An Introduction to Data Analysis using R

Dr Darren Kidney

Centre for Research into Ecological and Environmental Modelling (CREEM)
University of St Andrews

EMCSR, August 2014

Outline

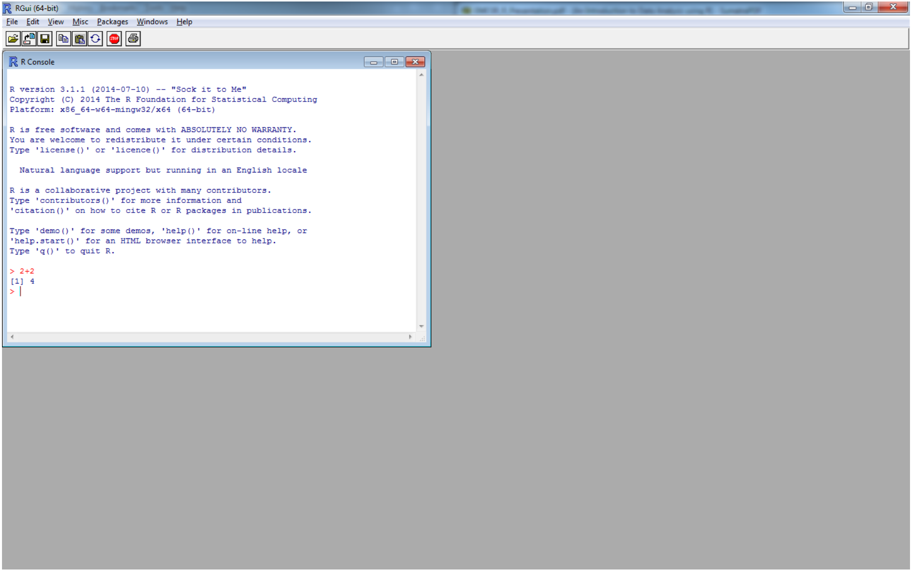
1.1 What is R?

- *"R is a language and environment for statistical computing and graphics"*
- Similar to the S language (which was developed at Bell Laboratories, US)
- R is Open Source and **free** (under the terms of the GNU Licence)
- Relatively simple programming language
- Large number of users and freely available extensions

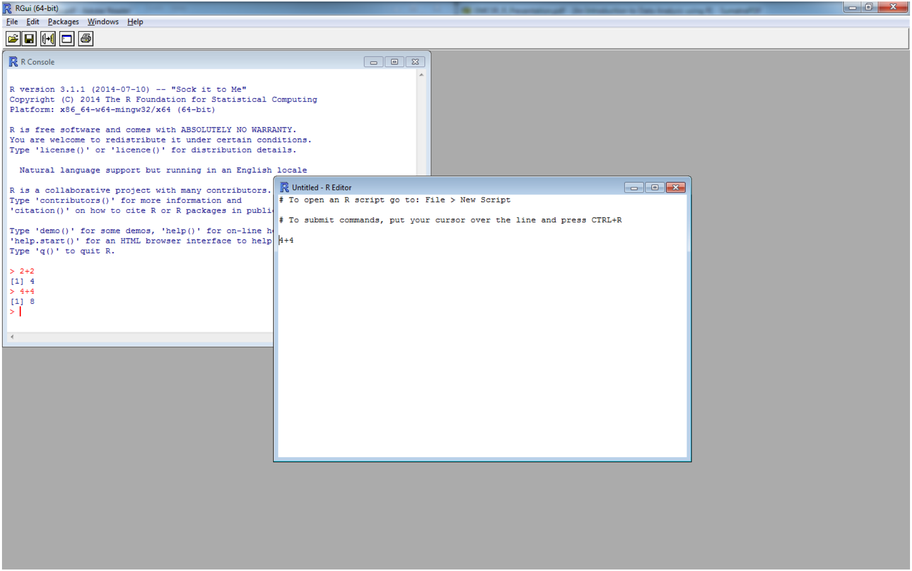
1.2 What does R do?

- Data handling and storage
- Mathematical operations and calculations (for a wide range of data types)
- Data analysis
- Publication quatlity plots

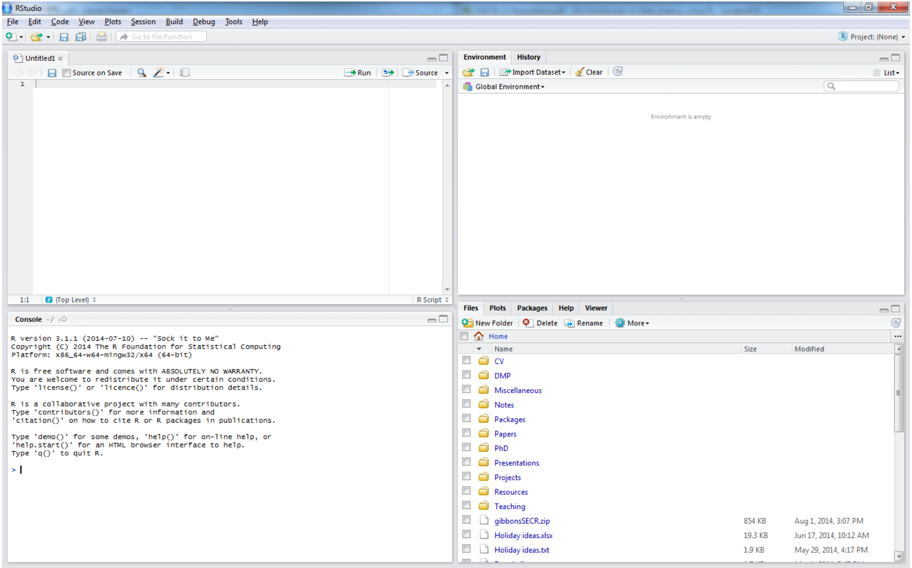
2. R basics


RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
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.
> 2+2
[1] 4
>

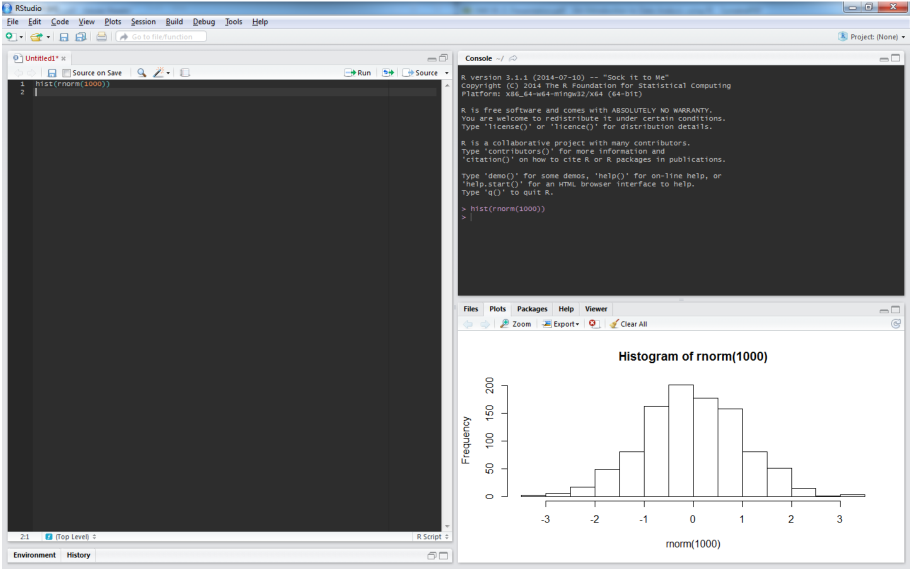
2. R basics


RGui (64-bit)
File Edit Packages Windows Help
R Console
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
Type 'demo()' for some demos, 'help()' for on-line h
'help.start()' for an HTML browser interface to help
Type 'q()' to quit R.
> 2+2
[1] 4
> 4+4
[1] 8
Untitled - R Editor
To open an R script go to: File > New Script
To submit commands, put your cursor over the line and press CTRL+R
4+4

2. R basics


RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
Untitled1
Source on Save
Run
Source
1:1
(Top Level)
R Script
Environment History
Import Dataset
Clear
List
Global Environment
Environment is empty
Console ~/
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
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.
Files Plots Packages Help Viewer
New Folder
Delete
Rename
More
Home
Name
Size
Modified
CV
DMP
Miscellaneous
Notes
Packages
Papers
PhD
Presentations
Projects
Resources
Teaching
gibbonsSECR.zip 854 KB Aug 1, 2014, 3:07 PM
Holiday ideas.xlsx 19.3 KB Jun 17, 2014, 10:12 AM
Holiday ideas.txt 1.9 KB May 29, 2014, 4:17 PM

2. R basics



RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
Project: (None)
Untitled1*
Source on Save
Run
Source
hist(rnorm(1000))
2:1
(Top Level)
R Script
Environment History
Console ~/
R version 3.1.1 (2014-07-10) -- "Sock it to Me"
Copyright (C) 2014 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
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.
> hist(rnorm(1000))
>
Files Plots Packages Help Viewer
Zoom Export Clear All
Histogram of rnorm(1000)
Frequency
0 50 100 150 200
-3 -2 -1 0 1 2 3
rnorm(1000)

2.1 Objects

- Data classes
  - integer
  - numeric (i.e. double)
  - character
  - logical
- Data structures
  - scalars
  - vectors
  - matrices
  - dataframes
  - lists

2. R basics

e.g. integer vector

```
> x = c(2L,4L,6L)
> x

[1] 2 4 6

> class(x)

[1] "integer"
```

e.g. numeric vector

```
> y = seq(from = 0, to = 1, by = 0.1)
> y

 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0

> class(y)

[1] "numeric"
```

e.g. logical matrix

```
> my.matrix = matrix(c(TRUE,FALSE), nrow=2, ncol=4)
> my.matrix # fills by column and recycles values

      [,1]  [,2]  [,3]  [,4]
[1,]  TRUE  TRUE  TRUE  TRUE
[2,] FALSE FALSE FALSE FALSE

> class(my.matrix)

[1] "matrix"

> typeof(my.matrix)

[1] "logical"
```

e.g. list

```
> my.list = list(x = letters[1:3],
+                y = matrix(1:2, 1, 2))
> my.list

$x
[1] "a" "b" "c"


$y
     [,1] [,2]
[1,]    1    2

> class(my.list)

[1] "list"

> typeof(my.list)

[1] "list"
```

e.g. data frame

```
> my.data.frame = data.frame(
+       x = letters[1:3],
+       y = 1,
+       z = 3:1)
> my.data.frame

  x y z
1 a 1 3
2 b 1 2
3 c 1 1

> class(my.data.frame)

[1] "data.frame"

> typeof(my.data.frame)

[1] "list"
```

2.2 Functions

Simple functions

```
> log(10)

[1] 2.302585
```

Compound functions

```
> exp(log(10))

[1] 10
```

Functions on objects

```
> x = matrix(1:9, nrow = 3)
> x

     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> log(x)

          [,1]     [,2]     [,3]
[1,] 0.0000000 1.386294 1.945910
[2,] 0.6931472 1.609438 2.079442
[3,] 1.0986123 1.791759 2.197225
```

Apply functions

```
> apply(x, 1, sum) # row sums
[1] 12 15 18
> apply(x, 2, mean) # column means
[1] 2 5 8
```

See also `tapply`, `sapply` and `lapply`.

Make your own functions

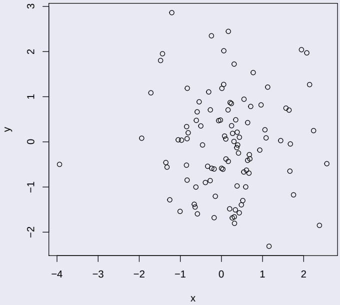
```
> myfunc = function(x,y=3){
+     result = 2 * x + y
+     return(result)
+ }
> myfunc(10) # using default value for y

[1] 23
```

2.3 Graphics

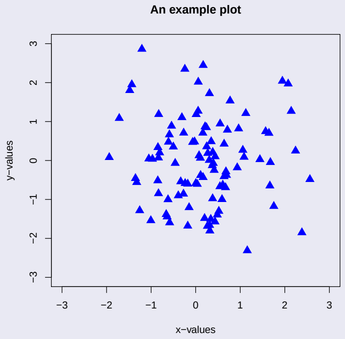
e.g. the plot function

```
> x = rnorm(100) ; y = rnorm(100)
> plot(x, y)
```



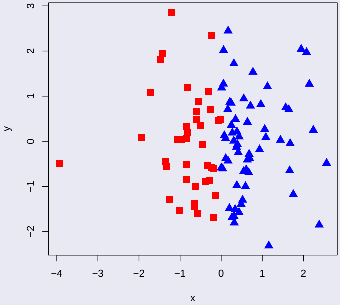
customising plots

```
> plot(x, y, pch = 17, col = "blue", cex = 1.5,
+       xlim = c(-3,3), ylim = c(-3,3),
+       ylab = "y-values", xlab = "x-values",
+       main = "An example plot",)
```



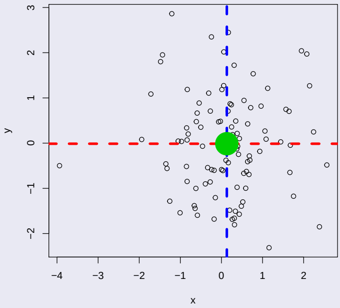
customising plots

```
> i = as.factor(x > 0)
> plot(x, y, cex = 1.5,
+       pch = c(15,17)[i],
+       col = c("red", "blue")[i])
```



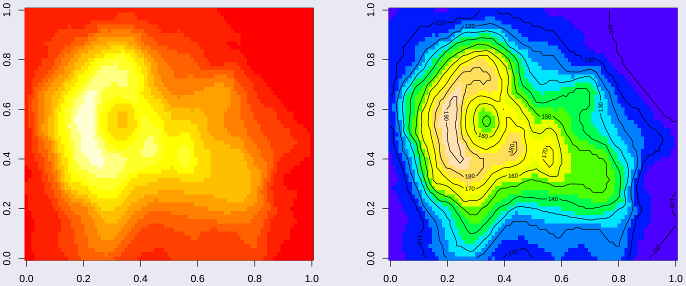
annotating plots

```
> plot(x, y)
> abline(v = mean(x), h = mean(y), lty = 2, lwd = 4,
+         col = c("red","blue"))
> points(mean(x), mean(y), pch = 19, col = 3, cex = 5)
```



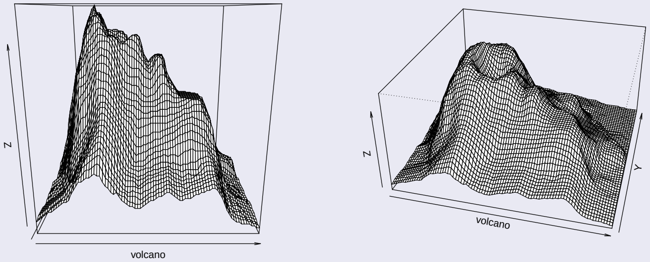
heat maps

```
> par(mfrow = c(1,2))
> image(volcano)
> image(volcano, col = topo.colors(10))
> contour(volcano, add = TRUE)
```



3D plots

```
> par(mfrow = c(1,2))
> persp(volcano)
> persp(volcano, phi = 30, theta = 15, expand = 0.5, d = 2)
```

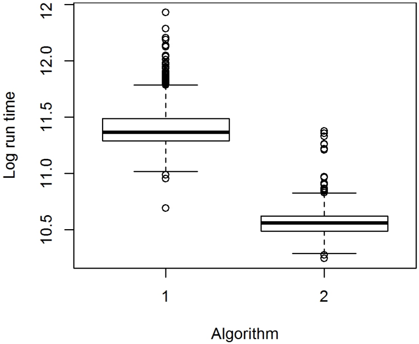


3.1 Example 1

- Suppose we want to compare the speed of two different algorithms
- We could run each algorithm once and compare the times...
- ...but this tells us nothing about how much the runtimes might vary from one run to the next
- It would be better to run each algorthm multiple times and compare the two sets of times
- Statistical analysis can help us make an objective decision

3. Data Analysis

Here are some real data. It looks pretty clear cut, but we'll analyse them anyway.



3. Data Analysis

We want to know:

- Is it plausible that the mean run time (i.e the long run average) for each algorithm is the same?
- In other words, is it plausible that the difference between the mean run times is zero?

We could answer this in two ways:

1. A two-sample t-test,
2. Or a one-way ANOVA

3. Data Analysis

First let's have a look at the data.

Algorithms data

```
> head(algorithms)

  algorithm runtime log.runtime
1         2   44659    10.70681
2         2   29701    10.29894
3         1  127845    11.75857
4         2   47603    10.77065
5         1   76642    11.24690
6         1   89245    11.39914

> attach(algorithms) # allows direct use of column names
```

3. Data Analysis

Now perform a t-test using the `t.test()` function.

Two-sample t-test

```
> a1 = log.runtime[algorithm == 1]
> a2 = log.runtime[algorithm == 2]
> t.test(a1, a2, var.equal = TRUE)

Two Sample t-test


data:  a1 and a2
t = 117.8057, df = 1998, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.8392051 0.8676192
sample estimates:
mean of x mean of y
 11.40930  10.55588
```

- The most important part of the output is the p-value
- We can access the p-value directly if we save the output (which happens to be a list) and extact the relevant component

Accessing the p-value

```
> results = t.test(a1, a2, var.equal = TRUE)
> results$p.value

[1] 0
```

- The p-value gives the probability of the result if the null hypothesis (i.e. no difference) were true
- In this case the p-value is extremely small (0.05 is the conventional cutoff)
- So the probability of observing a result like this if the null hypothesis were true is extremely small
- And we therefore reject the null hypothesis in favour of the alternative hypothesis that the means are different

When interpreting p-values always ask yourself:

What is the null hypothesis?

3. Data Analysis

Alternatively we could perform a one-way ANOVA using R's linear modelling capabilities.

One-way ANOVA

```
> fit1 = lm(log.runtime ~ algorithm, data = algorithms)
> summary(fit1)


...
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.262709   0.011560  1060.8   <2e-16 ***
algorithm   -0.853412   0.007244  -117.8   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The reliability of the results of the t-test and the ANOVA depends on the following assumptions about the data:

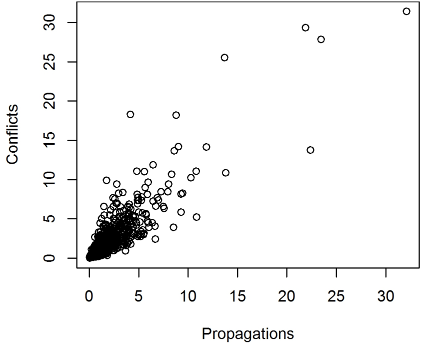
1. **Normally distributed**
2. **Constant variance**
3. **Independence**

You will get a chance to assess these assumptions in the practical.

3.2 Example 2

- Suppose we wanted to assess the influence of a continuous variable on algorithm speed, or some other type of continuous response

For example, consider the following set of artifical data (based on Figure 3. in Gent, 2013).



We might wish to know:

- Is there a linear relationship between the number of propagations has no relationship with mean number of conflicts?
- If so, what is the nature of that relationship?

We can try and answer these questions using **simple linear regression**.

3. Data Analysis

First let's have a look at the data.

Speedup data

```
> head(speedup)

       conf      prop
1 1.8662396 1.3703172
2 0.7953660 0.7339403
3 0.1689633 0.5044575
4 1.5296215 0.7363003
5 5.3021610 4.6651811
6 0.2300878 0.1102576

> attach(speedup) # allows direct use of column names
```

3. Data Analysis

Now lets perform a simple linear regression using R's linear modelling capabilities.

Simple linear regression

```
> fit2 = lm(conf ~ prop, data = speedup)
> summary(fit2)


...
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.11424    0.05141   2.222   0.0265 *
prop         1.05863    0.01815  58.335   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The estimate of the slope of the relationship is given by the `prop` coefficient

extracting parameter estimates

```
> coef(fit2)

(Intercept)        prop
  0.1142375   1.0586284

> coef(fit2)["prop"]

    prop
1.058628
```

- The p-value for `prop` is extremely small (much less than 0.05) which leads us to reject the null hypothesis.
- The null hypothesis in this case is that the true value of the slope is zero.

- In addition to estimates we can also provide a range of plausible values using 95% confidence intervals for the trye value of the parameters

confidence intervals for parameters

```
> confint(fit2)

                 2.5 %    97.5 %
(Intercept) 0.01335256 0.2151225
prop        1.02301708 1.0942397
```

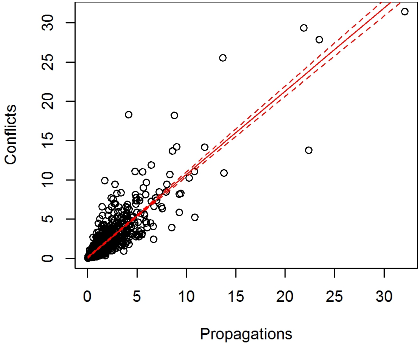
- Notice that zero does not fall inside either of these intervals.

- We can plot the estimated regression line along with a confidence region.

fitted regression line

```
> plot(prop, conf, xlab = "Propagations", ylab = "Conflicts")
> i = order(prop)
> preds1 = predict(fit2, speedup, interval = "confidence")
> lines(prop[i], preds1[i,"fit"], col = 2)
> lines(prop[i], preds1[i,"lwr"], col = 2, lty = 2)
> lines(prop[i], preds1[i,"upr"], col = 2, lty = 2)
```

3. Data Analysis



4. Advanced topics

- One of R's main strengths is it's flexibility.
- However one of it's main weaknesses is that it can be relatively slow (due to it being an interpreted, 4th generation language)
- This is particularly the case with `for` loops
- There are two main ways to speed up R: parallelistion and integrating C++ code

4.1 Rcpp

```
#include <RcppArmadillo.h>
using namespace Rcpp;
// [[Rcpp::depends(RcppArmadillo)]]
// [[Rcpp::export]]
arma::rowvec colProds(NumericMatrix x){
arma::mat X = arma::mat(x.begin(), x.nrow(), x.ncol(), false);
arma::rowvec col_prods = prod(X,0) ;
return col_prods ;
}
```

Sourcing an Rcpp file

```
> A = matrix(1:9, 3)
> apply(A, 2, prod)

[1]   6 120 504

> require(Rcpp)
> sourceCpp("src/colProds.cpp")
> colProds(A)

     [,1] [,2] [,3]
[1,]    6  120  504
```

Benchmarking - single run

```
> B = matrix(runif(1e7), 100) ; dim(B)

[1]    100 100000

> system.time(apply(B, 2, prod))["elapsed"]

elapsed
   0.36

> system.time(colProds(B))["elapsed"]

elapsed
      0
```

Benchmarking - multiple runs

```
> require(rbenchmark)
> benchmark(apply(B, 2, prod), colProds(B),
+           columns = c("test", "replications",
+                       "elapsed", "relative"),
+           order = "relative", replications = 10)

               test replications elapsed relative
2       colProds(B)           10    0.06    1.000
1 apply(B, 2, prod)           10    3.58   59.667
```

4.2 parallel

It's fairly easy to parallelise embarassingly parallel code.

Serial

```
> nloops = 80
> nseconds = 0.01
> system.time({
+     results = lapply(1:nloops, function(i){
+         Sys.sleep(nseconds)
+     })
+ })["elapsed"]

elapsed
   0.79
```

4.2 parallel

Parallel

```
> require(parallel)
> ncores = detectCores()
> myCluster = makeCluster(ncores)
> clusterExport(myCluster, "nseconds")
> system.time({
+     results = parLapply(myCluster, 1:nloops, function(i){
+         Sys.sleep(nseconds)
+     })
+ })["elapsed"]

elapsed
   0.11

> stopCluster(myCluster)
```

5. References

- Gent, I. P. 2013. Optimal Implementation of Watched Literals and More General Techniques. *Journal of Artificial Intelligence Research* **48** 231-252