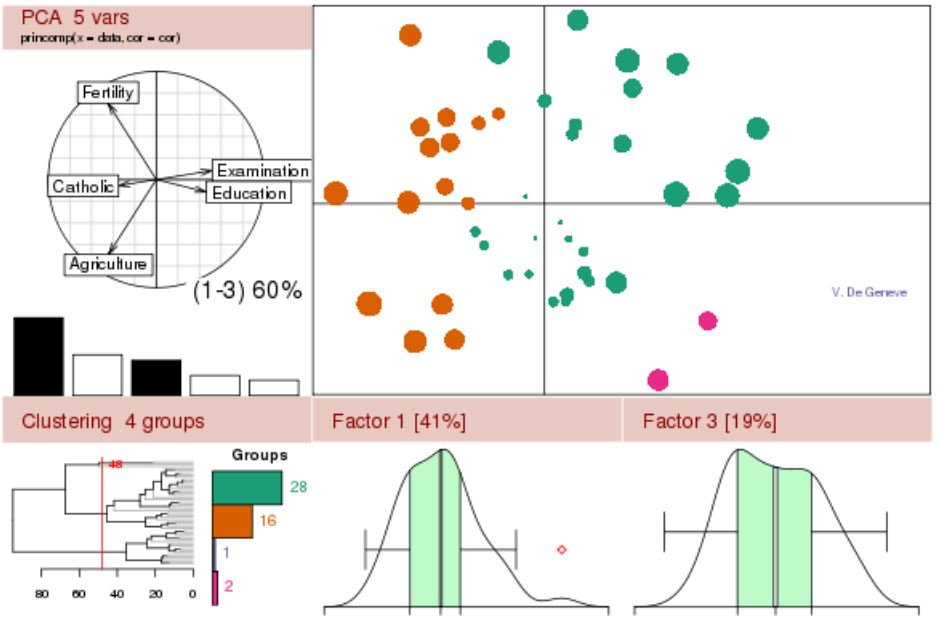
Imperial College London

diamond

A short introduction to the R statistical programming language



Diamond Light Source - April 2013 – PART 1

James Foadi

MPL, Imperial College London - Diamond Light Source Ltd, Oxfordshire

Starting R

Open a shell and type: “module load R”
“R”

```
R version 2.12.1 (2010-12-16)
Copyright (C) 2010 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-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

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.

[Previously saved workspace restored]

> 
```

Starting R

When R is running, most of the stuff is stored in the computer's active memory as a series of objects with a name.

The active memory is organised in different levels:

```
> ls()
character(0)
> ls(1)
character(0)
> ls(2)
  [1] "acf"                 "acf2AR"              "add1"
  [4] "addmargins"          "add.scope"           "aggregate"
  [7] "aggregate.data.frame" "aggregate.default"    "aggregate.ts"
……..
……..
```

Starting R

```
> ls(3)
 [1] "abline"          "arrows"          "assocplot"       "axis"
 [5] "Axis"            "axis.Date"       "axis.POSIXct"    "axTicks"
 [9] "barplot"         "barplot.default" "box"             "boxplot"
[13] "boxplot.default" "boxplot.matrix"  "bxp"             "cdplot"
…..............................
…..............................
```

```
> ls()
character(0)
> x <- 12.3
> y <- 8.7
> z <- x+y
> ls()
[1] "x" "y" "z"
> z
[1] 21
```

Starting R

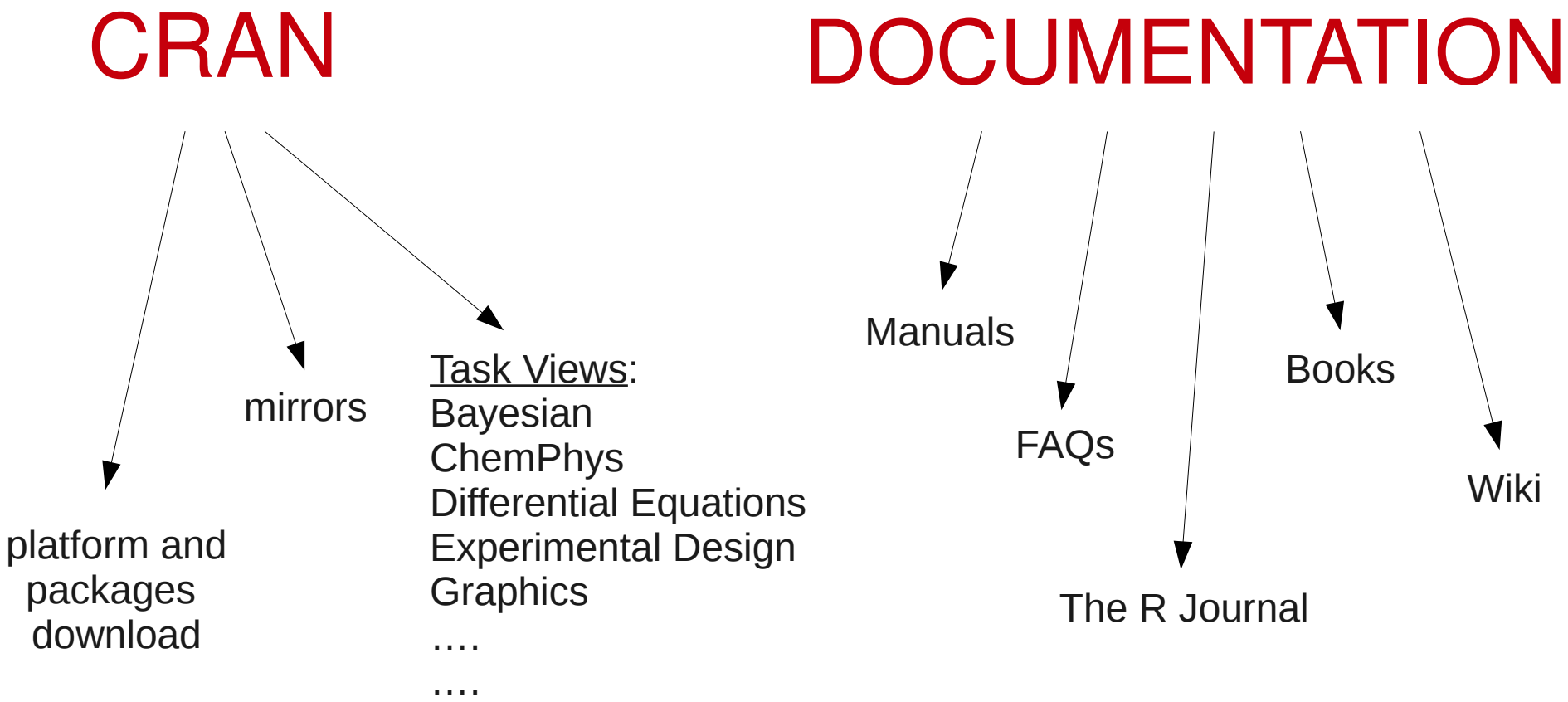
```
> q()
Save workspace image? [y/n/c]: y
```

Two “hidden” files have been created in the directory Where R was started: a).RData and b) .Rhistory
If next time you start R from within the same directory both files will be loaded and all previous content will be loaded in the active memory. This makes working with projects very easy.

```
> ls()
[1] "x" "y" "z"
```

Main reference for R

http://www.r-project.org



Day 1

- R as a very complete calculator
- R as a numerical simulator

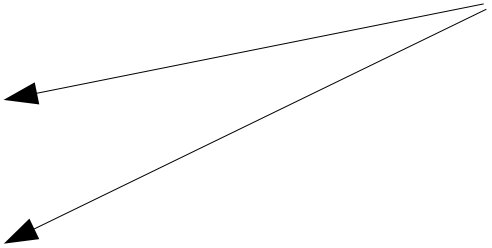
Day 2

- R for statistical modeling
- R for graphics and data visualization
- R as a programming language

R as a very complete calculator

The 4 (and more) operations:

```
> 12+22
[1] 34
> 12-60
[1] -48
> 30*110
[1] 3300
> 30/110
[1] 0.2727273
> 3^4
[1] 81
> 127%/%40
[1] 3
> 127%%40
[1] 7
```



Modular Arithmetics

R as a very complete calculator

Can we calculate quotient and remainder in one go?
Yes. We use a taylor-made "function".

```
> QandR <- function(dividend,divisor)
+ {
+  quotient <- dividend%/%divisor
+  remainder <- dividend %% divisor
+
+  return(c(quotient,remainder))
+ }
> QandR(127,40)
[1] 3 7
> tmp <- QandR(10,3)
> tmp[1]
[1] 3
> tmp[2]
[1] 1
```

R as a very complete calculator

Often we need to modify a function previously created. This can be done during an R session, but it is better to keep new code somewhere else.

With your preferred editor create an ascii file and call it “my_functions.R”. The first lines of this file look like this:

R as a very complete calculator

```
# Some useful functions used in this session

QandR <- function(dividend,divisor)
{
 quotient <- dividend%/%divisor
 remainder <- dividend %% divisor

 # Name slots with sensible names
 x <- c(quotient=quotient,remainder=remainder)

 return(x)
}
```

Comments

Named vector

R as a very complete calculator

The full content of any file with R code can be loaded With the function “source”:

```
> source("my_functions.R")
> tmp <- QandR(13,4)
> tmp
 quotient remainder
        3         1
> tmp[1]
quotient
       3
> tmp["quotient"]
quotient
       3
```

R as a very complete calculator

Other built-in functions (a lot of them!):

```
> sin(pi/2); cos(0)
[1] 1
[1] 1
> atan2(sqrt(2)/2,sqrt(2)/2)*180/pi
[1] 45
> exp(1)
[1] 2.718282
> y <- exp(2)
> y
[1] 7.389056
> log(y)
[1] 2
> ?log
```

R as a very complete calculator

Getting help to understand R commands (?):

```
> ?log
log
log                package:base                R Documentation

Logarithms and Exponentials

Description:

    'log' computes logarithms, by default natural logarithms, 'log10'
    computes common (i.e., base 10) logarithms, and 'log2' computes
    binary (i.e., base 2) logarithms.  The general form 'log(x, base)'
    computes logarithms with base 'base'.
……..
……..

Usage:

    log(x, base = exp(1))
    logb(x, base = exp(1))
    log10(x)
    log2(x)
```

R as a very complete calculator

```
> log(8,base=2)
[1] 3
> log10(1000)
[1] 3
> factorial(3)
[1] 6
> gamma(4)
[1] 6
> factorial(5)
[1] 120
> factorial(6)
[1] 720
> factorial(5.5)
[1] 287.8853
> gamma(6.5)
[1] 287.8853
```

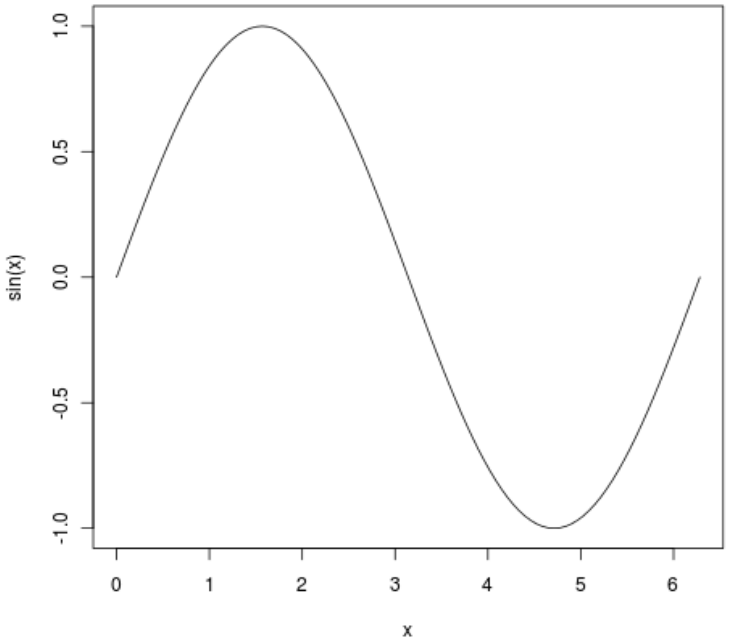
$$\Gamma(t+1)=t!$$

$$\Gamma(t)=\int_0^\infty x^{t-1}e^{-x}\,dx$$

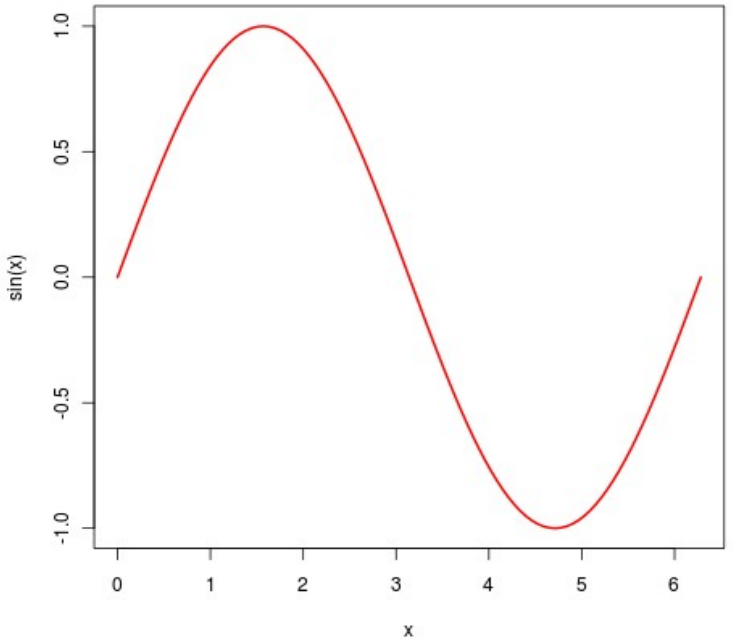
R as a very complete calculator

Any function can be plotted with the command "curve":

```
> curve(sin(x),from=0,to=2*pi)
```

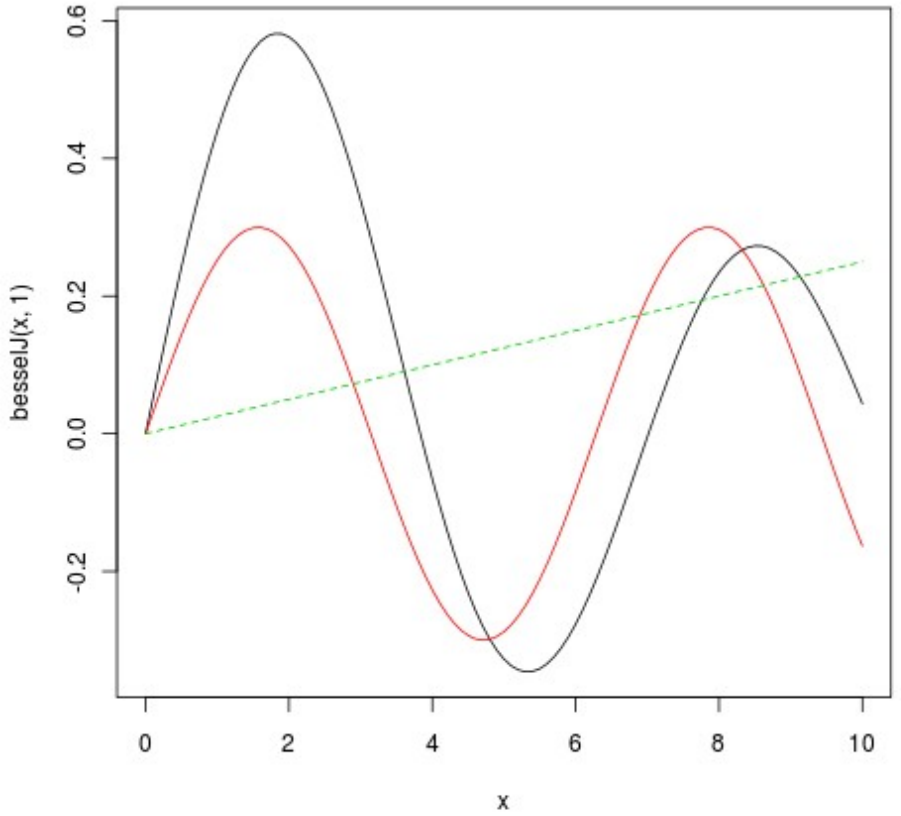



```
> curve(sin(x),from=0,to=2*pi,lwd=2,col=2)
```



R as a very complete calculator

```
> curve(besselJ(x,1),from=0,to=10)
> curve(0.3*sin(x),from=0,to=10,col=2,add=TRUE)
> curve(x/40,col=3,lty=2,add=TRUE)
```



Try it yourself!

Exercise 1

Write a function that takes in 2 numbers and returns their sum, difference and product as a named vector of length 3. Try to include the R code for this function in file “my_functions.R”. Later source this file and test your new function.

Exercise 2

Plot the three curves log(x), log(2x) and log(x/2) in the interval [1,10]. Use different colours and or thickness/types. Are some parts of the curves outside the plot?