Future developments

Work in progress with Gareth Roberts and Martin Sköld aims to improve convergence and mixing of the MCMC algorithm by using a more appropriate parameterisation.

Acknowledgements

Some of the C code in the package is based on code originally developed together with Rasmus Waagepetersen. We are grateful to Peter J. Diggle for giving encouragement and support to the development of the package. Ole acknowledges support from DINA, NERC and the EU TMR network. Paulo acknowledges CAPES/Brasil grant 1676/96-2.

Bibliography

O.F. Christensen and R.P. Waagepetersen. Bayesian prediction of spatial count data using generalized linear mixed models. *Biometrics*, 58:280–286, 2002. 26

P. J. Diggle, P. J. Ribeiro Jr, and O. F. Christensen. An introduction to model-based geostatistics. In M. B. Hansen and J. Møller, editors, *Spatial statistics and computational methods*. Springer Verlag, 2002. (to appear). 27

P. J. Diggle, J. A. Tawn, and R. A. Moyeed. Model-based geostatistics (with discussion). *Appl. Statist.*, 47:299–350, 1998. 26

Paulo J. Ribeiro, Jr. and Peter J. Diggle. geoR: A package for geostatistical analysis. *R News*, 1(2):14–18, 2001. 26

H. Zhang. On estimation and prediction for spatial generalised linear mixed models. *Biometrics*, 58:129–136, 2002. 26

Ole F. Christensen
Lancaster University, UK
o.christensen@lancaster.ac.uk

Paulo J. Ribeiro Jr
Universidade Federal do Paraná, Brasil
and *Lancaster University, UK*
Paulo.Ribeiro@est.ufpr.br

Querying PubMed

Web Services

by Robert Gentleman and Jeff Gentry

Introduction

While many view the world wide web (WWW) as an interactive environment primarily designed for interactive use, more and more sites are providing web services that can be accessed programmatically. In this article we describe some preliminary tools that have been added to the *annotate* package in the Bioconductor project www.bioconductor.org. These tools facilitate interaction with resources provided at the National Center for Biotechnology Information (NCBI) located at www.ncbi.nlm.nih.gov. These ideas represent only a very early exploration of a single site and we welcome any contributions to the project in the form of enhancements, new tools, or tools adapted to other sites providing web services.

We believe that web services will play a very important role in computational biology. In part this is because the data are complex and gain much of their relevance by association with other data sources. For example, knowing that there is a particularly high level of messenger RNA (mRNA) for some gene (or set of genes) does not provide us with much insight. However, associating these genes with the relevant scientific literature and finding common themes often does provide new insight into how these genes interact.

We can think of a cellular pathway as a set of genes that interact (through the proteins that they produce) to provide a particular function or protein. A second way of obtaining insight into the role of certain genes would be to examine the expression of mRNA for a set of genes in a particular pathway, or to take a set of genes and determine whether there is a particular pathway that contains (most of) these genes.

Both of these examples rely on associating experimental data with data that are available in databases or in textual form. These latter data sources are often large and are continually evolving. Thus, it does not seem practical nor prudent to keep local versions of them suitable for querying. Rather, we should rely on retrieving the data when it is wanted and on tools to process the data that are obtained from on-line sources.

It is important to note that most of the processes we are interested in can be carried out interactively. However, there are two main advantages to designing programmatic interfaces. First, interactive use introduces a rate limiting step. The analysis of genomic data needs to be high throughput. A second reason to prefer programmatic access is that it allows for the possibility of combining data from several sources, possibly filtered through online resources, to provide a new product. Another reason to prefer a programmatic approach is that it makes fewer mistakes and

gets bored or distracted somewhat less easily than a human.

In this article we describe the features added to the **annotate** package that facilitate interactions with abstracts made available from the PubMed archive supplied by the National Library of Medicine. There are many other functions in **annotate** such as `genbank` and `locuslinkQuery` that provide web services tools that we will not discuss here.

Querying PubMed

We have written very little code to provide a very powerful tool for analysing textual information but rather have relied on some well written tools to provide basic services. Our tools are still in development and are likely to change over the next few months as we gain experience but we believe that the ideas are of interest to many researchers in the area of Bioinformatics and computational biology.

We will focus on the analysis of microarray data but this is just to provide some specific examples, the ideas are much more widely applicable. A very simplified view of the biology is that DNA makes RNA which in turn is translated into proteins. Proteins and their interactions basically provide mechanisms for all cellular functions and hence are of great interest.

DNA microarray experiments give us a static view of the level of messenger RNA in a set of biological samples. We are often interested in finding sets of mRNA that are highly expressed or not expressed in particular subsets of our samples. A typical experiment involves the study of a small number of samples (usually less than 100) and a large number of genes (usually more than 10,000).

There are several reasons why we would like to bring more biologically relevant information into the data analytic process. This can be done in many ways, one of which we will discuss next. There is a large repository of biological knowledge available in the published scientific literature. We are becoming more capable of processing and dealing with that literature in some specific ways.

We have exploited connections, a publicly available resource PubMed and the **XML** package to construct a tool for obtaining PubMed abstracts in a standard format. This is possible since PubMed provides tools for downloading their data in XML format. This makes the processing on our end much simpler and less error prone.

Once we have the abstracts we can process them in many ways. We can provide the data analyst with a set of titles or of keywords or of any other components. In addition, if the abstract is provided we can do some plain text searching of it using tools such as `grep` and `regexp`. In some cases the text of the entire document is available and that also can be obtained electronically and processed in a similar fashion.

In the **annotate** package we have produced a simple interface from microarray data to the PubMed abstracts. A user can select a set of genes that they find interesting and map these to an appropriate set of identifiers. These identifiers are then used to obtain the PubMed identifiers for associated abstracts. Finally PubMed is queried and these abstracts are downloaded into a particular form than can be processed using standard tools in R.

The code for a simple example is given below. In this example we have simply selected some *interesting genes* and then retrieved the abstracts and searched them for the word *protein*.

```
library(Biobase)
library(annotate)
## load up our test data set
data(eset)
## generally these would come from a filtering
## or other selection procedure
int.genes <- geneNames(eset)[273:283]
absts <- pm.getabst(int.genes, "hgu95A")
pm.titles(absts)
## which abstracts mention the word protein
wh.Protein <- sapply(absts,
   function(x) pm.abstGrep("[Pp]rotein", x))
```

It would also be nice to go in the other direction and we are actively developing tools that will help a researcher go from a gene of interest to a set of related genes. These can then be used to examine the available data to see if it concurs with the scientific literature.

This is a work in progress in almost every sense. Our programs need to be further developed. PubMed is growing and expanding the data resources it provides. New text processing algorithms are being developed that we would like to adapt to R so that they can be used to provide context for some of the queries being made. We believe that tools of this nature will become essential for computational biology.

The amount of processing that can currently be done is limited by a lack of algorithms and available data. There are a large number of data bases that exist but for which there is no programmatic way to access the data they contain or to perform queries remotely. Essentially we hope that more database providers see themselves as web services providers rather than as interactive tool providers. Making a tool interactive introduces a rate limiting step in our search for high throughput analysis tools. On the algorithm front it will be essential to start exploring ways of providing contextual information. We are currently limited to testing whether a gene's name (in one of its many forms) appears but cannot easily tell whether it is a passing reference or if there is substantial information being provided.

The details

The ability of the query functions to interact with the NCBI databases is provided by a set of well documented utilities provided by NCBI that work through standard HTTP connections. We describe some of them here, but refer the interested reader to the NCBI web site for definitive documentation.

All of our query functions can have the results of the query returned as an R object or have it rendered in the user's web browser. The NCBI resources are queried through CGI scripts which take various '&' separated parameters.

The LocusLink functions `locuslinkByID` and `locuslinkQuery` provide the ability to perform LocusLink searches either via a set of specific ID values or an actual text search respectively. In the former case, the URL to use is simple, `http://www.ncbi.nih.gov/LocusLink/LocRpt.cgi?l=<id1>,<id2>,<id3>...`. For all comma separated lists we use the HTML symbol '%2c' to circumvent browser issues.

The ability to run a text search is slightly more complicated, as the LocusLink database is divided by species. The R function `locuslinkQuery` takes a text query, and a set of zero or more species (the default is `HS`, human). Each supplied species is pasted to the string '`&ORG=`', and then appended to the query itself. Finally this string is pasted to the base URL to yield: `http://www.ncbi.nih.gov/LocusLink/list.cgi?Q=<query>&ORG=<S1>&ORG=<S2>...`. This URL is then sent to the user's browser, and the proper LocusLink page is displayed. The interactions with LocusLink are limited because LocusLink does not provide for programmatic querying.

The `pubmed` and `genbank` functions both utilize a set of CGI tools known as *Entrez*. *Entrez* provides the same basic searching mechanisms that are available to the interactive user, as well as a set of utilities designed for downloading the results programmatically.

To render the output into the user's browser we use the `query` tool provided by *Entrez*. `query` understands either Pubmed ID values or Genbank accession numbers, although the structure of the query is slightly different depending on which is used. For Pubmed the option is,

```
cmd=Retrieve&db=<id1>,<id2>...
```

and for Genbank it is

```
cmd=Search&db=<acc1>,<acc2>...
```

In either case, this is attached to form the full URL `http://www.ncbi.nih.gov/entrez/query.fcgi?tool=bioconductor&cmd=Retrieve&db=<id1>...`. Note the use of the `tool` directive, which is requested by NCBI to mark automated usage of their *Entrez* tools to help them track usage information, etc.

Other forms of output are available if requests are made using the `pmfetch` tool provided by *Entrez*. Since XML is one of the formats available and R has the **XML** package (D. Temple Lang) we prefer to obtain the data in XML. The results of a query are a list of XML objects that can be further processed. The interactive tools discussed above (`pm.getAbst`) rely on `pubmed`. The `pmFetch` utility has four different options that can be set by the user. The first is what format to display the data in, and is noted in the URL by `report=`*type*, where *type* is one of a variety of options (e.g., brief, Medline, Docsum, xml). The next option determines what format the data should be rendered in (text, file, or html — which is the default) and is set using `mode=`*type*. Third is a field to set which NCBI database to retrieve the data from (PubMed, Protein, Nucleotide, or Popset), and is called with `db=`*type*. Lastly, one needs to specify which IDs to use, which can be either PubMed IDs (PMID), MEDLINE Identifiers (UI), or molecular biology database (GI) and this is done with the command `id=<id1>,<id2>,...` or `id=<id1>&<id2>&...`. Note that these four parameters can be in any order within the URL and are separated using the '&' symbol.

For these functions we are always using text mode and a display value of XML. The database flag is currently set to either PubMed or Nucleotide depending on if this is being constructed by `pubmed` or `genbank` and we are using the comma separated method for the ID flags. As an example, if one were to make the call within R:

```
pubmed("11780146","11886385","11884611",
       disp="data")
```

the actual URL constructed to be sent to NCBI will be:

```
http://www.ncbi.nih.gov/entrez/utils/pmfetch.
fcgi?report=xml&mode=text&tool=bioconductor&db=
PubMed&id=11780146\%2c11886385\%2c11884611
```

Opening an `http` connection to this URL would provide the requested data. This is stored in XML objects and a list returned to the user for further processing.

Acknowledgment

Robert Gentleman's work is supported in part by NIH/NCI Grant 2P30 CA06516-38.

Robert Gentleman
DFCI
rgentlem@jimmy.harvard.edu

Jeff Gentry
DFCI
jgentry@jimmy.harvard.edu

evd: Extreme Value Distributions

by Alec Stephenson

Extreme value distributions arise as the limiting distributions of normalized maxima. They are often used to model extreme behaviour; for example, joint flooding at various coastal locations.

evd contains simulation, distribution, quantile and density functions for univariate and multivariate parametric extreme value distributions.

It also provides functions that calculate maximum likelihood estimates for univariate and bivariate models.

A user's guide is included in the package. It can also be downloaded directly (in postscript or pdf) from http://www.maths.lancs.ac.uk/~stephena/.

Introduction

Let $X_1, \ldots, X_m$ be *iid* random variables with distribution function F. Let $M_m = \max\{X_1, \ldots, X_m\}$. Suppose there exists normalizing sequences a_m and b_m such that $a_m > 0$ and as $m \to \infty$

$$\Pr(Z_m \leq z) = [F(a_m z + b_m)]^m \to G(z)$$

for $z \in \mathbb{R}$, where G is a non-degenerate distribution function and $Z_m = (M_m - b_m)/a_m$ is a sequence of normalized maxima. It follows that the distribution function G is generalized extreme value, namely

$$G(z) = \exp\left[-\left\{1 + \xi\left(z - \mu\right)/\sigma\right\}_+^{-1/\xi}\right],$$

where (μ, σ, ξ) are location, scale and shape parameters, $\sigma > 0$ and $h_+ = \max(h, 0)$. The case $\xi = 0$ is defined by continuity.

Multivariate extreme value distributions arise in a similar fashion (see Kotz and Nadarajah (2000) for details). In particular, any bivariate extreme value distribution can be expressed as

$$G(z_1, z_2) = \exp\left\{-(y_1 + y_2)A\left(\frac{y_1}{y_1 + y_2}\right)\right\},$$

where

$$y_j = y_j(z_j) = \{1 + \xi_j(z_j - \mu_j)/\sigma_j\}_+^{-1/\xi_j}$$

for $j = 1, 2$. The dependence function A characterizes the dependence structure of G. $A(\cdot)$ is a convex function on $[0, 1]$ with $A(0) = A(1) = 1$ and $\max(\omega, 1 - \omega) \leq A(\omega) \leq 1$ for all $0 \leq \omega \leq 1$. The jth univariate marginal distribution is generalized extreme value, with parameters (μ_j, σ_j, ξ_j).

Parametric models for the dependence function are commonly used for inference. The logistic model appears to be the most widely used. The corresponding distribution function is

$$G(z_1, z_2; \alpha) = \exp\left\{-(y_1^{1/\alpha} + y_2^{1/\alpha})^\alpha\right\}, \qquad (1)$$

where the dependence parameter $\alpha \in (0, 1]$. Independence is obtained when $\alpha = 1$. Complete dependence is obtained as $\alpha \downarrow 0$. Non-parametric estimators of A also exist, most of which are based on the estimator of Pickands (1981).

Features

- Simulation, distribution, quantile, density and fitting functions for the generalized extreme value and related models. This includes models such as $[F(\cdot)]^m$ for a given integer m and distribution function F, which enable e.g. simulation of block maxima.
- Simulation, distribution, density and fitting functions for eight parametric bivariate extreme value models. Non-parametric estimates of the dependence function can also be calculated and plotted.
- Simulation and distribution functions for two parametric multivariate extreme value models.
- Linear models for the generalized extreme value location parameter(s) can be implemented within maximum likelihood estimation. (This incorporates the forms of non-stationary most often used in the literature.)
- All fitting functions allow any of the parameters to be held fixed, so that nested models can easily be compared.
- Model diagnostics and profile deviances can be calculated/plotted using `plot`, `anova`, `profile` and `profile2d`.

Application

The `sealevel` data frame (Coles and Tawn, 1990) is included in the package. It has two columns containing annual sea level maxima from 1912 to 1992 at Dover and Harwich, two sites on the coast of Britain. There are 39 missing maxima in total; nine at Dover and thirty at Harwich.

The maxima on both margins appear to be increasing with time. The following snippet fits the logistic model (1) with simple linear trend terms on each marginal location parameter.

```
data(sealevel) ; sl <- sealevel
tt <- (1912:1992 - 1950)/100
lg <- fbvlog(sl, nsloc1 = tt, nsloc2 = tt)}
```