APPENDIX B: TUTORIAL ON USING THE REO PACKAGE

STEVEN C. WALKER AND DONALD A. JACKSON

Contents

I. Introduction

In this appendix, we illustrate the use of our R package `reo`—for random effects ordination. The source code and manual for the functions in this package is available in Supplementary material 2. The easiest way to get using `reo` is simply to paste the source code into an R prompt. A tar archive of the package is also available from the first author. This archive can be incorporated into the package library of R installations on unix-based machines (e.g. linux; mac), using the following command in the shell / terminal: `R CMD INSTALL reo_1.0.tar.gz`. Unfortunately, a zip file is not yet available for Windows R installations. However, we plan to put our package on CRAN shortly, which will allow it to be loaded onto any (sufficiently recent) R installation. The package has two main functions: `factanal.predictive` for continuous data and `ltm.ecol` for presence / absence data. All of the data we use is available in Supplementary material 1, making our analyses fully replicable.

II. Using factanal.predictive

We begin by loading the `reo` library.

```
> library(reo)
```

which contains the `limn` data set that we used in our paper,

```
> limn
             area maxd  vol shore elev   pH   ca cond
3 Island     1.36 0.85 0.88  0.43 2.56 6.09 0.35 1.34
Austin       1.28 0.41 0.70  0.57 2.53 5.60 0.38 1.34
Bear         1.98 1.56 1.95  0.99 2.55 6.40 0.43 1.48
Bentshoe     1.23 0.93 0.84  0.60 2.53 5.59 0.39 1.51
Big East     2.18 1.51 1.95  1.34 2.51 5.70 0.44 1.56
Big Orillia  1.66 1.08 1.37  0.58 2.51 6.10 0.41 1.46
Bloody       1.26 0.79 1.55  0.28 2.61 6.61 0.46 1.49
Blue Chalk   1.70 1.34 1.67  0.65 2.53 6.50 0.43 1.45
Brady        1.95 1.06 1.61  0.98 2.52 6.21 0.57 1.53
```

```
Buchanan      0.95 1.10 0.64  0.20 2.61 6.06 0.38 1.41
Cinder        1.89 1.56 1.81  1.06 2.52 5.35 0.33 1.38
Clayton       1.00 0.91 0.43  0.26 2.60 5.40 0.37 1.41
Crosson       1.76 1.37 1.68  0.59 2.53 5.91 0.31 1.37
Dan           1.23 1.19 1.00  0.32 2.53 5.89 0.38 1.41
Ernest        1.04 0.70 0.28  0.36 2.57 5.25 0.33 1.35
Fletcher      2.41 1.37 2.31  0.48 2.61 6.70 0.44 1.51
Grindstone    1.51 1.44 1.55  0.59 2.55 6.27 0.43 1.69
Gullfeather   1.84 1.11 1.52  0.72 2.51 5.51 0.34 1.39
Harvey        1.75 1.14 1.39  0.68 2.55 6.39 0.45 1.48
Herb          1.76 1.20 1.39  0.88 2.57 5.12 0.33 1.34
Jill          1.04 0.57 0.38  0.32 2.51 6.20 0.53 1.49
Kawagama      3.45 1.86 3.79  2.02 2.53 6.00 0.46 1.48
Kimball       2.33 1.83 2.68  0.99 2.55 6.81 0.42 1.47
L.Fletcher    1.79 1.48 1.90  0.23 2.61 6.56 0.43 1.49
L.Louie       1.04 0.74 0.28  0.04 2.60 6.52 0.54 1.57
L.Orillia     1.41 0.83 0.95  0.53 2.51 5.81 0.42 1.46
L.Troutspawn 1.51 1.38 1.40  0.41 2.61 6.83 0.52 1.53
L.Wren        1.20 1.09 0.82  0.49 2.54 5.80 0.43 1.48
Livingstone   2.28 1.56 2.38  0.94 2.58 6.56 0.47 1.52
Louie         1.49 1.63 1.72  0.43 2.60 6.50 0.53 1.54
McDonald      1.40 0.48 0.48  0.40 2.54 5.20 0.38 1.35
McFadden      1.73 1.48 1.80  0.46 2.60 6.00 0.51 1.56
McKeown       1.38 1.09 1.00  0.52 2.54 5.71 0.37 1.41
Millichamp    1.00 0.90 0.59  0.15 2.63 6.03 0.35 1.40
Poker         1.32 1.31 1.12  0.58 2.53 5.99 0.39 1.43
Poorhouse     1.48 1.12 1.09  0.49 2.62 6.90 0.49 1.62
Porcupine     1.75 1.34 1.47  0.72 2.51 5.90 0.45 1.45
Raven         2.75 1.62 2.66  1.61 2.54 6.32 0.37 1.43
Redchalk      1.76 1.51 1.80  0.69 2.53 6.43 0.41 1.48
Ridout        1.67 1.31 1.50  0.80 2.54 5.82 0.40 1.44
S.McDonald    1.08 0.81 0.56  0.20 2.54 5.60 0.37 0.95
Saucer        0.78 0.88 0.38  0.00 2.53 5.75 0.47 1.47
Shoe          1.59 1.23 1.33  0.72 2.55 6.02 0.40 1.45
South Jean    1.81 1.09 1.37  0.88 2.53 6.00 0.40 1.41
Sugarbowl     0.85 1.00 0.34  0.00 2.51 5.80 0.53 1.47
Sunken        1.11 0.74 0.48  0.49 2.56 5.81 0.33 1.35
Teapot        1.53 1.03 1.13  0.67 2.51 6.00 0.46 1.48
Tingey        1.18 0.86 0.95  0.36 2.55 4.79 0.36 1.39
Troutspawn    2.00 1.14 1.71  0.84 2.61 6.31 0.45 1.48
Wolf          1.97 1.37 1.75  1.00 2.69 6.50 0.38 1.42
Wren          1.70 0.99 1.24  0.75 2.54 5.88 0.39 1.46
Wrist         1.58 1.32 1.34  0.68 2.53 5.06 0.34 1.36
```

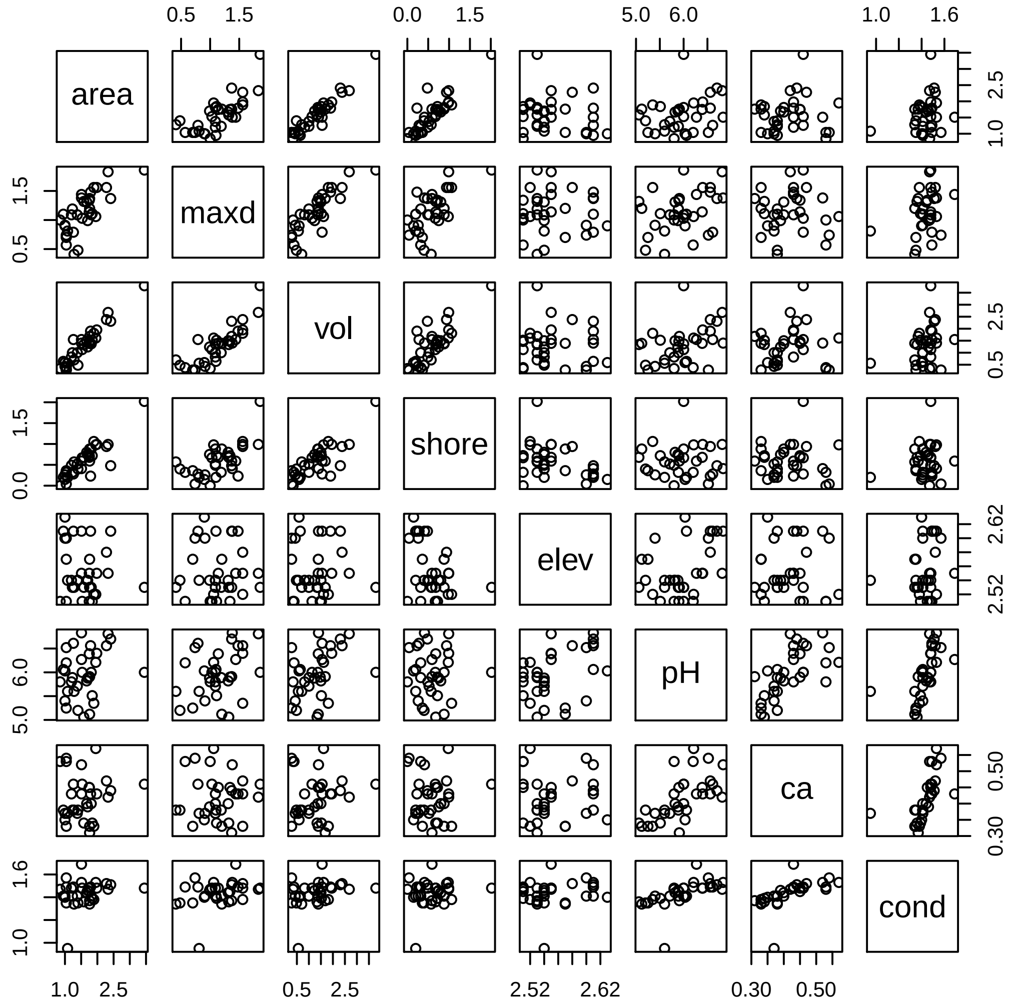
Note that if the source code is used (as opposed to the archive), then these data will have to be entered manually. We then split `limn` into validation and training data,

```
> valid <- c(4, 35, 8, 42, 5, 38, 26, 1, 32, 6, 43, 30, 39, 46, 49,
+     50, 48, 36)
> train <- setdiff(1:52, valid)
> limn.v <- limn[valid, ]
> limn.t <- limn[train, ]
```

This is the validation-training split that we randomly generated for the example in our paper.

As these data are continuous we consider fitting a factor analysis model to the training lakes. But the factor analysis model assumes multivariate normality. A first graphical check of this assumption can be made by looking at all possible pair-wise scatterplots of the data.

```
> pairs(limn.t)
```



As the pair-wise relationships display some obvious departures from linearity in the form of outliers (e.g. a lake with unusually low conductivity), we conclude that some outlier removal will be necessary. We use the `pcout` function in the `mvoutlier` package for this purpose.

```
> library(mvoutlier)
> pcout(limn.t)

$wfinal01
     Austin        Bear      Bloody       Brady    Buchanan      Cinder
          1           1           1           1           1           1
    Clayton     Crosson         Dan      Ernest    Fletcher  Grindstone
          1           1           1           1           1           0
 Gullfeather      Harvey        Herb        Jill    Kawagama     Kimball
          1           1           1           1           0           1
  L.Fletcher     L.Louie L.Troutspawn     L.Wren Livingstone    McDonald
          1           1           1           1           1           1
    McKeown  Millichamp   Porcupine      Ridout  S.McDonald  South Jean
          1           1           1           1           0           1
  Sugarbowl      Teapot        Wren       Wrist
          1           1           1           1

$wfinal
     Austin        Bear      Bloody       Brady    Buchanan      Cinder
  0.8741966   1.0000000   0.9260756   0.3717855   0.9214016   0.8366668
    Clayton     Crosson         Dan      Ernest    Fletcher  Grindstone
  0.5453381   0.9427834   1.0000000   0.6053479   0.6277903   0.1348551
 Gullfeather      Harvey        Herb        Jill    Kawagama     Kimball
  0.9620812   1.0000000   0.6959360   0.5601415   0.1575620   0.8508286
  L.Fletcher     L.Louie L.Troutspawn     L.Wren Livingstone    McDonald
  0.8754219   0.9178537   0.7460058   1.0000000   0.9727006   0.9277552
    McKeown  Millichamp   Porcupine      Ridout  S.McDonald  South Jean
  1.0000000   0.4756180   1.0000000   1.0000000   0.0400000   1.0000000
  Sugarbowl      Teapot        Wren       Wrist
  0.5675057   1.0000000   1.0000000   0.7762452

$wloc
     Austin        Bear      Bloody       Brady    Buchanan      Cinder
  1.0000000   1.0000000   0.9900225   0.3760376   0.9440209   0.8012576
    Clayton     Crosson         Dan      Ernest    Fletcher  Grindstone
  0.5341955   0.9726624   1.0000000   0.5567913   0.7450923   0.0000000
 Gullfeather      Harvey        Herb        Jill    Kawagama     Kimball
  0.9526015   1.0000000   0.7504432   0.6479301   0.6230086   0.9412347
  L.Fletcher     L.Louie L.Troutspawn     L.Wren Livingstone    McDonald
  0.9403072   1.0000000   0.7696601   1.0000000   0.9688461   0.9999937
    McKeown  Millichamp   Porcupine      Ridout  S.McDonald  South Jean
  1.0000000   0.5796331   1.0000000   1.0000000   0.0000000   1.0000000
  Sugarbowl      Teapot        Wren       Wrist
  0.8022376   1.0000000   1.0000000   0.7250102

$wscat
```

```
      Austin         Bear       Bloody        Brady     Buchanan       Cinder
  0.84274579   1.00000000   0.91690874   0.67792334   0.95574941   0.99355046
     Clayton      Crosson          Dan       Ernest     Fletcher   Grindstone
  0.83657942   0.95482894   1.00000000   0.92236783   0.73576018   0.59284461
 Gullfeather       Harvey         Herb         Jill     Kawagama      Kimball
  1.00000000   1.00000000   0.83691834   0.72470964   0.03200246   0.86600151
  L.Fletcher      L.Louie L.Troutspawn       L.Wren  Livingstone     McDonald
  0.89915436   0.89731714   0.89315944   1.00000000   0.99695367   0.90969988
     McKeown   Millichamp    Porcupine       Ridout   S.McDonald   South Jean
  1.00000000   0.64576121   1.00000000   1.00000000   0.00000000   1.00000000
   Sugarbowl       Teapot         Wren        Wrist
  0.59270674   1.00000000   1.00000000   0.99396967

$x.dist1
      Austin         Bear       Bloody        Brady     Buchanan       Cinder
   1.8867659    1.1011326    2.0730172    3.5246848    2.3305407    2.7396414
     Clayton      Crosson          Dan       Ernest     Fletcher   Grindstone
   3.2530070    2.1957419    1.4087339    3.2136071    2.8609097    5.0070413
 Gullfeather       Harvey         Herb         Jill     Kawagama      Kimball
   2.2946601    0.9698303    2.8498319    3.0499349    3.0956451    2.3416160
  L.Fletcher      L.Louie L.Troutspawn       L.Wren  Livingstone     McDonald
   2.3452471    1.1933683    2.8092891    1.7104232    2.2167646    1.8914416
     McKeown   Millichamp    Porcupine       Ridout   S.McDonald   South Jean
   1.8021063    3.1733937    1.2273625    1.1697343    8.3365575    0.6634245
   Sugarbowl       Teapot         Wren        Wrist
   2.7374132    1.5362141    1.7464071    2.9017500

$x.dist2
      Austin         Bear       Bloody        Brady     Buchanan       Cinder
   2.5004983    1.1377209    2.3204768    2.8007545    2.1939609    1.9860505
     Clayton      Crosson          Dan       Ernest     Fletcher   Grindstone
   2.5135362    2.1974707    1.5347777    2.3047240    2.7040453    2.9337612
 Gullfeather       Harvey         Herb         Jill     Kawagama      Kimball
   1.7179099    0.8957770    2.5128251    2.7230470    3.8898589    2.4492071
  L.Fletcher      L.Louie L.Troutspawn       L.Wren  Livingstone     McDonald
   2.3686444    2.3733970    2.3840112    1.1808021    1.9461722    2.3405580
     McKeown   Millichamp    Porcupine       Ridout   S.McDonald   South Jean
   1.0084458    2.8520737    1.3107836    0.7115855    5.0423570    0.9960534
   Sugarbowl       Teapot         Wren        Wrist
   2.9339704    1.4787665    1.3242796    1.9818342

$M1
33.33333%
 1.886766
```

```
$const1
[1] 4.520427

$M2
[1] 1.858655

$const2
[1] 4.100231
```

This function gives a great deal of output, allowing the analyst to explore the outlier structure of the data in detail. Here we focus on scatter outliers (component `wscat`; see main text for justification) and therefore further separate the training data into an outlier and non-outlier component as follows.

```
> not.out <- which(pcout(limn.t)$wscat > 0.25)
> limn.t.out <- limn.t[setdiff(1:nrow(limn.t), not.out), ]
> limn.t <- limn.t[not.out, ]
```

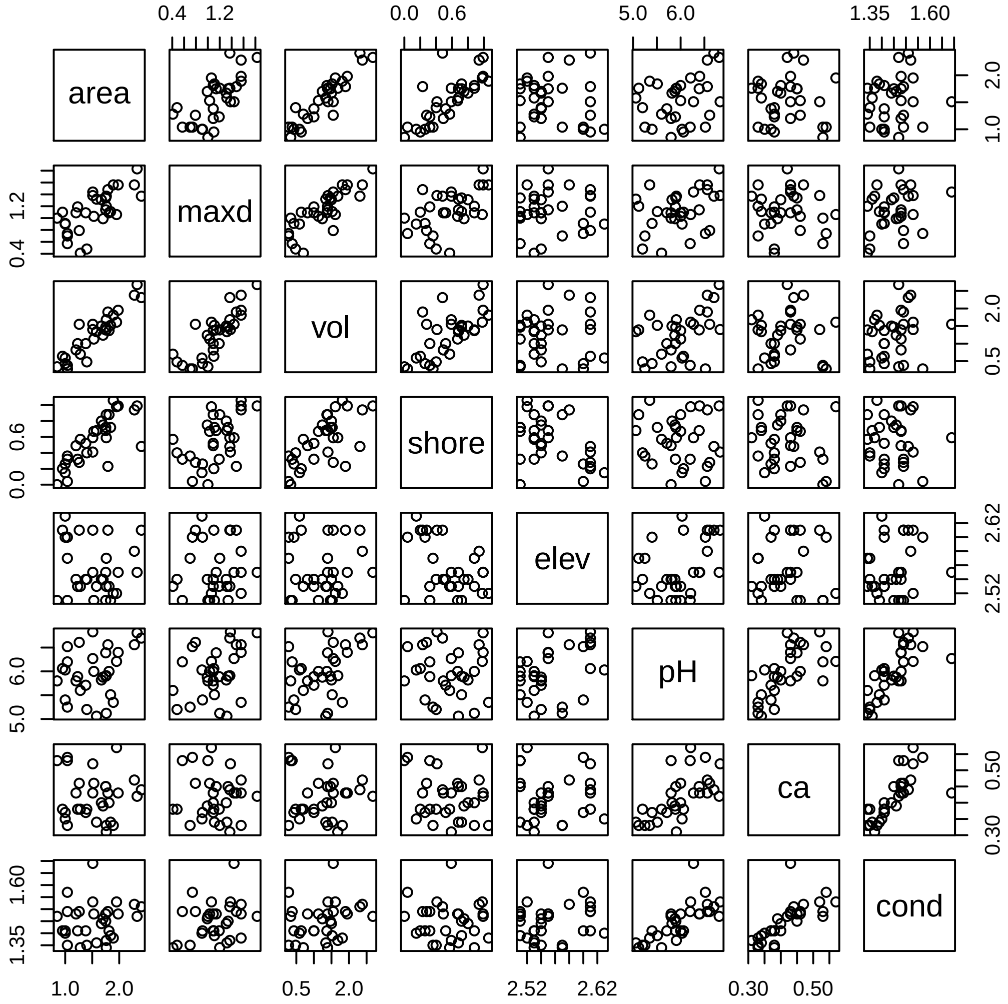
The first line stores the indices for the data that were not determined to be scatter outliers in `not.out`. We use a cut-off for the `wscat` index of 0.25 (the default in `pcout`). Now `limn.t.out` are the outlying training data and `limn.t` are the training data with the outliers removed. Here are the offending lakes.

```
> limn.t.out
```

```
           area maxd  vol shore elev  pH   ca cond
Kawagama   3.45 1.86 3.79  2.02 2.53 6.0 0.46 1.48
S.McDonald 1.08 0.81 0.56  0.20 2.54 5.6 0.37 0.95
```

and here is what the scatterplot matrix looks like with these two lakes removed.

```
> pairs(limn.t)
```



There still appears to be an outlier (unusually high conductivity value) but we give `pcout` the benefit of the doubt and proceed. Our general philosophy is to be somewhat liberal with initial outlier detection, preferring to examine the data in the light of fitted models—which are argued in the main text to be quite appropriate. If our models were clearly missing certain data features, we would recommend either cutting more outliers or—better still—making new models.

We use the `factanal.predictive` function in our `reo` package to fit a factor analysis model.

```
> limn.t.fa <- factanal.predictive(limn.t, 2)
```

The second argument indicates that a two-axis model is to be fitted. The name, `limn.t.fa`, refers to what is known in R-speak as an *object* of *class* `factanal.predictive`. Objects of this class contain lots of information on the fitted models that they represent. To extract this information, we have provided what are known in R-speak as *methods* for the `ltm.ecol` class of objects. Such methods are alternative versions of common functions that will be familiar to many R users (e.g. `print`, `predict`, etc.). Invoking the `print` method is particularly easy, and simply involves typing the name of the object into the command line and pressing enter.

```
> limn.t.fa
Call:
factanal(x = x, factors = factors, scores = "none")

Uniquenesses:
 area  maxd   vol shore  elev    pH    ca  cond
0.121 0.327 0.005 0.344 0.798 0.005 0.471 0.446

Loadings:
      Factor1 Factor2
area   0.937
maxd   0.812   0.116
vol    0.982   0.175
shore  0.753  -0.299
elev           0.445
pH     0.299   0.952
ca    -0.128   0.716
cond   0.131   0.733

               Factor1 Factor2
SS loadings      3.196   2.288
Proportion Var   0.400   0.286
Cumulative Var   0.400   0.685

Test of the hypothesis that 2 factors are sufficient.
The chi square statistic is 37.3 on 13 degrees of freedom.
The p-value is 0.000372
```

Alternatively, we could have typed the following to get the same output.

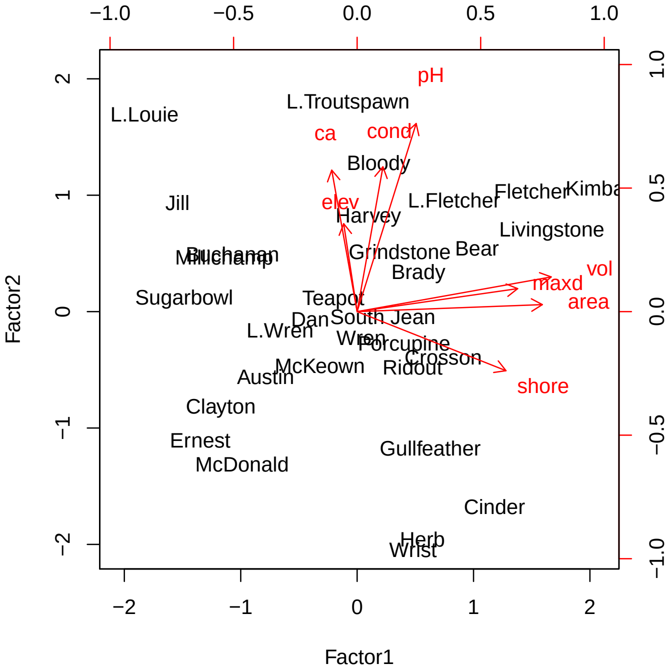
```
> print(limn.t.fa)
```

The first piece of information we get is simply the call to `factanal` that is made within the function `factanal.predictive`; this is because our function is simply a modification of `factanal` in the `stats` package that we have specifically tailored for predictive inference. Next we get the uniquenesses for each variable (i.e. the proportion of observed variance that is not explained by the axes). We see that elevation has a particularly large uniqueness, which reflects the fact that it is not well correlated with the other variables (see the scatterplot matrices above). The next piece of information is a table of loadings, which are the estimated **B** coefficients (described in the main text) divided by the observed standard deviation of the variable being summarized. These standardized coefficients (or loadings) describe how each axis summarizes each variable. Large positive (negative) values mean that the variable is positively (negatively) and linearly related to the axis. The next piece of information describes the amount of variation explained by the axes. The first row (SS loadings) expresses variation explained on a scale that goes from zero to the number of variables. The second row express identical information but from zero to one, with the third row being cumulative.

It is important to note that these measures are not sensitive to differences among variables in their observed variances—the measures are standardized. Finally, we have a significance test, which in this case is highly significant indicating that perhaps two axes is not sufficient to summarize the patterns of covariation in the training data. We will look at this possibility in more detail using information criteria below.

Please note that for this particular method (i.e the `print` method), we actually use the old method for `factanal` objects from the `stats` package. Because our `factanal.predictive` function is a modification of `factanal`, we felt that it was not necessary to modify the `print` method. However, we have written several other new methods that are specific to `factanal.predictive` objects. For example, we can produce an ordination diagram with our `biplot` method as follows.

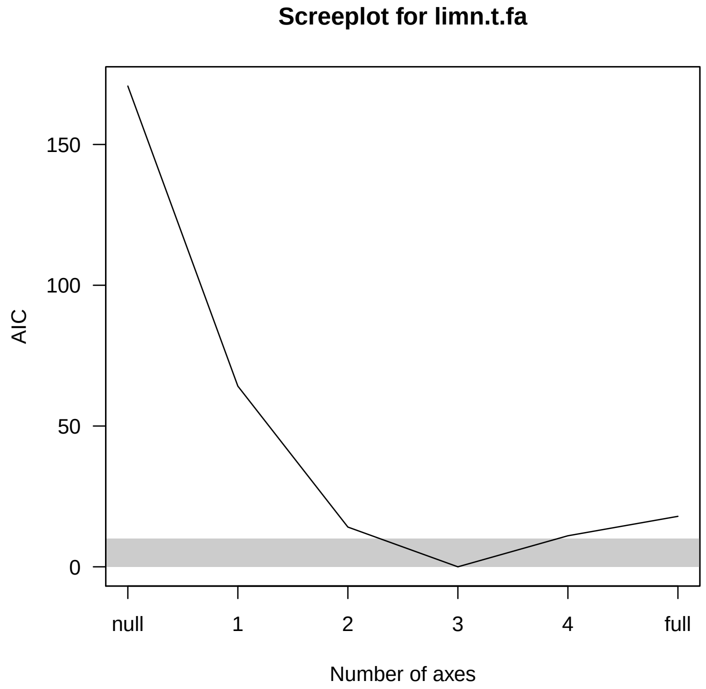
```
> biplot(limn.t.fa)
```



Here the names refer to the training lakes and the red arrows refer to the variables. This plot is interpreted in largely the same way that we would interpret a principal component analysis biplot. As noted in the main paper, we see that the first axis largely summarizes lake size and the second largely lake chemistry.

Now to assess whether other numbers of axes are more or less appropriate than this two axis model. We have developed a `screeplot` method for doing this.

```
> screeplot(limn.t.fa, stat = "AIC")
```



The second argument specifies that AIC should be used. The default is to correct AIC using the MAICc method we discuss and recommend in the main text (but this can be changed). Here the three axis model minimizes MAICc and so we recommend examining this model further, thereby illustrating some of the methods for exploring `factanal.predictive` objects.

```
> limn.t.fa <- factanal.predictive(limn.t, 3)
> limn.t.fa
```

```
Call:
factanal(x = x, factors = factors, scores = "none")

Uniquenesses:
 area  maxd   vol shore  elev    pH    ca  cond
0.088 0.316 0.005 0.161 0.418 0.124 0.045 0.317

Loadings:
      Factor1 Factor2 Factor3
area   0.936          -0.167
maxd   0.813           0.136
vol    0.985   0.144
shore  0.743  -0.125  -0.521
elev           0.102   0.755
pH     0.331   0.769   0.419
ca    -0.139   0.960  -0.116
cond   0.134   0.799   0.163
```

```
                Factor1 Factor2 Factor3
SS loadings       3.208   2.213   1.106
Proportion Var    0.401   0.277   0.138
Cumulative Var    0.401   0.678   0.816

Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 8.26 on 7 degrees of freedom.
The p-value is 0.31
```

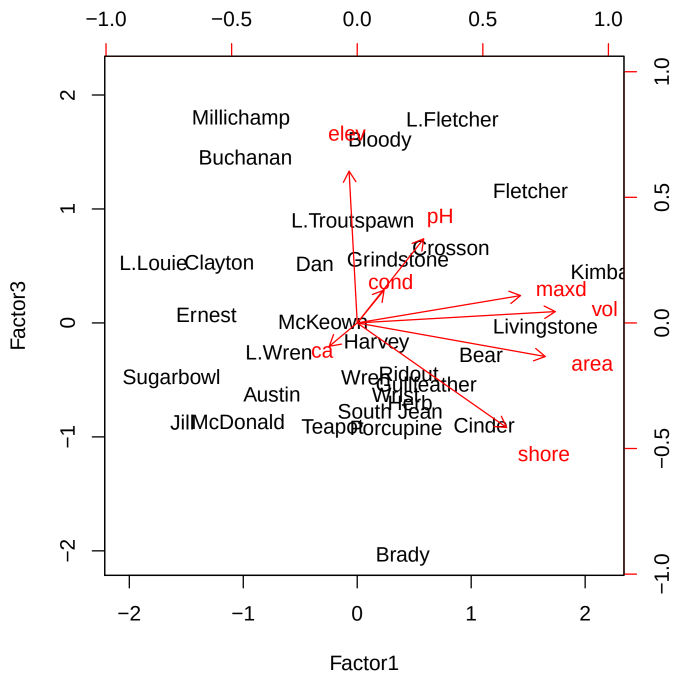
```
> deviance(limn.t.fa)
```

```
[1] -332.4640
```

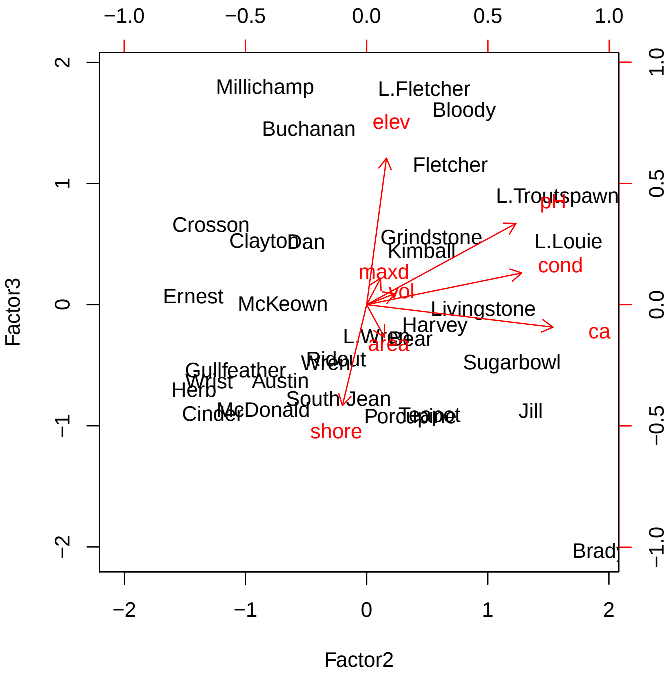
```
> AIC(limn.t.fa)
```

```
    MAICc
-232.7587
```

```
> biplot(limn.t.fa, axes = c(1, 3))
```



```
> biplot(limn.t.fa, axes = c(2, 3))
```



```
> predict(limn.t.fa)
$conditional.means
                   area
Austin        1.2687564
Bear          2.0080985
Bloody        1.5815321
Brady         1.9145971
Buchanan      1.0544543
Cinder        2.0110706
Clayton       1.0041917
Crosson       1.7742198
Dan           1.3603797
Ernest        0.9579385
Fletcher      2.0419003
Grindstone    1.6888355
Gullfeather   1.7710500
Harvey        1.6318338
Herb          1.7172096
Jill          1.0603853
Kimball       2.3895542
L.Fletcher    1.7626846
L.Louie       0.8497146
L.Troutspawn  1.5313707
L.Wren        1.3117253
Livingstone   2.2263371
```

```
McDonald     1.1327288
McKeown      1.4045637
Millichamp   0.9884217
Porcupine    1.7601324
Ridout       1.7541008
South Jean   1.6901084
Sugarbowl    1.0071733
Teapot       1.5477725
Wren         1.5818825
Wrist        1.6852763

$slope.coefs
              area
maxd   0.001421171
vol    0.575524039
shore  0.207463512
elev  -0.740933151
pH    -0.084932273
ca     0.504499023
cond  -0.080660185

$intercept.coefs
         area
[1,] 3.013609

$residual.covariance
           area
area 0.01753592
```

These results are discussed in the main text, but the last command using the `predict` method deserves some more discussion on how to use it in practice.

In our use of `predict` above, we used the default settings. However, the function is very versatile and can be used to make many different sorts of predictions via the `responses` and `predictors` arguments. Each of these are vectors of indices referring to different variables. For example, if we want to use variables one and two to predict variables three and four as responses we would use the following command.

```
> predict(limn.t.fa, responses = c(3, 4), predictors = c(1, 2))
$conditional.means
                   vol     shore
Austin       0.5655171 0.4716054
Bear         1.9631986 0.7801273
Bloody       0.7557037 0.4219085
Brady        1.6520766 0.8115355
Buchanan     0.5933880 0.2041502
Cinder       1.8659100 0.7258191
Clayton      0.5415340 0.2531355
```

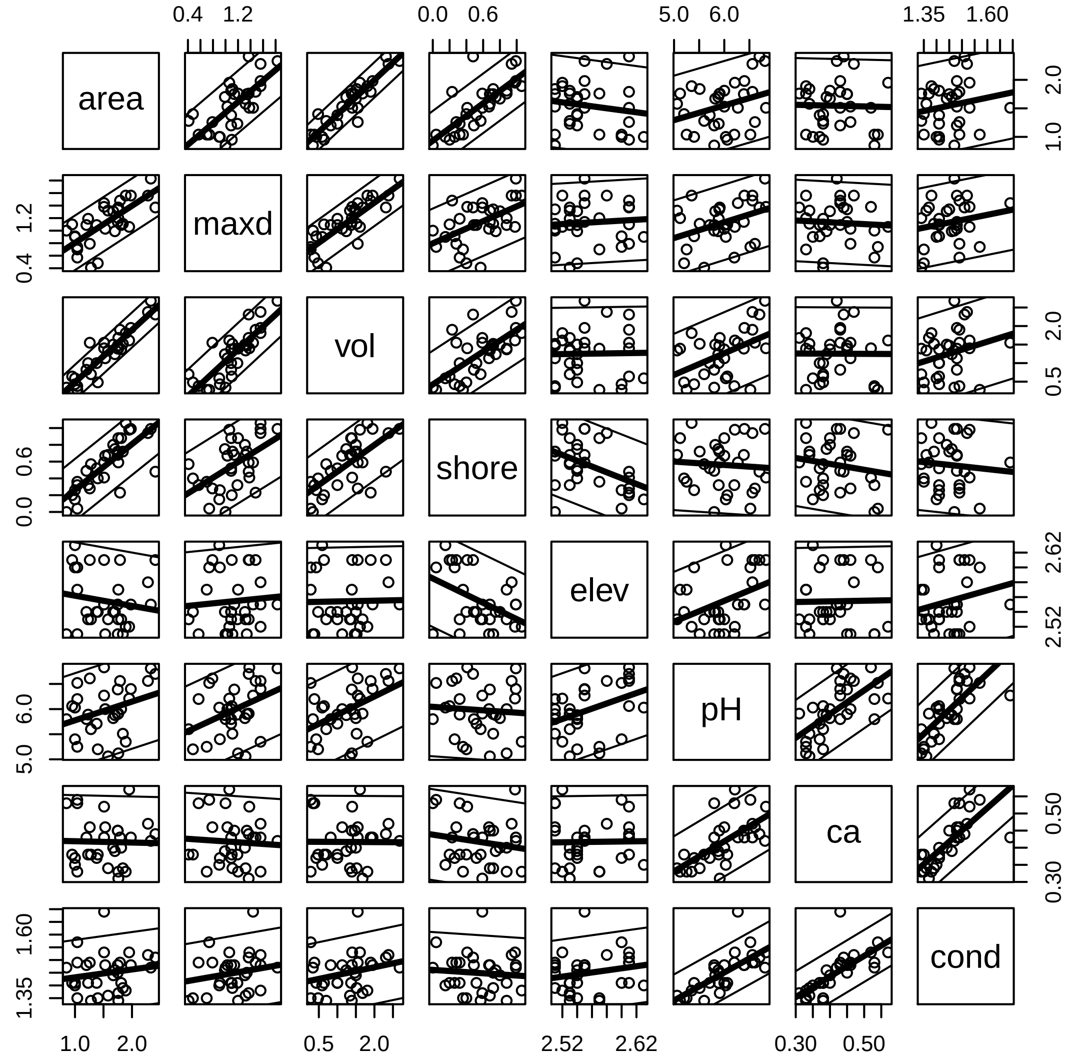
```
Crosson       1.6194789 0.6661881
Dan           0.9462281 0.3641970
Ernest        0.4677225 0.2980671
Fletcher      2.3221186 1.0584140
Grindstone    1.3882499 0.5084004
Gullfeather   1.5610376 0.7402077
Harvey        1.4804706 0.6829289
Herb          1.5247235 0.6830218
Jill          0.3952625 0.3109400
Kimball       2.4920369 0.9645900
L.Fletcher    1.7132208 0.6733984
L.Louie       0.4900179 0.2941062
L.Troutspawn 1.3548068 0.5143417
L.Wren        0.8580601 0.3559965
Livingstone   2.2874938 0.9611546
McDonald      0.7342521 0.5370848
McKeown       1.0526372 0.4646129
Millichamp    0.5359602 0.2541258
Porcupine     1.5919476 0.6631245
Ridout        1.4887473 0.6178212
South Jean    1.5174604 0.7240854
Sugarbowl     0.4295511 0.1537099
Teapot        1.1813417 0.5610679
Wren          1.3428136 0.6676109
Wrist         1.3970326 0.5625228

$slope.coefs
           vol      shore
area 1.0809840  0.6034245
maxd 0.5573849 -0.0990219

$intercept.coefs
           vol     shore
[1,] -1.046670 -0.260179

$residual.covariance
               vol        shore
vol    0.042720608 -0.002733137
shore -0.002733137  0.034893079
```

The output we get is a `list` with four components. The first (`conditional.means`) gives the point predictions for each of the response variables at each training lake; these are like points on a regression line (or plane). The next two components (`slope.coefs` and `intercept.coefs`) give the slopes and intercepts of these regressions. For example, the conditional means for `vol` are given by: -1.046670 + 1.0809840`area` + 0.5573849`maxd`. The last component (`residual.covariance`) gives the predicted covariance matrix of the residuals of these regressions. These predictions were all in the training data. To make genuine out-of-sample predictions, we specify `newdata` as in the following example.

```
> predict(limn.t.fa, limn.v, responses = c(3, 4), predictors = c(1,
+     2))

$conditional.means
                   vol     shore
Bentshoe    0.8013080 0.3899427
Poker       1.1104029 0.4066226
Blue Chalk  1.5378984 0.6329533
Saucer      0.2869960 0.1233528
Big East    2.1515261 0.9057633
Raven       2.8289994 1.2388228
L.Orillia   0.9401467 0.5084613
3 Island    0.8972452 0.4763097
McFadden    1.6483618 0.6371929
Big Orillia 1.3497389 0.6345620
Shoe        1.3576778 0.5774690
Louie       1.4725333 0.4775178
Redchalk    1.6975128 0.6523250
Sunken      0.5656868 0.3363460
Troutspawn  1.7507166 0.8337850
Wolf        1.8464856 0.7929072
Tingey      0.7082419 0.3667030
Poorhouse   1.1774572 0.5219847

$slope.coefs
           vol      shore
area 1.0809840  0.6034245
maxd 0.5573849 -0.0990219

$intercept.coefs
           vol     shore
[1,] -1.046670 -0.260179

$residual.covariance
              vol        shore
vol    0.042720608 -0.002733137
shore -0.002733137  0.034893079
```

Figure 3 in the main text was generated using the following single line.

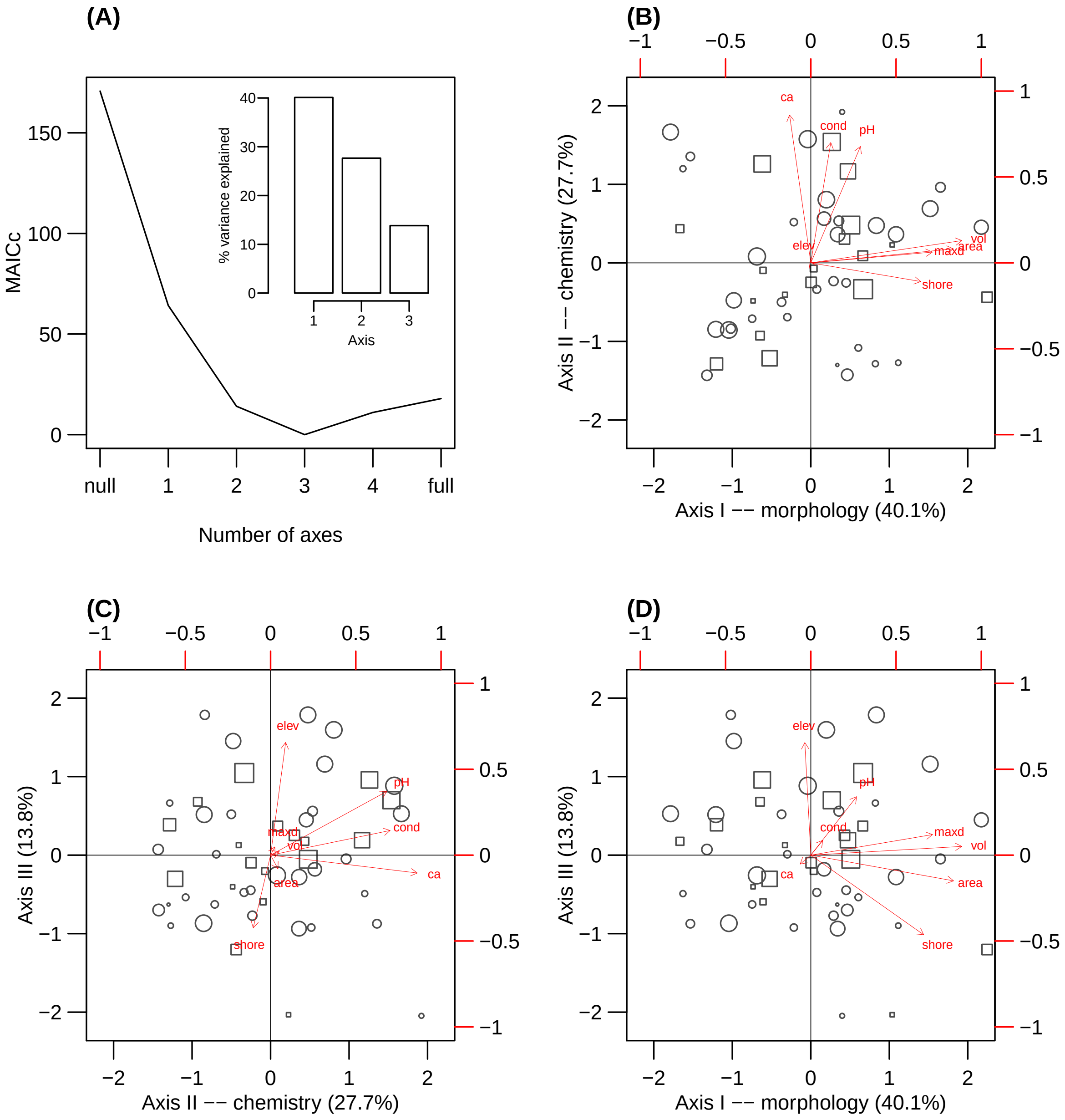
```
> pairs(limn.t.fa)
```



This `pairs` method is based on the `predict` method just described. In particular, it is the slopes and intercepts that `predict` produces that are used to make the regression lines in the graphs produced by this `pairs` method for `factanal.predictive` objects.

However, not all graphs in the manuscript can be made so easily. For example, Figure 2 was produced using the following code, which demonstrates that while the `biplot` and `screeplot` methods are good for data exploration, specially made plots may be necessary for publication purposes.

```
> dft <- par() # save the default graphics parameters to be reset after
        plotting.
> par(mfrow=c(2,2),mar=c(4,4,5,3),tck=-0.05,mgp=c(3,1,0)) # adjust
        graphics parameters
> d <- 3 # number of axes
> B <- limn.t.fa$loadings[] # extract standardized coefficients
> ve.d <- ve <- colSums(B^2)*(100/ncol(limn.t)) # variation explained
       by each axis
> # calculate point sizes for lakes (proportional to latitude
> valid.cex <- 5*((0.05+lat[valid,]-min(lat))/(0.5+max(lat)-min(lat)))
> train.cex <- 5*((0.05+lat[train,]-min(lat))/(0.5+max(lat)-min(lat)))
> # calculate axis scores
> xt <- scores(limn.t.fa)
> xv <- scores(limn.t.fa,limn.v)
>  # make screeplot
> screeplot(limn.t.fa,stat="AIC",main="",ylab="",grey.int=1)
> title(main="(A)",adj=0)
> title(ylab="MAICc",line=2.5)
> # make inset of variation explained
> par(new=TRUE,mar=c(10,11,5.5,2.6))
> hwid <- 0.4
> plot(1:d,ve.d,type="n",ylim=c(0,max(ve.d)),xlim=c(0,4),
+         xlab="",ylab="",axes=FALSE)
> for(i in 1:3){
+         rect(i-hwid,0,i+hwid,ve.d[i])
+ }
> par(tck=-0.05,mgp=c(3,0.2,0))
> axis(1,at=1:d,labels=1:d,cex.axis=0.7)
> par(mgp=c(3,0.5,0))
> axis(2,at=c(0,10,20,30,40),labels=c(0,10,20,30,40),las=1,cex.axis=0.7,
       line=-0.4)
> title(xlab="Axis",cex.lab=0.7,line=1)
> title(ylab="% variance explained",cex.lab=0.7,line=1)
> # make biplots
> fig.let <- c("(B)","(C)","(D)")
> par(mar=c(4,4,5,3),tck=-0.05,mgp=c(3,1,0))
> indices <- rbind(c(1,2,1),c(2,3,3))
> for(k in 1:3){
+         i <- indices[1,k]
+         j <- indices[2,k]
+         plot(limn.t.fa$scores[,c(i,j)],cex=train.cex,
+                 xlab="",ylab="",asp=1,las=1,col=grey(0.3),pch=1,
+                 xlim=c(-1,1)*max(abs(c(xt))),
+                 ylim=c(-1,1)*max(abs(c(xt))))
+         points(xv[,c(i,j)],pch=0,col=grey(0.3),cex=valid.cex)
+         title(main=fig.let[k],adj=0)
+         abline(h=0,lwd=0.5)
+         abline(v=0,lwd=0.5)
+         title(xlab=paste("Axis ",switch(i,"I -- morphology",
+                "II -- chemistry","III"),
+                 " (",round(ve[i],1),"%)",sep=""),line=2)
+          title(ylab=paste("Axis ",switch(j,"I -- morphology",
+                 "II -- chemistry","III"),
+                  " (",round(ve[j],1),"%)",sep=""),line=2)
+          par(new=TRUE)
+          plot(B[,c(i,j)],type="n",xlab="",ylab="",axes=FALSE,
+                  xlim=c(-1,1),ylim=c(-1,1))
+          text(B[,c(i,j)],labels=colnames(limn.t),cex=0.6,col="red")
+          axis(3,at=seq(-1,1,0.5),labels=seq(-1,1,0.5),col.ticks="red")
+          axis(4,at=seq(-1,1,0.5),labels=seq(-1,1,0.5),las=1,col.ticks="red")
+          for(l in 1:ncol(limn.t)){
+                  mp <- (sqrt(sum(B[l,c(i,j)]^2))-0.1)/sqrt(sum(B[l,c(i,j)]^2))
+                  arrows(0,0,mp*B[l,i],mp*B[l,j],lwd=0.3,length=0.05,col="red")
+          }
+ }
> par(dft) # restore original graphics parameters
```


(A)
MAICc
150
100
50
0
null
1
2
3
4
full
Number of axes
% variance explained
40
30
20
10
0
1
2
3
Axis
(B)
Axis II –– chemistry (27.7%)
Axis I –– morphology (40.1%)
ca
cond
pH
elev
vol
area
maxd
shore
(C)
Axis III (13.8%)
Axis II –– chemistry (27.7%)
elev
pH
cond
maxd
vol
area
ca
shore
(D)
Axis III (13.8%)
Axis I –– morphology (40.1%)
elev
pH
cond
maxd
ca
vol
area
shore
−2
−1
0
1
2
−1
−0.5
0
0.5
1

III. Using `ltm.ecol`

We now consider the latent trait model analysis of the `fish` data, which can be displayed using the following command if the `reo` package is loaded.

```
> fish
             PS YP WS CC BB NRD GS BT SB BNM CS PD FSD LB BNS B FM LT BS BD ID C
3 Island      0  0  0  0  1   1  0  0  0   0  0  1   0  0   0 0  1  0  0  0  0 0
Austin        0  1  1  0  0   0  0  0  1   0  0  0   0  0   0 0  0  0  0  0  0 0
Bear          1  1  1  1  0   0  1  0  1   1  0  0   0  0   1 1  0  1  0  0  0 0
Bentshoe      1  1  1  1  1   0  0  1  1   0  0  0   0  1   0 0  0  0  0  0  0 0
Big East      1  1  1  1  1   1  0  0  0   0  0  0   0  0   0 0  0  0  0  0  0 0
Big Orillia   1  1  1  0  0   0  0  1  0   0  0  0   0  1   0 0  0  0  0  0  0 1
Bloody        1  1  1  1  1   1  1  1  0   1  1  0   1  0   1 0  1  0  0  1  0 0
Blue Chalk    1  1  1  1  1   1  0  0  0   1  1  0   0  0   1 1  0  1  0  0  0 0
Brady         1  1  1  0  1   0  1  0  1   0  1  0   0  1   0 0  0  0  0  0  0 0
Buchanan      1  0  0  1  1   1  1  1  0   0  0  1   1  0   0 0  1  0  0  0  0 0
Cinder        1  1  1  1  0   1  1  0  0   0  0  0   0  0   0 0  0  0  0  0  0 0
Clayton       0  0  0  1  0   0  0  1  0   0  0  0   0  0   0 0  0  0  0  0  0 0
Crosson       1  1  1  1  1   0  1  1  0   0  0  0   0  0   1 0  0  0  0  0  0 0
Dan           1  1  1  1  1   1  0  0  0   0  0  0   0  0   0 0  0  0  0  0  0 0
Ernest        1  1  1  1  1   0  1  0  0   0  0  0   0  0   0 0  0  0  0  0  0 0
Fletcher      1  1  1  1  1   0  1  1  0   1  1  0   1  0   0 0  0  0  1  0  0 0
Grindstone    1  1  0  0  0   0  0  0  1   0  0  0   0  0   0 0  0  0  0  0  0 0
Gullfeather   1  1  1  1  0   0  1  0  0   0  0  0   0  1   1 0  0  0  0  0  0 0
Harvey        1  1  1  1  1   0  0  1  1   0  1  1   1  0   0 1  0  0  0  0  0 0
Herb          1  1  1  1  1   0  1  0  1   0  0  0   0  0   0 0  0  0  0  0  0 0
Jill          1  1  1  0  0   0  0  0  0   0  0  0   0  1   0 0  0  0  0  0  0 0
Kawagama      1  1  1  1  1   1  0  0  1   0  1  0   1  0   0 1  0  1  1  0  1 0
Kimball       0  1  1  1  1   1  0  1  1   0  0  0   1  0   0 1  0  1  1  0  1 0
L.Fletcher    1  1  1  1  1   1  1  0  0   1  1  1   1  0   0 0  0  0  0  0  0 0
L.Louie       1  1  1  1  1   1  1  0  0   0  0  0   0  0   0 1  1  0  1  1  0 0
L.Orillia     1  1  1  0  0   0  0  1  0   0  0  0   0  1   0 0  0  0  0  0  0 1
L.Troutspawn  1  0  1  1  0   1  0  1  0   1  0  1   0  0   0 0  1  0  0  1  0 0
L.Wren        1  1  1  1  1   0  1  0  1   0  0  0   0  0   0 1  0  0  0  0  0 0
Livingstone   1  1  1  1  0   1  0  1  1   1  1  1   1  0   1 0  0  1  0  0  0 0
Louie         1  1  1  1  1   1  0  1  0   1  1  0   1  0   1 0  0  0  0  0  0 0
McDonald      1  1  1  1  1   1  1  0  0   0  0  1   0  0   0 0  0  0  0  0  0 0
McFadden      1  1  1  1  1   0  1  0  0   1  1  1   1  0   0 0  1  1  0  0  0 0
McKeown       0  1  1  1  0   1  1  0  0   0  0  0   0  0   0 0  0  0  0  0  0 0
Millichamp    1  0  0  1  1   1  0  1  0   0  0  1   1  0   0 0  1  0  0  0  0 0
Poker         1  1  1  1  1   1  0  0  0   0  0  1   0  0   1 0  0  0  0  0  0 0
Poorhouse     1  0  1  1  1   1  1  1  0   1  1  0   0  0   0 0  1  0  1  1  0 0
Porcupine     1  1  0  0  1   0  0  0  0   1  0  0   0  1   0 0  0  0  0  0  0 0
Raven         1  1  1  1  1   0  1  0  1   1  1  0   0  0   1 1  0  1  0  0  1 0
Redchalk      1  0  1  1  1   1  1  1  0   1  1  1   1  0   1 1  0  1  0  1  0 0
Ridout        1  1  1  1  1   1  0  1  0   1  0  0   0  0   0 0  0  0  0  0  0 0
```

S.McDonald	1	1	0	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Saucer	1	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Shoe	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
South Jean	1	1	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Sugarbowl	1	1	1	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
Sunken	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Teapot	1	1	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1
Tingey	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Troutspawn	1	0	1	1	1	1	1	1	0	1	1	1	0	0	0	0	1	0	1	0	0	0
Wolf	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Wren	1	1	1	1	1	0	1	0	1	1	0	0	0	0	0	1	0	0	0	0	1	0
Wrist	1	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	LC	RB	FF	BM	M	RS	LW	NSS
3 Island	0	0	0	0	0	0	0	0
Austin	0	0	0	0	0	0	0	0
Bear	0	0	0	0	0	0	0	0
Bentshoe	0	0	0	0	0	0	0	0
Big East	0	0	0	0	0	0	0	0
Big Orillia	0	0	0	0	0	0	0	0
Bloody	0	0	0	0	0	0	0	0
Blue Chalk	0	0	0	0	0	0	0	0
Brady	0	0	1	0	1	0	0	0
Buchanan	0	0	0	0	0	0	0	0
Cinder	0	0	0	1	0	0	0	0
Clayton	0	0	0	0	0	0	0	0
Crosson	0	0	0	0	0	0	0	0
Dan	0	0	0	0	0	0	0	0
Ernest	0	0	0	0	0	0	0	0
Fletcher	0	0	0	0	0	0	0	0
Grindstone	0	0	0	0	0	0	0	0
Gullfeather	0	0	0	0	0	0	0	0
Harvey	0	0	0	0	0	0	0	0
Herb	0	0	0	0	0	0	0	0
Jill	0	0	0	0	0	0	0	0
Kawagama	1	1	0	0	0	0	1	1
Kimball	0	0	0	0	0	0	0	0
L.Fletcher	1	0	0	0	0	0	0	0
L.Louie	0	0	0	0	0	0	0	0
L.Orillia	0	0	0	0	0	0	0	0
L.Troutspawn	0	0	0	0	0	0	0	0
L.Wren	0	1	0	0	0	0	0	0
Livingstone	0	0	0	0	0	1	0	0
Louie	0	0	0	0	0	0	0	0
McDonald	0	0	0	0	0	0	0	0
McFadden	0	0	0	0	0	0	0	0

```
McKeown       0  0  0  0 0  0  0   0
Millichamp    0  0  0  0 0  0  0   0
Poker         0  0  0  0 0  0  0   0
Poorhouse     0  0  0  0 0  0  0   0
Porcupine     0  0  0  0 0  0  0   0
Raven         0  0  0  0 0  0  0   0
Redchalk      0  0  1  0 0  0  0   0
Ridout        0  0  0  0 0  0  0   0
S.McDonald    0  0  0  0 0  0  0   0
Saucer        0  0  0  0 0  0  0   0
Shoe          0  0  0  0 0  0  0   0
South Jean    0  0  0  0 0  0  0   0
Sugarbowl     0  0  0  0 0  0  0   0
Sunken        0  0  0  0 0  0  0   0
Teapot        0  0  0  0 0  0  0   0
Tingey        0  0  0  0 0  0  0   0
Troutspawn    0  0  0  0 0  0  0   0
Wolf          1  0  0  0 0  0  0   0
Wren          0  1  0  0 0  0  0   0
Wrist         0  0  0  0 0  0  0   0
```

We separate the same training lakes from `fish` that we did for `limn`.

```
> fish.t <- fish[train, ]
```

We now fit the training data to a series of latent trait models using our `ltm.ecol` function,

```
> fish.t.ltm <- ltm.ecol(fish.t, lambda = seq(0.1, 4.1, 1))
log-likelihood at initial values: -335.9
[1]       lambda   params/spp        iters     log-like
[1]          0.1          4.4           46       -255.8
[1]          1.1          3.4           27       -301.7
[1]          2.1          2.7           14       -326.6
[1]          3.1          1.5           30       -361.4
[1]          4.1            1           10       -374.7
```

A report is printed as a series of lines that appear slowly on the screen as `R` estimates the coefficients for each model. Each line corresponds to a different value of the regularization parameter, $\lambda = \{0.1, 1.1, 2.1, 3.1, 4.1\}$ (see first column in the report). These reports can be suppressed by setting the `verbose` argument to `FALSE`, but we recommend against this as they provide important information and reassure you that something is actually happening. The second column gives the number of non-zero coefficients per species. Notice that this decreases with λ because larger values of λ correspond to simpler models (i.e. with fewer coefficients). The simplest models use only a single intercept parameter for each species—these null models assume that species co-occurrences are not related to each other. Whenever the simplest models use more than one parameter per species, the models should be re-fitted using larger maximum λ values. The third column gives the number of iterations of the EM-algorithm that were required to reach convergence. The maximum number of iterations can be set using the `maxit` argument (default = 200). If the number of iterations is greater than the maximum (not the case here), then it might be a good idea to try setting larger values for `maxit`. If this results in prohibitively long computation times, then an option is to lower the precision used to judge convergence using the `digits` argument (default = 1) to `ltm.ecol`. There is a trade-off here: lower numbers mean that fewer iterations are required for convergence but also lead to less accurate approximations of the absolutely best estimates. The last column gives the log likelihood at convergence. This decreases with λ because simpler models always fit the data more poorly. However, the complex model may not provide optimal out-of-sample predictions, as we will soon see.

At this point, we would like to emphasize some practical issues that may arise. It will be common to have to re-run this function several times until an appropriate set of candidate λ values is decided on. An important consideration is computation time—more λ's takes more time. In particular, later we will calculate CVIC values, which require n-times the computational effort of simply fitting the models. See Section 10.1 of the main text for other guidelines on choosing a set of λ values. Another issue is that—perhaps surprisingly—one should not expect identical model fits for a particular value of λ within two different candidate sets; this is because our fitting procedure is sensitive to initial conditions and uses the fitted coefficients from the proceeding λ in the candidate set as initial values for the current λ. However, although the quantitative details of model fits can change from one candidate set to another, we have found that qualitative conclusions are quite robust. The situation here is similar to non-metric multidimensional scaling algorithms (e.g. `metaMDS` in the `vegan` package), which can be quantitatively (but usually not qualitatively) sensitive to initial conditions.

Now let's look at the fitted models. We may pass the `fish.t.ltm` object to the `print` function to obtain the following output.

```
> print(fish.t.ltm, which.lambda = 1)

Call:
ltm.ecol(Y = fish.t, lambda = seq(0.1, 4.1, 1))

Coefficients:
    (Intercept) axis I axis I^2 axis II axis II^2
PS   7.8        -0.1   -1.1     -0.2    -1.8
YP   9.8        -3.9   -2.3
WS   3.1        -0.3   -1.0      0.9
CC   5.7         4.6   -1.5      0.5
BB   2.2         0.5   -0.8      0.3    -0.3
NRD -0.3         4.7   -2.0     -0.5
GS   1.5               -1.2     -0.2    -0.3
BT  -1.4         3.3             0.9    -0.6
SB  -3.6        -2.7             8.0
BNM  0.0         0.6   -0.5      0.7    -0.9
CS  -0.1         0.8   -0.9      1.7    -1.0
PD  -2.0         4.8   -1.6     -0.1    -0.5
FSD -1.9         5.5   -2.2      2.4    -0.6
LB  -2.0        -5.8   -2.1     -1.3    -1.6
BNS -0.5         0.0   -0.5      0.3    -0.7
```

```
B   -1.5         0.4   -3.2      4.0
FM  -1.7         4.9   -1.4                -4.2
LT  -3.5         1.2   -1.0      2.9        0.0
BS  -1.8         5.5   -7.4      2.1
BD  -1.4         2.7   -1.2      0.0       -3.0
ID  -5.3               -2.8      4.7
LC  -2.0         2.7   -4.9      1.3       -0.1
RB  -3.7        -2.1   -4.1      4.3       -0.4
```

We used the `which.lambda` argument to specify the model to print—here we use the first model with $\lambda = 0.1$. The printed output reports the line of code used to fit the models (under the heading, 'Call'), and the fitted model coefficients in a table. The rows of the table correspond to species and the columns correspond to the intercept and linear and quadratic terms for each axis. Note that some coefficients are missing, because they were set to zero by the LASSO. For example, YP has no coefficients on the second axis, which implies that this axis summarizes no information about YP. We can also print the coefficients associated with the other values of λ.

```
> print(fish.t.ltm, which.lambda = 2)
```

```
Call:
ltm.ecol(Y = fish.t, lambda = seq(0.1, 4.1, 1))

Coefficients:
    (Intercept) axis I axis I^2 axis II axis II^2
PS   4.7         0.0   -0.3      0.2       -0.8
YP   4.1        -1.1   -0.5      0.7       -0.1
WS   2.1        -0.1   -0.3      0.7
CC   3.4         1.5   -0.7
BB   1.7         0.2   -0.3      0.3       -0.2
NRD -0.2         1.2    0.0
GS   1.3               -0.6                -0.3
BT  -1.1         1.5
SB  -0.8        -0.6             1.2
BNM -0.7         0.4             0.4       -0.3
CS  -0.8         0.2   -0.1      0.8       -0.2
PD  -1.3         1.3                       -0.2
FSD -1.5         1.2             0.9
LB  -2.1        -1.8   -0.2                -0.1
BNS -1.3                0.0                -0.1
B   -1.3               -0.4      1.4
FM  -2.2         1.4                       -0.2
LT  -2.8         0.1             1.4
BS  -2.3         0.2   -0.1      1.0
BD  -2.5         0.8                       -0.1
ID  -3.6                         1.7
LC  -3.0                         0.7
RB  -2.7                0.0      1.0
```

```
> print(fish.t.ltm, which.lambda = 3)
Call:
ltm.ecol(Y = fish.t, lambda = seq(0.1, 4.1, 1))

Coefficients:
    (Intercept) axis I axis I^2 axis II axis II^2
PS   2.7                                -0.3
YP   3.3        -0.6   -0.3      0.3    -0.1
WS   1.9               -0.2      0.5     0.0
CC   2.5         0.9   -0.4
BB   1.0         0.1   -0.1      0.1
NRD -0.2         0.8
GS   1.1               -0.4             -0.3
BT  -0.9         0.9
SB  -0.7        -0.3             0.7
BNM -0.9         0.2                    -0.1
CS  -1.2         0.1    0.0      0.4
PD  -1.4         0.9                     0.0
FSD -1.2         0.8             0.5
LB  -1.9        -1.0
BNS -1.5                                 0.0
B   -1.4               -0.1      0.8
FM  -2.3         1.0
LT  -2.4                         0.9
BS  -2.3                         0.8
BD  -2.5         0.6
ID  -2.9                         1.0
LC  -2.9                         0.5
RB  -2.5                         0.6
> print(fish.t.ltm, which.lambda = 4)
Call:
ltm.ecol(Y = fish.t, lambda = seq(0.1, 4.1, 1))

Coefficients:
    (Intercept) axis I axis I^2 axis II axis II^2
PS   2.0
YP   3.2               -0.5
WS   1.9               -0.3
CC   1.4         0.3
BB   0.7
NRD -0.3         0.5
GS   0.1                0.0
BT  -0.8         0.5
SB  -0.5               -0.1
BNM -1.0
```

```
CS  -1.2
PD  -1.4         0.6
FSD -1.1         0.5
LB  -1.4        -0.3
BNS -1.5
B   -1.3
FM  -2.1         0.8
LT  -2.0
BS  -2.0
BD  -2.4         0.3
ID  -2.3
LC  -2.8
RB  -2.3
```

```
> print(fish.t.ltm, which.lambda = 5)

Call:
ltm.ecol(Y = fish.t, lambda = seq(0.1, 4.1, 1))

Coefficients:
    (Intercept) axis I axis I^2 axis II axis II^2
PS   2.0
YP   2.0
WS   1.3
CC   1.3
BB   0.7
NRD -0.2
GS   0.0
BT  -0.7
SB  -0.6
BNM -1.0
CS  -1.2
PD  -1.2
FSD -1.0
LB  -1.3
BNS -1.5
B   -1.3
FM  -1.8
LT  -2.0
BS  -2.0
BD  -2.3
ID  -2.3
LC  -2.8
RB  -2.3
```

Notice that more and more coefficients are missing as λ is increased. The last model (with $\lambda = 4.1$) contains only intercept terms and is therefore a ‘null’ model.

Initially we hoped to be able to use existing software for fitting latent trait models, but found that existing tools were not up to the challenges posed by ecological data. Our `ltm.ecol` function was based on the `ltm` function in the `ltm` package, but with modifications that make it more useful for ecologists. A simple one-axis model without quadratic terms can be fitted with the following code.

```
> library(ltm)
> ltm(fish.t.ltm$Y ~ z1, IRT.param = FALSE)
Call:
ltm(formula = fish.t.ltm$Y ~ z1, IRT.param = FALSE)

Coefficients:
     (Intercept)      z1
PS         2.025   0.103
YP         2.207  -0.775
WS         1.372   0.192
CC        21.597  22.302
BB         0.862   0.698
NRD       -0.328   1.716
GS         0.015   0.288
BT        -1.000   1.415
SB        -0.608  -0.032
BNM       -1.121   0.756
CS        -1.409   1.046
PD        -1.534   1.304
FSD      -20.555  29.882
LB        -3.047  -2.723
BNS       -1.538   0.059
B         -1.651   1.123
FM        -2.282   1.350
LT        -3.060   1.898
BS        -3.357   2.206
BD        -2.707   1.029
ID        -3.593   2.019
LC       -14.004   9.442
RB        -2.637   0.925

Log.Lik: -327.527
```

This function uses the formula-style input familiar to users of `lm` and `glm`. On the left of the tilde is the observed data matrix and on the right is the symbol `z1`, which denotes a one axis logistic model with linear terms only. The first thing to notice about this fit is the very large coefficients for CC and FSD, which lead to predicted probabilities of occurrence that are very close to zero and one. Such probabilities are too certain and cross-validate poorly with ecological data. Furthermore, a one-axis linear model is not generally consistent with ideas in gradient analysis. While `ltm` allows us to fit two-axis linear and quadratic models, this over-fitting problem only becomes exacerbated as we see here.

```
> ltm(fish.t.ltm$Y ~ z1 + z2, IRT.param = FALSE)
Call:
ltm(formula = fish.t.ltm$Y ~ z1 + z2, IRT.param = FALSE)

Coefficients:
       (Intercept)             z1             z2
PS     8682958.400    2274726.519     973706.857
YP     4333745.333    4192892.684    2104987.935
WS    12541836.205   15810013.357    3284022.679
CC    38651088.864  -20881953.683  -15753423.771
BB     1224994.585   -5548414.960   -4792202.929
NRD    8844949.226  -11134854.525   -8497636.016
GS   -21272797.461    3810315.960    5293802.369
BT     6866117.013  -13329864.504  -14616116.456
SB   -36644199.677   40534811.352   16910734.170
BNM    4995780.061   10791516.738     773292.161
CS          -0.668          0.541         -0.333
PD    -1824211.670   -8109335.281   -3706103.460
FSD    8795431.680  -11006248.132  -15292969.401
LB   -13089245.746    4779235.436    9743884.394
BNS  -22506235.508    4250019.681    5550268.224
B    -19483826.775   23046472.754    3240351.341
FM     4572532.928  -38547212.334  -32002249.113
LT   -27819867.217     851468.404    7466117.507
BS    -7221023.862    9036141.092    -282702.126
BD     4426526.281   -5547974.985   -8258049.815
ID   -12156931.921   12077947.118   -4514760.381
LC   -35177424.664   40475824.193   -6157804.426
RB    -9506460.278    6285096.414   -8536089.647

Log.Lik: -2025.203
> ltm(fish.t.ltm$Y ~ z1 + I(z1^2) + z2 + I(z2^2), IRT.param = FALSE)
Call:
ltm(formula = fish.t.ltm$Y ~ z1 + I(z1^2) + z2 + I(z2^2), IRT.param = FALSE)

Coefficients:
     (Intercept)             z1      I(z1^2)           z2      I(z2^2)
PS      -9068149     -9029520.8    4959582.1   2492970.77    924923.85
YP      53877867     -2150473.5   -1275813.3  -6560274.37   -687292.23
WS       5557867      2231792.6    2384455.4  -1639673.01   -193040.14
CC      22938709      3436567.9   -6425602.3    630345.59   -442533.79
BB       1679719      1398845.8   -3385331.2   2469791.51    497718.04
NRD     21284052      4989236.1   -3499327.9    894305.03   -726165.41
GS       6328861      4572790.2   -6362202.4    579654.28    229141.27
BT      14192814      4406902.1   -4001912.3   -190162.06   -216300.03
```

```
SB      14097186       1223507.6    -534048.5  -1983588.72  -1208548.50
BNM    -33672642      -6448910.0    3861156.5   2217220.89   1161168.77
CS     -12109497      13526878.3   -5490538.8    804578.74    -89911.42
PD      -8324334       3770029.7   -5805767.8   1093438.69    263927.96
FSD      7313967       -384990.8  -11843033.7     62403.27   -292539.70
LB     -13683354      -1877526.5    5084193.7  -4163188.56   -391627.51
BNS    -45736031       6383188.0    3131491.1   2939011.11    656753.95
B        7297288       1548474.3   -4700924.6  -2052272.92   -578098.70
FM     -28755617     -11152921.6  -12499533.3   6179643.24   1657113.76
LT       7795098      -1752870.3   -8300991.6  -1721871.79   -147530.26
BS      -1997597      -4049973.6   -3814033.1    351696.61   -157459.01
BD      -7823435     -10068047.8   -9695955.4   5417409.84   -356441.34
ID      -4382425      22990483.7  -12757052.2  -4206887.01  -1140263.67
LC     902573073   -1869336684.1  799689705.7   3398585.27   -452781.33
RB       9874183       3998984.2   -4151299.9  -2010797.88   -590414.27

Log.Lik: -2271.514
```

Notice that the goodness-of-fit as measured by log-likelihood (i.e. `Log.Lik` in the model output) goes down as the models get more complex; this is a sign of a very bad model-fitting procedure for these data. It was this poor performance of the `ltm` function that led us to write `ltm.ecol`, which makes use of LASSO-based regularization. Our LASSO procedure also ensures that species cannot take their minimum probability of occurrence at intermediate values of the ordination space, which is often thought to be an ecologically questionable assumption. Another consequence of these extremely large coefficients is that AIC, which is calculated by `ltm`, is not appropriate because AIC requires that parameter estimates are numerically on the interior of the parameter space—not the case for `ltm` with the `fish` data. Our `ltm` uses CVIC, which is valid for virtually any fitting procedure.

To compare CVIC values for a number of λ values, add a `cvic=TRUE` argument to `ltm.ecol`.

```
> fish.t.ltm <- ltm.ecol(fish.t, cvic = TRUE, lambda = seq(0.1, 4.1,
+     1), verbose = FALSE)
```

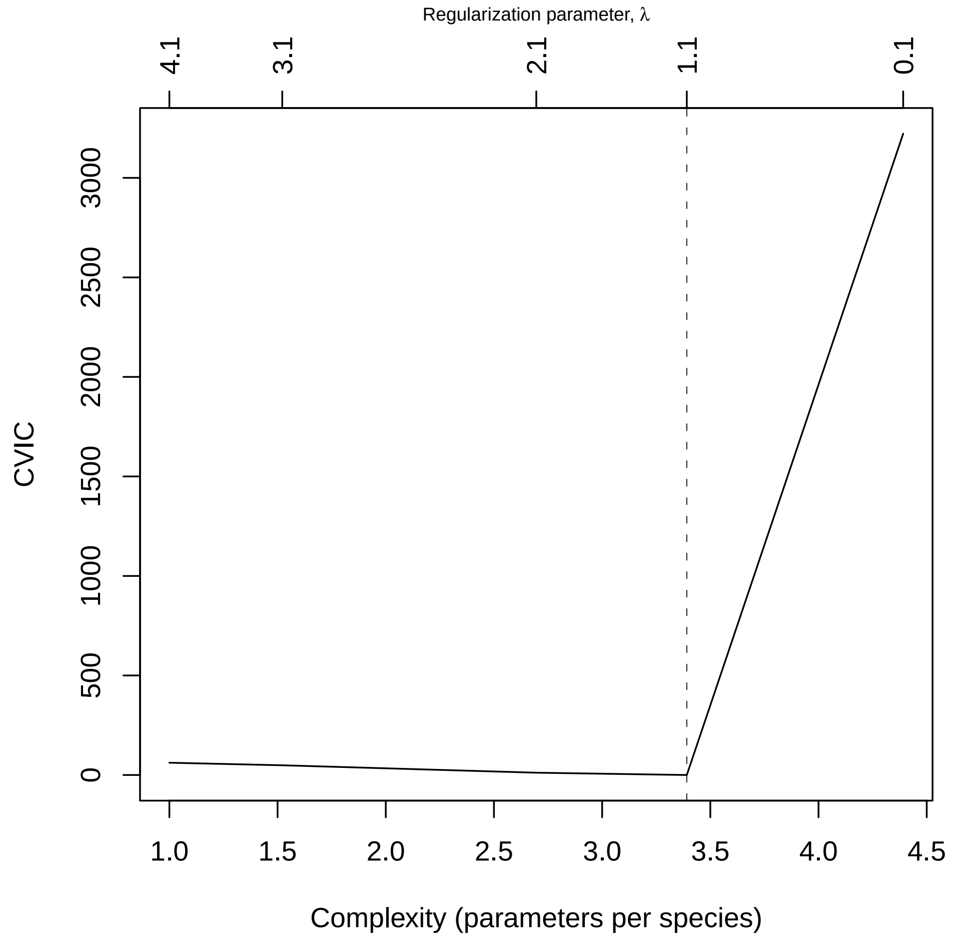
We set `verbose=FALSE` to suppress long reports that would take up many pages. But in practice, we recommend using these reports as they provide valuable information for monitoring the cross-validation procedure. Note that calculating CVIC in this way can take a fairly long time for larger data sets, and therefore we recommend making sure that the λ values used cover a wide range of model complexity before setting `cvic=TRUE` (see discussion above and in Section 10.1). We can print this fitted model object as follows.

```
> fish.t.ltm
Regularization path:
  lambda   CVIC
1    0.1 3961.0
2    1.1  738.9
3    2.1  750.3
4    3.1  787.6
5    4.1  800.5
```

```
 Optimal lambda:  1.1

Call:
ltm.ecol(Y = fish.t, cvic = TRUE, lambda = seq(0.1, 4.1, 1))

Coefficients:
    (Intercept) axis I axis I^2 axis II axis II^2
PS   4.7         0.0   -0.3      0.2    -0.8
YP   4.1        -1.1   -0.5      0.7    -0.1
WS   2.1        -0.1   -0.3      0.7
CC   3.4         1.5   -0.7
BB   1.7         0.2   -0.3      0.3    -0.2
NRD -0.2         1.2    0.0
GS   1.3               -0.6             -0.3
BT  -1.1         1.5
SB  -0.8        -0.6             1.2
BNM -0.7         0.4             0.4    -0.3
CS  -0.8         0.2   -0.1      0.8    -0.2
PD  -1.3         1.3                    -0.2
FSD -1.5         1.2             0.9
LB  -2.1        -1.8   -0.2             -0.1
BNS -1.3                0.0             -0.1
B   -1.3               -0.4      1.4
FM  -2.2         1.4                    -0.2
LT  -2.8         0.1             1.4
BS  -2.3         0.2   -0.1      1.0
BD  -2.5         0.8                    -0.1
ID  -3.6                         1.7
LC  -3.0                         0.7
RB  -2.7                0.0      1.0
```

There is some more information in this printout compared with the version without the CVIC calculations. This new information consists of a table of CVIC values for various values of λ, and the optimal λ value. The coefficients that are printed are for the optimal model. We can see this information in graphical form using a method we wrote for the `screeplot` function (The dotted vertical line indicates the optimal model with approximately 3.4 coefficients per species).

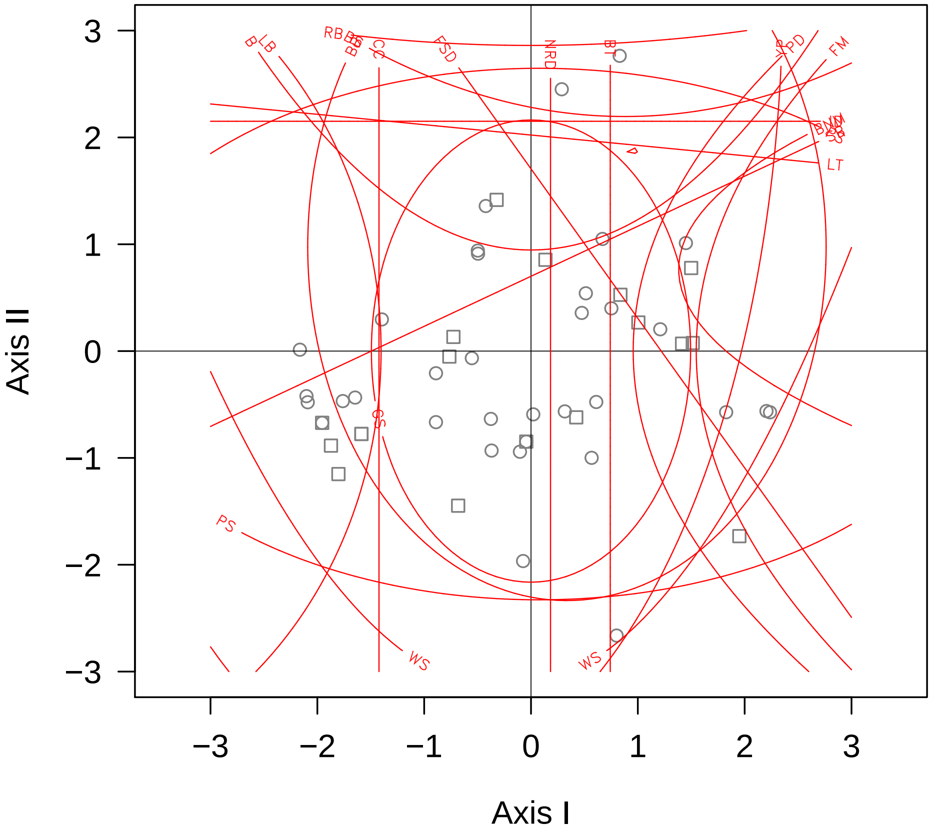
```
> screeplot(fish.t.ltm)
```



Notice the extremely poor performance of the $\lambda = 0.1$ model, which approaches the complexity of the models fitted by the `ltm` function that we found to perform poorly above.

We can explore this optimal model using the biplot function.

```
> fish.v <- fish[valid, colnames(fish.t.ltm$Y)]
> biplot(fish.t.ltm, fish.v)
```



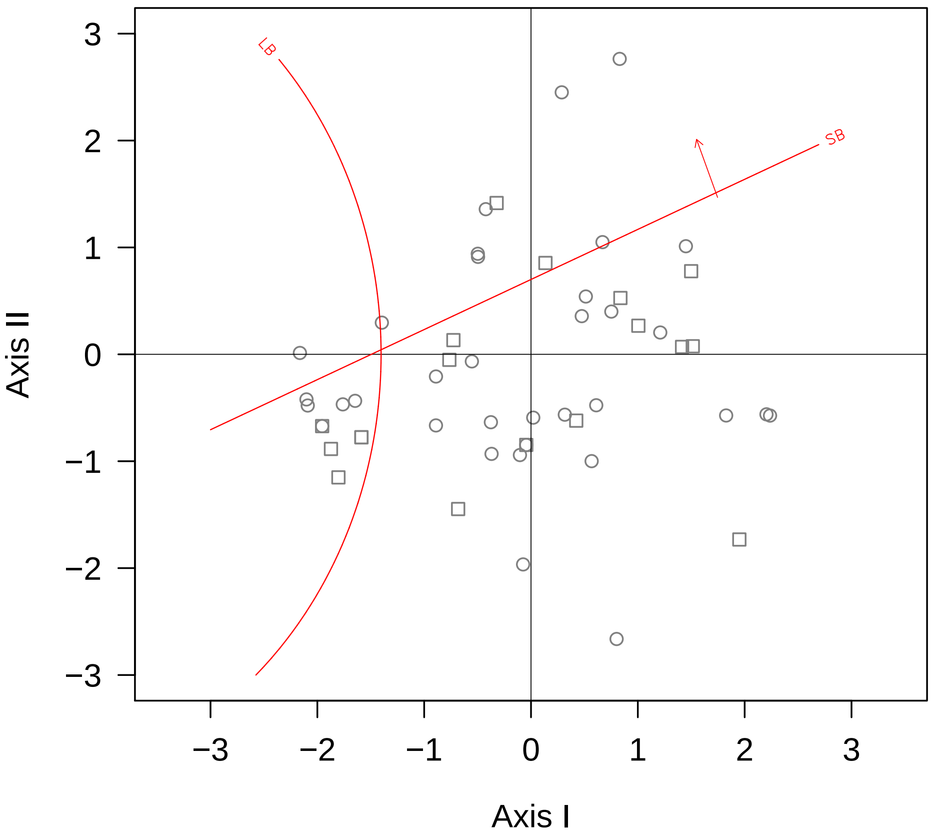
We supply `newdata` to the biplot function. These new data are the validation lakes that were not used to fit the model. They are plotted as squares whereas the circles represent the training lakes. Notice the subscripting technique that makes use of the `colnames` function in the first line, which ensures that only species that were used in the training data are retained in the validation data; this is important because the model only makes predictions about the species used to fit it. The red lines that represent the species are contours connecting points in the ordination space where probability of occurrence is $\frac{1}{2}$. We see a wide range of shapes, from ellipses to parabolas to lines, illustrating the diversity of responses that can be modeled—allowing a diversity of species responses to be represented if such diversity is present in the information provided by the training data. On the other hand, one obvious problem with this graph is that the large number of species makes it difficult to read. Another problem is that the directions of increasing probabilities of occurrence for some species are not clear. The code below allows us to overcome some of these difficulties.

```
> spp <- c(9, 14)
> bp.arws <- biplot(fish.t.ltm, fish.v, spp = spp, arrows = "input")
```

This code generates a biplot of the bass species (spp nine and fourteen) only. We do not show this plot because it is interactive (i.e. it requires user input using the mouse). By setting the `arrows="input"` argument, the `biplot` method plots each species one at a time.

The user is given an opportunity to draw an arrow for each species indicating the direction of increasing probability. Two clicks are required: the first for the tail of the arrow and the second for the head. If the user wishes to not plot an arrow for that species, one clicks above 3 on the y-axis. These arrow positions are stored in `bp1.arws`, which can be passed to another call to the `biplot` method as follows.

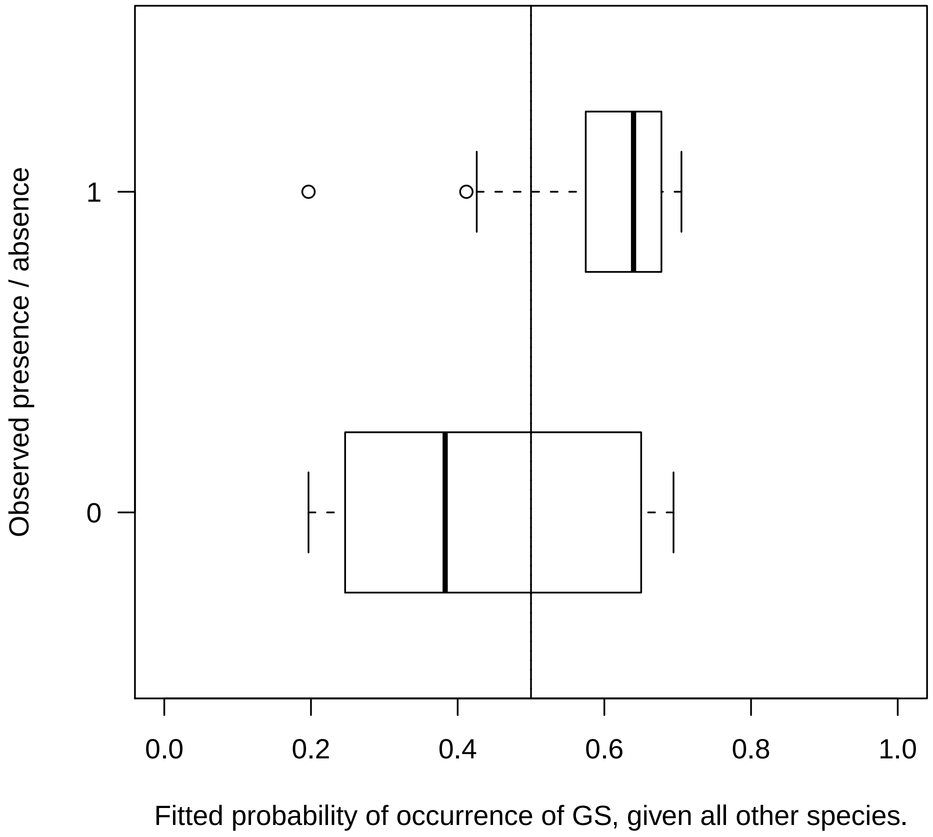
```
> biplot(fish.t.ltm, fish.v, spp = spp, arrows = bp.arws)
```



Here we drew an arrow indicating that SB decreases (increases) along axis one (two). This information can be read off of the coefficient table for SB, which gives negative (positive) linear term coefficients for axis one (two). LB does not require an arrow because its parabolic curvature is obvious—because our modified LASSO procedure constrains quadratic coefficients to be negative, higher probabilities are always inside of curved contours. We chose to make arrow drawing interactive and not automatic for two reasons: (1) sometimes the curvature is not obvious and it is difficult to automate when arrows are required and (2) human-guided arrow-placement can be important for readability.

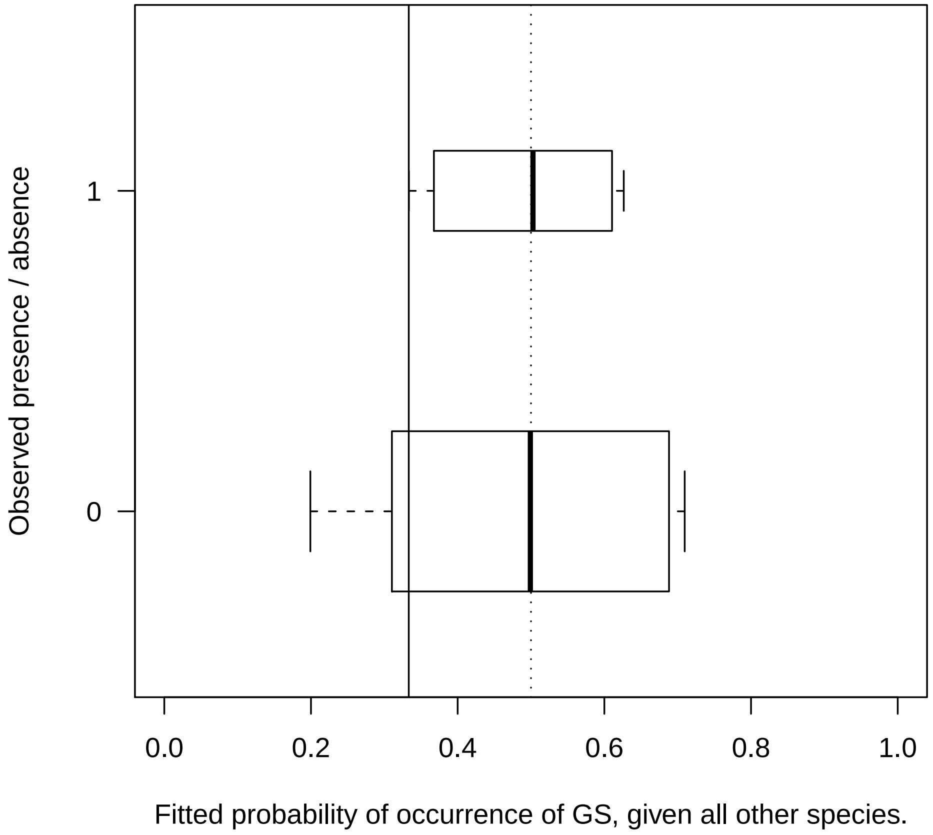
We can use the `boxplot` method to graphically explore model predictions. Here is the code to compare the observed presence / absence of species 7 (GS) against fitted probabilities of occurrence in the training data.

```
> sp <- 7
> boxplot(fish.t.ltm, responses1 = sp, xlab = paste("Fitted probability
+     of occurrence of ",
+     colnames(fish.t)[sp], ", given all other species.", sep = ""),
+     ylab = "Observed presence / absence")
```



The predictions for this species are much worse out of the training sample as we now show by specifying `newdata`.

```
> boxplot(fish.t.ltm, fish.v, responses1 = sp, xlab = paste("Fitted
+     probability of occurrence of ",
+     colnames(fish.t)[sp], ", given all other species.", sep = ""),
+     ylab = "Observed presence / absence")
```



But see the main text of the paper for examples of good out-of-sample prediction.

The `boxplot` method is based on the `predict` method, which can be used to generate any model-implied probability of occurrence or co-occurrence. We use the `responses1`, `responses0` and `predictors` arguments to specify different probabilities. The third of these arguments specifies which species to condition on. These species play the role of predictors, in that their patterns of co-occurrence determine the probabilities for the species playing the role of response. These patterns of co-occurrence of the predictor species are passed to `predict` either (1) through the data stored in the fitted model object that was used to fit the model (i.e. training data) or (2) through the `newdata` argument if it is provided. If `predictors=NULL` then no conditioning is done (analogous to an intercept only model in regression). The `responses1` and `responses0` arguments are used to specify the response species that are predicted to be present and absent respectively. For example, if we want to obtain the probability that species one is absent and species two is present given the patterns of occurrence of species three through ten in the validation lakes, we would input the following.

```
> predict(fish.t.ltm, fish.v, responses0 = 1, responses1 = 2, predictors = 3:10)
```

```
  Bentshoe       Poker  Blue Chalk      Saucer    Big East       Raven
0.07128117  0.04208394  0.02231566  0.11823551  0.04208394  0.02354065
```

```
  L.Orillia    3 Island    McFadden Big Orillia        Shoe       Louie
 0.10714363  0.09240949  0.01692038  0.10714363  0.20862564  0.02260226
   Redchalk      Sunken  Troutspawn        Wolf      Tingey   Poorhouse
 0.01716019  0.06133988  0.01716019  0.03529885  0.21434699  0.01716019
```

The variation in these probabilities among the validation lakes is due to variation in the presence / absence patterns of the predictor species. To see this, if we specify a vector of length zero for `predictors` (i.e. no predictors) we get the following.

```
> predict(fish.t.ltm, fish.v, responses0 = 1, responses1 = 2,
+         predictors = numeric(0))
```

```
   Bentshoe       Poker  Blue Chalk      Saucer    Big East       Raven
 0.04178084  0.04178084  0.04178084  0.04178084  0.04178084  0.04178084
  L.Orillia    3 Island    McFadden Big Orillia        Shoe       Louie
 0.04178084  0.04178084  0.04178084  0.04178084  0.04178084  0.04178084
   Redchalk      Sunken  Troutspawn        Wolf      Tingey   Poorhouse
 0.04178084  0.04178084  0.04178084  0.04178084  0.04178084  0.04178084
```

As expected, the variation has disappeared. In these cases, it is better to specify `predictors=NULL` as follows.

```
> predict(fish.t.ltm, fish.v, responses0 = 1, responses1 = 2,
+        predictors = NULL)
```

```
[1] 0.04178084
```

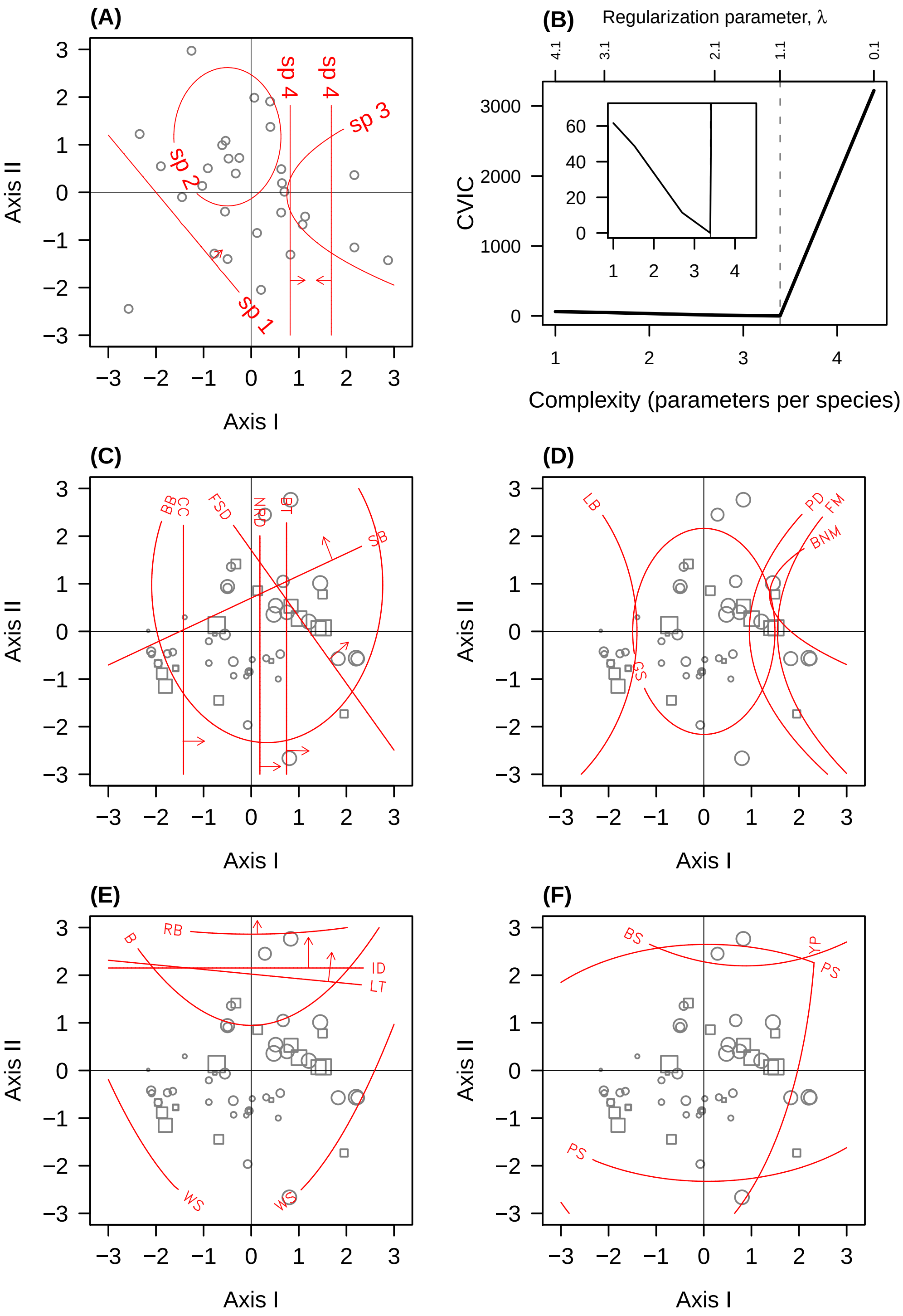
The possibilities are numerous. Here is one final example.

```
> predict(fish.t.ltm, fish.v, responses0 = c(10, 17), responses1 = c(4,
+     9), predictors = c(6, 21, 1, 19))
```

```
   Bentshoe       Poker  Blue Chalk      Saucer    Big East       Raven
 0.17489647  0.13298580  0.13298580  0.17489647  0.13298580  0.31650548
  L.Orillia    3 Island    McFadden Big Orillia        Shoe       Louie
 0.17489647  0.08142069  0.17489647  0.17489647  0.17489647  0.13298580
   Redchalk      Sunken  Troutspawn        Wolf      Tingey   Poorhouse
 0.13298580  0.10349341  0.20649377  0.17489647  0.17489647  0.20649377
```

```
> bp.arws1 <- biplot(fish.t.ltm, fish.v, spp = c(4, 5, 6, 8, 9, 13),
+     arrows = "input")
> bp.arws3 <- biplot(fish.t.ltm, fish.v, spp = c(3, 16, 18, 21, 23),
+     arrows = "input")
```

```
> par(mfrow = c(3, 2), mar = c(4, 4, 2, 2))
> plot(rnorm(30, sd = 1.3), rnorm(30, sd = 1.3), col = grey(0.5),
+     pch = 1, xlim = c(-3, 3), ylim = c(-3, 3), xlab = "Axis I",
+     ylab = "Axis II", asp = 1, las = 1, cex.lab = 1.2, cex.axis = 1.2)
> title(main = "(A)", adj = 0)
> abline(v = 0, lwd = 0.01)
> abline(h = 0, lwd = 0.01)
> x <- seq(-3, 3, 6/(75 - 1))
> X1 <- cbind(1, x, x^2) %x% rep(1, 75)
> X2 <- matrix(x, 75^2, 1)
> X1 <- cbind(X1, X2)
> X2 <- X2^2
> X <- t(cbind(X1, X2))
> B <- matrix(c(1.2, 0.6, 0, 0.5, 0, 1, -5, -5, 7, -3, -1.2, 1.6,
+     0, -0.1, -1, -5.5, 10, -4, 0, 0), 4, 5, byrow = TRUE)
> eta <- B %*% X
> for (j in 1:4) {
+     contour(x, x, t(matrix(eta[j, ], 75, 75)), nlevels = 1, zlim = c(0,
+         0), add = TRUE, labels = paste("sp", j), col = "red", lwd = 0.5,
+         lty = 1, method = "edge", labcex = 0.8)
+ }
> arrows(-0.8286392, -1.420022, -0.6064043, -1.21086, length = 0.05,
+     col = "red", lwd = 0.5)
> arrows(0.83, -1.845596, 1.13, -1.845596, length = 0.05, col = "red",
+     lwd = 0.5)
> arrows(1.6689526, -1.845596, 1.368953, -1.845596, length = 0.05,
+     col = "red", lwd = 0.5)
> par(mar = c(5, 4, 4, 1))
> screeplot(fish.t.ltm, las = 1, lwd = 2, cex.lab = 1.2, cex.axis = 1.2,
+     lambda.cex = 0.8, reg.cex = 0.7, yaxt = "n", xaxt = "n")
> axis(1, at = 1:4, labels = 1:4)
> axis(2, at = c(0, 1000, 2000, 3000), labels = c(0, 1000, 2000,
+     3000), las = 1)
> title(main = "(B)", adj = 0, line = 2.5)
> par(new = TRUE, mar = c(9, 7, 5, 7))
> screeplot(fish.t.ltm, print.reg = FALSE, las = 1, ylim = c(0, 70),
+     yaxt = "n", xaxt = "n", xlab = "", ylab = "")
> axis(2, at = c(0, 20, 40, 60), labels = c(0, 20, 40, 60), las = 1)
> axis(1, at = 1:4, labels = 1:4)
> par(mar = c(4, 4, 2, 2))
> biplot(fish.t.ltm, fish.v, spp = c(4, 5, 6, 8, 9, 13), arrows = bp.arws1)
> biplot(fish.t.ltm, fish.v, spp = c(7, 10, 12, 14, 17))
> biplot(fish.t.ltm, fish.v, spp = c(3, 16, 18, 21, 23), arrows = bp.arws3)
> biplot(fish.t.ltm, fish.v, spp = c(1, 2, 19))
```


(A)
Axis I
Axis II
sp 1
sp 2
sp 3
sp 4
sp 4
−3 −2 −1 0 1 2 3
(B)
Regularization parameter, λ
4.1 3.1 2.1 1.1 0.1
CVIC
0 1000 2000 3000
0 20 40 60
1 2 3 4
Complexity (parameters per species)
(C)
BB
CC
FSD
NRD
BT
SB
(D)
LB
PD
FM
BNM
GS
(E)
B
RB
ID
LT
WS
WS
(F)
BS
YP
PS
PS
Axis I
Axis II