R Basics

Richard Upton
3 August 2015

Course Outline

- Day 1: R basics
- Day 2: Introduction to *ggplot2* for Plotting
- Day 3: Writing and Applying Functions, Text Strings
- Day 4: Merging data, Dates
- Day 5: Statistics, Random Numbers and Stochastic Simulation

General Day Outline (approximate)

- 9:30 Review of homework
- 10:00 Lecture
- 10:30 Break
- 10:45 Hands-on
- 11:30 Self-guided homework set

About me

- 30 years as an experimental scientist
- 17 years as a data modeller
- 6 years as Pharmacometric Consultant (Projections, PKPDSystems)
- 5 years Professor of Pharmacometrics, University of South Australia
- R user since 2001

What is R ?

- A computer language for statistical computing and graphics
- Multi-platform (Windows, Mac, Linux)
- Open Source, GNU General Public License
- R is an interpreted language
  - Often run from a command-line (although GUI are available)
  - The commands are collected in a text file (script) and run sequentially
  - No "compiled" programs (*.exe)

Who makes R?

Base package

R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL

http://www.R-project.org/.

- Collaborative and Voluntary
- Provides Core functions
- A structured system for development
- A structured system for user contributed packages

Who makes R?

User contributed packages

- Over 5000 specialized packages
- Generally written by experts
- Frequently updated
- A package may have dependencies
- You can write your own package!

A brief history of R

- S: developed by Chambers at Bell Laboratories, 1975
  - Easier than doing statistics in Fortran
- S-plus: S with a windows GUI, 1988
  - A commercial implementation of S
  - S-plus and R are mostly compatible
  - R: Open source implementation of S
  - Ihaka and Gentleman (Auckland, 1995)
- R Core Core Team, Version 1.0
  - Late 90's

Other software vs R?

Excel

A different paradigm, poor error tracking, elementary statistics

SAS, SPSS, Stata

Similar, expensive, training legacy, smaller ecosystem, less graphics. SAS common for data management in clinical trials

Matlab

Similar, expensive, engineers don't think stochastically

Berkeley Madonna

Best for model building and exploration, no random effects

Who uses R?

- Statisticians
- Ecologists
- Molecular Biologists
- Data Miners
- Finance Analysts
- Pharmacometricians!

Why are these people using R?

Fast, efficient, error free data analysis

Program once, run twice…

Accessible software

Free *and* permanently available

Expert software

Implements latest methods & ideas

Reproducible research

All data made available
All calculations recorded permanently
Documentation so a "stranger" can follow the analysis

Getting R

Project home

http://www.r-project.org/

CRAN - Comprehensive R Archive Network

A network of mirror sites - use the closest mirror

Download and install base

e.g. Windows binaries

Install packages

Beware university firewalls and proxies
ackages can be installed from downloaded *.zip files

One version of R for One analysis!

Keep old versions on your computer

A Key Concept for Learning R

Getting help

- Functions & packages have extensive help with examples
- Many tutorials on the web
- Active mailing lists
- Blogs
- Textbooks
- Mentors
- Type "R" into Google, and it will know what you mean

Your first command

The prompt ">" indicates R is ready to receive a command.
Type 2+2 into the command console.
Pressing enter submits the command to R

```
> 2+2
```

```
[1] 4
```

[1] indicates the first entry in the first line of output from R
The answer "4" is the first number in the output line.
This is handy for multiple line answers.

Your second and third commands

Enter these two commands.
Pressing enter after each line to submit them to R.

```
> result <- 2+2
> result
```

```
[1] 4
```

The sum of 2 and 2 has been assigned to an object called "result".
Typing result will show the contents of the object.
It's a single number, 4.

R is case sensitive

These are all different objects:

- Result
- result
- RESULT

Inputing commands

Type into the console

line by line, inefficient

Cut and paste from a text file

collect commands in text file (a script)
give it a *.R extension

Source a script from a complete text file

commands in file are processed from top to bottom

"Integrated Development Environments" for R

RStudio is popular - http://www.rstudio.com
We will use the Windows GUI and a text editor

Text editors

R scripts are written with a text editor

- There are many to choose from
- Try to find one with:
  - syntax highlighting
  - line numbers
  - compare files
  - search and replace across files
  - "launching" of scripts to other software
- We will use Notepad++ - http://notepad-plus-plus.org

More about objects

```
> result1 <- 2+2
> result2 <- 4+4
> ls()
```

```
[1] "result1" "result2"
```

There are two objects in the workspace, result1 and result2
ls() is an inbuilt function.
This lists the objects in the current workspace.
Functions are called by name followed by brackets with the function arguments.
ls is called without arguments here, so the brackets are empty

Objects can be over-written

This is useful, but be careful.
Assignment (<-) works from right to left.

```
> result1 <- 2+2
> result1
```

```
[1] 4
```

```
> result1 <- result1+10
> result1
```

```
[1] 14
```

Objects in R can be of different types

- Numeric
- Character
- Logical
- Vector
- Factor
- List
- Matrices
- Dataframe

Numeric vector basics

Vectorised operations are preferred in R
These are quick to code and computationally fast

```
> vector1 <- c(1,2,3,4)
> vector1
```

```
[1] 1 2 3 4
```

```
> vector2 <- vector1*10
> vector2
```

```
[1] 10 20 30 40
```

Note that every element of vector1 has been multiplied by 10

Numeric vector basics

R vector operations use recycling

```
> vector1 <- c(1,2,3,4)
> vector1
```

```
[1] 1 2 3 4
```

```
> vector2 <- c(0,10)
> vector3 <- vector1 + vector2
> vector3
```

```
[1]  1 12  3 14
```

Note that vector2 was reused twice because it is shorter than vector1

Accessing the elements of a vector

By index

```
vector1 <- c(10.1,23.4,3,49.7)
vector1[1]
```

```
[1] 10.1
```

```
vector1[3]
```

```
[1] 3
```

Accessing the elements of a vector

By name

```
vector1 <- c("AUC"=10.1,"Cmax"=23.4,"tmax"=3)
vector1["AUC"]
```

```
 AUC
10.1
```

```
vector1["tmax"]
```

```
tmax
   3
```

Dataframe basics

Dataframes

- The closest R has to a spreadsheet
- A dataframe is defined by rows and columns
- A column must have a single type of data
- But a dataframe can have columns of different types of data
- Usually read from a file

Dataframe - reading a file

exampledata.csv is a file in our working directory

find the working directory with getwd()
set the working directory with setwd()
file paths use forward slashes, not back slashes
e.g. setwd("P:/OSUcourse/Rcourse_Day1/0_Rdemo")

```
> exampledf <- read.csv("exampledata.csv")
> exampledf
```

```
   Study Subject Dose   AUC
1 PhaseI       1   10  34.3
2 PhaseI       2   20  61.9
3 PhaseI       3   30  80.2
4 PhaseI       4   40 111.5
```

Dataframe - column & row indicies

```
> exampledf
```

```
   Study Subject Dose   AUC
1 PhaseI       1   10  34.3
2 PhaseI       2   20  61.9
3 PhaseI       3   30  80.2
4 PhaseI       4   40 111.5
```

Dataframe - column & row indicies

```
> #First row
> exampledf[1,]
```

```
   Study Subject Dose  AUC
1 PhaseI       1   10 34.3
```

```
> #Fourth column
> exampledf[,4]
```

```
[1]  34.3  61.9  80.2 111.5
```

Dataframe - element indicies

```
> #Second row, fourth column
> exampledf[2,4]
```

```
[1] 61.9
```

```
> #Second row, first column
> exampledf[2,1]
```

```
[1] PhaseI
Levels: PhaseI
```

Dataframe - using columns by name

Using a function on a column

```
> exampledf$AUC
```

```
[1]  34.3  61.9  80.2 111.5
```

```
> mean(exampledf$AUC)
```

```
[1] 71.97
```

Dataframe - using columns by name

Calculations using 2 columns

```
> exampledf$CL <- exampledf$Dose/exampledf$AUC
> exampledf$CL
```

```
[1] 0.2915 0.3231 0.3741 0.3587
```

Logical operators

operator	meaning
==	equal to
!=	not equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
&	and
vertical line	or

Dataframe basics - subsetting

Subsetting a column

```
> exampledf$Dose
```

```
[1] 10 20 30 40
```

```
> exampledf$Dose[exampledf$Dose < 30]
```

```
[1] 10 20
```

Dataframe basics - subsetting

Subsetting a dataframe

```
> exampledf[exampledf$Dose < 30,]
```

```
   Study Subject Dose  AUC     CL
1 PhaseI       1   10 34.3 0.2915
2 PhaseI       2   20 61.9 0.3231
```

Factor basics

Categorical data are represented by factors in R

Factors are important in plots, data summaries and statistics
Don't mix up categorical(discrete) and continuous data
By default, R will import text data as factors

Factors have levels (names) and levels have an order

Default order is alphabetical
The names of factors can be text numbers! This will catch you out one day
Factors are represented internally as integers with names

Factor basics

Categorical data starts as a vector of text

```
> catdata <- c("Vehicle","Treatment1","Vehicle","Treatment2","Vehicle","Treatment1")
> catdata
```

```
[1] "Vehicle"    "Treatment1" "Vehicle"    "Treatment2" "Vehicle"
[6] "Treatment1"
```

```
> class(catdata)
```

```
[1] "character"
```

Factor basics

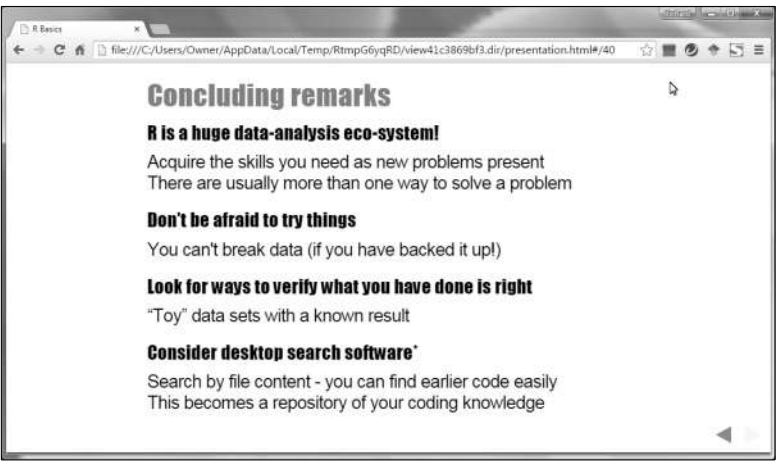
Turning text into a factor

```
> catdataf <- as.factor(catdata)
> catdataf
```

```
[1] Vehicle    Treatment1 Vehicle    Treatment2
Vehicle    Treatment1
Levels: Treatment1 Treatment2 Vehicle
```

```
> class(catdataf)
```

```
[1] "factor"
```


R Basics
file:///C:/Users/Owner/AppData/Local/Temp/RtmpG6yqRD/view41c3869bf3.dir/presentation.html#/40
Concluding remarks
R is a huge data-analysis eco-system!
Acquire the skills you need as new problems present
There are usually more than one way to solve a problem
Don't be afraid to try things
You can't break data (if you have backed it up!)
Look for ways to verify what you have done is right
"Toy" data sets with a known result
Consider desktop search software*
Search by file content - you can find earlier code easily
This becomes a repository of your coding knowledge