Calling R Externally : from command line, Python, Java, and C++

Leo Pekelis

February 2nd, 2013, Bicoastal Datafest, Stanford University

What is R? Why and why not use it?

- "R is a language and environment for statistical computing and graphics… R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible.""
  - *http://www.r-project.org/*
- R's main selling point is the massive amount of libraries allowing you to perform almost any statistical procedure in a single command
  - *There are currently 4273 available packages on CRAN, all independently tested, and generally peer reviewed.*
- R is great for performing analysis on a dataset, and presenting findings in a static set of graphics
- R is not so great for:
  - *Writing dynamic API, such as a web app*
  - *High end graphics*
  - *looping and iteration. This is devilishly slow in R, and in fact, most iterative algorithms call C externally from inside R.*
  - *Programmers who do not want to learn the eccentricities of a new language*

Do you want a crash course on R?

- Quick-R, http://www.statmethods.net/
  - *A useful website with concise explanations of some of the basic (and more advanced) features of R.*
  - *Designed to get you running statistical analysis quickly, and painlessly.*
- Sources to search for answering specific R related questions
  1. *"R (my question)" into Google*
  2. *http://stackoverflow.com/*
  3. *R mailing list archive - http://tolstoy.newcastle.edu.au/R/*
  4. *The documentation for any function can be called by ?function (e.g. ?mean).*
     - One of the top links in google will also generally return this.

Calling R from (your favorite programming language).

- Command Line
  - *If you can read from and write to the command line, a few simple commands can run R scripts and extract the output.*
- Python
  - *The package **rpy2** provides low-level interface to R, and access to all R libraries.*
  - *http://rpy.sourceforge.net/rpy2.html*
- Java
  - ***JRI** "is a Java/R Interface, which allows to run R inside Java applications as a single thread. Basically it loads R dynamic library into Java and provides a Java API to R functionality. It supports both simple calls to R functions and a full running REPL (read-eval-print loop)."*
  - *http://www.rforge.net/JRI/*
- C++
  - ***Rcpp** "provides matching C++ classes for a large number of basic R data types. Hence, a package author can keep his data in normal R data structures without having to worry about translation or transfering to C++. … the data structures can be accessed as easily at the C++ level, and used in the normal manner."*
    - http://dirk.eddelbuettel.com/code/rcpp.html
  - ***RInside** C++ classes to embed R in C++ applications.*
    - The RInside packages makes it easier to have 'R inside' your C++ application by providing a C++ wrapper class providing the R interpreter.
    - Makes use of **Rcpp**
    - http://dirk.eddelbuettel.com/code/rinside.html

What's to come ...

- The following will give more details on using each of the solutions defined in the previous slide.
- I do not pretend to be an expert. The man and help pages will be your friends.

Running R from Command Line

- General command line syntax, `R [options] [<infile] [>outfile]`
- Required & Useful Options
  - *`--vanilla`, evokes R such that it doesn't save data upon exiting (required), and doesn't start with any user profile information*
  - *`--max-ppsize=N`, max size of pointer protection stack. Can be increased up to 100,000 for large and complicated computations*
  - *`--args [arg1] [arg2] ...`, can specify input arguments after with spaces in between*
    - The arguments are then returned as a character vector by `commandArgs(TRUE)` within R
- An example may be something like: `R --vanilla '--args 1 10 TRUE' < myscript.R > outfile.txt`
  - *Note `outfile.txt` will get the console results of running the code in `myscript.R`*
  - *To have external access to data structures created while running the script incorporate `write.csv()` or `save()` into `myscript.R`*

Running R from Python: rpy2

- **rpy2** interacts with **R** in a number of different ways
- *high level interface* - designed to facilitate the use of R by Python programmers. R objects are exposed as instances of Python-implemented classes, with R functions as bound methods to those objects in a number of cases.
  - *`robjects.r` class instances one R session on loading the module.*
  - *Then Python commands interact with the R session, sending commands, and extracting varables.*
  - *e.g.* `robjects.globalenv['foo'] = 1.2` *- stores the value 1.2 in a new variable* `foo` *in R*
    - `foo = robjects.r['foo']` - extracts the variable `foo` from R into a Python variable of the same name
    - `foo[0]` - call to the first element of Python `foo`, prints 1.2
  - *Sending commands to R is handled similarly.*
- *read-eval-print loop* (REPL), where interactivity is important.
- *numpy* - a signature numerical package in Python can share objects with R through a subpackage of rpy2
- *low level interface* - similar to R's C level API. Used for performance optimization or in developing high level interactions.
- Full documentation: http://rpy.sourceforge.net/rpy2/doc-2.3/html/index.html

Running R from JAVA: JRI

- Start an R instance with `Rengine re=new Rengine(args, false, new TextConsole());`
  - *`TextConsole()` is a class which implements R callbacks. From looking at examples, it needs to be defined manually.*
- Then R commands are evalauted with `re.eval("R command")` and the output is the result returned by R
  - *e.g. Load dataset `iris` and store it in a JAVA subclass for representing R objects*

```
REXP x;
re.eval("data(iris)",false);
System.out.println(x=re.eval("iris"));
```

- Documentation is not great, but a couple example files can be found in the tar ball.
  - *http://www.rforge.net/JRI/files/*

Running R from C++: RCpp and Rinside

- In **RCpp**, as before, R data types have dedicated classes
  - e.g. `Rcpp::NumericVector` *class for (you guessed it) numeric vectors in R*
- Main use case is: "existing R code may be replaced byequivalent C++ code in order to reap performance gains."
- But can still call R code from C++. The following example samples 10 random normals with `sd=100`

```
Environment stats("package:stats");
Function rnorm = stats["rnorm"];
return rnorm(10,Named("sd", 100.0));
```

- **Rinside** - not a lot of documentation, but the source does come with 6 examples
  - *including a light web-app using the Wt toolkit …*

Webapps with R

If you are ever interested in creating a webapp with R, there are at least a few approaches.

1. **Shiny** allows for easy web applications using only R
    - *http://rstudio.github.com/shiny/tutorial/#*
2. **Rserve** is a TCP/IP server which allows other programs to use facilities of R
    - *Client-side implementations are available for popular languages such as C/C++, PHP and Java.*
    - *Rserve supports remote connection, authentication and file transfer.*
    - *Typical use is to integrate R backend for computation of statstical models, plots etc. in other applications.*
    - *http://www.rforge.net/Rserve/index.html*
3. Wt (pronounced as witty) is a C++ library for developing web applications.
    - *"The API is widget-centric and uses well-tested patterns of desktop GUI development tailored to the web. To the developer, it offers abstraction of web-specific implementation details, including client-server protocols, event handling, graphics support, graceful degradation (or progressive enhancement), and URL handling."*
    - *http://www.webtoolkit.eu/wt*