Median :	0.06496	Median :	-0.0557	Median :	-0.069537
Mean :	0.3682	Mean :	0.21119	Mean :	0.1967	Mean :	0.001575
3rd Qu.:	0.8927	3rd Qu.:	0.89506	3rd Qu.:	1.1681	3rd Qu.:	0.276196
Max. :	1.3536	Max. :	2.11254	Max. :	1.8597	Max. :	1.112259

Y

Min.	:2.296
1st Qu.:	2.738
Median :	3.381
Mean :	4.292
3rd Qu.:	4.855
Max. :	8.409

```
> library(leaps)
> x = matrix(rnorm(40),10,4)
> x = data.frame(x)
> y = 3 + x$X1 + 2*x$X2 + rnorm(10)
> subset = leaps(x=x,y=y,nbest=1)
> subset
```

```
$which
      1      2      3      4
1 FALSE TRUE FALSE FALSE
2  TRUE TRUE FALSE FALSE
3  TRUE TRUE  TRUE FALSE
4  TRUE TRUE  TRUE  TRUE
```

```
$label
[1] "(Intercept)" "1"          "2"          "3"          "4"
```

```
$size
[1] 2 3 4 5
```

```
$Cp
[1] 15.890414  2.822094  4.113123  5.000000
```

```
> house = read.table("housing.data",header=T)
```

```
> names(house)
[1] "crim"      "zn"        "indus"     "chas"      "nox"       "rm"        "age"
[8] "dis"       "rad"       "tax"       "ptratio"   "b"         "lstat"     "medv"
```

```
> dim(house)
[1] 506 14
```

```
> house.lin = lm(log(medv)~.,data=house)
> summary(house.lin)
```

```
Call:
lm(formula = log(medv) ~ ., data = house)
```

```
Residuals:
      Min       1Q   Median       3Q      Max
-0.73361 -0.09747 -0.01657  0.09629  0.86435
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
```

```

(Intercept)  4.1020423  0.2042726  20.081  < 2e-16  ***
crim         -0.0102715  0.0013155  -7.808  3.52e-14  ***
zn           0.0011725  0.0005495   2.134  0.033349  *
indus        0.0024668  0.0024614   1.002  0.316755
chas         0.1008876  0.0344859   2.925  0.003598  **
nox         -0.7783993  0.1528902  -5.091  5.07e-07  ***
rm           0.0908331  0.0167280   5.430  8.87e-08  ***
age          0.0002106  0.0005287   0.398  0.690567
dis         -0.0490873  0.0079834  -6.149  1.62e-09  ***
rad          0.0142673  0.0026556   5.373  1.20e-07  ***
tax         -0.0006258  0.0001505  -4.157  3.80e-05  ***
ptratio     -0.0382715  0.0052365  -7.309  1.10e-12  ***
b            0.0004136  0.0001075   3.847  0.000135  ***
lstat       -0.0290355  0.0020299 -14.304  < 2e-16  ***
---
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

```

```

Residual standard error: 0.1899 on 492 degrees of freedom
Multiple R-Squared:  0.7896, Adjusted R-squared:  0.7841
F-statistic: 142.1 on 13 and 492 DF,  p-value: < 2.2e-16

```

```

> house$rad = as.factor(house$rad)
> levels(house$rad)
[1] "1" "2" "3" "4" "5" "6" "7" "8" "24"
> house.lin = lm(log(medv) ~ ., data=house)
> summary(house.lin)

```

```

Call:
lm(formula = log(medv) ~ ., data = house)

```

```

Residuals:
    Min       1Q   Median       3Q      Max
-0.73163 -0.10074 -0.01271  0.09153  0.86209

```

```

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  4.0078476  0.2188374  18.314  < 2e-16  ***
crim        -0.0102270  0.0013164  -7.769  4.75e-14  ***
zn           0.0015096  0.0005698   2.649  0.008333  **
indus        0.0027748  0.0025634   1.082  0.279575
chas1        0.1002734  0.0347642   2.884  0.004096  **
nox         -0.7728738  0.1569242  -4.925  1.16e-06  ***
rm           0.0868708  0.0169635   5.121  4.39e-07  ***
age          0.0001981  0.0005327   0.372  0.710086
dis         -0.0511276  0.0081318  -6.287  7.21e-10  ***
rad2         0.0833711  0.0595060   1.401  0.161838
rad3         0.1766617  0.0537784   3.285  0.001094  **
rad4         0.1002350  0.0478207   2.096  0.036595  *
rad5         0.1344828  0.0486359   2.765  0.005908  **
rad6         0.0961606  0.0589625   1.631  0.103565
rad7         0.2075191  0.0632793   3.279  0.001115  **
rad8         0.1939560  0.0600758   3.229  0.001329  **
rad24        0.3504602  0.0720382   4.865  1.55e-06  ***
tax         -0.0005052  0.0001569  -3.221  0.001365  **
ptratio     -0.0370069  0.0058188  -6.360  4.67e-10  ***
b            0.0004148  0.0001072   3.871  0.000123  ***
lstat       -0.0291645  0.0020395 -14.300  < 2e-16  ***
---

```

```

Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1

```

Residual standard error: 0.189 on 485 degrees of freedom
 Multiple R-Squared: 0.7946, Adjusted R-squared: 0.7861
 F-statistic: 93.81 on 20 and 485 DF, p-value: < 2.2e-16

```
> house[1:10,]
      crim   zn indus chas   nox   rm   age   dis rad tax ptratio   b
1  0.00632 18.0  2.31   0 0.538 6.575 65.2 4.0900  1 296   15.3 396.90
2  0.02731  0.0  7.07   0 0.469 6.421 78.9 4.9671  2 242   17.8 396.90
3  0.02729  0.0  7.07   0 0.469 7.185 61.1 4.9671  2 242   17.8 392.83
4  0.03237  0.0  2.18   0 0.458 6.998 45.8 6.0622  3 222   18.7 394.63
5  0.06905  0.0  2.18   0 0.458 7.147 54.2 6.0622  3 222   18.7 396.90
6  0.02985  0.0  2.18   0 0.458 6.430 58.7 6.0622  3 222   18.7 394.12
7  0.08829 12.5  7.87   0 0.524 6.012 66.6 5.5605  5 311   15.2 395.60
8  0.14455 12.5  7.87   0 0.524 6.172 96.1 5.9505  5 311   15.2 396.90
9  0.21124 12.5  7.87   0 0.524 5.631 100.0 6.0821  5 311   15.2 386.63
10 0.17004 12.5  7.87   0 0.524 6.004 85.9 6.5921  5 311   15.2 386.71
  lstat medv
1   4.98 24.0
2   9.14 21.6
3   4.03 34.7
4   2.94 33.4
5   5.33 36.2
6   5.21 28.7
7  12.43 22.9
8  19.15 27.1
9  29.93 16.5
10 17.10 18.9
```

```
> house[1:10,c(1,7,9)]
      crim   age rad
1  0.00632 65.2  1
2  0.02731 78.9  2
3  0.02729 61.1  2
4  0.03237 45.8  3
5  0.06905 54.2  3
6  0.02985 58.7  3
7  0.08829 66.6  5
8  0.14455 96.1  5
9  0.21124 100.0 5
10 0.17004 85.9  5
```

```
> house[1:10,-c(1,7,9)]
      zn indus chas   nox   rm   dis tax ptratio   b lstat medv
1 18.0  2.31   0 0.538 6.575 4.0900 296   15.3 396.90  4.98 24.0
2  0.0  7.07   0 0.469 6.421 4.9671 242   17.8 396.90  9.14 21.6
3  0.0  7.07   0 0.469 7.185 4.9671 242   17.8 392.83  4.03 34.7
4  0.0  2.18   0 0.458 6.998 6.0622 222   18.7 394.63  2.94 33.4
5  0.0  2.18   0 0.458 7.147 6.0622 222   18.7 396.90  5.33 36.2
6  0.0  2.18   0 0.458 6.430 6.0622 222   18.7 394.12  5.21 28.7
7 12.5  7.87   0 0.524 6.012 5.5605 311   15.2 395.60 12.43 22.9
8 12.5  7.87   0 0.524 6.172 5.9505 311   15.2 396.90 19.15 27.1
9 12.5  7.87   0 0.524 5.631 6.0821 311   15.2 386.63 29.93 16.5
10 12.5  7.87   0 0.524 6.004 6.5921 311   15.2 386.71 17.10 18.9
```

```
> (house$zn>10)[1:10]
[1] TRUE FALSE FALSE FALSE FALSE FALSE TRUE TRUE TRUE TRUE
> !(house$zn>10)[1:10]
[1] FALSE TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE
```

```
> sum(house$rad==2)
```

```

[1] 24
> house[house$rad==2, c(1, 7, 9)]
      crim  age rad
2    0.02731 78.9  2
3    0.02729 61.1  2
57   0.02055 35.7  2
89   0.05660 86.3  2
90   0.05302 63.1  2
91   0.04684 66.1  2
92   0.03932 73.9  2
96   0.12204 57.8  2
97   0.11504 69.6  2
98   0.12083 76.0  2
99   0.08187 36.9  2
100  0.06860 62.5  2
121  0.06899 69.7  2
122  0.07165 84.1  2
123  0.09299 92.9  2
124  0.15038 97.0  2
125  0.09849 95.8  2
126  0.16902 88.4  2
127  0.38735 95.6  2
197  0.04011 34.1  2
198  0.04666 36.6  2
199  0.03768 38.3  2
202  0.03445 38.4  2
203  0.02177 15.7  2

> x = 0
> for(i in 1:506) {
+ x = x + 1/506*house$medv[i]
+ }
> x
[1] 22.53281

> x = 0
> k = 0
> for(i in 1:506) {
+ if(house$medv[i]<25) {
+ k = k+1
+ x = x + house$medv[i]
+ }
+ }
> x
[1] 6829.4
> k
[1] 374
> x/k
[1] 18.26043
> sum( house$medv[house$medv<25] )/sum(house$medv<25)
[1] 18.26043

> apply(house[, c(1, 2)], 2, var)
      crim      zn
73.98658 543.93681

> make.mean = function(x) {
+ N = length(x)
+ m = 0
+ for(i in 1:N) {

```

```

+ m = m + x[i]
+ }
+ return(m/N)
+ }
> make.mean
function(x) {
N = length(x)
m = 0
for(i in 1:N) {
m = m + x[i]
}
return(m/N)
}
> make.mean(house$medv)
[1] 22.53281

> ls()
[1] "house"          "house.lin"      "housing.data"  "i"              "k"
[6] "make.mean"      "subset"         "x"             "y"

> source("/usr/class/stats315a/WWW/tutorial.r")
> ls()
[1] "a"              "b"              "blah1"         "blah2"         "c"
[6] "hmm"           "house"          "house.lin"     "housing.data"  "i"
[11] "k"             "make.mean"      "subset"        "x"             "y"
> summary(hmm)
      a                b                c
Min.   :-2.24890   Min.    : 2.0   Min.    : 3.324
1st Qu.: -0.36254   1st Qu.: 26.5   1st Qu.: 27.299
Median : 0.07473   Median : 51.0   Median : 51.028
Mean    : 0.10078   Mean    : 51.0   Mean    : 52.650
3rd Qu.: 0.59666   3rd Qu.: 75.5   3rd Qu.: 78.099
Max.    : 1.62418   Max.    :100.0   Max.    :103.976

> blah1
function(data,index)
{
  N = nrow(data)
  m = 0
  for(i in 1:N) {
    m = m + N/data[i,index]
  }
  return(1/m)
}

> blah1(hmm,1)
[1] 0.0002630308

> summary(blah2(hmm))

Call:
lm(formula = c ~ ., data = data)

Residuals:
    Min       1Q   Median       3Q      Max
-0.44024 -0.19457 -0.05491  0.19565  0.55370

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 1.348277   0.083234   16.20  <2e-16 ***

```

```
a          2.077227  0.049125  42.28  <2e-16 ***
b          1.001818  0.001413  709.04  <2e-16 ***
```

```
---
```

```
Signif. codes:  0 "***" 0.001 "**" 0.01 "*" 0.05 "." 0.1 " " 1
```

```
Residual standard error: 0.2874 on 47 degrees of freedom
Multiple R-Squared: 0.9999, Adjusted R-squared: 0.9999
F-statistic: 2.515e+05 on 2 and 47 DF,  p-value: < 2.2e-16
```

```
> q()
```

```
Save workspace image? [y/n/c]: y
```

```
Process R finished at Sun Jan 25 12:01:19 2004
```