Getting Started with R

140.776 Statistical Computing

August 21, 2011

Reading

- **An Introduction to R (Venables and Smith), Chapter 2-6**, from http://www.r-project.org

Getting Started with R

Open your R, type:

```
> 1+1
```

```
> x <- read.table("Speed_Ex.txt",sep="\t",header=TRUE)
> x
```

```
                    STATE INCREASe FATALITIESCHANGE
1                  Alaska       No           -29.0
2             Connecticut       No            -4.4
3    Dist. of Columbia          No           -80.0
4                  Hawaii       No           -25.0
5                 Indiana       No           -13.2
6                Kentucky       No             3.4
7               Louisiana       No            -5.4
```

Set working directory

```
> getwd()
```

Set working directory

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

> getwd()
[1] "C:/Users/jihk/doc/courses/Computing2010/lecture2"

> list.files()

> x <- read.table("Speed_Ex.txt",sep="\t",header=TRUE)
```

Speed limits and fatalities

	STATE	INCREASe	FATALITIESCHANGE
1	Alaska	No	-29.0
2	Connecticut	No	-4.4
3	Dist. of Columbia	No	-80.0
4	Hawaii	No	-25.0
5	Indiana	No	-13.2
6	Kentucky	No	3.4
7	Louisiana	No	-5.4

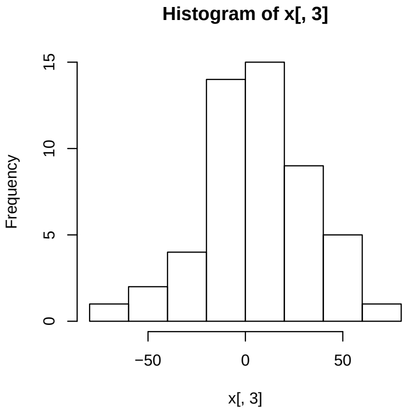
Speed limits and fatalities

- 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.

Speed limits and fatalities

```
> hist(x[,3])
```

Speed limits and fatalities

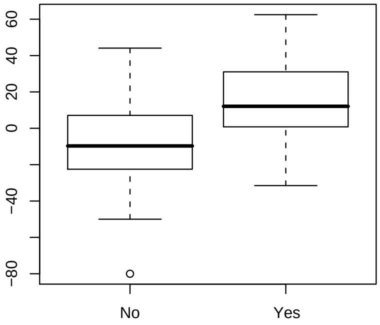


Speed limits and fatalities

- During 1996, 32/51 states increased speed limits.

```
> boxplot(x[,3]~x[,2])
```

Speed limits and fatalities



Speed limits and fatalities

```
> 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
```

Speed limits and fatalities

```
> class(x)
[1] "data.frame"

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

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

> class(x[,3])
[1] "numeric"
```

Objects

R operates on *objects*:

- vectors
- matrices
- factors
- lists
- data frames
- functions

Vectors

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)

Speed limits and fatalities

```
> 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*.

Create a numeric vector

You can do the same thing using the following commands:

```
> 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:

```
> 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())
```

Vector arithmetic

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:

```
> 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
```

Maximum and minimum

- max and min select the largest and smallest values in their arguments, even if they are given several vectors.

Examples:

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

> max(x,y)
[1] 5
```

Maximum and minimum

- pmax and pmin return element-wise maximum and minimum.

Examples:

```
> pmax(x,y)
[1] 5 4 3 4 5
```

Sorting

Elements in a vector can be sorted:

```
> 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<-read.table("sortdata.txt",header=FALSE)

> dim(x)
[1] 1000    2

> y<-x[,1]

> y[1]
[1] 0.87765
```

$y_{(3)} + y_{(176)} + y_{(872)} = ?$

Exercise

```
> 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
```

Generating regular sequences

- 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

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

> n:1
[1] 5 4 3 2 1
```

Generating regular sequences

The function seq() provides a more general approach:

```
> 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
```

Generating regular sequences

Another useful function is rep():

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

> rep(x,times=2)

> rep(x,each=2)
```

Numbers

- Numbers in R a generally treated as numeric objects with double precision
- To explicitly specify an integer, you can use the L suffix

Example:

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

> x<-1L
> is.integer(x)
[1] TRUE
```

Numbers

- To work with complex numbers, supply an explicit complex part:

```
> sqrt(-9+0i)
[1] 0+3i
```

Exercise

```
> x<-read.table("sortdata.txt",header=FALSE)

> sum(log(x[,2]-x[,1]))
```

Numbers

- 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)