PROCEEDINGS OF SCIENCE

Data analysis using the Gnu R system for statistical computation

James Simone*
Fermi National Accelerator Laboratory, Batavia, Illinois 60510, USA
E-mail: simone@fnal.gov

R is a language system for statistical computation. It is widely used in statistics, bioinformatics, machine learning, data mining, quantitative finance, and the analysis of clinical drug trials. Among the advantages of R are: it has become the standard language for developing statistical techniques, it is being actively developed by a large and growing global user community, it is open source software, it is highly portable (Linux, OS-X and Windows), it has a built-in documentation system, it produces high quality graphics and it is easily extensible with over four thousand extension library packages available covering statistics and applications. This report gives a very brief introduction to R with some examples using lattice QCD simulation results. It then discusses the development of R packages designed for chi-square minimization fits for lattice n-pt correlation functions.

XXIX International Symposium on Lattice Field Theory
July 10 – 16 2011
Squaw Valley, Lake Tahoe, California

*Speaker.

PoS(Lattice 2011)048

1. What is R?

GNU R is a software environment for statistical computing, data analysis and for producing publication-quality graphics. R is widely used among statisticians developing statistical software and is used for data analysis in diverse fields of research and commerce. R is highly portable and extensible through a package system which also extends documentation via a `help()` function and example vignettes that include code and often sample data. Many state-of-the-art statistical techniques and domain specific code modules are available freely through web-based software repositories.

R is also an interpreted language, a dialect of S language created by John Chambers at Bell Labs for statistical computing. R is being developed by the *R Core Development Team* which includes the original developers Ross Ihaka and Robert Gentleman as well as Chambers. The original developers, influenced by the Scheme language made R lexically scoped, meaning a function only has access to variables visible at the time of the function definition. R is dynamically typed meaning that an object's type is a property of the object and not the variable its assigned to. Function definitions are another type that variables can hold. R has object oriented features including generic functions. For example, the behavior of `print()` function depends upon the type of the argument. Lexical scoping, functions, dynamical typing and object orientation together promote an expressive programming style.

2. Getting and running R

The main R website, *The R Project for Statistical Computing* [1], contains links to *The Comprehensive R Archive Network* (CRAN) [2] mirror sites worldwide. There are precompiled binary distributions for Linux, OS-X and Windows as well as source code distributions. R can be run interactively from the shell command line. Interactive mode is a convenient way to build simple plots, do simple analyses, load and test packages or read R documentation. Tab completion is available and it will list possible completions matching a partially typed identifier name. On quitting an R session, its possible to save the current context, including data and functions, which can be reloaded at restart. It is also possible to develop R scripts and interact with R from within the Emacs editor using ESS mode. R scripts may also be executed in batch mode. R computations can be done from within Python using, for example, the `Rpy2.py` module. The statistical capabilities of R are also a part of Sage [3], an open source platform for mathematics.

The R community has developed GUI environments for R, available for Linux (Tcl/Tk based), OS-X (Aqua/Coca) and Windows. Figure 1 is a screen capture of the OS-X GUI showing windows for script and data editing, output graphics (PDF) and R package management.

R supports "literate programming" which allows TEX documents to embed R code. This promotes "reproducible research", allowing runnable code to be distributed within a scientific paper, important in fields such as clinical drug trials. The preprocessor can split R code and document files, useful for developing both code and documentation. Preprocessing

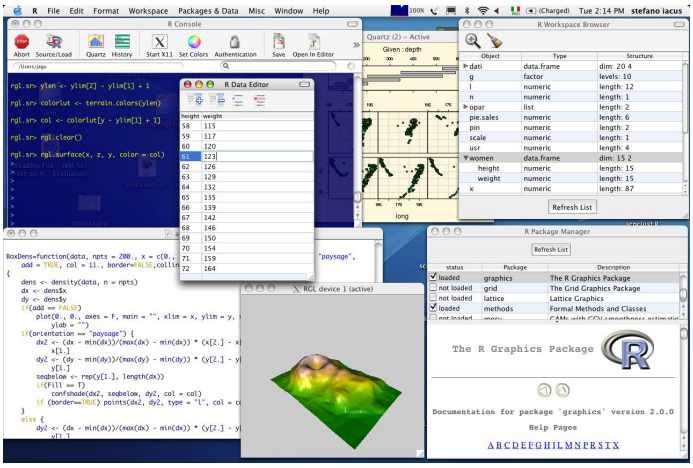

Figure 1: The R Aqua GUI for Mac OS-X.

can execute the R code and output TEX formatted code and R output results including any generated PDF plots. The R examples in this paper were generated in this way.

3. R features: package libraries, help and graphics

CRAN [2], the Comprehensive R Archive Network, a group of mirrored archive sites around the world, is the main mechanism for package distribution. The package system makes it easy to install and update R libraries. Currently, there are around three thousand R packages available. The scope of available packages is broad since R is the de-facto standard among statisticians developing statistical software. Packages consist of libraries of R functions, data sets and documentation. Libraries may wrap code written in languages such as C/C++ and FORTRAN. Typically packages are distributed in both binary form for Linux/OS-X/Windows and in source form. R comes with tools to help developers create new packages. These tools encourage the development and distribution of documentation.

Some domain specific packages are distributed via third-party web sites. For example, BioConductor [4] distributes packages for the bioinformatics community including open source data access, metadata annotation, analysis and specialized graphics. The BioConductor core team is based at the Fred Hutchinson Cancer Research Center and has collaborators world-wide. Their aim is to provide a common software platform that enables the rapid development and deployment of a broad range of plugable, well-documented, scalable, and interoperable software. BioConductor serves as a model of what could be developed for the lattice community.

The simplest way of getting help with e.g. some `function` is to type `help(function)`. This will return documentation on the function in a style similar to Unix `man` pages. Help commonly includes several simple code examples illustrating the function usage. R encourages the use of vignettes designed to provide further documentation such as HOW-TO recipes for using the package. The function `browseVignettes()` starts a web browser pointing it to a file listing available vignettes.

R easily produces publication-quality (in PDF) graphics of many types and styles, including many domain-specific plots. Many graphics are influenced by the ideas of Tufte [5] and others who advocate expressive and clear presentation of statistical data. There are three main graphics systems the 'base' system, included as a standard package, and two others that implement structured descriptions of graphics. The base system includes a generic `plot` function specialized to many object types. The *Lattice* graphics package implements the Trellis Graphics visualization framework based upon W.S. Cleveland's book *Visualizing Data* [6]. It is good for making panels of plots conditioned on sets of categorical parameters. The `ggplot2` package [7] is a system based upon Wilkinson's book *the Grammar of Graphics* [8]. Graphs are built by adding elements such as data mappings, geometry, statistic functions, facets etc. It simplifies plotting error bounds on points and uncertainty bands on fit curves.

4. A brief tour of R

This section briefly illustrates some R features through an example exploration of some bootstap data. The reader is referred to on-line documentation [2],books [9] and the *the R Journal* [10] for in-depth guides to using R.

The example bootstraps distributions are $\phi = f_{D_q}\sqrt{m_{D_q}}$, where f_{D_q} is the leptonic decay constant of a D_q meson with light quark of mass m_q. These data were written to separate files in ASCII by a custom C++ fitter. The function `dataFrameFromBoots` reads the (three) bootstrap files and labels the data by $m_q = 0.024, 0.03, 0.0415$. Data are returned in a `data.frame` object having columns containing the variables and rows of (correlated) resamplings of each variable:

```
> phi <- dataFrameFromBoots(bFiles, cNames)
> summary(phi)
     0.024             0.03            0.0415
 Min.   :0.1543   Min.   :0.1592   Min.   :0.1672
 1st Qu.:0.1575   1st Qu.:0.1617   1st Qu.:0.1691
 Median :0.1581   Median :0.1622   Median :0.1696
 Mean   :0.1581   Mean   :0.1622   Mean   :0.1696
 3rd Qu.:0.1586   3rd Qu.:0.1627   3rd Qu.:0.1701
 Max.   :0.1613   Max.   :0.1651   Max.   :0.1719
```

Tabulating means, std. deviations and quantiles (5% and 95%) is straightforward:

```
> cbind(mean = mean(phi), sd = sd(phi), t(sapply(phi, quantile, probs = c(0.05,
+     0.95))))
            mean           sd        5%       95%
0.024  0.1580667 0.0008711096 0.1566551 0.1594960
0.03   0.1621974 0.0007893725 0.1608956 0.1634911
0.0415 0.1696112 0.0007010243 0.1684707 0.1707906
```

`sapply` applies `quantile` to each variable of `phi`. The quantile vectors are transposed (`t`) and bound (`cbind`) with columns of the `mean` and `sd`.

The sample correlation matrix among bootstrap distributions is:

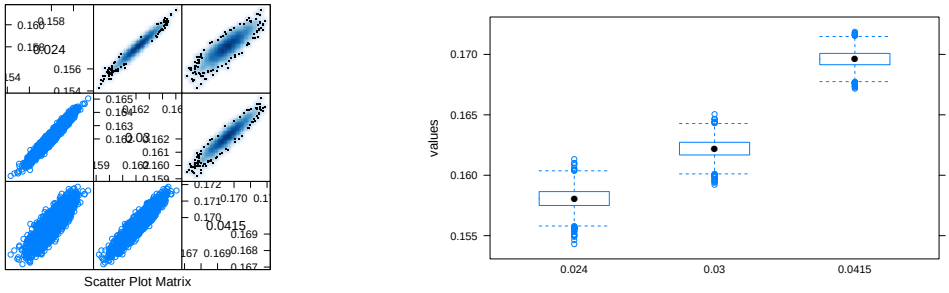

Figure 2: Plots of bootstrap distributions showing correlations and outliers.

```
> cor(phi)
           0.024      0.03    0.0415
0.024  1.0000000 0.9739778 0.8610807
0.03   0.9739778 1.0000000 0.9442953
0.0415 0.8610807 0.9442953 1.0000000
```

Correlations among distributions can be visualized in Lattice graphics xy scatter plots:

```
> plotpwscat <- splom(phi, upper.panel = panel.smoothScatter, lower.panel = panel.xyplot,
+     as.matrix = TRUE)
```

Figure 2 (left plot) shows the matrix of pairwise scatter plots of the bootstrap values. Different plot functions are used above and below the diagonal. Above, samples are smoothed to a color density with only the outliers shown as individual points. Below, all data appear as separate points.

A boxplot, `bwplot`, depict the quartiles, median, and outliers a distribution. Here, `stack` transforms `phi` into a data frame with two columns: the data values `values`, and an index `ind` numbering the distribution each belongs to. Figure 2 (right plot) shows the plot.

```
> plotbw <- bwplot(values ~ ind, data = stack(phi), horizontal = FALSE)
```

A `densityplot` is a smoothed alternative to binning data in a histogram. The data are convolved with a narrow Gaussian kernel function. In Figure 3 (left) the $m_q = 0.03$ bootstraps (in blue) are overlaid (in red) by a Gaussian (`dnorm`) with the same mean and width. The plot indicates the distribution is close to Gaussian.

```
> plotden <- densityplot(phi$"0.03", type = "density", panel = function(x,
+     ...) {
+     panel.densityplot(x, ...)
+     panel.mathdensity(dmath = dnorm, col = "red", args = list(mean = mean(x),
+         sd = sd(x)))
+ })
```

The quantile-quantile (QQ) plot compares quantiles of the data distribution to quantiles of a Gaussian distribution. It is better for visualizing deviations from in the distribution tails. The QQ plot on the right of Fig. 3 indicates the data are nearly Gaussian.

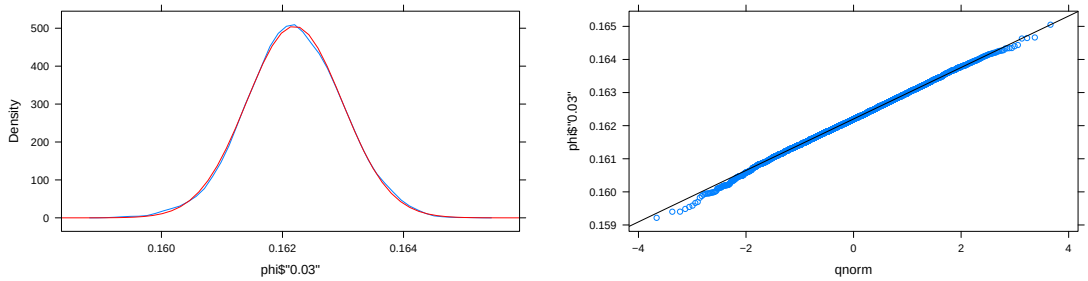

Figure 3: Graphical comparisons of a bootstrap distribution to the normal distribution.

```
> plotqq <- qqmath(phi$"0.03", panel = function(x, ...) {
+     panel.qqmath(x, ...)
+     panel.qqmathline(x, ...)
+ })
```

5. χ^2 minimization: n-pt data

R has least-squares fitting of variables from a `data.frame` for both linear and non-linear models, specified as `formula` objects. Unfortunately, R does not have chi-square minimization that includes a full data correlation matrix. R does have suitable minimizers for non-linear functions of many variables. It also has built-in support for bootstrap resampling procedures. Hence, it's reasonable to develop a fitter for lattice n-point data leveraging existing packages. Functionality that needs to be added includes: access to the n-pt data, (augmented) chi-square functions and an efficient specification of the model functions.

The n-pt data to be fit and associated meta-data are stored in relational databases. While R has packages to access SQL databases, the lattice data schema requires data decompression and table joins, which would not be simple to implement and optimize in R. Instead, the access functions are built by wrapping an existing C/C++ database code. Fortunately, the `Rcpp` package greatly simplifies the task of exposing C++ functions in R. The `Rcpp` design is modeled after boost/python which is already being used to access data in Python. Work has begun to make the lattice data container into an R S4 class and implement generic functions such as `plot` and `summary`. The goal is to build a complete R package.

A simple test of constrained chi-square minimization was done in R. A single heavy-light 2-pt function was fit over 14 timeslices to a model function having six parameters. No implementation of the model gradient function was provided, hence, gradients were approximated numerically. The `optimx` package was used for the minimization since it provides a common interface to many suitable minimizer algorithms. Specifying a list of algorithms produces a table comparing timings, the minimum chi-square value and counts of the number of function and gradient evaluations. Table 1 shows that the `ucminf` algorithm converges fast with fewer function and gradient evaluations.

min. χ^2	method	Nfunc	Ngrad	converged	time (sec)
15.323	CG	300	49	0	0.153
10.106	spg	2078	NA	1	2.382
8.961	L-BFGS-B	70	70	0	0.135
8.959	BFGS	201	14	0	0.071
8.958	Nelder-Mead	335	NA	0	0.076
8.958	ucminf	26	26	0	0.037
8.958	newuoa	2038	NA	0	0.526
8.958	bobyqa	1787	NA	0	0.498
8.958	nlm	NA	NA	0	0.057
8.958	nlminb	60	150	0	0.048

Table 1: A test of minimizer methods available in R.

Prototyping a simple fitter has helped guide the design of a more general fitter package. One area where optimization is desirable is the specification of the model functions. Although R expressions are naturally vector, an engine that interfaces to faster compiled code for both the model function and gradient is desirable. The prototype 2-pt fitter also included bootstrap analysis. The `boot` package provides extensive support for bootstrap error analysis. When used with the `multicore` package, processing is done in parallel.

6. Summary

R has many appealing features as a data analysis platform: portability, very good help/documentation, extensibility, support for a wide range of data formats, it's the de-facto standard platform for statistical analysis software and it has expressive publication-quality graphics. Domain-specific packages are need to take full advantage of R.

References

[1] http://www.r-project.org/, "The r project for statistical computing."

[2] http://cran.r-project.org, "The comprehensive r archive network (cran)."

[3] http://www.sagemath.org/, "The sage mathematics software system."

[4] http://www.bioconductor.org/, "Bioconductor: open source software for bioinformatics."

[5] E. R. Tufte, *The Visual Display of Quantitative Information.* Graphics Pr, ISBN: 978-0961392147, 2 ed., 2001.

[6] W. Cleveland, *Visualizing Data.* Hobart Press, ISBN:978-0963488404, 1993.

[7] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis.* Springer, ISBN: 978-0387981406, 2 ed., 2009.

[8] L. Wilkinson, *The Grammar of Graphics.* Statistics and Computing. Springer, ISBN: 978-0387987743, 1 ed., 1999.

[9] http://www.r-project.org/doc/bib/R-jabref.html, "Searchable r book index."

[10] http://journal.r-project.org/, "The r journal."