Getting Started with R

140.776 Statistical Computing

August 21, 2011
An Introduction to R (Venables and Smith), Chapter 2-6, from http://www.r-project.org
Open your R, type:

```r
> 1+1
```
> x <- read.table("Speed_Ex.txt",sep="\t",header=TRUE)
> x
<table>
<thead>
<tr>
<th>State</th>
<th>Increase</th>
<th>Fatalities Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alaska</td>
<td>No</td>
<td>-29.0</td>
</tr>
<tr>
<td>Connecticut</td>
<td>No</td>
<td>-4.4</td>
</tr>
<tr>
<td>Dist. of Columbia</td>
<td>No</td>
<td>-80.0</td>
</tr>
<tr>
<td>Hawaii</td>
<td>No</td>
<td>-25.0</td>
</tr>
<tr>
<td>Indiana</td>
<td>No</td>
<td>-13.2</td>
</tr>
<tr>
<td>Kentucky</td>
<td>No</td>
<td>3.4</td>
</tr>
<tr>
<td>Louisiana</td>
<td>No</td>
<td>-5.4</td>
</tr>
</tbody>
</table>
Set working directory

> getwd()
Set working directory

```r
> setwd("C:/Users/jihk/doc/courses/Computing2010/lecture2")

> getwd()

> list.files()

> x <- read.table("Speed_Ex.txt",sep="\t",header=TRUE)
```
<table>
<thead>
<tr>
<th></th>
<th>State</th>
<th>Increased?</th>
<th>Fatalities Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alaska</td>
<td>No</td>
<td>-29.0</td>
</tr>
<tr>
<td>2</td>
<td>Connecticut</td>
<td>No</td>
<td>-4.4</td>
</tr>
<tr>
<td>3</td>
<td>Dist. of Columbia</td>
<td>No</td>
<td>-80.0</td>
</tr>
<tr>
<td>4</td>
<td>Hawaii</td>
<td>No</td>
<td>-25.0</td>
</tr>
<tr>
<td>5</td>
<td>Indiana</td>
<td>No</td>
<td>-13.2</td>
</tr>
<tr>
<td>6</td>
<td>Kentucky</td>
<td>No</td>
<td>3.4</td>
</tr>
<tr>
<td>7</td>
<td>Louisiana</td>
<td>No</td>
<td>-5.4</td>
</tr>
</tbody>
</table>
The National Highway System Designation Act was signed into law in 1995.

It abolished the federal mandate of 55 mph speed limits.

The data show percentage changes in interstate highway traffic fatalities from 1995 to 1996.
> hist(x[,3])
Speed limits and fatalities

Histogram of x[, 3]

x[, 3]

Frequency

−50 0 50

0 5 10 15

140.776 Statistical Computing  Getting Started with R
During 1996, 32/51 states increased speed limits.

> boxplot(x[,3]~x[,2])
Speed limits and fatalities

-80 -40 0 20 40 60

No Yes

140.776 Statistical Computing
Getting Started with R
Speed limits and fatalities

```r
> t.test(x[,3]~x[,2])

Welch Two Sample t-test

data:  x[, 3] by x[, 2]
t = -2.7722, df = 28.248, p-value = 0.009747
alternative hypothesis: true difference in
means is not equal to 0
95 percent confidence interval:
  -38.799538  -5.833028
sample estimates:
  mean in group No mean in group Yes
    -8.563158     13.753125
```
> class(x)
[1] "data.frame"

> class(x[,1])
[1] "factor"

> class(x[,2])
[1] "factor"

> class(x[,3])
[1] "numeric"
R operates on *objects*:

- vectors
- matrices
- factors
- lists
- data frames
- functions
The simplest objects are *vectors* which can have one of the following modes:

- numeric (double, integer)
- complex
- logical (TRUE/FALSE)
- character
- raw (hold raw bytes)
> is.vector(x[,3])
Create a numeric vector

> x <- c(1.1, 2.2, 3.3, 4.4, 5.5)
> x
[1] 1.1 2.2 3.3 4.4 5.5

- The first line is an assignment using the function `c()`.
- In this example, `c()` takes five arguments and concatenates them into a vector.
- The second line is an expression.
You can do the same thing using the following commands:

```r
> y = c(1.1, 2.2, 3.3, 4.4, 5.5)
> y
[1] 1.1 2.2 3.3 4.4 5.5

> assign("z", c(1.1, 2.2, 3.3, 4.4, 5.5))
> z
[1] 1.1 2.2 3.3 4.4 5.5

> c(1.1, 2.2, 3.3, 4.4, 5.5)->u
> u
[1] 1.1 2.2 3.3 4.4 5.5
```
Create a numeric vector

You can also concatenate two vectors:

```r
> w<-c(x,0,y)
> w
[1] 1.1 2.2 3.3 4.4 5.5 0.0 1.1 2.2 3.3 4.4 5.5
```
Check objects in the current workspace

> ls()
Remove objects

> rm(x)
> ls()

> rm(list=ls())
Vectors can be used in arithmetic expressions:

- +, -, *, /
- log, exp, sin, cos, tan, sqrt
- max, min, mean, median, sum, prod, . . .

Operations are performed element by element:

```r
> x
[1] 1 2 3 4 5

> x^2
[1] 1 4 9 16 25
```

\[ x_1^2 + x_2^2 + \ldots + x_5^2 = ? \]
Vector arithmetic

> sum(x^2)
> [1] 55
max and min select the largest and smallest values in their arguments, even if they are given several vectors.

Examples:

```r
> x
[1] 1 2 3 4 5
> y
[1] 5 4 3 2 1
> max(x,y)
[1] 5
```
pmax and pmin return element-wise maximum and minimum.

Examples:

```r
> pmax(x, y)
[1] 5 4 3 4 5
```
Elements in a vector can be sorted:

```r
> x <- c(3, 8, 4, 2, 9, 10)
> x
[1] 3 8 4 2 9 10

> sort(x)
[1] 2 3 4 8 9 10

> y <- sort.int(x, index.return = TRUE)
> y
$x
[1] 2 3 4 8 9 10
$ix
[1] 4 1 3 2 5 6
Exercise

\[ x <- \text{read.table("sortdata.txt", header=FALSE)} \]

\[ \text{dim}(x) \]
\[ [1] 1000 2 \]

\[ y <- x[,1] \]

\[ y[1] \]
\[ [1] 0.87765 \]

\[ y(3) + y(176) + y(872) = ? \]
\begin{verbatim}
\> z <- sort(y)
\> z[3] + z[176] + z[872]
[1] -3.079099
\> (z[500] + z[501]) / 2
[1] 0.02903422
\> median(y)
[1] 0.02903422
\end{verbatim}
1:5 does the same thing as c(1,2,3,4,5)

Example:

> n<-5

> 1:n-1

> 1:(n-1)
Generating regular sequences

- The colon operator has high priority within an expression
- You can generate a decreasing sequence

```r
> 2*1:n
[1] 2 4 6 8 10

> n:1
[1] 5 4 3 2 1
```
The function `seq()` provides a more general approach:

```r
> seq(from=-1,to=2,by=0.5)
[1] -1.0 -0.5 0.0 0.5 1.0 1.5 2.0

> seq(from=-1,by=0.5,length=7)
[1] -1.0 -0.5 0.0 0.5 1.0 1.5 2.0

> x<-rnorm(4)
> x
[1] -2.3247806 -1.5598637 -0.3470389 -0.1820149
> seq(along.with=x)
[1] 1 2 3 4
```
Another useful function is `rep()`:

```r
generating_regular_sequences
> x <- c(1, 2, 3)
> x
[1] 1 2 3

> rep(x, times = 2)

> rep(x, each = 2)
```
Numbers in R are generally treated as numeric objects with double precision.

To explicitly specify an integer, you can use the L suffix.

Example:

```r
> x <- 1
> is.integer(x)
[1] FALSE

> x <- 1L
> is.integer(x)
[1] TRUE
```
To work with complex numbers, supply an explicit complex part:

```r
> sqrt(-9+0i)
[1] 0+3i
```
> x<-read.table("sortdata.txt",header=FALSE)

> sum(log(x[,2]-x[,1]))
- NaN represents an undefined value ("not a number") (e.g. 0/0 gives you NaN), it can also be thought of as a missing value
- There is a special number Inf which means infinity (e.g. 1/0 = Inf; Inf can be used in calculation, 1/Inf = 0)