A USER'S GUIDE TO THE PHYLOGENETIC REGRESSION PROGRAM, PHYLO.GLM, VERSION 1.03

by Alan Grafen

Table of Contents

1 INTRODUCTION

The program implements the phylogenetic regression (A. Grafen, 1989 Philosophical Transactions of the Royal Society B, **326**:119-157 - henceforth "the source paper"), a statistical technique which allows the hypothesis testing facilities of general linear models to be applied validly to cross-species data.

The user will need a nodding acquaintance with GLIM3, a statistical computer language, in which the program is written. GLIM exists on most computers, both mainframe and personal, and is available from NAG (Numerical Algorithms Group - NAG Ltd, Wilkinson House, Jordan Hill Road, OXFORD, United Kingdom OX2 8DR. Telephone number: Oxford (0865) 511245. The North American office is NAG Inc., 1400 Opus Place Suite 200, Downers Grove, Illinois 61505-5702, USA. Telephone number: 708-971-2337 (FAX 2706)). Two recent good books about GLIM are: M.J.R. Healy, *GLIM: an introduction* (Clarendon Press, 1988), and M. Aitkin, D. Anderson, B. Francis and J. Hinde, *Statistical modelling in GLIM* (Oxford University Press, 1989). The version of GLIM *must be at least 3.77* (in December 1989 the most recent version) - for syntactical reasons the program could not have been written in earlier versions, not even 3.12.

The phylogenetic regression is a small part of my work, and I cannot promise to help users of this program with any urgency or in any detail. I have therefore made this documentation as simple and complete as possible. There is an illustration of *nearly every feature* in the example sessions at the end of this documentation. If any actual errors are discovered in the program, then of course I will try to fix these as soon as possible. Otherwise, I do not plan any further versions of this program. Writing software is not my job, and I supply this program only because inventing an unimplemented method is intellectually unsatisfying. If anyone else wants to write a better program, I shall have no objections.

The program is written in GLIM. The syntax and control structures of GLIM mean the program cannot provide a fully "user-friendly" service, and this raises the question why *is* the program written in GLIM. The program developed along with the method itself, and many of the results about the phylogenetic regression were checked in numerical examples. The simulations reported in the source paper were also run using the program. For none of these purposes was user-friendliness important. While a program designed from the start for general use would not be written in GLIM, it was feasible to add as much user-friendliness as possible to the existing program. I would never have undertaken the much larger task of starting from scratch. The best way to implement the phylogenetic regression would be as an option in one of the large packages: SAS, SPSS or BMDP.

I would be grateful if any papers using analyses produced by this program acknowledged that fact. The possibility exists for users to tinker with the program - where this has occurred I would like this fact too to be mentioned in the paper. On the subject of acknowledgments, I am grateful to various people who have tried out earlier versions of this program, and made suggestions for improvement. Paul Harvey has been a keen user from an early stage. Jeremy John, Graham Stone and Daniel Promislow also made useful suggestions. William Kirk has pointed out bugs and helpfully requested extra facilities.

2 HOW TO USE THIS DOCUMENTATION

Use of the program is first explained fully in the logical order of steps that need to be taken. The preliminary steps involve preparing data and a phylogeny, and these will not involve GLIM, but some text editor. Once the stage of using GLIM is reached, it will probably be helpful to work through the explanation simultaneously with one of the example sessions that are given at the end of the documentation. Then you will see a concrete example of the GLIM commands and replies, and this should make the general explanation easier to understand.

Further information, for more advanced use, can be found in the file "Technical details" supplied on the same disk as the manual. This file should rarely be needed. It begins with an account of its contents, so look there if you feel you need information not supplied in this manual.

3 HOW TO USE THE PROGRAM

3.1 What you need to have before using the program

Before using the program you will need to have two things. First the dataset, with the variables you will want to use. There should be one line for each species. The species can be in any order. The files containing the data must be text files (ASCII files) and all the data must be in numeric form. Numbers must be separated by spaces or carriage returns, not tabs. The default input format in GLIM is "FREE", and this means that no alignment of columns is necessary. GLIM does not handle characters. In particular, this means that missing values must be coded with a numerical value. It will be most convenient if the same numerical value codes for "missing" in all the variables to be used in the analyses.

The second thing you will need is a representation of the working phylogeny. There are two ways the program allows you to specify the working phylogeny: the single vector method and the taxonomic levels method.

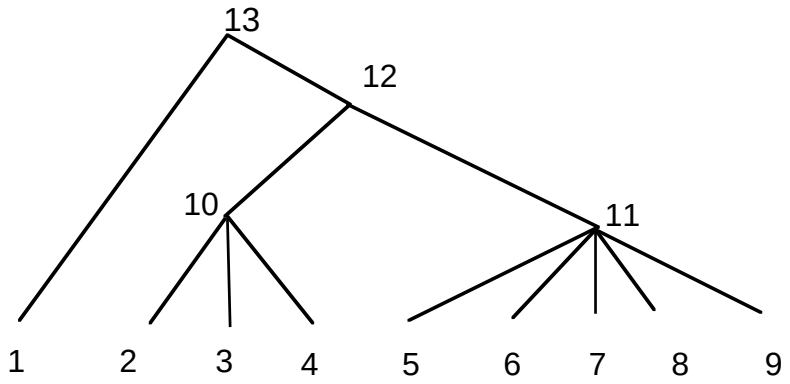
The taxonomic levels method. This method uses variables to represent the taxa to which each species belongs. So there might be one variable for genus, one for family and one for order. The taxa must be coded by positive integers (i.e. 1, 2, 3 …). The integers representing the different taxa at one level must be unique within the higher level taxon to which they belong. So for example, the three genera within a family must be coded by different integers. The four genera in the next family can (but need not) re-use those integers. It will frequently occur that a species is the only species in its genus, or a family is the only family in its superfamily - this causes no problems. There can be up to eighty-one vectors carrying the taxonomic levels. This will be the easier method when the conventional taxonomy is being used as the working phylogeny.

The following mini-example shows an order with two families, three genera and nine species. One family is monospecific, as perforce is the genus it contains. The other family has two genera, with three and five species in.

FA	GE	SP
1	1	1
2	1	2
2	1	3
2	1	4
2	2	5
2	2	6
2	2	7
2	2	8
2	2	9

The third column is used only to make the example easier to understand - it is not required as one of the taxonomic vectors.

The single vector method. When the working phylogeny derives from a genuine attempt at a phylogeny, you will most likely not already have variables coding the phyletic units to which each species belongs. Here it may be more convenient to use the single vector method. To implement this method, start with a drawing of the working phylogeny. The first task is to number each node using consecutive integers. Each species node should be numbered according to its position in the dataset. The higher nodes should be numbered in accordance with the principle that the number of each node must be lower than the number of its parent node. The root of the tree will therefore have the highest number. There are many ways of numbering the nodes compatibly with these principles, and all are equally good. In the example just given for the taxonomic vectors method, the drawing would look like this:



The second task is to write down a list of numbers which define the phylogeny. It is most convenient to write two lists in adjacent columns. The left hand column contains the integers 1, 2, and so on up to one less than the number of the root. The right hand column contains the number of the parent node of the node denoted on the left. The required vector is the right hand column, which now contains a full description of the working phylogeny. This method can represent any working phylogeny. For the example, the phylogeny drawn above looks like

Node	Parent	Comment
1	13	This species is connected directly to the root
2	10	These three species
3	10	belong to a genus
4	10	with node number 10
5	11	These five
6	11	species belong
7	11	to a genus
8	11	with node
9	11	number 11
10	12	This genus (“10”) belongs to node 12
11	12	So does this genus (“11”)
12	13	This node, containing genera 10 and 11, belongs to the root
		The root has no parent, and so has no entry here

I know from experience that this process is quite error-prone, even if in this mini-example it seems beguilingly simple. The result should be checked carefully.

With the data and the phylogeny in suitable datafiles, it is possible to use the program.

3.2 The initial setting up of the analysis

If you do not know how to get into GLIM, ask somebody. Once inside GLIM the first thing to do is to read in the program file, whose name is PHYLO.GLM. This can be done as follows:

```
$INP 31 $  ! Any number between 11 and 99 will do
```

and the computer will prompt you for the name of the file with

```
Filename?
```

and you complete the line by typing the name:

```
Filename?    PHYLO.GLM
```

The computer may then spend a little while reading in the program. The next thing to do is to read in your data.

3.2.1 Data

The first step is to choose names for each of your variables. The only complication here is to avoid *name conflicts*. If you use the same name as I have already used, or use during the program, then problems arise. All of the names I use end with an underscore “_”. All you have to do is not end any of your names with an underscore, and all will be well. This principle applies to all vector and macro names you have to choose for whatever purpose during your GLIM session. *Name conflicts* can arise in scalars too. The program uses only the special scalars %z1 through %z9, that are conventionally reserved for use inside macros. You are free to use the ordinary scalars %a to %z. If none of this stuff about name conflicts makes sense to you, or you don't even know what a scalar is, don't worry - it's not you who would ever have chosen a clashing name. It's only keen GLIMmers who might have.

You should note that GLIM recognizes only the first four letters of a name, and so to avoid name conflicts with yourself, ensure that none of your names share the same first four letters. The wise policy is to use no more than four letters in a name anyway.

Suppose, then, that the dataset has 127 species in it, that your variable names in order of appearance in the file are MASS for body mass, BR for brain size, HAB for habitat type, and BMR for basal metabolic rate, and that the file is called BORING.DAT. Then the following instructions will read in the data:

```
$UNITS 127 $     !To tell GLIM how many datapoints

$DATA MASS BR HAB BMR $   ! Variables in order

$DINP 11 $       !Ask GLIM to read the data
```

The point of the 11 is that GLIM needs a number here, greater than ten and less than a hundred, which you must use in case you want to use the same file later in the session. It must be a different number from the one you used in connection with the program (in the example above this was 31). GLIM will prompt you for the name of the file:

```
Filename?
```

and you reply by typing on the same line so it looks like:

```
Filename?   BORING.DAT
```

GLIM then reads from the file and, if all goes well, your four variables will now have values. You can check by doing

```
$LOOK MASS BR HAB BMR $
```

and you'll see what values GLIM thinks they have.

The main problem that arises on input is having characters in the file - usually tabs. Remove them with an editor, replacing them with spaces, and try again. Another common problem is having a datafile that is more than 80 characters wide. GLIM initially expects input channels to have a width of 80 or less. If you want, you can specify a larger width (up to 132) by giving it after the channel number when you $DINP, thus in the example above you would have said

```
$DINP 11 132 $
```

If you have a lot of data, you can read in multiple files. Suppose you have variables A1 to A10 in file ADATA, and variables B1 to B10 in BDATA. Then just do

```
$DATA A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 $

$DINP 11 $

Filename?   ADATA

$DATA B1 B2 B3 B4 B5 B6 B7 B8 B9 B10 $

$DINP 12 $   ! You must choose a DIFFERENT number

Filename?   BDATA
```

You should now have your data all present and correct.

3.2.2 Phylogeny

The second step is to create your phylogeny, and the way you do this depends on the method you have chosen.

Taxonomic levels method. With this method, you should have read in the taxonomic level vectors with your data. If you have nine or fewer taxonomic level vectors, now define a macro called TX_ which contains the names of the vectors containing the taxonomic levels in order, with the highest level vector first, and the lowest level last. For example, if the vectors are called OR, SPFA, FA, and GE , (for order, superfamily, family and genus), then define TX_ by

```
$MACRO TX_ OR SPFA FA GE $ENDMAC
```

Then do

```
$USE MPH_ $
```

Your phylogeny has now been created and recorded by the program. As a side effect, your taxonomic vectors will have been altered so that within each vector, each taxon has a unique integer (not just unique within the next higher level taxon).

If you have more than nine levels, then the method is very similar. You must define a macro TAX_ to contain the names of up to nine macros. Each of these macros can contain the name of up to nine taxonomic level vectors. The vectors should again be in order, so that the first vector of the first macro is the highest level, and the last vector of the last macro is the lowest. For example, if your levels are OR, INOR, PVOR, SPFA, FA, SBFA, TR, SBTR, GE and SBGE, (for order, infra-order, parv-order, superfamily, family, subfamily, tribe, subtribe, genus and subgenus) then the following definitions would work:

```
$MACRO TX1 OR INOR PVOR SPFA FA $ENDMAC

$MACRO TX2 SBFA TR SBTR GE SBGE $ENDMAC

$MACRO TAX_ TX1 TX2 $ENDMAC
```

Then, as in the simpler case, do

```
$USE MPH_ $
```

and your phylogeny will have been created and recorded by the program. As in the case with few taxonomic vectors, they will have been altered so that within each vector, each taxon has a unique integer (not just unique within the next higher level taxon).

The program's output contains information about individual nodes in the phylogenetic tree, which it refers to by number. The number of a species is just its sequence number in the dataset. To find the numbers of higher nodes, do

```
$USE WHO_ $
```

This gives a list of the higher node numbers. For each higher node it gives one species that is included in the node, and one species that is “just” not included. For the root of the tree, which includes all species, the unincluded species is given as number zero. You may find it convenient to enter the numbers on a drawing of the phylogeny. Another way to check the phylogeny is to do

```
$LOOK PHY_ $
```

PHY_ is the internal representation of the phylogeny, whose structure is explained in the "Single vector method" subsection of §3.1 above.

Single vector method. The second column of your phylogeny datafile contains the phylogeny in the form the program requires it. Accordingly all you need to do is to read it into the variable called PHY_ . If you used the method recommended earlier, you can discard the first column by defining your data list thus:

```
$DATA PHY_ PHY_ $
```

With this method, you will already have a drawing of the phylogeny with its higher node numbers, but it is still recommended to

```
$USE WHO_ $
```

as a way of checking that you have entered the phylogeny correctly. It is a fiddly business, and worth checking before doing analyses.

3.2.3 Automatic missing values

The program provides a way of omitting species because they have missing values. You must choose one numerical value which is to represent "missing" in every variable, including the y-variable. The program will automatically exclude from an analysis any species which has that numerical value in any of the variables included in that analysis. You specify that value by setting OPT_(1) as follows:

```
$CA OPT_(1)=-100 $
```

where I have assumed -100 is the special value. The special value must be different from zero. The program interprets OPT_(1)=0, which is the default, as an instruction to omit its check on missing values. Because categorical variables may be missing, and the special missing value needs to be put into each of the dummies, it is essential to set OPT_(1) before creating categorical variables or interactions involving categorical variables. If you change OPT_(1) during a session, all categorical variables and interactions involving categorical variables must be redefined before being used in analyses.

The safest and simplest way is therefore i) to set OPT_(1) to the special value as soon as you have read in PHYLO.GLM, and ii) to read in the data with the missing values already converted to the special value.

3.2.4 Categorical variables

If you have categorical x-variables, then you need to convert them into a form the program can use. Macros are provided to help you do this. Suppose you have a factor called G. (It need not be defined in GLIM as a factor - but it should code categories using integers starting with one.) You must choose a name for a macro to contain the design variables for G, say you choose F. Then define F by

```
$MACRO F G2 G3 G4 G5 $ENDMAC
```

where I have assume there are 5 levels to G. There should always be one less design vector than there are levels. It is essential that the top level is actually used (G=5 in the example), but not that lower levels are actually used (so there need be no species with G=3). The reason is that the program counts how many vectors to fill by the highest (non-missing) value in the factor The names of the design vectors are arbitrary: they are chosen here to index the design variables by the corresponding value of G. To create the design vectors, do

```
$USE EXF_ G F $
```

putting the factor first, and macro containing the design variables second. If OPT_(1) is non-zero, then its value will be taken as missing, and any missing value in a factor will be transferred to each of the dummy variables. This method works for variables with ten or fewer levels. With more levels, there is a way of "cheating" using the factor by factor interaction macro IFF_, described below.

3.2.5 Interactions

In order to use interactions, it is necessary first to create variables to represent them. There are three cases:

i) interactions between two continuous variables. The new variable is created simply by multiplying together the two base variables. For example, if A and B are continuous variables, then you can create a variable AB to represent the interaction as follows:

```
$CALC AB=A*B $
```

ii) interactions between a continuous variable and a factor. The variable is created in a way similar to the creation of a macro to represent a categorical variable. Create a macro with as many variable names as one less than the number of levels in the categorical variable. If a macro G has five levels, then you might define a macro FC by

```
$MACRO FC GC1 GC2 GC3 GC4 $ENDMAC
```

Then you would make FC contain the variables representing the interaction between G and a continuous variable C as follows:

```
$USE IFC_ G C FC $
```

This method works for factors with ten or fewer levels. It is essential that the top level of G is actually used - in this example, that there is at least one species with G=5.

iii) interactions between factors. If the variables have m and n levels, then a total of (m-1)×(n-1) vectors will be needed to represent the interaction. Place this many vectors, in groups of nine or less, into a series of macros. Then place those macro names (nine or fewer of them) into yet another macro. For example, if G has five levels, and H has four levels, then the following definitions would be appropriate:

```
$MACRO K1 L1 L2 L3 L4 L5 L6 $ENDMAC

$MACRO K2 L7 L8 L9  L10 L11 L12 $ENDMAC

$MACRO M K1 K2 $ENDMAC
```

L1 to L12 are the vector names. K1 and K2 are the intermediate macro names, and M is the top level macro representing the interaction. Once this is done, you can create the interaction thus:

```
$USE IFF_ G H M $
```

It is essential that the top level of each factor is actually used - in this example, that there is at least one species with G=5, and at least one species with H=4.

You can use IFF_ to create the dummy variable for a single factor with more than ten levels. To do this, construct a fictional factor each of whose values is 2. If FBIG has fifteen levels, then the following instructions would create the dummy variables:

```
$CALC FICF=2 $

$MAC FB1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 $ENDMAC

$MAC FB2 Q9 Q10 Q11 Q12 Q13 Q14 Q15 $ENDMAC

$MAC FB FB1 FB2 $ENDMAC

$USE IFF_ FBIG FICF FB $
```

How to use interactions of all kinds is described below.

3.2.6 Path segment length rules

As explained in the source paper, the variance-covariance structure of the data is described by assigning a length to each path segment in the phylogenetic tree. The program provides three ways to assign those lengths: the default "Figure 2" method (the name refers to Figure 2 of the source paper), the taxonomic levels method and the fully general method, which will be described in turn. The method to be used is determined by the first (and only) argument to the macro GO_. When unspecified, GLIM assumes that the argument to a macro is the same as last time. GO_'s first argument is pre-set to produce the Figure 2 method. The method can be changed from one analysis to the next.

Figure 2 method. This is the default method. If there are *n* species altogether, and a node has exactly *i* species below it in the phylogeny, then the height of that node is *(i-1)/(n-1)*. Species therefore have height zero and the root has height one. If no action is taken, this method will be used. To switch back after using another method, do

```
$ARG GO_ %A $
```

%A is not altered by this - so it is safe to use it yourself if you want. (All that is important is that the length of the first argument equals one.)

Taxonomic levels method. This method is available only if the phylogeny was created by the taxonomic vectors method. If there were *v* taxonomic vectors, then there are *v+2* taxonomic levels. The extra two are the root, at the top of the tree, and the species level at the bottom. The method allocates the species level a height of zero, and so there are *v+1* remaining heights to specify. You do this by placing *v+1* values in a vector. The values should be positive, as they must be strictly above the species height of zero, and they should be increasing, as the heights are read from the lowest level to the highest level. Multiplying the heights by a positive constant has no consequence for the analysis. Suppose there were 15 taxonomic vectors. Then there are 16 heights to specify, and this could be done as follows

```
$ASS Z= 1.1,2.4,3,5,7,9,14,18,26,30,32,35,41,46,51,53 $
```

The name of the vector need not be Z. All that matters is that you specify it as the first argument to GO_ before the analysis, by

```
$ARG GO_ Z $
```

The program knows to use the taxonomic levels method by the length of this vector. If you get it wrong, by specifying too many or too few levels, the default "Figure 2" method will be used instead. (So it may useful to ask for confirmation of the node height method used by setting OPT_(2) - see below.) The method stays in force through subsequent analyses until you explicitly change the argument of GO_. You can alter the values of the vector in between analyses, and this change will affect the program. This would be useful if you wanted to check what difference was made by assuming different sets of node heights.

Fully general method. This method allows you to specify a different height for every single node. To do this you must supply a vector that contains in its *i*th element the height of the *i*th node. If you have used the single vector method of entering the phylogeny, then you will already know the names of the higher nodes. Otherwise you should

```
$USE WHO_ $
```

to tell you the names. If there are *n* nodes altogether, then your vector must of length *n*. (This will be one element longer than the length of PHY_, as PHY_ has entries only for nodes with a parent node i.e. PHY_ omits the root.) Having defined your vector, presumably having prepared it in a file beforehand, you choose the fully general method by setting the first argument of GO_ to this vector. If the vector is NH, then

```
$ARG GO_ NH $
```

will set the node height method to "fully general" until it is explicitly changed again. It is by checking the length of the vector that the program knows to use the fully general method. (So it may useful to ask for confirmation of the node height method used by setting OPT_(2) - see below.) Changing the content of the vector between analyses will change the heights used in the analysis. So if the root is node number 207, and its initial height is 35.8, you can find the consequences of increasing the height of the root node to 47.2 by

```
$CALC NH(207)=47.2 $
```

and performing another analysis.

The species need not have the same height. Parent nodes should of course be higher than their daughters. Use of this method allows study of the consequences of assuming different error rates in different path segments. Using real dates in years would investigate the assumption of constant divergence. An increase in the rate of divergence within a taxon could be represented by assigning the heights of nodes according to "year equivalents".

The heights in this method should not be negative, as the power transformation used in fitting ρ makes sense only for positive and zero heights. Multiplying the heights by a positive constant has no consequence for the analysis, as the program internally scales the heights so that the root has a height of one. Adding a constant to the heights *does* matter. The power transformation used in fitting ρ will produce a different family of sets of path segment lengths if a constant is added to all the heights, even though the member with $\rho=1$ will be common to them all.

You should now have the data and the phylogeny known to GLIM, have prepared your factors and interactions, chosen your path segment length method and be ready to test hypotheses.

3.3 Preparing to test hypotheses

Before testing a hypothesis, you need to specify various logical parts of the hypothesis. These are listed below under the GLIM identifiers which contain the necessary information.

`YV_` This macro must contain the name of the y-variable.

`C_` This macro must contain the names of the x-variables being controlled for, separated by spaces. `C_` can contain up to 9 variables. If more are required, or if factors, or interactions involving factors, are being controlled for, then see below. To control for nothing, you must still define this macro - just put nothing in it but a space.

`T_` This macro must contain the names of the x-variables being tested for addition to the model, separated by spaces. `T_` can contain up to 9 variables. If more are required, or if factors, or interactions involving factors, are being tested, then see below.

`SPI_` It should be of the same length as your data vectors. A species should have a 1 if you wish to include it in the analysis, and a 0 if you wish to exclude it. Note that missing values can be handled automatically.

For example, to test for the effects of `DIET` and `HAB` on `BRSZ`, controlling for `BDSZ`, and including all species, you would do

```
$MAC YV_ BRSZ $ENDMAC

$MAC C_ BDSZ $ENDMAC

$MAC T_ DIET HAB $ENDMAC

$CA SPI_=1 $
```

If instead you wanted to include only those species for which `SMPS` was greater than 50, and for which `WMBD` was 1, you would instead specify

```
$CA SPI_=(SMPS>50)&(WMBD==1) $ !The double == is essential
```

(“`=`” means “is assigned the value”, and “`==`” means “equals”).

These four elements are needed in every analysis. You will have to define them before the first analysis. After that they will retain their values into subsequent analyses until you change them. If you have a lot of variables to control or test for, then you may have to use `CON_` and `TST_` in place of `C_` and `T_` . `CON_` is a macro which contains the names of macros which contain the names of x-variables to be controlled for. By default, `CON_` contains only `C_` . But if you wanted more than 9 x-variables controlled for, then you have to divide them between macros in groups of 9 or less. Suppose you called them `C1`, `C2`, and `C3`. Then you need to define `CON_` by

```
$MACRO CON_ C1 C2 C3 $ENDMAC
```

(Notice all my names end in an underscore, and none of yours do. You can't choose `C1_`, `C2_` and `C3_` in case I have already used those names for something else in the program. You could have chosen `C_` for one of them.)

Another reason you may wish to use `CON_` and `TST_` is if you have used factors. A factor name is the name of a macro containing its design variables. Thus

if C_ contains the ordinary variables you wish to control for, and you wish in addition to control for the factor F, then define CON_ by

```
$MAC CON_ C_ F $ENDMAC
```

The same applies if you are using an interaction between a categorical variable and a continuous variable.

When using an interaction between two categorical variables (or a categorical variable whose design vectors were created using IFF_), the situation is slightly different. If M is the higher level macro representing the interaction, then it would be added to CON_ as #M. If M is the interaction between G and H, whose macros are GF and HF, then controlling for GF and HF and their interaction would be achieved by

```
$MACRO CON_ GF HF #M $ENDMAC
```

If you use CON_ and TST_, it is important to understand the way the program uses them. The program actually never looks at C_ directly. It always looks at CON_, but CON_ is set up in the first place so that it contains just C_. In my program there is the macro definition:

```
$MAC CON_ C_ $ENDMAC
```

and what you are doing when you use CON_ is over-riding my definition. It follows that if you have used CON_ for some analyses, but then want to return to the simpler use of C_, then all you have to do is to repeat this definition, and the original state will be regained. What the program does when it looks at CON_ is to look in CON_ for names of macros, and then to look in each of those macros for names of vectors to use in the model. Placing a vector name directly in CON_ will cause a failure, as will placing a macro name in C_. This explains why the categorical variables created by EXF_ and IFC_ must be placed in CON_ - they are the names of macros that contain vectors. The categorical variables created by IFF_ are one level up - they are macros that contain macros that contain vectors. So they must be placed in CON_, but with a # in front of them. The # causes them to be replaced by their contents whenever CON_ is used. So that the program thinks that CON_ contains the list of macros that contain vectors, rather than the single macro that contains the names of macros that contain vectors. This information may be useful to more advanced users.

The same remarks apply *mutatis mutandis* to TST_ and T_.

3.4 Testing hypotheses

Once YV_, C_ (or CON_), T_ (or TST_) and SPI_ have been set up, you perform the phylogenetic regression by typing

```
$USE GO_ $
```

The program will then spend a little while thinking before giving you the answers. To test another hypothesis, you need only redefine whichever of YV_, C_ (or CON_), T_ (or TST_) and SPI_ you wish to change. The output of the program is controlled by macros, as described in the next section.

To do three analyses with the same data, but using the Figure 2 method, the taxonomic levels method with level heights Z, and the fully general method with node heights NH, you could do:

```
$USE GO_ %A $

$USE GO_ Z $
```

```
$USE GO_ NH $
```

The argument given remains in force until explicitly changed, so now giving

```
$USE GO_ $
```

would result in the fully general method being used with NH as the node heights, even if the y-variable, or the the control or test variables, or SPI_, had been changed in between. Specifying the argument as part of the $USE statement is a usually preferable alternative to the equivalent use of $ARG to specify the arguments and then $USE to invoke the macro.

4 THE OUTPUT OF THE PROGRAM

Each time you $USE GO_ the program will provide you with information about the phylogenetic regression of YV_ on TST_, controlling for CON_, using the species indicated by SPI_. If OPT_(1) is non-zero, then species will be excluded if any of the variables in the analysis take the same value as OPT_(1).

Most of the output of the program is under your control. The values of OPT_(2) to OPT_(14) determine which output you receive (although in most cases GLIM prepares the output anyway, and just throws it away if you don't want it). The values of OPT_(20) to OPT_(24) allow you to interrupt the program at specified places, and so obtain more information. If the value of an element of OPT_ is one, you receive the corresponding output item (or interruption), if it is zero you do not. The user is quite free to alter the settings from their default values. Here are the default values of OPT_ and the information supplied by each element.

OPT_ element	Default	Information or interruption supplied
OPT_(2)	0	The node height method used in the analysis.
OPT_(3)	1	The total number of species, the number omitted through use of SPI_, and the number omitted because of missing values.
OPT_(4)	0	The optimal value of rho, and the maximized value of the log-likelihood, in the long regression on the control variables only.
OPT_(5)	0	The parameter estimates and deviance of the long regression on the control variables only.
OPT_(6)	1	The parameter estimates and deviance of the long regression on the control and test variables.
OPT_(7)	0	The parameter estimates and deviance of the short regression on the control variables only.
OPT_(8)	0	The parameter estimates and deviance of the short regression on the control and test variables.
OPT_(9)	1	A plot derived from the short regression. How to interpret the plots is discussed below.

OPT_(10)	1	A listing of the data used in the plot, and an influence measure that indicates the relative importance of each higher node in determining the significance of the test variable.
OPT_(11)	1	The y-variable, the controlled for variables, the test variables, and the final F-ratio of the phylogenetic regression, with its degrees of freedom.
OPT_(12)	0	A page throw before the (text before the) plot
OPT_(13)	0	When using either the taxonomic levels or fully general method of determining node heights, prints out the values of the vector containing the heights.
OPT_(14)	0	A calculation of the total degrees of freedom in the short regression from the total number of higher nodes, minus the number omitted because of omitted species, minus the number omitted for lacking a phylogenetic degree of freedom.
OPT_(15)	0	Gives a breakdown of the total degrees of freedom in the short regression into control DF, additional fitted parameters DF, test DF and residual DF.
OPT_(16)	0	Gives the mean of each variable as calculated using the efficient weights. This allows the best fit equation in the long regression to be calculated.
OPT_(17)	0	Performs an analysis for fixed ρ. The value of ρ should be put in OPT_(17), but a value of zero causes the automatic search by maximum likelihood. When OPT_(17) is not zero, SC__(10) should be set to zero, as there is no extra parameter being estimated.
OPT_(18..19)		NOT USED
OPT_(20)	0	Reduces the text printed at each interruption of the execution of the program
OPT_(21)	0	Interrupts the program just after the fitting of the long regression on control variables only.
OPT_(22)	0	Interrupts the program just after the fitting of the long regression on control and test variables.
OPT_(23)	0	Interrupts the program just after the fitting of the short regression on control variables only.
OPT_(24)	0	Interrupts the program just after the fitting of the short regression on control and test variables.

Some information you cannot avoid:

i) If any denominator degrees of freedom are lost, the program will tell you how many and which higher nodes are involved.

ii) If any numerator degrees of freedom are lost, the program will tell you how many.

Further information is provided in "left-over" vectors. After each analysis, the vector SPU_ is left over. It is the same length as the number of species. It contains a 1 for species included in the most recent analysis, and a 0 for species excluded. Species with a 1 in SPI_ and a 0 in SPU_ were excluded for having missing values.

The other left-overs are for enthusiasts only, and the notation refers to the appendix of the source paper. WL__ contains the diagonal entries of the inverse of the matrix C. WF__ contains the weights used to find the averages at higher nodes, as in Figure 4 of the source paper. B_ contains the node heights before ρ-transformation, for all nodes except the root, scaled so that the root has a height of one. QRS_ contains the linear contrasts used to form the short from the long regression. These are the entries in the matrix GC^{-1} collapsed to take advantage of its block diagonal structure. All these left-overs can be deleted without harm - they are deleted by the program and recreated anyway at each use of GO_. Because they are deleted before the point of maximum use of identifiers, deleting them does not help in reducing the number of identifiers used.

Another way in which output can be controlled is by use of the GLIM directive $ACC, which controls the number of significant figures printed of all numbers. GLIM's default value is 4, and the program does not use $ACC. Hence the user can change the accuracy of all output to give at least 7 significant figures by

```
$ACC 7 $     ! The highest value allowed is 9
```

This will control the output until overridden by the next $ACC directive.

If you decide to take advantage of program interruptions to obtain more information, be very careful - you can easily mess the whole thing up. It is essential in this case to consult the section on "Interrupting the program".

4.1 Further explanations

4.1.1 Long and short regressions

This section explains what is meant by "long regression" and "short regression", terms which are used in the output of the program. The dataset begins as you enter it, with one datapoint for each species. Internally, it undergoes two transformations before the final F-ratio is produced, first into the long regression, then into the short regression.

The long dataset has one datapoint for each node in the phylogeny except the root, including each of the species. It has more datapoints than the species dataset, hence its name. Each datapoint has the average values of all the species below its corresponding node, expressed as a deviation from the corresponding average at its parent node. (The creation of the long regression is the process of "hanging on the tree" described in §3(a) of the source paper.). The long dataset and the long regression are used to perform a regression that is valid in the face of *recognized phylogeny*, but not *unrecognized phylogeny*, that is equivalent to the *standard regression*. For explanations of the italicized terms, see the source paper. For present purposes, it is important that the long regression on the control variables alone is used to fit ρ, by maximum likelihood simultaneously with the regression parameters. It is the log-likelihood from this regression that is reported along with the maximizing value of ρ when OPT_(4)=1. (You can fix ρ by placing the desired value in OPT_(17). To restore automatic fitting, set OPT_(17)=0. If your value of ρ is *a*

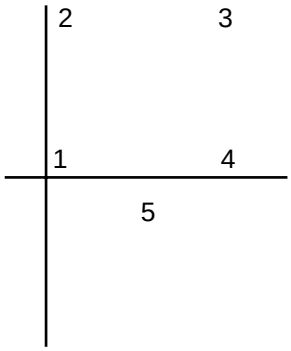
priori, then you should set SC__(10)=0, so that no degree of freedom is subtracted on account of the fitting of ρ.)

The path segment lengths implied by this value of ρ are then used to create the short dataset. This has one datapoint for each higher node (including the root). The idea is that each higher node should contribute one independent unit of information to the final statistical test. Theorem 3 of the source paper shows that the short regression on control variables alone has the same parameter estimates and deviance as the long regression on control variables alone. The F-ratio of the phylogenetic regression is the F-ratio for adding the test variables to the control variables in a regression on this short dataset.

The short regression provides biassed parameter estimates for test variables, because the linear contrasts used to form it from the long dataset are data-dependent. For parameter estimates, it is therefore best to use the estimates from the long regression. These are unbiassed, but their standard errors cannot be trusted.

4.1.2 Plots and influence

The plot provided has on the x-axis the residual in the short regression on the control and test variables. On the y-axis is the net reduction in squared residual brought about by inclusion of the test variable. “Net” means that a certain amount of reduction would be expected by chance, and this chance reduction is taken as the zero point. (Both sets of residuals are divided by the square root of their respective residual degrees of freedom before the difference is taken. Under the null hypothesis, each difference is expected to be zero.) Points badly fitted by the control and test variables have high x-values. Points that contribute positively towards significance of the test variables have high values on the y-axis. Points that resist significance have negative values on the y-axis.



In the figure to the right, point 1 is well fitted by the control and test variables (low on x-axis), but was also well fitted by the control variables alone, as it contributed little to significance (near zero on y-axis). Point 2 contributed a lot to significance of the test variables, and is well fitted by the control and test variables - it has therefore been satisfactorily explained by the test variables where it had not been before. Point 3 contributed much to the significance of the test variables, but is still comparatively poorly fitted. Point 4 was poorly fitted by control variables only, and the test variables didn't help much, as it contributed little to their significance. Point 5 is middling on residuals after the control and test variables, but contributed negatively to the significance of the test variables. This doesn't necessarily mean that its residual is actually larger once the test variables are included in the model, though it may be. It just means that the reduction in the residual was less than would be expected from the degrees of freedom of the models alone. Whether a point has a negative or positive value on the x-axis is irrelevant - only its magnitude matters.

The influence measure provided in this case is the y-variable from this plot, scaled so that the maximum magnitude is 100, and truncated to an integer to ensure easy reading. If the F-ratio is exactly equal to one, then the sum of the influence measures will be close to zero. If the F-ratio is less than one, then the sum of the influence measures will be negative. If the F-ratio is greater than one, then the sum of the influence measures will be positive.

The scaling of the influence measure means that it gives no overall indication of significance. For this, the p-value of the final F-ratio should be used. The influence measure does show the relative influence of the different nodes in determining that significance.

4.1.3 Parameter estimates and model deviances

The phylogenetic regression produces as its main result an F-ratio, as the primary problem with comparative data has been finding valid p-values. The values of slopes, and especially their signs, are important too. Two ways of finding a parameter estimate are possible in the program, one the fully correct method, and the other a quicker approximation. The quick approximation is to look at the parameter estimates of the long regression with the test variables (`OPT_(6)=1`). This method is a very good approximation to the fully correct method. The reason for its slight imperfection is that the value of ρ used in the long regression is the one found by fitting ρ simultaneously with the control variables only. The best method is to use the value of ρ fitted simultaneously with the control and test variables. To achieve this, simply include the control and test variables as the control variables in a further regression (and test for some arbitrary variable), and use the parameter estimates for the long regression with the control variables only (`OPT_(5)=1`). The difference is likely always to be small, and if a variable is significant, it would be astonishing if the sign of the parameter estimates differed between the two methods. The whole point of the phylogenetic regression is that significance tests in the long regression cannot be trusted - this means that the standard errors of the parameter estimates cannot be trusted.

The parameter estimates in the long regression allow a best fit equation to be calculated. Ignore Z0__, this is a syntactical artifice to placate GLIM. The best fit equation is derived in the usual way, except that by the time a variable arrives at the long regression it has been expressed as a deviation from its mean. To write the best fit equation therefore requires knowing the mean for each variable. This is not the simple mean across species, but the efficient mean using the branch lengths of the tree. The program will list these means for you if you set OPT_(16)=1. With categorical variables, you will need to understand how the dummy variables are created, and this follows GLIM conventions as explained in the third example session. But notice that these dummy variables too need to have their mean put back in order to derive the best fit equation.

The short regressions are supplied just for the curious, without (`OPT_(7)=1`) and with (`OPT_(8)=1`) the test variables. Notice that the estimates in the short regression without test variables are the same as the estimates in the long regression without test variables, and the deviances are the same too. This follows from Theorem 3 in the appendix to the source paper. It is important to know that the estimates of the test variables in the short regression are biassed, and are more extreme than the unbiassed estimate provided by the long regression (see §5(d) of the source paper).

4.1.4 Phylogenetic degrees of freedom

These are discussed in §3(e) of the source paper. The basic idea of the phylogenetic regression is not to use species as independent datapoints, but instead to

use the higher nodes in the phylogeny. This means that the number of independent datapoints is the number of higher nodes, including the root. Apart from this basic reduction in numbers of degrees of freedom, compared to the number of species, there are two things that can happen to degrees of freedom because of the phylogenetic aspect. If these things do happen, they are reported by the program under the headings "Phylogenetic degrees of freedom in the numerator" and "Phylogenetic degrees of freedom in the denominator".

The “degrees of freedom in the numerator” refers to the numerator of the F-ratio, and is the number of degrees of freedom in the test variables. Usually there are as many degrees of freedom in the test variables as there are test variables. If one of the test variables can be expressed as a linear combination of other test variables, then the number of degrees of freedom will be less than this. This situation is sometimes called multi-collinearity. If there is multi-collinearity in the species regression in the first place, then there will also be in the phylogenetic regression. But it can happen that variables which are not collinear in the species regression are collinear in the short regression. It is only these extra losses that are reported by the program. Such extra losses are likely to arise when variables differ from each other only at a few higher nodes. When degrees of freedom are lost in the numerator, it means that the variables involved are phylogenetically too restricted in their variation to deserve the full degrees of freedom.

“Degrees of freedom in the denominator” refers to the denominator of the F-ratio. The short regression is formed by condensing the information at a higher node into one datapoint, and to do the condensing it uses the residuals from the (long) regression on the control variables. If these residuals are all zero, then there is no unexplained variation at that node, and no condensing can be done. That higher node must therefore be dropped from the short regression. (Its parent and daughter nodes may still be retained.) When this occurs, the program reports which nodes have been dropped.

There are two ways loss of denominator degrees of freedom is likely to occur. What has to happen is that the fitted and observed values are identical for all the daughters of a higher node. The first reason is that a categorical variable has been included in the regression that takes different values for nearly all of a higher node's daughters and is unvarying elsewhere. Fitting this categorical variable can cause the observed and fitted values to be identical at that node. Any set of variables that differ only at a higher node can have the same effect. This is like, in an ordinary regression, including a categorical variable that is uniform except for one datapoint. Effectively, that datapoint is deleted from the regression.

The second way to lose a denominator degree of freedom is to have a y-variable that “just happens” to take the same value for all the species in a genus, or whose average values “just happen” to be the same for all the daughters of some higher node. If the error in the regression really were normally distributed, this “just happening” would never arise. Often y-variables are treated as continuous, but in fact take a few discrete values. In this case, the “just happening” may arise quite often.

Notice that the loss of a denominator degree of freedom means the loss of a datapoint. This introduces a new way for test variables to become collinear. They may become collinear when a datapoint that prevents collinearity is excluded.

The degrees of freedom possessed by the variables controlled for are subject to just the same considerations as the degrees of fredom possessed by the test variables. These changes are not reported by the program, as they will usually not be of major interest. They may cause puzzlement, however, as the degrees of freedom may not seem to add up correctly. There will seem to be too many altogether if the

control variables are collinear - because these degrees of freedom will remain in the denominator instead of being soaked up by the control variables.

Losses of degrees of freedom in this way do not invalidate the F-ratio. They are detected by the program and appropriate action is taken. But they will happen in rather unusual circumstances, and may be caused by a mistake of some sort. So it is worthwhile working out where and why the degrees of freedom have been lost.

Notice that sometimes degrees of freedom will be lost in the long regression before moving to the short regression. For example, when there is collinearity between some of the control variables, or between control and test variables, in the long regression itself. This has nothing special to do with phylogeny, and is not commented on by the program.

To track down the explanation for the degrees of freedom in the final F-test, it will help to set `OPT_(14)` for an account of the total degrees of freedom in the short regression, and `OPT_(15)` for how that total is divided up. (The degree of freedom allotted to ρ is discussed in §5(b) of the source paper.) Loss of degrees of freedom in the long regression, in control and test variables, can be investigated by setting `OPT_(5)` and `OPT_(6)`. The variables excluded for collinearity will have the word "`aliased`" instead of a standard error in the list of parameter estimates. Setting `OPT_(7)` and `OPT_(8)` will supply the same information for the short regression. Losses present in the long regression will occur in the short regression too, but are not related to phylogeny. Additional losses in the short compared to the long regression represent degrees of freedom lost for phylogenetic reasons.

4.2 Interrupting the program

By setting `OPT_(21)` to `OPT_(24)`, you can cause the program to suspend itself in any of four places in between your call of `GO_`, and the usual return of control once `GO_` has finished. Once the program is suspended, you can ask for further output using ordinary GLIM commands. You then continue execution of the program by typing `$RETURN $`, or

```
$RET $
```

for short. One purpose of this section is to explain what those four places in the program are, what information you can obtain and how to obtain it. The other is to warn that you can easily disrupt the program - the interrupted program is in a delicate state, and must be treated with caution.

The information you can obtain includes i) values of the transformed variables in the long and short regressions ii) further details of the long and short regressions, such as fitted values and covariance matrices.

The places of interruption are first described, and then how to extract information for long and short regressions.

4.2.1 Places of interruption

There are four places in the execution of `GO_` at which you can cause the program to be suspended. These are

Place 1	just after the LONG regression on CONTROL variables only
Place 2	just after the LONG regression on CONTROL and TEST variables
Place 3	just after the SHORT regression on CONTROL variables only

Place 4 just after the SHORT regression on CONTROL and TEST variables

Interruptions at Places 1 to 4 are brought about by the setting to 1 of OPT_(21) to OPT_(24), respectively. You can interrupt at any number of Places in one run of GO_. You can even change OPT_(21) to OPT_(24) during one interruption, to bring about an interruption later in the same run. You might like to know in that case that the actual order of arrival at the Places is not 1,2,3,4; but 2,1,3,4. The long regression on control and test variables happens first.

The details of the corresponding model are available at each Place using the usual GLIM directive $DISP, and the system vectors %FV and so on. Consult the GLIM manual for further details on the information available. I have supplied through the program nearly all the information I think you should need, so you should use further information on your own responsibility.

At each interruption, there is by default a dire warning printed on the screen, along with information about which Place you are at. To avoid all except the barest information, you can set OPT(20) to 1.

4.2.2 Extracting and interpreting information

4.2.2.1 The long regression

Your y-variable and x-variables can be accessed in GLIM directives by their names in the usual way. Thus one can $LOOK at them and $PRINT them. One can use them in $CALCULATE and $ASSIGN directives. The simplest way to take a copy of a variable, suppose it is called BDSZ, is to pick a new name you haven't used before, say BSL, and to

```
$ASSIGN BSL=BDSZ $
```

The new variable will continue to exist after you have $RETURNed control to the program, and can be used in any way you wish once GO_ has finished.

The left-over variable ON_ contains the names of the nodes in the same order as these fitted values, so that

```
$LOOK %YV %FV ON_ $
```

during the interruption will show y-values, fitted values and the node name. (Don't put ON_ first, because it is one element longer than the others (the root has a name, but no fitted value), and $LOOK will give you an error message.) ON_ takes a zero value for omitted species.

Some points should be noted.

1) The variables contain the mean of all the species below or at a node, expressed as a deviation from the mean of all the species below the parental node. This means the variables are not immediately interpretable as heights, or weights or brain sizes; only as differences in heights, or weights or brain sizes.

2) Some species are omitted from the long regression. If a species was omitted because it had a missing value for one of the x-variables, then its fitted value is meaningless. Omitted species are also omitted from all the averaging and differencing done to create the variables of the long regression. Omitted species are indicated by having a zero value in the

weighting vector for the long regression, WL__. They are also indicated in the left-over vector SPU_.

3) To see which higher nodes in the original, full phylogeny correspond to the elements of vectors in the long regression, use ON_. ON_ is a vector that contains the name of the higher node for each datapoint in the long regression. ON_ changes whenever the set of included species changes, for example when a new variable is added to the model that has extra missing values. It is sensible, therefore, to take a copy of the current ON_ at the same time as you take copies of any long vectors.

4) If you calculate residuals using the y-variable and the fitted values (found in %FV), you should be aware that the residuals need *standardizing* before they can be compared. To standardize, multiply the simple residuals (%YV-%FV) by %SQRT(WL__). Then the magnitudes do reflect the extent of deviation from the fitted model. WL__ is a left-over vector, and so this step need not be performed during the interruption.

5) You should not try to perform the long regression yourself with your extracted vectors, unless you really know what you are doing. For example, the degrees of freedom won't seem to add up, because to improve efficiency I have dropped a factor from the regression which I can show analytically will not affect the deviance or parameter estimates - but GLIM does not know this, and so the omitted factor's degrees of freedom misleadingly appear in the residual.

You can obtain further details of the fitted model in the long regression, such as fitted values (in %FV) and the covariance matrix of the parameters (by $DISP C $). You should notice that in one respect the long regression is not an implementation of the standard regression mentioned in the source paper. As already mentioned in note (5), a factor has been dropped for reasons of efficiency. This means that the residual degrees of freedom are inflated, and so the residual mean square will be underestimated. As a result, the standard errors and parameter covariance matrix will be wrong because the scale factor is wrong. But they are wrong anyway because they ignore unrecognized phylogeny. (An informed user could make the necessary corrections to solve the scale factor problem - the phylogenetic regression itself is the solution to the more major problem of unrecognized phylogeny.)

4.2.2.2 The short regression

Your y- and x-variables are still of the length appropriate to the long regression, but only the first so many elements are used in the short regression. The number of elements used in the short regression is contained in SC__(3). Thus in order to $LOOK at the elements of the short regression, we make use of the facility to place two numbers in between $LOOK and the vectors to be looked at. These numbers specify a subrange of the vectors. For example,

```
$CALC %A=SC__(3) $ ! Store the value in a scalar (needn't be %A)

$LOOK 1 %A Y X1 X2 %FV WSH_ L3__$  !  Use to specify the subrange
```

will show the short regression's values of Y, X1 and X2 (which I have supposed are the names of variables you have included in the analysis), %FV (the fitted values in the most recent fit), and the weighting vector WSH_, which will contain a zero for any datapoints omitted because they lack a phylogenetic degree of freedom. Also included is L3__, explained below.

To copy the values of the short regression, we need to create vectors of the right length, and copy across the top SC__(3) values. Suppose we want to put the short regression values of Y and X1 into new variables called SY and SX1, the fitted values into a new variable called SFV, and the weighting vector into SWSH. This can be achieved as follows.

```
$CALC %B=SC__(3) $ ! Store the value in a scalar (needn't be %B)

$VAR %B SY SX1 SFV $!Create the new vectors with the right length

$CALC SY=Y(%CU(1)) $!Copy the top %B values of Y.

$CALC SX1=X1(%CU(1))$ ! Copy the top %B values of X1.

$CALC SFV=%FV(%CU(1))$ ! Copy the top %B values of %FV.

$CALC SWSH=WSH_(%CU(1)) $ ! Copy the top %B values of WSH_.
```

(These $CALC statements copy the first %B values because the length of the left hand side vectors is %B, and this determines the length of the subscripting vector represented by %CU(1), which could be taken to be of any length.) The new variables SY, SX1 and SFV will continue to exist after you have allowed GO_ to complete its action by $RETURNing, and you can use them in any way you want.

The vector L3__ at Place 4 contains, for each datapoint of the short regression, the name of the corresponding higher node. At Place 3, L3__ is being used for another purpose, but the names are the same at Places 3 and 4.. It will usually be worth copying L3__ at the same time as other short vectors. L3__ is likely to be different for analyses that do not include exactly the same set of species.

Some points should be noted.

1) Some of the first SC__(3) datapoints may be omitted from the short regression. The value of variables at these omitted points is meaningless. The variable WSH_ is of the long regression length, and contains 1 for included datapoints, and 0 for unincluded datapoints. You should create a short version of WSH_ along with short versions of your other vectors, so you know which datapoints are meaningless.

2) The meaning even of the meaningful datapoints is not transparent. For one thing, multiplying by -1 all the variables in any datapoint will not affect the result. See §3 of the source paper for some kind of explanation.

3) You should not perform the short regression yourself with the extracted vectors unless you really know what you are doing. For example, the short regression is performed without a constant (GLIM needs to be tricked into allowing this) - and including the constant renders it meaningless.

4) In contrast to the long regression, the residuals are already standardized. And the residual degrees of freedom are correct.

5) Remember that the parameter estimates, in the short regression on control and test variables, are biassed. This bias is mentioned in §5(d) of the source paper.

5 HINTS ON USING THE PROGRAM

5.1 Control files, dumping and restoring

If you are likely to analyse your dataset in more than one GLIM session, it is probably worth putting the preliminary commands into a file, so that the data is read and the phylogeny is created automatically. This means when you make a mistake, you can edit the required changes into your control file, rather than type the whole thing in again at the terminal, and introduce errors into what you got right the first time.

Once the data is in, and transformed as necessary, and the phylogeny has been created, it is worthwhile to $DUMP. This need be done only once. Then in any future session, $RESTORE will take you to exactly to that state very quickly. It also allows a recovery from errors that crash my macros. This is quite easy to do, for the careless, and it can leave your data-vectors in quite a mess. One solution is to quit GLIM, and start all over again. If you have $DUMPed, on the other hand, you need only type

```
$REWIND 3 $   ! Assuming 3 is your dump channel

$RESTORE  $   !  Do $ENV C $ to find what it really is
```

and you will have recovered. You will have been re-instated to exactly the same state as when you gave the corresponding $DUMP command. If you $DUMP more than once in the same session, you should consult the GLIM manual so you know what you are doing. (Successive $DUMPs without $REWINDs store successive dumps in the dump file. After a $REWIND, successive $RESTOREs will restore you to those successive $DUMPs.)

An easy slip to make that wipes out your session so far is to type $END by mistake. (I do this when I think a $USE statement is a $MACRO statement, and try to finish it off with $ENDMAC.) This ends the session, and GLIM cheerfully tells you so. It is in such cases that having used $DUMP at a convenient time saves on temper.

5.2 Crashes and other problems

The sign that the program has crashed is an error message from GLIM. For example, if you have run out of identifiers, the message will say that the directory is full. Control will return to you, the terminal or the batch job, immediately. You will have left the program at some unknown place, and the environment you find yourself in is very unfriendly. It is likely that the vectors used in the model do not have the values you gave them. This is because I store the original values away, make free with the vectors during the program, and put the original values back only at the end. In short, there is nothing useful you can do except

```
$REWIND 3 $! Assuming 3 is your dump channel

$RESTORE $!  Do $ENV C $ to find out what it really is
```

on the charitable assumption you did $DUMP at a useful time.

The important thing is to work out why you crashed, so you can avoid it next time. The easiest ways to cause a crash are as follows.

1) Defining too many vectors. This is discussed under SPACE MANAGEMENT below. You will know this has happened because

GLIM reports that the directory is full, something like this (where it happens precisely depends how many vectors extra you have):

```
** directory full, at [! $ca dr_=]

   on level 5  from macro FT3_


Maximum number of user-defined identifiers allowed is 100.  Use $ENV D to list

the directory then delete unwanted identifiers and try again.
```

2) Including the same vector twice in YV_, C_ and T_. You will know this has happened because you will get a complaint like this from GLIM:

```
** invalid or mixed lengths, at [__)=%%z2 $]

   on level 8  from macro PSH_


The vector G has length 156.  This conflicts with the length 127 used elsewhere

in this directive.
```

where G is the vector that is repeated, and 127 is the number of species in your dataset.

3) Including ten items in a macro that is supposed to have nine or less (e.g. C_, T_, TX_, CON_, TST_, TAX_). If you include a #*macro* in such a macro, the #'d element must count for as many items as are included in *macro*, not just as one. You will know this has happened because GLIM will complain that once the previous directive was completed, it looked for the next directive, but instead found - and then it will give in square brackets the offending tenth item in the list.

4) Having provided too many dummy vectors in IFF_, IFC_ or EXF_. The problem will not arise during the use of those macros, but only when you use one of the created variables in CON_ or TST_. You will know this has happened because the GLIM error message will complain that one of the variable names you supplied as a dummy has not been declared, or has not been given values.

5) Not having defined YV_ or TST_ or CON_ or SPI_. You can check if this is the reason by

```
$PRI YV_ $

$PRI #CON_ $  ! The # is important.

$PRI #TST_ $  ! The # is important.

$PRI SPI_ $
```

If any one is undefined, $PRI will complain when asked to print it. (The # before CON_ and TST_ asks for a printing not of CON_ or

TST_ but of the macros whose names are contained in CON_ and TST_. This therefore checks both levels of definition at once.)

If the program does crash, you may be puzzled by the fact that although you type commands to GLIM, it doesn't reply (except for occasional error messages). The reason is that the program runs with output switched off most of the time, and so is likely to crash in that state. To switch output back on, do

```
$USE OON_ $
```

and GLIM will speak to you again.

5.3 Space management

5.3.1 The number of user identifiers

The aspect of space management that is most problematic is the restriction placed by GLIM on the number of user identifiers. User identifiers are the vectors and macros you define, *and* those I have defined in the program. The usual limit is 100. My program has about 50 identifiers defined as soon as you $INPUT it. It creates about another 30 transiently during analyses, and most of these disappear by the time control returns to you. That leaves only 20 identifiers for the user. In many cases, the user will want to have more than 20 identifiers available to her.

The best solution is to use a version of GLIM that allows more user identifiers, and this is possible if GLIM is being used on a mainframe. GLIM is supplied in source code form to mainframe purchasers, and is designed to allow alteration of some parameters of the system, including the number of user identifiers allowed. This documentation includes a sheet that you can send to your computer centre, to explain exactly what you would like done. Unfortunately, GLIM for PCs is not supplied in source code form. Some mainframe users may not be able to persuade their computer centre to mount a larger version of GLIM. For some users, therefore, the only solution is to manage space carefully.

The main principle is to delete unneeded vectors and macros after use. The macro DLS_ deletes macros in the program that are no longer needed after the phylogeny has been created. So just before you are ready to $USE GO_, you can save space by

```
$USE DLS_ $

$DEL DLS_ $
```

This deletes about ten vectors and macros. If you used the taxonomic levels method for creating the phylogeny, you can delete the vectors that carried the taxonomic information. If you created factors or interactions using IFF_ or IFC_, you can delete the original factors.

To tell whether you have enough space left, do

```
$ENV U $
```

and one of the figures given is the number of user identifiers left. Just before the first use of GO_ (i.e. after declaration of YV_, C_, T_ and SPI_), there should be about twenty-six left for the program to run. Just before subsequent uses of GO_, there should be about eighteen user identifiers left. Provided, you have $DUMPed, nothing disastrous happens if you run the program with too few identifiers left - the program just crashes and you need to recover with

```
$REWIND 3 $    ! Assuming 3 is your dump channel

$RESTORE $     !  $ENV C $ will say what it really is
```

Remember that you may need to `$USE OON_ $` to see your own output, as the program runs mainly with output switched off, and is likely to crash in that state.

5.3.2 Total space

Each version of GLIM has a limit to the total amount of space available in a session, and you might run up against this if you have very large datasets. This limit can be relaxed for mainframes by local computer staff in the same way as the number of user identifiers.

5.3.3 The number of model vectors

As well as a limit to the total number of identifiers that can be declared at any one time, there is a limit of 30 to the number of vectors, apart from the constant, that can be included in a model at one time. In normal use of GLIM, this is reasonably generous because a factor counts as just one "model vector". Unfortunately, the program needs to create its own design vectors (dummy variables), which it does when you use the macros `EXF_`, `IFF_`, and `IFC_`. It is easy, therefore, to run out of model vectors. For example, it is possible to test for the interaction of two factors with 5 levels each, but not of factors of 6 levels each. Unfortunately the limit of 30 is not amendable by local computer staff, and advice from NAG is that the next release of GLIM *may* relax this limit. There is no simple way round this problem.

5.4 Support

Support is logically of three kinds. The first is educational - helping people to use the program and understand the documentation. I wish to do very little of this. This documentation is intended to be complete and clear, and the program is intended to be correct. I cannot spend much time helping people to use the program, as I have work to do. My main research area is neither statistics nor comparative biology. The second kind is correcting bugs in the program or mistakes in the documentation - if necessary, I will do this. The third kind is upgrading the program. I plan not to do this, as the best way to implement this kind of analysis is as part of a major statistical package. Tinkering in GLIM is not a sensible approach in the long run. If a new version of GLIM appears, it is more than likely that the program will fail owing to syntactical changes. I will then have to consider whether it is worth amending the program, taking into account the extent of use of the method, and whether there are other implementations by then. I would obviously be unhappy if people wanted to use the phylogenetic regression, but no implementation was available.

If I receive letters asking for help or advice, I shall reply to them as my time permits. If all this seems unfriendly, remember any commitment would be to an unknown number of people with an unknown number of problems of unknown severity. I will do my best to help, but my help divided by an unknown number may turn out to be quite small.

6 WHICH PATH SEGMENT LENGTH RULE SHOULD I CHOOSE?

The phylogenetic regression will give different F-ratios depending on how you specify the path segment lengths. This may seem undesirable at first sight. The reason is that in reducing all the data to one number, decisions have to be made about how to combine data from different parts of the tree. No method can avoid this arbitrariness. But comparative data is clearly worth analyzing, and so this

arbitrariness has to be accepted as one extra source of uncertainty, along with the unreliability of data, the fact that nature may have played a joke on you, the usually tenuous connection between the theory of interest and the measurements actually to hand, and so on. Indeed, it is exactly the same kind of uncertainty as arises in ordinary regressions, in which there is an arbitrary choice to be made as to how to weight the datapoints against each other. Most people do not even realize there is a choice to be made, but use an unweighted regression because that's what the package offers by default, a choice that usually has no special justification. I am not recommending you to the same blindness, only arguing that comparative analyses suffer this fundamental problem just as other kinds of analysis do - this besetting sin, at least, is not unique.

The first point is that usually choice of weights will not make much difference to the F-ratio. Second, the major dimension of variation is whether much weight should go near the root or near the species tips - and this dimension is automatically taken care of by the fitting of ρ. Third, multiplying all the heights by a positive constant makes no difference, because the program automatically scales them so the root has a height of one. Fourth, adding a constant does make a difference because a different family of path segment lengths will be generated by varying ρ, even though the member with $\rho=1$ will be common to them all.

A sensible course of action would then seem to be as follows. If the taxonomic vector method of specifying the phylogeny is used, then use the taxonomic levels method. If dates of divergence for these levels are known roughly, then use those dates for the heights of the levels. If not, then use 1, 2, 3 …. If the single vector method of specifying the phylogeny is used, then use the default "Figure 2" method.

These recommendations are for normal use, where the phylogeny is of no interest in itself, and the focus of interest is on the interrelationship of the variables. If there is reason to think that the phylogeny has a strong effect, or that there has been much faster evoluion in one part of the tree than in another, then the program provides facilities for exploration. The fully general method allows each node height to be specified separately.

If there is serious disagreement between different phylogenies, then this means the conclusions about the interrelationships of variables are seriously in doubt. This is likely to arise, for example, when some parts of the phylogeny show a strong positive relationship between two variables, and other parts show a strong negative relationship. By choosing how to weight these parts, one can engineer either a net positive, or negative, or zero effect. In such a case, probably there is a relationship, but it may be non-linear, or it may involve an interaction with another variable.

One caution should be given. Having found any significant relationship, it is almost certainly possible to find by systematic exploration some set of path segment lengths that removes it. The mere existence of such a set of lengths is no cause for concern. Finding such lengths by exploration is the logical equivalent of correlating hundreds of x-variables with y, picking the most highly significant, and claiming to have found a significant relationship. Although finding the lengths is a way of diminishing significance, rather than enhancing it, it is similarly cheating because of the data-dependent choices made. Serious doubt should be caused by phylogeny only if pre-specified and biologically reasonable sets of path segment lengths give substantially different answers.

7 PUBLISHING ANALYSES

When reporting analyses using the phylogenetic regression, it is important to give details of which phylogeny was used, and how path segment lengths were assigned. Otherwise the analysis is unrepeatable even if the same dataset is available.

8 EXAMPLE SESSIONS

One extremely useful feature of GLIM is that it produces a transcipt file for each session, that records all your typing and all its replies. You need not hurriedly scribble down numbers as you use GLIM. Instead you can print out the transcript file after the session is over. Often it is convenient to edit before printing, as GLIM repeats the entire contents of data files and all the boring bits, as well as the parts you want.

The GLIM transcript file from four example sessions are shown here. As usual, the lines preceded by [o] are output from the computer, while those preceded by [i] are input from the user. Comments on the right hand side following exclamation marks were inserted by me as explanation afterwards. The first uses a dataset of Mark Ridley, and uses the single vector method of supplying the phylogeny. The second uses a dataset of Paul Harvey, and uses the taxonomic levels method. The third and fourth use a dataset invented with the aid of GLIM.

The transcript files have been slightly edited, so that the output looks more like what happens on the screen. Do not, therefore, be worried by minor discrepancies. The initial welcoming message that occurs on the screen, for example, does not appear in the transcript file - and I have inserted it in Session 4 only. The main difference is that the transcript file echoes all the data input files, while the screen does not - I have deleted these echoes throughout.

Almost every feature of the program is demonstrated in one or other of these programs. (One exception is the fixing of ρ by the user.) The following lists show which features are demonstrated in the four sessions.

Session 1. Single vector method of entering the phylogeny. Data input. Deleting unnecessary identifiers. Altering output options. Automatic missing values. Using only a subset of the species in an analysis.

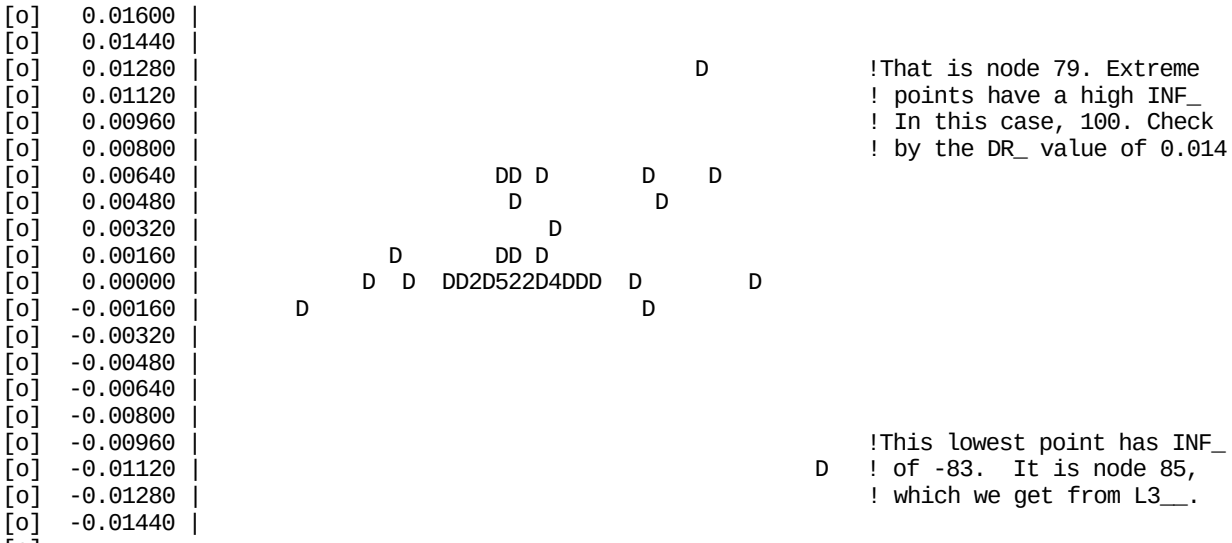
Session 2. Data input with a wide input file. Taxonomic vector method of entering the phylogeny. Use of `WHO_` to identify higher nodes. Transformations. Automatic missing values. Use of `$DUMP` and `$RESTORE`. Identifying points in plots. Looking at fitted values. The taxonomic levels method of path segment length determination. Exploring the effects of varying the heights of the levels. Crashing and recovering using `$REWIND` and `$RESTORE`. Having to switch output back on after a crash.

Session 3. Data input. Taxonomic vector method of entering the phylogeny, with more than nine taxonomic vectors. Creation and use of categorical variables. Creation and use of interactions. Use of `CON_` instead of `C_`. Loss of numerator and denominator degrees of freedom. "Fully general" method of path segment length determination.

Session 4. Use of a control file. Interrupting the program and obtaining intermediate results. Definition of macros by the user to save effort and error. The files used in Sessions 3 and 4 are included on the disk. They are DATA.DAT, PHYLOG1.DAT, PHYLOG2.DAT and RESYNTH.CON.

8.1 Session 1

```
[o] GLIM 3.77 update 2 (copyright)1985 Royal Statistical Society, London
[o]
[i] ? $inp 11 $
[i] File name? phylo.glm                    !The name of the program
[i] ? $units 97 $                           !There are 98 nodes altogether
[i] ? $data phy_ phy_ $dinp 14 $            !Read in the phylogeny.
[i] File name? long.phy                     ! from the file long.phy
[i] ? $units 56 $data d f i p s $dinp 15 $        !Now read in the data
[w] -- model re-initialised                       ! from long.dat
[i] File name? long.dat
[i] ? $use dls_ $                                 ! and delete unnecessary identifiers
[i] ? $del dls_ $                                 !dls_ has just become unnecessary!
[i] ? $env u $                                    !Check on space
[o] Usage:          used   left
[o]   data space   11737 188263
[o]   identifiers     42    158                   !This is the crucial line.  There
[o]   model vecs.      1     30                   ! should be about 25 identifiers
[o]   model terms      1    129                   ! left.  This is an expanded version
[o]   PCS levels       1     15                   ! of GLIM, so there's no problem anyway
[o]
[i] ? $mac yv_ d $endmac                          !The y-variable is to be d
[i] ? $mac c_ $endmac                             !Control for nothing
[i] ? $mac t_ f $endamc                           !Test for f
[i] ? $ca spi_=1 $                                !Include all species
[i] ? $use go_ $                                  !And off we go_
[o]
[o]
[o] Numbers of species:
[o]                          Total:    56
[o]
[o]               Omitted by spi_:     0
[o]    Omitted for missing values:     0
[o]          Included in analysis:    56
[o]
[o]
[o] Below are the parameter estimates from the long regression on control and test
[o] variables.  Their standard errors cannot be trusted.  These are good quick
[o] approximations to the best estimates for the test variables.  The deviance is
[o] also given: the changes should be ignored.
[o]
[o]          estimate        s.e.     parameter
[o]     1       0.000     aliased     ZO__
[o]     2      0.3796     0.04670     F
[o]     scale parameter taken as  0.02722
[o]
[o] Deviance is  2.613  on   96 d.f.   from  97 observations
[o]   change is -1.799 for   -1 d.f.
[o]
[o]
[o]
[o] In the following plot, on the x-axis is RCT_, which is proportional to the
[o] residuals in the short regression with control and test variables.  On the
[o] y-axis is DR_, the net difference in squared residuals in the short regression
[o] between before and after addition of the test variables, allowing for the
[o] change in residual degrees of freedom.  Points badly fitted by control and test
[o] variables have extreme x-values.  Influential points in contributing to
[o] significance have high y-values.
[o]                                         !We can identify points using the printout of
[o]                                         ! variables below
[o]
```



That is node 79. Extreme points have a high INF_ In this case, 100. Check by the DR_ value of 0.014

This lowest point has INF_ of -83. It is node 85, which we get from L3__.

The values displayed below are from the short regression. L3__ is the name of the higher node as given by WHO_. WSH_=1 for higher nodes included in the short regression. WSH_=0 for each node that is excluded because it lacks a phylogenetic degree of freedom. RCT_ and DR_ are the x and y variables on the plot just given. INF_ is an influence measure. RCO_ is proportional to the residual in the short regression on the control variables only.

	L3__	WSH_	RCO_	RCT_	DR_	INF_
1	57.00	1.000	0.038453	0.038930	-0.0000369651	0.000
2	58.00	1.000	0.012305	0.012458	-0.0000037852	0.000
3	59.00	1.000	0.011956	0.012105	-0.0000035738	0.000
4	60.00	1.000	0.033838	0.034259	-0.0000286256	0.000
5	61.00	1.000	0.008490	0.008596	-0.0000018022	0.000
6	62.00	1.000	0.015845	0.016042	-0.0000062768	0.000
7	63.00	1.000	0.036914	-0.028949	0.0005246600	3.000
8	64.00	1.000	0.052296	0.052945	-0.0000683705	0.000
9	65.00	1.000	0.076905	0.011539	0.0057812566	42.000
10	66.00	1.000	0.064258	0.029534	0.0032568832	24.000
11	67.00	1.000	0.004881	0.028244	-0.0007739138	-5.000
12	68.00	1.000	0.037085	0.056831	-0.0018544242	-13.000
13	69.00	1.000	0.023939	-0.005400	0.0005438883	4.000
14	70.00	1.000	0.013099	-0.046923	-0.0020301787	-15.000
15	71.00	1.000	0.047079	0.022718	0.0017003157	12.000
16	72.00	1.000	0.003076	0.003114	-0.0000002366	0.000
17	73.00	1.000	0.026148	0.026473	-0.0000170927	0.000
18	74.00	1.000	0.012305	0.012458	-0.0000037852	0.000
19	75.00	1.000	0.033522	0.033939	-0.0000280936	0.000
20	76.00	1.000	0.050500	-0.021480	0.0020888473	15.000
21	77.00	1.000	0.013811	0.001932	0.0001870169	1.000
22	78.00	1.000	0.026148	0.026473	-0.0000170927	0.000
23	79.00	1.000	0.136891	0.072270	0.0135162249	100.000
24	80.00	1.000	0.094493	0.056375	0.0057507665	42.000
25	81.00	1.000	0.079981	0.014653	0.0061822981	45.000
26	82.00	1.000	0.018124	-0.017173	0.0000335635	0.000
27	83.00	1.000	0.086134	0.087204	-0.0001854762	-1.000
28	84.00	1.000	0.024706	0.025013	-0.0000152591	0.000
29	85.00	1.000	0.025760	0.109385	-0.0113014244	-83.000
30	86.00	1.000	0.035407	0.010996	0.0011327419	8.000
31	87.00	1.000	0.090663	0.059031	0.0047350391	35.000
32	88.00	1.000	0.027686	0.028030	-0.0000191626	0.000
33	89.00	1.000	0.019369	0.019609	-0.0000093788	0.000
34	90.00	1.000	0.013069	0.013232	-0.0000042702	0.000
35	91.00	1.000	0.021533	0.021801	-0.0000115922	0.000
36	92.00	1.000	0.012305	0.012458	-0.0000037852	0.000

```
[o]  37   93.00   1.000   0.005518   0.005587  -0.0000007613     0.000
[o]  38   94.00   1.000   0.110812   0.076468   0.0064319051    47.000
[o]  39   95.00   1.000   0.066979   0.016253   0.0042220047    31.000
[o]  40   96.00   1.000   0.084056   0.024412   0.0064693689    47.000
[o]  41   97.00   1.000   0.005359   0.016057  -0.0002291052    -1.000
[o]  42   98.00   1.000   0.045624   0.015036   0.0018555267    13.000
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: d
[o] Controlling for:
[o]     Testing for: f
[o]
[o] F      = 33.7237                        !A highly significant result
[o]  1,40
[o]
[i] ? $ca opt_(6)=opt_(9)=opt_(10)=0 $      !Switch some output off
[i] ? $mac c_ i $endmac                     !Control for i
[i] ? $use go_ $                            !Another analysis
[o]
[o]
[o] Numbers of species:
[o]                          Total:    56
[o]
[o]               Omitted by spi_:     0
[o]    Omitted for missing values:     0
[o]          Included in analysis:    56
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: d
[o] Controlling for: i
[o]     Testing for: f
[o]
[o] F      = 32.2318
[o]  1,39
[o]
[i] ? $ca opt_(1)=-1 $                      !Set the automatic missing value to -1
[i] ? $mac c_ i s $endmac                   ! because we are controlling now for s
[i] ? $use go_ $                            ! which has missing values
[o]
[o]
[o] Numbers of species:
[o]                          Total:    56
[o]
[o]               Omitted by spi_:     0
[o]    Omitted for missing values:    25          !25 species omitted here
[o]          Included in analysis:    31
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: d
[o] Controlling for: i s
[o]     Testing for: f
[o]
[o] F      = 17.0139
[o]  1,20
[o]
[i] ? $ca spi_=(i==0) $                     !Let's look only at species with i==0
[i] ? $mac c_ s $endmac                     !Note = means "is assigned" and == means
[i] ? $use go_ $                            ! "is equal to".
[o]
[o]
```

```
[o] Numbers of species:
[o]                          Total:    56
[o]
[o]               Omitted by spi_:    32
[o]    Omitted for missing values:    10
[o]          Included in analysis:    14
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: d
[o] Controlling for: s
[o]     Testing for: f
[o]
[o] F      = 17.1913
[o]  1,7
[o]
[i] ? $stop                                    !And there we leave it.
```

8.2 Session 2

```
[o] GLIM 3.77 update 2 (copyright)1985 Royal Statistical Society, London
[o]
[i] ? $inp 11 $                                          !Fine because 10<11<99
[i] File name? phylo.glm                                 !The name of the program
[i] ? $units 127 $                                       !The number of datapoints
[i] ? $data a b c d e f g h i j k $        !11 variables, imaginatively named
[i] ? $dinp 14 100 $                       !harvey.dat is more than 80 chars wide, but
[i] File name? harvey.dat                         ! less than 100.  14 is not equal to 11
[i] ? $mac tx_ a b c $endmac                      !The taxonomic vectors, highest level
[i] ? $use mph_ $                                 ! first, then make the phylogeny
[i] ? $use who_ $                                 !This provides the following output:
[o]
[o]
[o] The first column tp__ is the name of a higher node, which is identified by
[o] giving one species that belongs to that node (in tp1_) and one species that
[o] "just" fails to belong to it (in tp2_).
[o]
[o]       TP__     TP1_      TP2_
[o]   1   128.0     9.000     5.000
[o]   2   129.0    15.000    13.000        ! This output identifies the higher nodes,
[o]   3   130.0    20.000    13.000        ! so that you know which number given by
[o]   4   131.0    30.000    24.000        ! the program corresponds to which node in
[o]   5   132.0    49.000    47.000        ! your phylogeny.
[o]   6   133.0    55.000    47.000
[o]   7   134.0    59.000    63.000        !There is a unique higher node that contains
[o]   8   135.0    78.000    75.000        ! the species given in TP1_, but not the
[o]   9   136.0    89.000    87.000        ! species given in TP2_, but whose parent
[o]  10   137.0    93.000   101.000        ! node contains both.
[o]  11   138.0    97.000   101.000
[o]  12   139.0    99.000   101.000
[o]  13   140.0   106.000    92.000
[o]  14   141.0   111.000   110.000
[o]  15   142.0   115.000   117.000
[o]  16   143.0   118.000   120.000
[o]  17   144.0   121.000   120.000
[o]  18   145.0     5.000    11.000
[o]  19   146.0    13.000     1.000
[o]  20   147.0    24.000    30.000
[o]  21   148.0    33.000     1.000
[o]  22   149.0    37.000    41.000
[o]  23   150.0    45.000    42.000
[o]  24   151.0    47.000    42.000
[o]  25   152.0    63.000    42.000
[o]  26   153.0    67.000    42.000
[o]  27   154.0    70.000     1.000
[o]  28   155.0    75.000    74.000
[o]  29   156.0    85.000     1.000
[o]  30   157.0    87.000    92.000
[o]  31   158.0   101.000    92.000
[o]  32   159.0   104.000    92.000
[o]  33   160.0   110.000    92.000
[o]  34   161.0   113.000    92.000
[o]  35   162.0   117.000    92.000
[o]  36   163.0   120.000    92.000
[o]  37   164.0     3.000     1.000
[o]  38   165.0    11.000     1.000
[o]  39   166.0    30.000     1.000
[o]  40   167.0    41.000     1.000
[o]  41   168.0    42.000     1.000
[o]  42   169.0    74.000     1.000
[o]  43   170.0    82.000     1.000        !The root has an excluded species of zero,
[o]  44   171.0    92.000     1.000        ! as it includes all species.
[o]  45   172.0     1.000     0.000        !
[i] ? $ca e=%if(e==-100,-100,%log(e)) $           !Transform e, f, g, h, k.  The missing
[w] -- invalid function/operator argument(s)      ! value is -100, and we must be careful
[i] ? $ca f=%if(f==-100,-100,%log(f)) $           ! to maintain it through the
[w] -- invalid function/operator argument(s)      ! transformations.  The warning occurs
```

```
[i] ? $ca g=%if(g==-100,-100,%log(g)) $           ! because GLIM evaluates both halves of
[w] -- invalid function/operator argument(s)      ! the IF condition, including some
[i] ? $ca h=%if(h==-100,-100,%log(h)) $           ! log(-100)'s, before discarding the
[i] ? $ca k=%if(k==-100,-100,%log(k)) $           ! unwanted ones.
[i] ? $mac yv_ e $endmac                    !The y-variable is to be e
[i] ? $mac c_ f $endmac                     !We control for f
[i] ? $mac t_ g $endmac                     !We test for g
[i] ? $ca spi_=1 $                          !We include all species
[i] ? $ca opt_(1)=-100 $                    !The missing value is -100
[i] ? $dump $                               !We $DUMP to reduce the impact of crashing
[w] -- program dump completed
[i] ? $ca opt_(1)=0 $                       !As an illustration, we “forget” to exclude
[i] ? $use go_ $                            ! missing values, and do an analysis
[o]
[o]
[o] Numbers of species:
[o]                          Total:   127
[o]
[o]               Omitted by spi_:     0
[o]    Omitted for missing values:     0
[o]          Included in analysis:   127
[o]
[o]
[o] Below are the parameter estimates from the long regression on control and test
[o] variables.  Their standard errors cannot be trusted.  These are good quick
[o] approximations to the best estimates for the test variables.  The deviance is
[o] also given: the changes should be ignored.
[o]
[o]          estimate        s.e.     parameter
[o]     1       0.000     aliased     ZO__
[o]     2    -0.06368     0.04030     F
[o]     3      0.8828     0.03752     G
[o]     scale parameter taken as  298.0
[o]
[o] Deviance is   50363.  on  169 d.f.   from  171 observations
[o]   change is -164954. for   -1 d.f.
[o]
[o]
[o]
[o] In the following plot, on the x-axis is RCT_, which is proportional to the
[o] residuals in the short regression with control and test variables.  On the
[o] y-axis is DR_, the net difference in squared residuals in the short regression
[o] between before and after addition of the test variables, allowing for the
[o] change in residual degrees of freedom.  Points badly fitted by control and test
[o] variables have extreme x-values.  Influential points in contributing to
[o] significance have high y-values.
[o]
[o]
[o]
[o]     680.0 |
[o]     640.0 |                     D       !This point has INF_==100, confirmed because
[o]     600.0 |                             ! DR_ is 648.  It is the root, 173, which we
[o]     560.0 |                             ! can tell from L3__.  All these variables
[o]     520.0 |                             ! we find in the table of values below.
[o]     480.0 |
[o]     440.0 |               D
[o]     400.0 |
[o]     360.0 |               D
[o]     320.0 |                                       D
[o]     280.0 |
[o]     240.0 |                D
[o]     200.0 |                2                             D
[o]     160.0 |                52                D
[o]     120.0 |                                 D
[o]      80.0 |               D D4
[o]      40.0 |               23DD  D
[o]       0.0 |        2       D 9
[o]     -40.0 |                                     D
[o]     -80.0 |
```

```
[o] ----------:---------:---------:---------:---------:---------:---------:
[o]        -8.00     -4.00      0.00      4.00      8.00     12.00     16.00
[o]
[o]
[o] The values displayed below are from the short regression.  L3__ is the name of
[o] the higher node as given by WHO_.  WSH_=1 for higher nodes included in the
[o] short regression.  WSH_=0 for each node that is excluded because it lacks a
[o] phylogenetic degree of freedom.  RCT_ and DR_ are the x and y variables on the
[o] plot just given.  INF_ is an influence measure.  RCO_ is proportional to the
[o] residual in the short regression on the control variables only.
[o]
[o]       L3__    WSH_      RCO_        RCT_           DR_        INF_
[o]   1   128.0   1.000    0.06726    0.11188851    -0.0079948     0.000
[o]   2   129.0   1.000   15.32681   -0.81959766   234.2392120    36.000
[o]   3   130.0   1.000    4.82649   -0.79235393    22.6672039     3.000
[o]   4   131.0   1.000    0.14084   -0.00059033     0.0198352     0.000
[o]   5   132.0   1.000    5.19734   -1.06953299    25.8684177     3.000
[o]   6   133.0   1.000    8.36022   -0.04855061    69.8909683    10.000
[o]   7   134.0   1.000    6.58998    8.02543640   -20.9798546    -3.000
[o]   8   135.0   1.000   12.39671   -0.82059598   153.0051575    23.000
[o]   9   136.0   1.000    0.21781    0.06821215     0.0427886     0.000
[o]  10   137.0   1.000   12.27409   -0.38882628   150.5019989    23.000
[o]  11   138.0   1.000   12.10913   -0.70225966   146.1379852    22.000
[o]  12   139.0   1.000   12.22530   -0.61611152   149.0782623    23.000
[o]  13   140.0   1.000    0.16264    0.08406164     0.0193842     0.000
[o]  14   141.0   1.000    0.08669    0.09488149    -0.0014882     0.000
[o]  15   142.0   1.000    0.04422   -0.01735289     0.0016541     0.000
[o]  16   143.0   1.000    0.01048    0.01799326    -0.0002138     0.000
[o]  17   144.0   1.000    8.38900   -0.00008719    70.3753204    10.000
[o]  18   145.0   1.000    7.98267   -1.22429311    62.2241936     9.000
[o]  19   146.0   1.000    6.29514   -1.10601342    38.4054756     5.000
[o]  20   147.0   1.000   20.50464    9.90808773   322.2701721    49.000
[o]  21   148.0   1.000   13.61895   -0.86532521   184.7271118    28.000
[o]  22   149.0   1.000   13.81551    6.80617952   144.5442352    22.000
[o]  23   150.0   1.000    0.16954   -0.02424331     0.0281566     0.000
[o]  24   151.0   1.000   17.83969   11.19276047   192.9768524    29.000
[o]  25   152.0   1.000    6.04152    1.16057384    35.1530495     5.000
[o]  26   153.0   1.000   13.97747   -0.97288620   194.4232330    30.000
[o]  27   154.0   1.000    4.94474   -0.76961553    23.8581009     3.000
[o]  28   155.0   1.000    9.65673   -0.39331010    93.0976639    14.000
[o]  29   156.0   1.000   12.16695   -0.80287260   147.3901825    22.000
[o]  30   157.0   1.000    9.09843   -0.10080893    82.7711945    12.000
[o]  31   158.0   1.000    7.63078   -0.44288164    58.0327339     8.000
[o]  32   159.0   1.000   12.07192   -0.69192821   145.2524719    22.000
[o]  33   160.0   1.000    0.13604    0.00116157     0.0185042     0.000
[o]  34   161.0   1.000   12.41313   -0.55374366   153.7792664    23.000
[o]  35   162.0   1.000    8.29061   -0.02174419    68.7338104    10.000
[o]  36   163.0   1.000   18.98673   -1.16605031   359.1362000    55.000
[o]  37   164.0   1.000    4.93692   -0.63947624    23.9642220     3.000
[o]  38   165.0   1.000    4.13916   -0.79514986    16.5004158     2.000
[o]  39   166.0   1.000    3.07927   -4.36826706    -9.5998459    -1.000
[o]  40   167.0   1.000    5.05368   -4.47499895     5.5140991     0.000
[o]  41   168.0   1.000   12.29784    6.28595257   111.7236023    17.000
[o]  42   169.0   1.000    4.79944   -0.11933664    23.0203400     3.000
[o]  43   170.0   1.000    0.04296   -0.00018008     0.0018457     0.000
[o]  44   171.0   1.000   21.27913   -1.04710066   451.7048340    69.000
[o]  45   172.0   1.000   25.46013    0.38069388   648.0734253   100.000
[o]
<<<< We can pick out any individual point in the graph, and find which node it corresponds
to, because both the y- and x- coordinates are given in the $LOOK values. >>>>
[o]
[o]
[o]
[o]
[o]      y-variable: e
[o] Controlling for: f
[o]     Testing for: g
[o]
[o] F      = 460.158                                  !A massive F-ratio,
```

```
[o]  1,42                                                    ! which can usually be arranged
[o]                                                          ! by including missing values
[i] ? $ca opt_(10)=opt_(6)=0 $ca opt_(1)=-100 $use go_ $  !Now lets do it properly and
[o]                                                          ! exclude them.  At the same
[o]                                                          ! cut down on the output.
[o] Numbers of species:
[o]                         Total:   127
[o]
[o]              Omitted by spi_:     0
[o]    Omitted for missing values:    89          !So most species had missing values!
[o]          Included in analysis:    38
[o]
[o]
[o] In the following plot, on the x-axis is RCT_, which is proportional to the
[o] residuals in the short regression with control and test variables.  On the
[o] y-axis is DR_, the net difference in squared residuals in the short regression
[o] between before and after addition of the test variables, allowing for the
[o] change in residual degrees of freedom.  Points badly fitted by control and test
[o] variables have extreme x-values.  Influential points in contributing to
[o] significance have high y-values.
[o]
[o]
[o]
[o]   0.02800 |               D
[o]   0.02400 |
[o]   0.02000 |
[o]   0.01600 |
[o]   0.01200 |                     D
[o]   0.00800 |                           D
[o]   0.00400 |                         D
[o]   0.00000 |         DD D                          D
[o]  -0.00400 |             DD       D
[o]  -0.00800 |
[o]  -0.01200 |                  D  D    D
[o]  -0.01600 |
[o]  -0.02000 |
[o]  -0.02400 |
[o]  -0.02800 |                                              D
[o]  -0.03200 |
[o]  -0.03600 |
[o]  -0.04000 |
[o]  -0.04400 |
[o]  -0.04800 |                                        D
[o] ----------:---------:---------:---------:---------:---------:---------:
[o]        -0.160    0.000     0.160     0.320     0.480     0.640     0.800
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: e
[o] Controlling for: f
[o]     Testing for: g
[o]
[o] F      = 0.202928                          !Not so impressive - such is life.
[o]  1,13
[o]
[i] ? $loo spu_ $                          !Let's check which species were actually
[o]        SPU_                            ! included in that analysis.  SPU_ is left
[o]    1   0.000                           ! over after each analysis, and contains
[o]    2   1.000                           ! "1" for included species, and "0" for
[o]    3   0.000                           ! species that are excluded (either because
[o]    4   0.000                           !  they have SPI_==1, or because they have
[o]    5   0.000                           ! missing values in one of the variables
[o]    6   0.000                           ! included in the analysis.  So species 2 was
[o]    7   0.000                           ! in, but the rest of 1 to 9 were out.
[o]    8   0.000
<<< Units 9 to 119 deleted to save space  >>>
[o]  120   1.000
```

```
[o]  121   1.000                          !120, 121, 123, 125, 127 were included
[o]  122   0.000                          !122, 124, 126 were excluded
[o]  123   1.000
[o]  124   0.000
[o]  125   1.000
[o]  126   0.000
[o]  127   1.000
[i] ? $ca opt_(9)=opt_(3)=0 $             !Now we decide to limit output again, and
[i] ? $ass z=1,2,3 $use go_ z $           ! to use the taxonomic levels method of
[o]                                       ! path segment length determination
[o]
[o]
[o]
[o]
[o]      y-variable: e
[o] Controlling for: f
[o]     Testing for: g
[o]
[o] F      = 0.202928                     !But this is suspiciously similar to last time
[o]  1,13                                 !We foolishly specified only 3 heights instead
[o]                                       ! of 4.  Fix, and ask for confirmation with
[i] ? $ca opt_(2)=1 $ass z=1,2,3,4 $use go_ $     ! OPT_(2)=1 of the node height method.
[o]
[o]
[o] Using the taxonomic levels as node heights.  The heights for each level are
[o] taken from the vector Z.
[o]
[o]                                       !This time, we are getting the right method.
[o]
[o]
[o]
[o]      y-variable: e
[o] Controlling for: f
[o]     Testing for: g
[o]
[o] F      = 0.0851183
[o]  1,13
[o]
[i] ? $mac t_ g h k $endmac               !Test for three variables at once, for fun.
[i] ? $ca opt_(9)=opt_(10)=opt_(2)=0 $    ! Switch output off, and explore the effects
[i] ? $ass z=1,6,7,10 $use go_ $          ! changing the node heights.  Because we
[o]                                       ! haven't cancelled the use of z and the node
[o]                                       ! heights vector (which we'd do by $USE GO_ %A)
[o]                                       ! altering z alters the node heights.
[o]
[o]
[o]      y-variable: e
[o] Controlling for: f
[o]     Testing for: g h k
[o]
[o] F      = 0.423207                     !Not very exciting!
[o]  3,11
[o]
[i] ? $ca opt_(2)=opt_(13)=1 $            !Try again with a different z.  Ask for
[i] ? $ass z=1,6,7,17 $use go_ $          ! confirmation AND a printout of levels.
[o]
[o]
[o] Using the taxonomic levels as node heights.  The heights for each level are
[o] taken from the vector Z.
[o]
[o]
[o] The values of Z are:
[o]
[o]    1.000   6.000   7.000   17.00      !We got this by specifying opt_(13)=1
[o]
[o]
[o]
[o]
[o]
```

```
[o]      y-variable: e
[o] Controlling for: f
[o]     Testing for: g h k
[o]
[o] F      = 0.346742                    !Not much difference, really.
[o]  3,11
[o]
[i] ? $mac c_ f g $endmac                !Now control for g, and "forget" to remove
[i] ? $use go_ $                         ! it from the test variables.
[o]
[o]
[o] Using the taxonomic levels as node heights.  The heights for each level are
[o] taken from the vector Z.
[o]
[o]
[o] The values of Z are:
[o]
[o]    1.000   6.000   7.000   17.00       !OK till now, but then we crash:
<<<< Here we crashed, and the following few lines appeared on the screen >>>
<<<< but not in the transcript file.  I have re-instated them to show >>>
<<<< what happened.                                                   >>>

*** invalid or mixed lengths, at [__)=%%z2 $]
  on level 8  from macro PSH_

The vector G has length 142.  This conflicts with the length 127 used elsewhere
in this directive.                              !Pretty scary, huh?

    ? $use oon_ $                               !Then I switched output back on
<<<< Now back to the transcript file  >>>
[i] ? $rewind 3 $                        !To recover, we must rewind our dump file
[i] ? $restore $                         ! and restore (check if your number is 3
[w] -- program restarts from previous dump ! with $env c $ ).
[i] ? $pri yv_ :c_ :t_ $                        !I've forgotten what the y-variable
[o] e                                           ! and stuff was when we dumped, so
[o] f                                           ! let's look at it, and the output
[o] g                                           ! options too.
[i] ? $loo opt_ $
[o]        OPT_
[o]   1  -100.000
[o]   2     0.000
[o]   3     1.000
[o]   4     0.000
[o]   5     0.000
[o]   6     1.000
[o]   7     0.000
[o]   8     0.000
[o]   9     1.000
[o]  10     1.000
[o]  11     1.000
[o]  12     0.000
[o]  13     0.000
[o]  14     0.000
[o]  15     0.000
[o]  16     0.000
[o]  17     0.000
[o]  18     0.000
[o]  19     0.000
[o]  20     0.000
[o]  21     0.000
[o]  22     0.000
[o]  23     0.000
[o]  24     0.000
[i] ? $ca opt_(3)=opt_(6)=opt_(9)=opt_(10)=0 $     !Now fix what needs changing for
[i] ? $mac c_ f g $endmac                         ! our present purpose, and
[i] ? $mac t_ h k $endmac                         ! perform our analysis
[i] ? $use go_ $
[o]
[o]                                               !We have used the Figure 2 method
```

```
[o]                                          ! because that's what we were using
[o]                                          ! when we $DUMPed
[o]
[o]      y-variable: e
[o] Controlling for: f g
[o]     Testing for: h k
[o]
[o] F      = 1.22132
[o]  2,11
[o]
[i] ? $stop                                  !The end
```

8.3 Session 3

```
[o] GLIM 3.77 update 2 (copyright)1985 Royal Statistical Society, London
[o]
[i] ? $inp 11 $
[i] File name? phylo.glm                          !The name of the program
[i] ? $units 100 $                                !The number of datapoints
[i] ? $data a b c d e f g h i j k l $dinp 14 $    !12 taxonomic vectors, on channel 14
[i] File name? phylog1.dat                        !The name of the file for channel 14
[i] ? $data m n o p q r s t u v w x $dinp 15 $    !12 more taxonomic vectors, on ch 15
[i] File name? phylog2.dat                        !The name of the file for channel 15
[i] ? $mac tax_ tx1 tx2 tx3 $endmac               !The names of macros
[i] ? $mac tx1 a b c d e f g h i $endmac          ! that contains the names of the
[i] ? $mac tx2 j k l m n o p q $endmac            ! taxonomic vectors  in order,
[i] ? $mac tx3 r s t u v w x $endmac               ! highest level first
[i] ? $use mph_ $                                 !Create the phylogeny
[i] ? $use who_ $                                 !Identify the higher nodes
[o]
[o]
[o] The first column tp__ is the name of a higher node, which is identified by
[o] giving one species that belongs to that node (in tp1_) and one species that
[o] "just" fails to belong to it (in tp2_).
[o]
[o]       TP__     TP1_     TP2_
[o]   1   101.0    1.000    3.000
[o]   2   102.0    3.000    1.000
[o]   3   103.0    5.000    7.000

<< Intermediate nodes deleted to save space  >>>

[o]  69   169.0   17.000   13.000
[o]  70   170.0   13.000    9.000
[o]  71   171.0    9.000    5.000
[o]  72   172.0    5.000    1.000                              !The taxonomic vectors
[o]  73   173.0    1.000    0.000                              ! have served their
[i] ? $del a b c d e f g h i j k l m n o p q r s t u v w x $    ! purpose, so delete them
[w] -- $data list abolished                                    ! to save space
[i] ? $data f g p q $dinp 16 $                          !Read in the data on channel 16
[i] File name? data.dat                           !The name for the file on channel 16
[i] ? $mac ff f2 f3 f4 f5 $endmac                       !A macro for the factor f
[i] ? $use exf_ f ff $                                  !Construct the design variables
[i] ? $mac gf g2 g3 g4 g5 $endmac                       !A macro for the factor g
[i] ? $use exf_ g gf $                                  !Construct the design variables
[i] ? $mac i i2 i3 i4 i5 $endmac                  !A macro for the f times g interaction
[i] ? $mac i2 i22 i23 i24 i25 $endmac             !Each item in i is the name of a macro
[i] ? $mac i3 i32 i33 i34 i35 $endmac             ! which contains the names of vectors
[i] ? $mac i4 i42 i43 i44 i45 $endmac             ! which are to contain the design
[i] ? $mac i5 i52 i53 i54 i55 $endmac             ! variables for the interaction
[i] ? $use iff_ f g i $                   !This creates the design variables
[i] ? $dump $                                           !After all that work, better $DUMP
[w] -- program dump completed
[i] ? $mac yv_ p $endmac                  !The first analysis is of p on q
[i] ? $mac c_ $endmac                     ! controlling for nothing
[i] ? $mac t_ q $endmac
[i] ? $ca spi_=1 $                        ! and including all species
[i] ? $ca opt_(3)=opt_(6)=opt_(9)=opt_(10)=0 $    !Restrict output to F-ratio only
[i] ? $use go_ $
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: p
[o] Controlling for:
[o]     Testing for: q
[o]
[o] F      = 2.19091
[o]  1,71
[o]
```

```
[i] ? $mac tst_ ff $endmac                !Next, test for the factor f, so use the
[i] ? $use go_ $                          ! macro that contains the design variables
[o]                                       !Notice it goes in tst_, not in t_, because
[o]                                       ! ff is a macro, not a vector.
[o]
[o]
[o]
[o]      y-variable: p
[o] Controlling for:
[o]     Testing for: f2 f3 f4 f5          !This lists the design vectors, not the macro
[o]
[o] F      = 1.75214                      !Pretty non-significant
[o]  4,68
[o]
[i] ? $mac con_ ff $endmac                !Now control for f
[i] ? $mac tst_ gf $endmac                ! and test for g
[i] ? $use go_ $
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: p
[o] Controlling for: f2 f3 f4 f5
[o]     Testing for: g2 g3 g4 g5
[o]
[o] F      = 2.34447                      !Nearly significant at 5%
[o]  4,64
[o]
[i] ? $mac con_ ff gf $endmac             !Next, test for the f by g interaction.  We
[i] ? $mac tst_ #i $endmac                ! control for f and g, of course.  Then we
[i] ? $ca opt_(6)=1 $                     ! put #i in tst_.  The # is because i is a
[i] ? $use go_ $                          ! macro that contains macros that contains
[o]                                       ! vectors.  The # effectively replaces i
[o]                                       ! with its contents in tst_.
[o] Below are the parameter estimates from the long regression on control and test
[o] variables.  Their standard errors cannot be trusted.  These are good quick
[o] approximations to the best estimates for the test variables.  The deviance is
[o] also given: the changes should be ignored.
[o]
[o]          estimate        s.e.     parameter
[o]     1       0.000     aliased     ZO__
[o]     2     -0.1237      0.1699     F2
[o]     3      0.1136      0.1699     F3          !We asked for this output with opt_(6)
[o]     4      0.3628      0.1699     F4          ! set to one.  Because of our
[o]     5     0.09019      0.1520     F5          ! sympathetic choice of names for our
[o]     6     -0.2256      0.1690     G2          ! design vectors, these have the same
[o]     7    -0.05751      0.1699     G3          ! intepretation as if they really were
[o]     8    -0.06031      0.1699     G4          ! factors in GLIM.  So F2 corresponds
[o]     9     -0.1711      0.1474     G5          ! F(2), G3 to G(3), and I24 to
[o]    10      0.3128      0.2390     I22         ! F(2).G(4).  If you don't know what
[o]    11    -0.01486      0.2403     I23         ! the correspondents mean, consult
[o]    12      0.3695      0.2403     I24         ! the GLIM manual or a statistician
[o]    13      0.2585      0.2085     I25         ! who has one.
[o]    14     -0.1306      0.2390     I32
[o]    15    -0.07221      0.2403     I33         !We could have chosen any names for
[o]    16    -0.08755      0.2403     I34         ! the design vectors.  It is not
[o]    17     0.04458      0.2084     I35         ! necessary to name them
[o]    18     -0.4108      0.2390     I42         ! sympathetically, just helpful.
[o]    19    -0.07721      0.2402     I43
[o]    20     -0.1974      0.2402     I44
[o]    21     -0.1715      0.2084     I45
[o]    22     -0.1804      0.2138     I52
[o]    23    -0.09326      0.2149     I53
[o]    24     -0.1476      0.2149     I54
[o]    25      0.1006      0.1894     I55
[o]     scale parameter taken as  0.04338
[o]
[o] Deviance is  6.4196  on  148 d.f.   from  172 observations
```

```
[o]   change is -0.1342 for   -4 d.f.
[o]
[o]
[o]
[o]                                        !I'm sorry that the list of variables tested
[o]                                        ! for spills over so ungainlily.  I can't see
[o]                                        ! a good way to stop it
[o]      y-variable: p
[o] Controlling for: f2 f3 f4 f5 g2 g3 g4 g5
[o]     Testing for: i22 i23 i24 i25 i32 i33 i34 i35 i42 i43 i44 i45 i52 i53 i54
[o] i55
[o]
[o] F      = 0.608666                      !Non-significant
[o]  16,48
[o]
[i] ? $mac yv_ g $endmac                   !Now I set the y-variable to be g, which
[i] ? $mac con_ c_ $endmac                 ! is a factor.  This is to illustrate what
[i] ? $mac c_ $endmac                      ! happens when the y-variable can take only
[i] ? $mac tst_ t_ $endmac                 ! a few discrete values
[i] ? $mac t_ q $endmac
[i] ? $ca opt_(6)=0 $                      !Notice I've redefined tst_ and con_ as well
[i] ? $use go_ $                           ! as t_ and c_.  If tst_ doesn't contain t_
[o]                                        ! then putting q in t_ wouldn't work
[o] PHYLOGENETIC DEGREES OF FREEDOM IN THE DENOMINATOR
[o]
[o] 16 nodes were omitted as lacking a phylogenetic degree of freedom.  The numbers
[o] of those higher nodes are:
[o]
[o]    103   106   109   112   115   118   121   124   127   130   133   136   139
[o]    142   145   148
[o]
[o]                                         !All these nodes have no variation at them
[o]                                        ! i.e. each daughter has the same y-value
[o]                                        ! Such nodes must be omitted from the
[o]                                        ! short regression, and so its sample size
[o]                                        ! diminishes.
[o]
[o]      y-variable: g
[o] Controlling for:
[o]     Testing for: q
[o]
[o] F      = 0.659356
[o]  1,55
[o]
[i] ? $mac c_ p $endmac
[i] ? $use go_ $
[o]
[o]                                        !When we control for p, we lose no degrees
[o]                                        ! of freedom.  This is because after
[o]                                        ! controlling for p, the residuals of g are
[o]                                        ! not identical within any node.  We regain
[o]      y-variable: g                     ! all our degrees of freedom
[o] Controlling for: p
[o]     Testing for: q
[o]
[o] F      = 0.602974                      !It's still non-significant, though
[o]  1,70
[o]
[i] ? $ca fac=1+(%cu(1)>=3)+(%cu(1)>=5) $  !fac is a factor that doesn't vary within
[i] ? $mac facf fc2 fc3 $endmac            ! the three daughters of the root node, but
[i] ? $use exf_ fac facf $                 ! takes a different value in each daughter
[i] ? $mac c_ $endmac                      !We control for nothing, and test fac
[i] ? $mac tst_ facf $endmac
[i] ? $use go_ $
[o]
[o] PHYLOGENETIC DEGREES OF FREEDOM IN THE DENOMINATOR    !Same as before now we aren't
[o]                                                       ! controlling for p
[o] 16 nodes were omitted as lacking a phylogenetic degree of freedom.  The numbers
[o] of those higher nodes are:
```

```
[o]
[o]    103   106   109   112   115   118   121   124   127   130   133   136   139
[o]    142   145   148
[o]
[o]
[o]                                                          !The new feature is here
[o]                                                          !fac has 2 degrees of freedom
[o] PHYLOGENETIC DEGREES OF FREEDOM IN THE NUMERATOR         ! in the long regression
[o]                                                          ! because it has 3 levels
[o] 1 degree of freedom was lost in the numerator            ! But when condensation to
[o]                                                          ! the short regression occurs
[o]                                          ! the two design variables turn out to be
[o]                                          ! collinear.  This is because all their
[o]                                          ! variation is restricted to one higher node,
[o]                                          ! in this case the root.  In general, the root
[o]                                          ! can "sustain" one degree of freedom for
[o]      y-variable: g                       ! a variable, while other nodes can sustain 2
[o] Controlling for:
[o]     Testing for: fc2 fc3
[o]
[o] F      = 0.558405                        !Still non-significant
[o]  1,55
[o]
[i] ? $mac ifcp if2 if3 $endmac              !Now I define the interaction between
[i] ? $use ifc_ fac p ifcp $                 ! fac and p, placing the design variables
[i] ? $mac con_ facf c_ $endmac              ! in the macro ifcp.
[i] ? $mac c_ p $endmac                      !We control for facf and p, of course, and
[i] ? $mac tst_ ifcp $endmac                 ! test for ifcp.
[i] ? $use go_ $
[o]
[o] PHYLOGENETIC DEGREES OF FREEDOM IN THE DENOMINATOR
[o]
[o] One node was omitted as lacking a phylogenetic degree of freedom.  The number
[o] of that higher node is: 173
[o]
[o]                                          !I wasn't expecting that either, but on
[o]                                          ! reflection it should indeed happen.  I'm
[o]                                          ! glad the program is checking all this for
[o]                                          ! me.
[o]      y-variable: g
[o] Controlling for: fc2 fc3 p
[o]     Testing for: if2 if3
[o]
[o] F      = 1.00634                         !Significance never seems to happen with this
[o]  2,68                                    ! dataset.  I wonder why this is.
[o]
<<<  This F-ratio seems to have too many degrees of freedom.  The reason is that two of
the control variables have lost their degrees of freedom for phylogenetic reasons, as
$CALC OPT_(14)=opt_(15)=1 $ and $USE GO_ would confirm.  See the section on “Phylogenetic
degrees of freedom” in the manual.  >>>
[i] ? $ca ipq=p*q $                          !Now I create the interaction between p and q
[i] ? $mac con_ c_ $endmac
[i] ? $mac c_ p q $endmac                    !Control for p and q, of course, while testing
[i] ? $mac tst_ t_ $endmac                   ! for the interaction.  Note tst_ needs to be
[i] ? $mac t_ ipq $endmac                    ! reset again to t_, and con_ to c_.  The
[i] ? $use go_ $                             ! interaction of two continuous variables is
[o]                                          ! a continuous variable.
[o]
[o]
[o]
[o]
[o]      y-variable: g
[o] Controlling for: p q
[o]     Testing for: ipq
[o]
[o] F      = 0.727927                        !Well I never
[o]  1,69
[o]
[i] ? $ass nh=b_,1 $                         !Now I change tack, and show how to use the
```

```
[i] ? $ca opt_(2)=1 $use go_ nh $          ! "fully general" method of assigning node
[o]                                       ! heights.  b_ is a left-over vector with the
[o]                                       ! old heights for the non-root nodes, scaled
[o] Individual node heights taken from the vector NH.     ! so the root is one.  Hence
[o]                                       !b y $ASSigning nh=b_,1, we should get exactly
[o]                                       ! the same answer as before.  (Note opt_(2)=1
[o]                                       ! got us confirmation of the node height
[o]                                       ! method.)
[o]                                       !Normally if you use this method, you'd read
[o]      y-variable: g                    ! the heights from a file specially
[o] Controlling for: p q                  ! prepared.
[o]     Testing for: ipq
[o]
[o] F      = 0.727927                     !Indeed, exactly the same as before
[o]  1,69
[o]
[i] ? $ca %a=%cu(nh==nh) :%a $            !Check how many elements there are in nh,
[o]           173.0                       ! which tells us the number of the root
[i] ? $ca nh(173)=1.1 $use go_ $          !Tinker with the height of the root, leaving
[o]                                       ! the rest unchanged, and analyse again.  Note
[o]                                       ! we don't need to re-specify to use nh.
[o] Individual node heights taken from the vector NH.
[o]
[o]
[o]
[o]
[o]      y-variable: g
[o] Controlling for: p q
[o]     Testing for: ipq
[o]
[o] F      = 0.671883                     !An altered F-ratio
[o]  1,69
[o]
[i] ? $ca nh(173)=1.2 $ca opt_(2)=0 $use go_ $     !Tinker again with the root height
[o]
[o]
[o]
[o]
[o]      y-variable: g
[o] Controlling for: p q
[o]     Testing for: ipq
[o]
[o] F      = 0.669845                     !Another altered F-ratio
[o]  1,69
[o]
[i] ? $use go_ %a $                       !Finally, return to the Figure 2 method
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: g
[o] Controlling for: p q
[o]     Testing for: ipq
[o]
[o] F      = 0.727927                     ! to get the original result again
[o]  1,69                                 !The reason all the results are NS is that
[o]                                       ! both the continous variables come from
[i] ? $stop                               ! GLIM's random number generator.  Mind you,
! most comparative methods wouldn't let that stop them giving you significance.
```

8.4 Session 4

First, here is the file containing the GLIM code that does all the work for us. The great advantage is that we can get it right by editing the file. This allows us to retain the bits that were right, rather than run the risk of getting them wrong next time we type them at the terminal.

```
<<<  The file starts on the next line, and is called “RESYNTH.CON”  >>>
$inp 11 $
$units 100 $
$data a b c d e f g h i j k l $dinp 14 $
$data m n o p q r s t u v w x $dinp 15 $
$mac tax_ tx1 tx2 tx3 $endmac
$mac tx1 a b c d e f g h i $endmac
$mac tx2 j k l m n o p q $endmac
$mac tx3 r s t u v w x $endmac
$use mph_ $
$use who_ $
$del a b c d e f g h i j k l m n o p q r s t u v w x $
$data f g p q $dinp 16 $
$mac ff f2 f3 f4 f5 $endmac
$del f2 f3 f4 f5 $
$use exf_ f ff $
$mac gf g2 g3 g4 g5 $endmac
$use exf_ g gf $
$mac i i2 i3 i4 i5 $endmac
$mac i2 i22 i23 i24 i25 $endmac
$mac i3 i32 i33 i34 i35 $endmac
$mac i4 i42 i43 i44 i45 $endmac
$mac i5 i52 i53 i54 i55 $endmac
$use iff_ f g i $
$ca spi_=1 $
$ca opt_(3)=opt_(6)=opt_(9)=opt_(10)=0 $
$dump $
$return $
<<<  That was the control file “RESYNTH.CON”  >>>
```

Next, here is the GLIM session that uses the control file.

```
[o] GLIM 3.77 update 2 (copyright)1985 Royal Statistical Society, London
[o]                                        !Choose a different number from any of the
[i] ? $inp 29 $                            ! numbers used within the control file.
[i] File name? resynth.con                 !The name of the control file
[i] File name? phylo.glm                   !You need to remember the order in which the
                                           ! control file calls on other files.  It first
                                           ! wants the program file.
This program implements the phylogenetic regression (see A. Grafen 1989 Phil
Trans R Soc Lond B).  The accompanying documentation is intended to be
complete.  Both may be copied and distributed provided no charge is made.  This
is a trial version only.
                                           !Of course, YOU will have the final version
                        Copyright Alan Grafen 1989.



[i] File name? phylog1.dat                 !Next the first phylogeny file
[i] File name? phylog2.dat                 ! then the second phylogeny file
[o]                                        !Then, the output from the control file’s
[o]                                        ! call of WHO_.
[o] The first column tp__ is the name of a higher node, which is identified by
[o] giving one species that belongs to that node (in tp1_) and one species that
[o] "just" fails to belong to it (in tp2_).
[o]
[o]       TP__     TP1_     TP2_
[o]   1   101.0    1.000    3.000
[o]   2   102.0    3.000    1.000
<< Units 3 to 70 deleted to save space >>
[o]  71   171.0    9.000    5.000
[o]  72   172.0    5.000    1.000
```

```
[o]  73   173.0    1.000    0.000
[w] -- $data list abolished
[i] File name? data.dat                    !Then it wants the data file
[w] -- program dump completed              !Wisely, you get the control file to $DUMP
[i] ? $mac yv_ g $endmac                   !Now define the y-, control and test variables.
[i] ? $mac c_ p $endmac
[i] ? $mac t_ q $endmac
[i] ? $ca opt_(21)=opt_(22)=1 $            !Ask to interrupt at Places 1 and 2
[i] ? $use go_ $                           ! and off we go_
[o]
[o] ******************************************  !This is the dire warning
[o]
[o] You have asked to interrupt the program.  Take care.  It is safe to LOOK and
[o] PRINT, and not much else.  You are strongly advised to consult the manual about
[o] the state of execution of the program.  Do RETURN to continue execution.
[o]
[o] You have interrupted at Place 2.
[o]   The Places are:
[o]      Place 1.  just after LONG regression on CONTROL only.
[o]      Place 2.  just after LONG regression on CONTROL and TEST.
[o]      Place 3.  just after SHORT regression on CONTROL only.
[o]      Place 4.  just after SHORT regression on CONTROL and TEST.
[o]
[o] %FV (the fitted values, but see the manual) and your own model variables may be
[o] of interest.  Note that these will always have the length of the LONG
[o] regression.  Only the first sc__(3) values of the vectors are used in the SHORT
[o] regression.  So do for example "CALC %A=SC__(3)" and "LOOK 1 %A WSH_ Y X1 X2
[o] %FV" to see the first sc__(3) values only (the rest are effectively garbage).
[o]
[o] ****************************************
[o]                                        !We copy all the elements in the long
[i] ? $ass lg=g $                          ! regression.  Not only the variables
[i] ? $ass lfv2=%fv $                      ! (g, p and q) but also the node names
[i] ? $ass lp=p $                          ! (on_) and the regression weights (wl__).
[i] ? $ass lq=q $                          ! Also the fitted values (%fv)
[i] ? $ass ln=on_ $
[i] ? $ass lw=wl__ $
[i] ? $ret $                               !Then $RETURN control to the program
[o]
[o] ******************************************  !Dire warnings again
[o]
[o] You have asked to interrupt the program.  Take care.  It is safe to LOOK and
[o] PRINT, and not much else.  You are strongly advised to consult the manual about
[o] the state of execution of the program.  Do RETURN to continue execution.
[o]
[o] You have interrupted at Place 1.
[o]   The Places are:
[o]      Place 1.  just after LONG regression on CONTROL only.
[o]      Place 2.  just after LONG regression on CONTROL and TEST.
[o]      Place 3.  just after SHORT regression on CONTROL only.
[o]      Place 4.  just after SHORT regression on CONTROL and TEST.
[o]
[o] %FV (the fitted values, but see the manual) and your own model variables may be
[o] of interest.  Note that these will always have the length of the LONG
[o] regression.  Only the first sc__(3) values of the vectors are used in the SHORT
[o] regression.  So do for example "CALC %A=SC__(3)" and "LOOK 1 %A WSH_ Y X1 X2
[o] %FV" to see the first sc__(3) values only (the rest are effectively garbage).
[o]
[o] ****************************************
[o]
[i] ? $ass lfv1=%fv $                      !The only thing that's different is the
[i] ? $ret $                               ! fitted values, so collect them too,
[o]                                        ! and $RETURN control to the program
[o]
[o]
[o]
[o]
[o]      y-variable: g
[o] Controlling for: p
```

```
[o]     Testing for: q
[o]                                         !You can see how much the program monkeys
[o] F      = 0.602974                       ! around with your variables by printing g and
[o]  1,70                                   ! lg.  g has been returned to its original
[o]                                         ! state.  lg shows what g was like during the
[i] ? $pri g $                              ! long regression.
[o]    1.000   2.000   3.000   4.000   5.000   5.000   1.000   2.000   3.000
[o]    4.000   5.000   5.000   1.000   2.000   3.000   4.000   5.000   5.000
[o]    1.000   2.000   3.000   4.000   5.000   5.000   1.000   2.000   3.000
[o]    4.000   5.000   5.000   1.000   2.000   3.000   4.000   5.000   5.000
[o]    1.000   2.000   3.000   4.000   5.000   5.000   1.000   2.000   3.000
[o]    4.000   5.000   5.000   1.000   2.000   3.000   4.000   5.000   5.000
[o]    1.000   2.000   3.000   4.000   5.000   5.000   1.000   2.000   3.000
[o]    4.000   5.000   5.000   1.000   2.000   3.000   4.000   5.000   5.000
[o]    1.000   2.000   3.000   4.000   5.000   5.000   1.000   2.000   3.000
[o]    4.000   5.000   5.000   1.000   2.000   3.000   4.000   5.000   5.000
[o]    1.000   2.000   3.000   4.000   5.000   5.000   1.000   2.000   3.000
[o]    4.000
[i] ? $pri lg $
[o]  -0.5000  0.5000 -0.5000  0.5000  0.      0.     -0.5000  0.5000 -0.5000
[o]   0.5000  0.      0.     -0.5000  0.5000 -0.5000  0.5000  0.      0.
[o]  -0.5000  0.5000 -0.5000  0.5000  0.      0.     -0.5000  0.5000 -0.5000
[o]   0.5000  0.      0.     -0.5000  0.5000 -0.5000  0.5000  0.      0.
[o]  -0.5000  0.5000 -0.5000  0.5000  0.      0.     -0.5000  0.5000 -0.5000
[o]   0.5000  0.      0.     -0.5000  0.5000 -0.5000  0.5000  0.      0.
[o]  -0.5000  0.5000 -0.5000  0.5000  0.      0.     -0.5000  0.5000 -0.5000
[o]   0.5000  0.      0.     -0.5000  0.5000 -0.5000  0.5000  0.      0.
[o]  -0.5000  0.5000 -0.5000  0.5000  0.      0.     -0.5000  0.5000 -0.5000
[o]   0.5000  0.      0.     -0.5000  0.5000 -0.5000  0.5000  0.      0.
[o]  -0.5000  0.5000 -0.5000  0.5000  0.      0.      -1.500 -0.5000  0.5000
[o]    1.500  -1.717  0.2833   1.621  -1.879  0.0916   1.592  -1.709  0.2911
[o]    1.619  -1.881  0.0859   1.586  -1.699  0.3007   1.616  -1.884  0.0788
[o]    1.579  -1.687  0.3130   1.612  -1.888  0.0697   1.570  -1.670  0.3297
[o]    1.607  -1.893  0.0578   1.558  -1.646  0.3545   1.603  -1.897  0.0418
[o]    1.542  -1.603  0.3970   1.602  -1.898  0.0218   1.522  -1.504  0.4956
[o]    1.633  -1.867  0.0279   1.528 -0.9721  0.1048  0.3629 -0.4738  0.0802
[o]   0.2950 -0.3552  0.0612  0.2515 -0.2967  0.0496  0.2223 -0.2600  0.0421
[o]   0.2012 -0.2342  0.0367  0.1851 -0.2148  0.0328  0.1724 -0.1995  0.0298
[o]   0.1619
[i] ? $ca lr1=(lg-lfv1)*%sqrt(wl__) $       !Calculate standardized residuals from the
[i] ? $ca lr2=(lg-lfv2)*%sqrt(wl__) $       ! long regression on control only (lr1), and on
[i] ? $loo lg lp lq lw ln lr1 lr2 $         ! control and test variables (lr2), then LOOK.
[o]          LG         LP         LQ         LW       LN        LR1        LR2
[o]    1  -0.50000  -0.049050  -0.2255500   9.398     1.000  -1.58703  -1.7705522
[o]    2   0.50000   0.049050   0.2255500   9.398     2.000   1.58703   1.7705522
[o]    3  -0.50000  -0.026250  -0.2529500   9.398     3.000  -1.56183  -1.7685590
[o]    4   0.50000   0.026250   0.2529500   9.398     4.000   1.56183   1.7685590
[o]    5   0.00000   0.009600  -0.0135000   9.398     5.000   0.01061  -0.0007717
[o]    6   0.00000  -0.009600   0.0135000   9.398     6.000  -0.01061   0.0007717
<< Units 7 to 167 deleted to save space >>
[o]  168   0.03280  -0.002017   0.0147918   9.263   168.000   0.09760   0.1097185
[o]  169   0.17238  -0.012547   0.0673762   9.286   169.000   0.51151   0.5668634
[o]  170  -0.19947   0.018671  -0.0434557   9.308   170.000  -0.58803  -0.6241107
[o]  171   0.02975   0.002640   0.0417148   9.329   171.000   0.09378   0.1278055
[o]  172   0.16194   0.025925  -0.0006480   9.349   172.000   0.52372   0.5223693
[i] ? $ca opt_(21)=opt_(22)=0 $             !Cancel interruptions during long regression
[i] ? $ca opt_(23)=opt_(24)=1 $             !Switch on short regression interruptions
[i] ? $ca opt_(20)=1 $                      !Switch off the dire warnings.
[i] ? $use go_ $
[o]
[o] You have interrupted at Place 3.        !Much more civilized!
[i] ? $ca %a=sc__(3) $                      !The number of elements in the short regression
[i] ? $var %a sg sp sq wsh sn sfv1 sfv2 $   !Define the vectors we want to remember
[i] ? $ca sg=g(%cu(1)) $                    !Transfer the relevant values of variables in
[i] ? $ca sp=p(%cu(1)) $                    ! the short regression into our new variables
[i] ? $ca sq=q(%cu(1)) $                    !Take the regression weights (wsh) in
[i] ? $ca wsh=wsh_(%cu(1)) $                ! case any datapoints have been omitted for
[i] ? $ca sfv1=%fv(%cu(1)) $                ! lacking a phylogenetic degree of freedom
[i] ? $ret $                                ! as well as the variables (g, p and q) and
```

```
[o]                                        ! the fitted values (sfv1).  Then $RETURN
[o] You have interrupted at Place 4.       !The next interruption
[i] ? $ca sn=l3__(%cu(1)) $                !Take the node names (kept in l3__, available
[i] ? $ca sfv2=%fv(%cu(1)) $               ! only at Place 4, and the fitted values (sfv2)
[i] ? $ret $                               !Then $RETURN
[o]
[o]
[o]
[o]
[o]
[o]      y-variable: g
[o] Controlling for: p
[o]     Testing for: q
[o]
[o] F      = 0.602974                      !See how the short regression values are
[o]  1,70                                  ! quite different from the species values (g)
[o]                                        ! and long regression values (lg) printed out
[i] ? $pri sg $                            ! above.
[o]    2.168   2.168  0.       2.168   2.168  0.       2.168   2.168  0.
[o]    2.168   2.168  0.       2.168   2.168  0.       2.168   2.168  0.
[o]    2.168   2.168  0.       2.168   2.168  0.       2.168   2.168  0.
[o]    2.168   2.168  0.       2.168   2.168  0.       2.168   2.168  0.
[o]    2.168   2.168  0.       2.168   2.168  0.       2.168   2.168  0.
[o]    2.168   2.168  0.       5.233   4.165   4.620   2.885   2.781   3.760
[o]    2.541   2.389   3.356   2.330   2.168   3.100   2.180   2.037   2.920
[o]    2.071   1.931   2.780   1.986   1.849   2.671   1.914   1.781   2.579
[o]    1.851
[i] ? $ca sr1=sg-sfv1 $                    !Calculate the residuals.  No standardization
[i] ? $ca sr2=sg-sfv2 $                    ! needed.  LOOK at the short regression
[i] ? $loo sg sp sp wsh sn sr1 sr2 $       ! variables, including node names and weights
[o]        SG       SP        SP       WSH     SN       SR1       SR2
[o]   1   2.168   0.21265   0.21265   1.000   101.0   2.24440   2.521248
[o]   2   2.168   0.11381   0.11381   1.000   102.0   2.20876   2.520882
<<< Units 3 to 70 deleted to save space >>>
[o]  71   1.781  -0.12117  -0.12117   1.000   171.0   1.73775   1.852230
[o]  72   2.579  -0.38009  -0.38009   1.000   172.0   2.44205   2.404704
[o]  73   1.851  -0.12793  -0.12793   1.000   173.0   1.80455   1.730167
[i] ? $macro lget $                        !You can define a macro to do your collecting
[i] $MAC? $ass lg=g:lp=p:lq=q:ln=on_:lw=wl__:lfv2=%fv $   ! for you.  Here I define lget.
[i] $MAC? $$endmac                         !Nothing happens when I define the macro.
[i] ? $ca spi_=%lr(1) $                    !Let's omit some species, to illustrate the
[i] ? $pri spi_ $                          ! importance of node name vectors and weights
[o]   0.      0.      0.      0.       1.000   1.000  0.      0.      0.      0.
[o]   0.       1.000   1.000  0.      0.       1.000   1.000   1.000  0.      0.
[o]   0.      0.       1.000   1.000   1.000  0.       1.000  0.      0.      0.
[o]    1.000  0.       1.000   1.000   1.000  0.      0.       1.000   1.000  0.
[o]    1.000  0.      0.       1.000   1.000  0.      0.      0.       1.000  0.
[o]    1.000   1.000  0.       1.000   1.000  0.       1.000  0.       1.000
[o]    1.000  0.       1.000   1.000   1.000  0.       1.000   1.000   1.000  0.
[o]    1.000  0.      0.      0.       1.000  0.       1.000   1.000  0.
[o]    1.000   1.000   1.000  0.       1.000   1.000  0.      0.      0.
[o]    1.000  0.       1.000   1.000   1.000  0.       1.000   1.000  0.      0.
[o]    1.000   1.000  0.
[i] ? $ca opt_(23)=opt_(24)=0 $            !Cancel short regression interrupts, and
[i] ? $ca opt_(22)=1 $                     ! ask to interrupt at Place 2.
[i] ? $use go_ $
[o]
[o] You have interrupted at Place 2.
[i] ? $use lget $                          !Now I use the macro LGET, and its instructions
[i] ? $ret $                               ! are carried out now.  Then I $RETURN
[o]
[o]
[o]
[o]                                        !Below, we $LOOK at the variables relevant to
[o]                                        ! the long regression.  Notice that LW is zero
[o]      y-variable: g                     ! for omitted species.  LN, containing the
[o] Controlling for: p                     ! of the nodes, starts skipping nodes after
[o]     Testing for: q                     ! 100, corresponding to the nodes in the
[o]                                        ! original phylogeny that no longer exist as
```

```
[o] F      = 2.65169                        ! nodes once some species are omitted.
[o]  1,32
[o]
[i] ? $loo lg lp lq ln lw lfv2 $
[o]           LG          LP          LQ          LN       LW       LFV2
[o]    1   1.0000000   0.5889000   0.5248000     1.000   0.000  -0.698465
[o]    2   2.0000000   0.6870000   0.9759000     2.000   0.000  -1.188888
[o]    3   3.0000000   0.1397000   0.0065000     3.000   0.000  -0.044325
[o]    4   4.0000000   0.1922000   0.5124000     4.000   0.000  -0.578828
[o]    5   0.0000000   0.0096000  -0.0135000     5.000   8.242   0.011299
[o]    6   0.0000000  -0.0096000   0.0135000     6.000   8.242  -0.011299
[o]    7   1.0000000   0.9467000   0.1551000     7.000   0.000  -0.414602
[o]    8   2.0000000   0.6375000   0.7250000     8.000   0.000  -0.917481
[o]    9   3.0000000   0.1528000   0.1934000     9.000   0.000  -0.240096
[o]   10   4.0000000   0.8238000   0.9187000    10.000   0.000  -1.166911
[o]   11   5.0000000   0.5979000   0.9073000    11.000   0.000  -1.094321
[o]   12   1.6594183  -0.2731842   0.4045198    12.000   1.022  -0.342476
<<< Units 13 to 95 deleted to save space >>>
[o]   96   5.0000000   0.9701000   0.8014000    96.000   0.000  -1.085677
[o]   97   1.0000000   0.3259000   0.5418000    97.000   0.000  -0.645091
[o]   98  -0.5000000  -0.2918000   0.0920500    98.000   8.242  -0.016061
[o]   99   0.5000000   0.2918000  -0.0920500    99.000   8.242   0.016061
[o]  100   4.0000000   0.2398000   0.6785000   100.000   0.000  -0.762499
[o]  101   1.4636893  -0.2595734  -0.2468683   103.000   1.065   0.323860
[o]  102   1.6777301  -0.1205478   0.0021186   109.000   1.131   0.030300
[o]  103   1.9209008   0.0877799   0.0796757   112.000   1.161  -0.105603
[o]  104   0.2063863   0.0498690   0.0596577   117.000   1.247  -0.074799
[o]  105   0.4570236  -0.2964752  -0.1640052   126.000   1.431   0.248572
[o]  106   1.5634167  -0.1576955  -0.2252229   130.000   1.594   0.274147
[o]  107   0.3288188  -0.0954850   0.1386416   132.000   1.719  -0.116877
[o]  108  -1.8502648   0.1619604   0.1838154   134.000   1.877  -0.232706
[o]  109  -2.0081880   0.1560185   0.1534492   140.000   2.364  -0.199871
[o]  110   1.2433677   0.0585627   0.1740348   142.000   2.784  -0.194787
[o]  111  -1.8525894   0.1364181   0.0910217   146.000   3.874  -0.130378
[o]  112  -1.1826642  -0.1686479  -0.1256876   149.000   6.300   0.174718
[o]  113   0.3300748   0.0812661   0.0547594   150.000   6.087  -0.078220
[o]  114  -0.1494565   0.0002728   0.0031577   151.000   9.618  -0.003321
<<< Units 115 to 132 omitted to save space
[o]  132   0.1787577  -0.0267293   0.0940413   169.000   7.444  -0.089527
[o]  133  -0.1970694   0.0324429  -0.0480400   170.000   8.608   0.040672
[o]  134  -0.1957290   0.0347108   0.0330119   171.000   7.961  -0.043307
[i] ? $macro sget $                          !Define a macro to grab info about short
[i] $MAC? $del sg sp sq sn sw sfv2 sr2 $    ! regression, including weights and node
[i] $MAC? $ca %a=sc__(3) $var %a sg sp sq sn sw sfv2 sr2 $       ! names.  Also
[i] $MAC? $ca sg=g(%cu(1)) :sp=p(%cu(1)) :sq=q(%cu(1)) $         ! $LOOK at them there
[i] $MAC? $ca sn=l3__(%cu(1)) :sw=wsh_(%cu(1)) $                 ! and then.
[i] $MAC? $ca sfv2=%fv(%cu(1)) :sr2=sg-sfv2 $
[i] $MAC? $loo sg sp sq sn sw sfv2 sr2 $
[i] $MAC? $$endmac
[i] ? $ca opt_(21)=opt_(22)=0 $              !Cancel interruptions of long regression
[i] ? $ca opt_(24)=1 $                       !Call for interruption at Place 4
[i] ? $use go_ $                             ! and go_
[o]
[o] You have interrupted at Place 4.
[i] ? $use sget $                            !$USE the macro we defined
[o]          SG          SP          SQ        SN       SW       SFV2        SR2
[o]    1   0.00000   0.038977  -0.054812   103.0   1.000   0.05890  -0.05890
[o]    2   0.00000   1.591371   0.013804   109.0   1.000  -0.38756   0.38756
[o]    3   0.00000   0.711133  -0.300247   112.0   1.000   0.20678  -0.20678
[o]    4   2.03007  -0.555021  -0.861967   117.0   1.000   1.19809   0.83198
[o]    5   2.03007  -0.079579   0.463262   126.0   1.000  -0.55594   2.58601
[o]    6   0.00000   0.988035  -0.044662   130.0   1.000  -0.17462   0.17462
[o]    7   2.03007  -0.248887  -1.057666   132.0   1.000   1.36950   0.66057
[o]    8   2.03007  -1.677649  -0.523758   134.0   1.000   1.04002   0.99005
[o]    9   2.03007   1.186372  -0.456563   140.0   1.000   0.28999   1.74008
[o]   10   0.00000   1.288079  -1.061523   142.0   1.000   1.01650  -1.01650
[o]   11   2.03007  -1.207485  -1.511996   146.0   1.000   2.15604  -0.12597
[o]   12   2.03007   1.184748  -0.373736   149.0   1.000   0.18766   1.84241
[o]   13   4.13740   0.323761   0.583949   150.0   1.000  -0.79949   4.93689
```

```
[o]  14   4.73855  -0.515746  -0.344967   151.0   1.000   0.54784   4.19071
[o]  15   0.96467  -0.001761  -0.020381   152.0   1.000   0.02568   0.93899
[o]  16   2.46752   0.091116   0.146495   153.0   1.000  -0.20287   2.67039
[o]  17   3.81183  -0.460971  -0.345599   154.0   1.000   0.53587   3.27597
[o]  18   2.11899   0.575512  -0.738340   155.0   1.000   0.78161   1.33738
[o]  19   0.82076  -0.068413  -0.192432   156.0   1.000   0.25455   0.56621
[o]  20   3.32724  -0.112646  -0.461956   157.0   1.000   0.59907   2.72817
[o]  21   1.59571  -0.283474   0.290547   158.0   1.000  -0.29431   1.89001
[o]  22   2.15463  -0.225203  -0.315690   159.0   1.000   0.44390   1.71073
[o]  23   3.39159  -0.114891  -0.170231   160.0   1.000   0.23784   3.15375
[o]  24   2.51930   0.042571  -0.028997   161.0   1.000   0.02605   2.49325
[o]  25   0.04567   0.132690  -0.490058   162.0   1.000   0.57681  -0.53114
[o]  26   2.45567   0.554888   0.343273   163.0   1.000  -0.55485   3.01051
[o]  27   1.17775   0.245466  -0.083549   164.0   1.000   0.04646   1.13129
[o]  28   2.02087   0.426434  -0.060382   165.0   1.000  -0.02439   2.04526
[o]  29   2.30942  -0.142142   0.244718   166.0   1.000  -0.27037   2.57980
[o]  30   2.02221  -0.262092  -0.002819   167.0   1.000   0.06451   1.95770
[o]  31   2.20929   0.100959   0.091638   168.0   1.000  -0.13714   2.34643
[o]  32   1.90924  -0.137182   0.002411   169.0   1.000   0.02894   1.88029
[o]  33   2.39590   0.183538   0.221884   170.0   1.000  -0.31787   2.71377
[o]  34   1.77461  -0.292148   0.432601   171.0   1.000  -0.46844   2.24305
[o]  35   1.60802  -0.285169  -0.271211   172.0   1.000   0.40270   1.20532
[i] ? $ret $                              !The values above give the variables that
[o]                                       ! were used in the short regression.  If
[o]                                       ! any phylogenetic degrees of freedom had
[o]                                       ! been lost in the denominator, they would
[o]                                       ! have shown up as sw==0.
[o]
[o]      y-variable: g
[o] Controlling for: p
[o]     Testing for: q
[o]
[o] F      = 2.65169
[o]  1,32
[o]
[i] ? $use go_ $                          !One last trick to demonstrate.  If you
[o]                                       ! forget that you are in the middle of an
[o] You have interrupted at Place 4.      ! interruption, and try to $USE GO_ again,
[i] ? $use go_ $                          ! I cunningly prevent you.
[o]
[o] You may not USE GO_ while you are interrupting a previous GO_.  If you have
[o] done no irreperable damage, reset everything as it was at the start of the
[o] interrupted GO_, and RETURN.  Otherwise, it is probably best to restart the
[o] session.
[i] ? $ret $                              !We hadn't altered yv_, or con_ or tst_
[o]                                       ! or spi_ or anything important.  So we can
[o]                                       ! just $RETURN and all will be OK.
[o]
[o]
[o]
[o]      y-variable: g
[o] Controlling for: p
[o]     Testing for: q
[o]
[o] F      = 2.65169
[o]  1,32
[o]
[i] ? $stop                               !       Happy hunting!
```

9 Draft Letter to Computing Service

This is a draft of a letter you might send to your computing service, to ask them to mount a more useful version of GLIM.

Dear System Manager,

I would be grateful if you would pass this letter on to the person in charge of the implementation of the statistical package GLIM.

My research requires the application of a specialized statistical technique, called the "Phylogenetic Regression". The software implementing this technique is distributed free and is written in GLIM 3.77. The program can be used for small analyses with the standard implementation of GLIM, but a limit is imposed by the number of user identifiers permitted. In the standard implementation, this is 100. GLIM is supplied in source form to mainframe purchasers, and the implementation notes accompanying GLIM explain how to modify the source to relax this limit.

It would be of considerable help to me in my research if a version of GLIM could be made available which allowed 200 user identifiers. I hope very much that this will be possible. Would you be so kind as to let me know?

Yours faithfully,

10 REVISION HISTORY

Version 1 of the Phylogenetic Regression was released to the world in January 1990. All previous versions were trial versions only. Too many bugs and problems were corrected between trial versions and Version 1 to be worth listing.

Version 1.01 of the Phylogenetic Regression was released in May 1990, in response to a problem reported by Dr William Kirk. The macro MPH_ produced an incorrect phylogeny from taxonomic levels vectors in a special combination of circumstances. The main ingredient was that a taxon should be unchanged in membership for at least three taxonomic levels (for example a monospecific family). This showed itself in a GLIM error when WHO_ was called to establish the identity of higher nodes. If you call WHO_ in Version 1 and don't get the error, then the bug has not struck. Version 1.01 solves the problem completely, and makes no other changes.

The manual was revised in August 1990, with small changes. i) The description of the example phylogenetic tree in section 3.1 was corrected. ii) The method of performing an analysis with fixed rho did not work, and has been amended. This method is described in the separate file Technical Details, under the description of the uses of elements 8, 10, 24 and 25 of the vector sc__ in section 6. iii) This section was added to contain the revision histories of the program and manual.

Version 1.02 of the Phylogenetic Regression was released in October 1990, in response to requests from Dr William Kirk. Two facilities were added. 1) OPT_(16) now causes the program to print out the (phylogenetically efficiently weighted) means of all the variables in the analysis. This allows the parameter estimates from the long regression to be used to construct a best fit equation. 2) OPT_(17) now short-circuits the fitting of rho, and instead uses whatever (non-zero) value is found in OPT_(17). This second facility improves and replaces the previous method of fixing rho.

The manual and technical details were revised in October 1990 to describe the new facilities. Also, the address of NAG's North American office is given.

Version 1.03 was released in March 1991, in response to a potentially serious problem reported by Dr William Kirk (again!). The macro MPH_ creates the phylogeny from taxonomic levels vectors. If the species were arranged in a regular order, so that species from the same genus were all together, and genera in the same family were all together, and so on, then MPH_ worked correctly. When species from different genera were interspersed, however, MPH_ worked incorrectly: although it worked out the correct phylogeny, it did so for the species in a different order to the one in which they appeared in the datafile. This makes nonsense of any analyses performed. The problem has been fixed, and MPH_ now works correctly for arbitrary ordering of the species.

Very minor alterations to the manual were also made in March 1991, and the area code for the North American distributors was updated in August 1992.