## Manual for

# bms_runner

***BayesTraits*-related script for finding functional associations among genes for Linux and Mac OS X**

## version 1.4 (3 March 2011)

Manual updated, August 2012



Information: Daniel Barker, db60@st-andrews.ac.uk

Download: http://biology.st-andrews.ac.uk/cegg/downloads.aspx

## Introduction

This manual describes *bms_runner*, a high-level computer program (script) to assist the search for functional associations among genes (Barker et al. 2007).

The separate program *BayesTraits*, formerly known as *BayesMultistate and Discrete* (Pagel et al. 2004; http://www.evolution.reading.ac.uk) can, among other things, seek correlated gain and loss of traits on a phylogenetic tree using the maximum likelihood approach of Pagel (1994). The script described here helps tailor this use of *BayesTraits* to the prediction of functional associations among genes, and helps apply it on a large scale.

*BayesTraits* (Pagel et al. 2004) can be used to seek correlated gain and loss of two discrete traits on a phylogenetic tree of species. For finding functional associations among genes, the traits represent presence and absence of genes from species' genomes (Barker & Pagel 2005). It is unlikely that the origin of a specific gene would occur more than once, and separate gains of the gene in different species by horizontal transfer are assumed to be rare, especially among eukaryotes. This suggests genes have a very low rate of gain. But once gained, a gene may be lost many times among the descendants of the species within which the gene originated. The rate of loss may be low or high and will vary according to the gene. The default likelihood models of independent (uncorrelated) and dependent (correlated) evolution implemented within *BayesTraits* are deliberately very general, and do not model this situation accurately. The models assume *a priori* that gains and losses of a trait are equally likely, and the software will allow modeling of multiple gains of a trait on a phylogenetic tree. It is, however, possible to fix the rate of gain of the trait at a low value. *BayesTraits* allows this modification of the models with its *restrict* command.

It is difficult to know which low value for rate of gains is appropriate. This will depend on the phylogeny and data at hand. Because of this, the *bms_runner* script will try several values for the rate of gains, and will assess their performance on a small test set of known functionally linked genes and known non-functionally linked genes provided by the user. This allows use of the 'ML-constrained' approach of Barker et al. (2007).

The user provides a positive training set of pairs of genes (or gene products) known to be functionally linked, and a negative training set of pairs of genes (or gene products) known not to be functionally linked. The script *bms_runner* can then predict functional links among these pairs without applying our prior knowledge of the genes, using *BayesTraits*. At the training stage, *bms_runner* performs the analysis leaving the rates of gain free to vary, and then fixing the rates of gain at various values. It then uses prior knowledge (known functional links and known non-functional links) to output the specificity and sensitivity of the results with these different rates of gain.

Once trained with representative data, *bms_runner* may then be used to predict functional links among a larger set of pairs of genes for which functional relationships are unknown. This is the biologically interesting stage, and allows prediction of novel functional links among genes. However, for accuracy, the previous stage of training, to decide the appropriate value at which rates of gain must be fixed, is important.

*bms_runner* may be used without options or just with the `-help` option. It will then output brief guide to possible options (to standard error or standard output, respectively) and exit. Options are described in detail below.

## Citation

The citation for *bms_runner* is Barker et al. (2007). Please also cite *BayesTraits* (Pagel et al. 2004), without which *bms_runner* is useless.

The likelihood approach for seeking correlated change among discrete traits was developed by Pagel (1994). It has been applied to prediction of functional associations between proteins by Barker & Pagel (2005) and Barker et al. (2007).

## System requirements

*bms_runner* requires a Pentium-compatible computer running the Linux operating system, or a PowerPC-based or Intel-based Apple computer running Mac OS X. The same version of *bms_runner* will run on any of these systems.

*BayesTraits* must also be present. The appropriate version of *BayesTraits* for your computer (i.e. *BayesTraits* for Linux, OS X PPC, or OS X Intel) may be downloaded from http://www.evolution.reading.ac.uk.

*bms_runner* requires that the Perl module `Statistics::Distributions` has been installed. This is available from the Comprehensive Perl Archive Network (CPAN, http://www.cpan.org). We have tested *bms_runner* with `Statistics::Distributions` version 1.02. One method of installing the latest release of `Statistics::Distributions` is as follows. First enter this command at the shell prompt:

```
perl -MCPAN -e shell
```

If you have installed Perl modules in this way before, you will see a `cpan>` prompt. If this is the first time, you will be asked various questions first. Mostly these are straightforward, but if you are not an administrator on your computer, you will wish to specify a non-default installation location for Perl modules. To do this, when asked for parameters for the `perl Makefile.PL` command, specify the directory you wish to use as a `PREFIX`. E.g. if you reply as follows, modules will be installed in the directory `/home/me/perl_libs`:

```
PREFIX=/home/me/perl_libs
```

Anyway, to download, test and install the necessary package, type this at the `cpan>` prompt:

```
install Statistics::Distributions
```

After successful installation, press CTRL-D to exit. If you installed in a non-default location, you should set your `PERL5LIB` environment variable so the library is found. If you type

```
echo $PERL5LIB
```

at the Linux or OS X command prompt and it shows nothing (i.e. the `PERL5LIB` variable contains nothing as yet), and if you are using the `bash` shell, at these lines in your `~/.bash_profile` file:

```
PERL5LIB=/path/to/folder
export PERL5LIB
```

If `echo $PERL5LIB` shows it does contain some folders already, instead add these lines in your `~/.bash_profile` file:

```
PERL5LIB=$PERL5LIB:/path/to/folder
export PERL5LIB
```

In either case, replace `/path/to/folder` with the path of the directory containing `Statistics`. Next time you start a command window, `PERL5LIB` will be set to the value you have specified and *bms_runner* should be able to find the module.

*bms_runner* is non-graphical and should usually be run from a command prompt, e.g. perhaps an interactive shell running through *xterm* or via a remote connection such as *ssh*, for Linux; or *Terminal* or a remote connection such as *ssh*, for Mac OS X. Since the script does not require interactive input, it can also be conveniently run through a job submission system on a high-performance computer cluster.

## Training

Usage:

*bms_runner* -train -positive <pos_file> -negative <neg_file> -tree <tree_file> -species <spp_file> -patterns <patterns_file> [ -exe <executable_file> ] [ -lrfile <lr_output_file> ] [ -try <rate1,rate2,...> ] [ -tmpdir <dir> ]

E.g.:

```
bms_runner -train -positive pos.txt -negative neg.txt \
-tree treefile.nex -species spp.txt \
-patterns distributions.txt
```

To save the summary of training results rather than print them to the screen, just redirect standard output to a file in the usual way, e.g.:

```
bms_runner -train -positive pos.txt -negative neg.txt \
-tree treefile.nex -species spp.txt \
-patterns distributions.txt > training_res.txt
```

Note that if the output file already exists, its previous contents will be destroyed.

<pos_file> is a text file giving a list of IDs of pairs of genes that are known to be linked. Genes are identified by an ID tag of the user's choice, which might be the gene name, the protein name, a database accession number or an arbitrary tag devised by the user. Each line gives IDs of a pair of genes known to be functionally linked, separated by a tab. E.g. part of <pos_file> might look like this:

```
10383786	6320190
10383786	6322750
L43A	L30
6324445	6325359
```

It is not acceptable to list a gene as functionally linked to itself, so the IDs on each line must not be the same. The same ID can occur in different lines, though.

<neg_file> is a text file giving a list of pairs of genes that are known *not* to be functionally linked. The format is the same as for <pos_file>.

<tree_file> is a Nexus-format file giving the phylogenetic tree for the species in the study, which must be prepared in advance by the user. The *bms_runner* script assumes there is a single, rooted phylogenetic tree, and that it has branch lengths. This will often be a maximum likelihood tree, but could be any kind of tree with branch lengths (though an inaccurate tree is likely to reduce the quality of results from *bms_runner*). All branch lengths must be greater than zero, the tree must be bifurcating, and the tree must be correctly rooted. Many phylogeny reconstruction programs will give a zero-length branch between the ingroup and the outgroup, which is inappropriate here. Before running *bms_runner*, the user should either remove the outgroup from the tree (as in Barker et al. 2007), or reposition the root at a more realistic position between the ingroup and outgroup. In the absence of evidence suggesting otherwise, it might be reasonable to place the root mid-way between ingroup and outgroup, so the lengths of the initial branches leading to ingroup and outgroup are identical (as in Barker & Pagel 2005).

<spp_file> is a text file listing the species in <tree_file>, in the exact sequence used in the cross-species distribution patterns of genes provided by the user. One species is given per line, and the names must precisely match those in <tree_file>. E.g.

<spp_file> might look like this:

```
s_cerevisiae
s_paradoxus
s_mikatae
s_bayanus
c_glabrata
k_lactis
e_gossyppi
c_albicans
d_hansenii
y_lipolytica
m_grisea
g_zeae
n_crassa
a_nidulans
s_pombe
c_neoformans
u_maydis
e_cuniculi
h_sapiens
d_melanogaster
c_elegans
```

to indicate that the first position in each gene's cross-species distribution pattern represents s_cerevisiae, the second position represents s_paradoxus, and so on until the final position, which represents c_elegans.

<patterns_file> is a text file giving the cross-species distribution patterns of all the genes in <pos_file> and <neg_file>. It may also give the distribution patterns of other genes, but these will be ignored. Each line gives a gene ID and a binary number representing the cross-species distribution pattern. Within the distribution pattern, '1' indicates presence of the gene and '0' indicates absence of the gene. The ID and distribution pattern are separated by a tab. Each digit of the binary number represents a different species. *bms_runner* infers labels for these digits from <species_in_order_file>, which lists the name of the first species in the distribution pattern, followed by the second species, and so on. E.g. with the <species_in_order_file> above,

```
10383786  11110100000000000000
```

indicates that the gene with accession number 10383786 is present in s_cerevisiae, s_paradoxus, s_mikatae, and s_bayanus, absent in c_glabrata, present in k_lactis, and absent in e_gossyppi, c_albicans, d_hansenii, y_lipolytica, m_grisea, g_zeae, n_crassa, a_nidulans, s_pombe, c_neoformans, u_maydis, e_cuniculi, h_sapiens, d_melanogaster and c_elegans. These binary cross-species distribution patterns are often referred to as 'phylogenetic profiles' (Pellegrini et al. 1999) and must be prepared by the user prior to running *bms_runner*.

<executable> is an optional parameter giving the executable binary file of

*BayesTraits*, which will be launched by the script. This may be a full path, a path relative to the directory in which the script is launched (beginning with a period '.'), or a file name. If a file name, it will be searched for according to the `PATH` environment variable. If <executable> is not specified using the `-exe` option, the default of `BayesTraits` is used.

<lr_output_file> is an optional parameter giving the file to store the results of *BayesTraits* analyses performed during training. If <lr_output_file> is not specified using the `-lrfile` option, these details will be lost and it will then be impossible to perform any post-processing of these results yourself. Hence, it is recommended that a file be specified.

<rate1,rate2,...> is an optional parameter list, giving a comma-separated list of values to use for the rate of gain parameter *r* (Barker et al. 2007). If only one value is required, this may be given without any commas. Rates of gain should be non-negative real numbers (for the 'ML-constrained' method of Barker et al. 2007), or else `unrestricted` to estimate rates of gain by maximum likelihood (the 'ML-unconstrained' model). If no rates are specified using the `-try` option, the following default is used:

```
-try unrestricted,0.1,0.2,0.4,0.8,1.5,3,6
```

<dir> is an optional parameter diving the directory where temporary files should be stored. If no temporary directory is specified using the `-tmpdir` option, the system default will be used. When running on a high-performance computer cluster, it will reduce the load on the internal network, and may increase the speed of *bms_runner*, if temporary files are stored in a directory local to each node.

A tab-delimited table is output to standard output. Its columns are as follows:

| | |
|---|---|
| `gains` | rate of gain *r*, or `unrestricted` if not fixed; |
| `lr_cut` | cutoff for the likelihood ratio statistic *LR* (i.e. the minimum *LR* taken to indicate a prediction of functional linkage); |
| `p_cut` | assumed *p*-value corresponding to this *LR* cutoff (i.e. the maximum *p*-value taken to indicate a prediction of functional linkage); |
| `preds` | number of predictions of functional linkage at this cutoff; |
| `spec` | specificity at this cutoff; and |
| `sens` | sensitivity at this cutoff. |

The Likelihood ratio statistic, *LR*, is calculated as follows (Pagel 1994):

$$LR = 2[\ln L(D) - \ln L(I)],$$

where $L(D)$ is the likelihood of the model of dependent evolution (correlated gain and loss), and $L(I)$ is the likelihood of the model of independent evolution (uncorrelated gain and loss). According to theory, it is always the case that $LR \geq 0$, and higher values of *LR* indicate greater evidence for the dependent model compared to the independent model.

In the table summarizing the results of training, *bms_runner* uses a range of 50 *LR* cut-offs, ranging from the minimum *LR* encountered to a value close to the maximum *LR* encountered, with a decreasing interval from one cut-off to the next. This gives a larger number of cut-offs at high *LR*s (low *p*-values), which tend to be the most interesting.

*p*-values are obtained from *LR* on the assumption that the null distribution of *LR* is $\chi^2$ with degrees of freedom equal to the difference in number of free parameters between the dependent and independent models, four degrees of freedom in our case. In practice, this assumption is unlikely to be correct (Pagel 1994; Pagel 1997; Barker & Pagel 2005). *p*-values are also unlikely to be comparable across different settings for *r*. Even without these problems, caution is necessary when interpreting *p*-values from multiple tests. It may be best to regard the *p*-values reported by *bms_runner* as an alternative kind of score, without any probabilistic interpretation. Compared to *LR*, it may occasionally be convenient that *p* has clear upper limits at both ends of its range, i.e. $0 \leq p \leq 1$.

## Searching for functional associations

Usage:

bms_runner -search -i <pairs_file> -gains <gains_rate>|unrestricted -tree <tree_file> -species <spp_file> -patterns <patterns_file> [ -exe <executable_file> ]
[ -tmpdir <dir> ]

E.g.

```
bms_runner -search -i pairs_of_interest.txt \
-gains 0.4 -tree treefile.nex -species spp.txt \
-patterns distributions.txt > searching_res.txt
```

or

```
bms_runner -search -i pairs_of_interest.txt \
-gains unrestricted -tree treefile.nex -species spp.txt \
-patterns distributions.txt > searching_res.txt
```

The training step (above) gives a table of different rates of gain, and the associated specificity and sensitivity at each of those at various score cut-offs, where score is the likelihood ratio statistic, *LR* (see above). The user should then chose a given rate of gain (*r* in the 'ML-constrained' method of Barker et al. 2007), or `unrestricted` if no constraint on the rate of gain is desired (the 'ML-unconstrained' method), and use this in a search for functional associations among genes. For most purposes, the rate of gain and *LR* cut-off should be chosen to give high specificity in combination with reasonable sensitivity.

<pairs_file> gives pairs of genes between which the user wishes to seek functional

associations. Its format is the same as for <pos_file> and <neg_file>, above. However, in <pairs_file>, there may be novel pairings of unknown functional significance (as well as, or instead of, known functionally linked and non-functionally linked pairs). *bms_runner* will use *BayesTraits*, and a user-specified rate of gain of genes, to statistically detect functional linkage in each of the pairs in <pairs_file>.

The format of <tree_file>, <species_in_order_file> and <patterns_file> is as for the training step, above.

<executable_file> and <dir> are optional, as for the training step, above.

Output of the search is a tab-delimited text file giving output from *BayesTraits*, including rate parameters and log likelihoods of the independent and dependent models and *LR* and the *p*-value for each pair. This has the same format as the output file from training (if one was stored using the `-lrfile` option), except now there is no `test_set` column. Note that the *p*-value is obtained from *LR* using an assumption which is unlikely to be correct (see above).

All comparisons and results are reported, whether or not they are significant. Where *LR* equals or exceeds the cut-off chosen after the training step, this may be interpreted as a predicted functional link between the genes. Where *LR* is lower than the cut-off, there is no prediction of functional linkage. If the training data were representative of the data being searched for functional links, these predictions will have approximately the specificity and sensitivity suggested by the training step, for the rate of gains used and cut-off applied. (However, it may generally be safer to assess sensitivity and specificity by searching for functional links among known data that were *not* used for training.)

For details of rare situations that may cause difficulty in interpreting results, please see 'Training', above. In all such cases *bms_runner* conservatively outputs an *LR* of 0, and the corresponding *p*-value of 1.

## Quirks and pitfalls

*bms_runner* avoids redundant computation by reducing pairs of cross-species distribution patterns to a non-redundant set. It then duplicates these results as necessary to give results for the full, potentially redundant set of pairs in the input. Consequently, results for pairs of genes or proteins may appear in the results file (lines in the file giving *LR* values) in different sequence to the input.

Following one established approach in bioinformatics (Baldi and Brunak 2001; von Mering et al. 2003; Barker et al. 2007), *bms_runner* defines and sensitivity and specificity as:

sensitivity = count(true positives)/[count(true positives) + count(false negatives)],

specificity = count(true positives)/[count(true positives) + count(false positives)].

Confusingly, an alternative definition of specificity is also in use, different from the above (e.g. Sokal and Rohlf 1995; cf. Baldi and Brunak 2001:162–163). To evaluate performance, rather than just use the table summarizing training output by *bms_runner*, it is often helpful to keep the full *BayesTraits* results, via the `-lrfile` option of *bms_runner*, and post-process these using a separate statistics package or script. This allows the user to apply whichever performance measures and terminology are preferred.

*bms_runner* will report significant *LR* values, irrespective of whether the correlation between cross-species distribution patterns is positive or negative. For prediction of functional linkage between genes products, positive correlations are of most interest, since without both gene products being present in the same organism they cannot be functionally linked. However, negatively correlated patterns are of interest for predicting analogues (Morett et al. 2003). Currently, *bms_runner* users have to make their own decision as to how to deal with negatively correlated patterns and this may involve some pre- and/or post-processing of data. Possibilities include omitting such pairs as unlikely to indicate functional linkage (Barker & Pagel 2005), or including them, without any special treatment, if they are rare at high values of *LR* (Barker et al. 2007). If aiming to make functional predictions on the basis of analogy, it may make sense to omit *positively* correlated pairs.

In the case of a comparison between two patterns which consist only of '1', i.e. both represent a gene universally present across the species in the study, *BayesTraits* will not be run and most columns in the output file (giving *LR* values) will be empty. Similarly, *BayesTraits* will not be run in the case of a comparison between two patterns which only consist of '0'. In such cases, an *LR* of 0 the corresponding *p*-value of 1 will be output to the file specified with `-lrfile`. (Whether or not this option is used, sensitivity and specificity will be assessed using these *LR*s.) Care must be taken to correctly interpret the empty columns if post-processing the output file.

Due to difficulties estimating model parameters accurately, likelihoods output by *BayesTraits* might very occasionally suggest a slightly negative *LR*. 'Negative' *LR* is interpreted and reported as 0 by *bms_runner*. On other extremely rare occasions, fitting a model may fail entirely, and this is indicated by *BayesTraits* reporting a log likelihood of –999999 for the model in question (independent or dependent). Again, in such cases *bms_runer* reports an *LR* of 0. Any 'log likelihoods' of –999999 must not be taken literally if, for example, calculating the mean likelihood across a study.

## Example files

To illustrate use of *bms_runner*, example files are provided in the `example_files` directory. The tree and cross-species distribution patterns are from the larger analysis of Barker et al. (2007). But these example files are not intended to draw biological conclusions. They are just intended to show the correct format of the input files, and to see *bms_runner* in action on your system. For a larger amount of training data, see the supplementary data of Barker et al. (2007).

The example files are:

| | |
|---|---|
| `pos_pairs.txt` | 'known positive' pairs, i.e. pairs of IDs of functionally linked gene products; |
| `neg_pairs.txt` | 'known negative' pairs, i.e. pairs of IDs of gene products unlikely to be functionally linked; |
| `pairs_of_interest.txt` | pairs of IDs of gene products among which we wish to seek functional links; |
| `species_in_order.txt` | list of species in the order in which they appear in cross-species distribution patterns in `traitmat.txt`, below; |
| `traitmat.txt` | cross-species distribution patterns of genes in `pos_pairs.txt`, `neg_pairs.txt` and `pairs_of_interest.txt`; and |
| `tree_rooted.nex` | Nexus-format rooted phylogeny of species, with branch lengths. |

To use these files in a training run, you could place these files, `bms_runner` and `BayesTraits` in the current directory. Then type the following command at the Linux or Mac OS X command prompt:

```
./bms_runner -train -positive pos_pairs.txt \
-negative neg_pairs.txt -tree tree_rooted.nex \
-species species_in_order.txt -patterns traitmat.txt \
-exe ./BayesTraits -lrfile training_lrs.txt \
> training_results.txt
```

The backslash characters ('\') just tell the system to carry on reading this command from the next line. To avoid typing the command from scratch, you could copy the above three lines from this document and paste them in at the Linux or Mac OS X command prompt. Press RETURN at the end of the command (i.e. after `training_results.txt`).

After a few minutes, the command should complete (indicated by the return of the Linux or Mac OS X command prompt) and you will have two output files:

`training_results.txt` – tab-delimited text giving rate-of-gain parameter (*r* of Barker et al. 2007) or else `unrestricted`, LR statistic cut-off, *p*-value cut-off, number of predictions of functional linkage, specificity and sensitivity.

`training_lrs.txt` – LR statistics and *p*-values of the comparisons performed to obtain the summary in `training_results.txt`.

By looking at `training_results.txt`, you should now select a value of the gains parameter *r* and an LR statistic cut-off that give acceptable sensitivity and specificity. This small example is not sufficient to choose *r* unambiguously. However, on the basis of `training_results.txt` it seems that $r = 6$ and a LR statistic cut-off of around 15 might be appropriate. In a 'real' study, if it seemed like the

highest value of *r* tried might be the best, it would be a good idea to re-run training with an even higher value to check if that is even better. Similarly, if the smallest fixed value of *r* seems best, it would be sensible to re-run the training step with an even smaller value. However, negative or zero values should not be used.

We will skip re-running the training step, and go on to predict functional links among the pairs of genes listed in `pairs_of_interest.txt`. This can be done with the following command:

```
./bms_runner -search -gains 6 -i pairs_of_interest.txt \
-tree tree_rooted.nex -species species_in_order.txt \
-patterns traitmat.txt -exe ./BayesTraits \
> search_results.txt
```

In this example, the cross-species distribution patterns for `pairs_of_interest.txt` are given in the same file as the patterns for `pos_pairs.txt` and `neg_pairs.txt`. It is not necessary to use the same file in training and searching. Separate files can be used so long as they contain all the cross-species distribution patterns required for the step in question. Additional distribution patterns are allowed and will be ignored.

After a few seconds the command will complete. Wherever a pair of genes has an LR statistic (`lr_stat`) of at least the cut-off we decided upon above, we would predict a functional link. None of the three pairs in `pairs_of_interest.txt` meets this criterion so we would conclude that none of these pairs represents a functional link.

The value of *r* (the rate of gain in the ML-constrained model; or 'unrestricted' if the ML-unconstrained model is more appropriate to the study), and the appropriate cut-off, will vary from study to study due to the nature of the input, and also due to the requirements of the user. It is important to perform the training step first, with your phylogeny and cross-species distribution patterns, and interpret the results of the training step according to your aim. For initial predictions to be confirmed by other methods, it may be acceptable to sacrifice some specificity in the interests of greater sensitivity, but if the predictions are to be treated as a final result, this may not be acceptable.

## References


Baldi, P, and Brunak, S (2001) *Bioinformatics: The Machine Learning Approach*, 2nd edition. Cambridge (MA): MIT Press.

Barker, D, Meade, A and Pagel, M (2007) Constrained models of evolution lead to improved prediction of functional linkage from correlated gain and loss of genes. *Bioinformatics* **23**: 14–20.

Barker, D and Pagel, M (2005) Predicting functional gene links from phylogenetic-statistical analyses of whole genomes. *PLoS Comput Biol* **1**: e3.

von Mering, C, Zdobnov, EM, Tsoka, S, Ciccarelli, FD, Pereira-Leal, JB, Ouzounis, CA and Bork, P (2003) Genome evolution reveals biochemical networks and

functional modules. *Proc Natl Acad Sci U S A* **100**: 15428–15433.
Morett, E, Korbel, O, Rajan, E, Saab-Rincon, G, Olvera, L, Olvera, M, Schmidt, S, Snel, B and Bork, P (2003) Systematic discovery of analogous enzymes in thiamin biosynthesis. *Nat Biotechnol* **21**: 790–795.
Pagel, M (1994) Detecting correlated evolution on phylogenies: A general method for the comparative analysis of discrete characters. *Proc R Soc Lond B Biol Sci* **255**: 37–45.
Pagel, M (1997) Inferring evolutionary processes from phylogenies. *Zool Scr* **26**: 331–348.
Pagel, M, Meade, A and Barker, D (2004) Bayesian estimation of ancestral character states on phylogenies. *Syst Biol* **53**: 673–684.
Pellegrini, M, Marcotte, EM, Thompson, MJ, Eisenberg, D and Yeates, TO (1999) Assigning protein functions by comparative genome analysis: Protein phylogenetic profiles. *Proc Natl Acad Sci U S A* **96**: 4285–4288.
Sokal, RR and Rohlf, FJ (1995) *Biometry*, 3rd edition. New York: W. H. Freeman and Company.