mvabund analysis of Skokholm data

Philip Dixon

9/5/2021

mvabund analyses of Skokholm dataset

The general concept is to create a mvabund object, a matrix with species composition data. Rows = sites and columns = species. Functions in the mvabund library are designed to use an mvabund object as the "response" variable.

The main modeling function is manyglm(), to fit glm models with various distributions, including Poisson and negative binomial, to each species and package the results in a useful way. Alternatives include manylm() to fit models with gaussian=normal responses and manyany() to fit any R modeling function. The example at the very end uses manyany() to fit a spline model, gam(count ~ s(year), family='nb') to each species. This is included only to show what is possible. It is way beyond what you need to know.

This document illustrates two uses of manyglm():

- comparing species composition between two groups of samples (1930's and 1960's) and * fitting regression models, specifically a linear regression on year.

We read the data and create a new data frame with just the 1930's and 1960's data for all species. The data manipulation is done using functions in the dplyr library. Base R functions could be used, but dplyr code is usually easier to read.

```
skokholm <- read.table('../skokholm.txt', header=T)

# keep only 1930's and 1960's data

decades <- skokholm %>%
  mutate(decade = 10*floor(year/10)) %>%
  filter(decade %in% c(1930, 1960))
```

create the mvabund object with just the species composition data We need to remove year and decade, so the data frame has only species information. Then, mvabund() converts the data frame to a matrix and checks that it is a reasonable matrix of species composition, i.e. no negative values.

```
# remove year and decade (the only two non-species composition variable)
temp <- decades %>% select(-year, -decade)

# then convert to a mvabund object
decades.mva <- mvabund(temp)

# check by printing the column names (only want the species variables)
# the mvabund object is a matrix, not a data frame, so need to use dimnames(), not names()
dimnames(decades.mva)[[2]]
```

```
##  [1] "Ho"   "Ap"   "Oo"   "Aa"   "Sv"   "Vv"   "Co"   "Cm"   "Cc"   "Pm"
## [11] "Cr"   "Es"   "Tm"   "Gc"   "Bb"   "Ma"   "Ap.1" "Hr"   "Sc"   "As"
## [21] "Fp"   "Cu"   "Ra"   "Er"   "Sr"   "Cx"   "Pp"   "Ac"   "Gg"
```

Because we want to compare two groups, convert decade (numeric) to a factor named decade.f.

```
decades <- decades %>%
  mutate(decade.f = factor(decade))
```

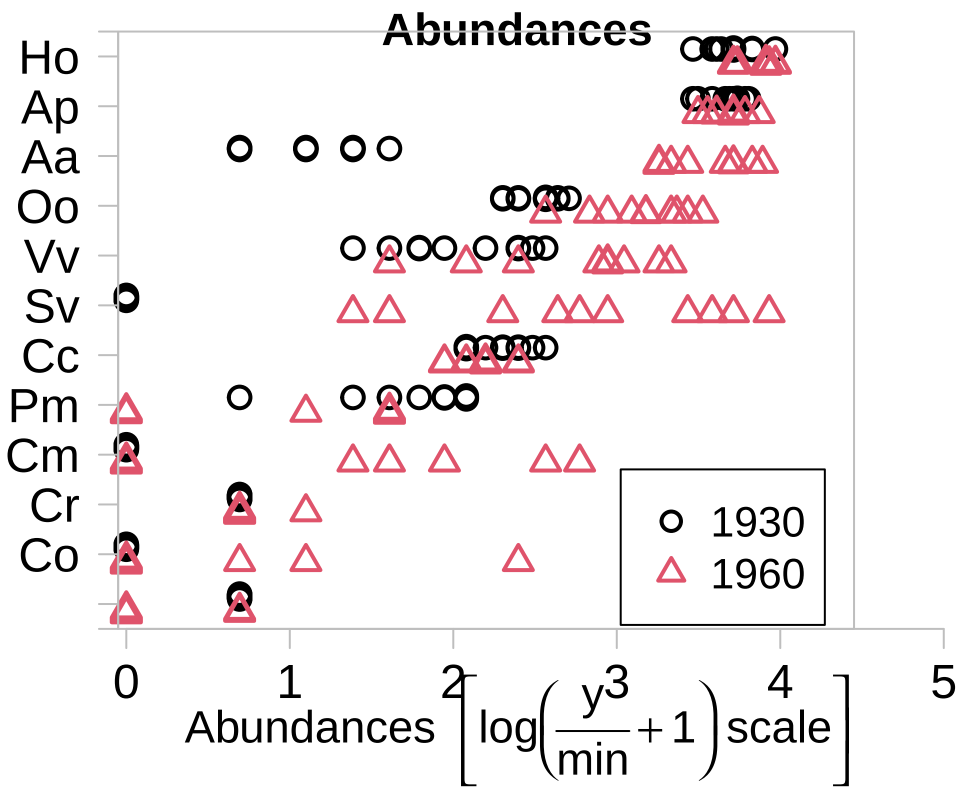
The mvabund library provides various potentially useful plots:

```
# plot the data, separate observations by decade
#  only shows the most abundant species
plot(decades.mva ~ decades$decade.f)
```

```
## Overlapping points were shifted along the y-axis to make them visible.

##
##  PIPING TO 2nd MVFACTOR

## Only the variables Ho, Ap, Aa, Oo, Vv, Sv, Cc, Pm, Cm, Cr, Co, Gc were included in the plot
## (the variables with highest total abundance).
```

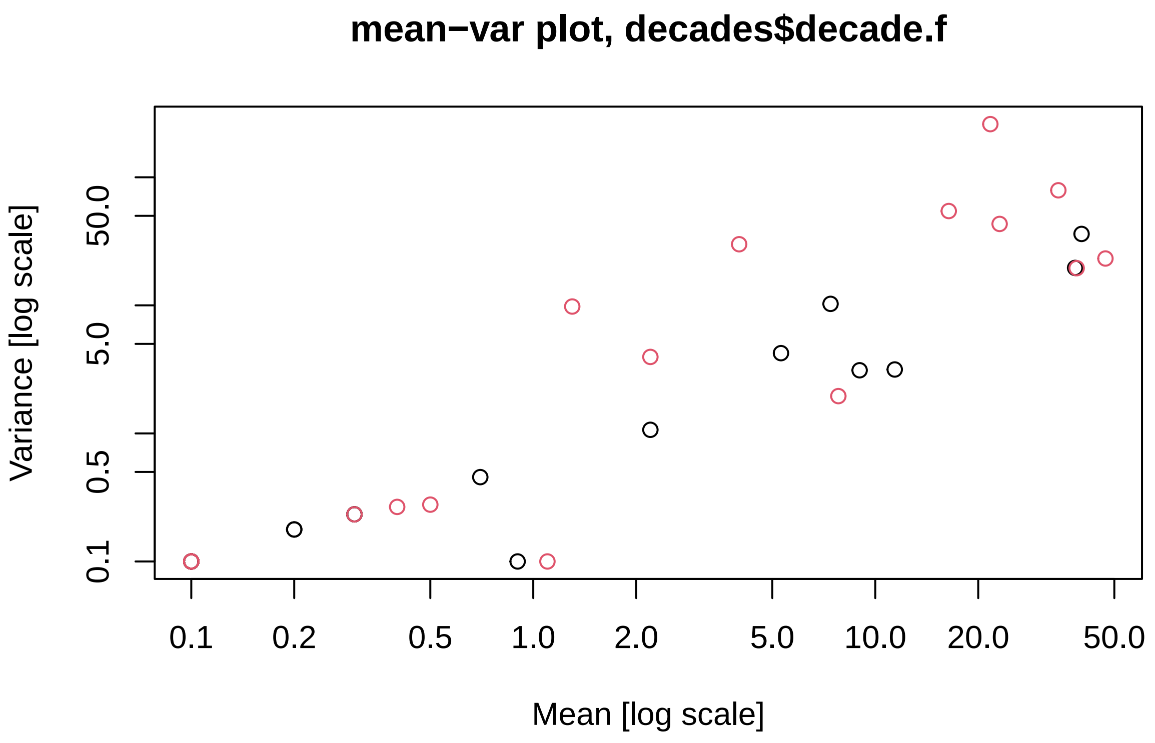


```
# look at mean vs variance for all species in both periods
meanvar.plot(decades.mva ~ decades$decade.f)
```

```
## START SECTION 2
## Plotting if overlay is TRUE

## using grouping variable decades$decade.f 19 mean values were 0 and could
##                                         not be included in the log-plot

## using grouping variable decades$decade.f 21 variance values were 0 and could not
##                                         be included in the log-plot
```



```
## FINISHED SECTION 2
```

Using manyglm()

manyglm() fits the glm to all species. The default distribution is a negative binomial distribution. family= will specify a new distribution. Currently, manyglm() provides binomial, poisson, negative.binomial, and Gamma distributions. manylm() fits normal distributions and manyany() provides access to just about any modeling function and error distribution.

Each species is assumed to have a different overdispersion when using a negative binomial distribution. However all observations for a species are assumed to have the same overdispersion.

The "response" variable is the mvabund object. The X variables (right hand side of the formula can be any R formula. data= specifies where to find the variables used in the right-hand side of the formula. I usually don't try to put the mvabund object (response variable) in the data frame. It's a separate object.

```
decades.glm <- manyglm(decades.mva ~ decade.f, data=decades)

decades.glm
```

```
##
## Call:  manyglm(formula = decades.mva ~ decade.f, data = decades)
## [1] "negative.binomial"
##
## Nuisance Parameter(s) phi estimated by the PHI method.
##    Ho     Ap     Oo     Aa     Sv     Vv     Co     Cm     Cc     Pm     Cr
## 0.007  0.007  0.009  0.027  0.550  0.119  6.181  3.264  0.007  0.007  0.006
##    Es     Tm     Gc     Bb     Ma   Ap.1     Hr     Sc     As     Fp     Cu
## 0.007  0.007  0.006  0.007  0.007  0.006  0.007  0.007  0.006  0.007  0.006
##    Ra     Er     Sr     Cx     Pp     Ac     Gg
## 0.006  0.006  0.006  0.006  1.050  1.050  0.006
##
## Degrees of Freedom: 19 Total (i.e. Null); 18 Residual
##
##                      Ho        Ap        Oo        Aa        Sv        Vv
## 2*log-likelihood:   -124.649  -119.002  -112.289  -101.642   -80.479  -119.700
## Residual Deviance:    12.526     9.214    20.091    14.905    10.762    20.138
## AIC:                 130.649   125.002   118.289   107.642    86.479   125.700
##                      Co        Cm        Cc        Pm        Cr        Es
## 2*log-likelihood:    -25.914   -46.167   -84.693   -85.731   -41.289   -15.330
## Residual Deviance:     6.423     9.527     5.326    32.505     0.676     7.330
## AIC:                  31.914    52.167    90.693    91.731    47.289    21.330
##                      Tm        Gc        Bb        Ma        Ap.1      Hr
## 2*log-likelihood:    -20.380   -33.224   -19.897   -23.537   -19.829   -13.224
## Residual Deviance:     7.766     7.224     1.897    11.537    11.829     7.224
## AIC:                  26.380    39.224    25.897    29.537    25.829    19.224
##                      Sc        As        Fp        Cu        Ra        Er
## 2*log-likelihood:    -17.043    -6.605   -10.438    -6.605    -6.605    -6.605
## Residual Deviance:    11.043     4.605     6.438     4.605     4.605     4.605
## AIC:                  23.043    12.605    16.438    12.605    12.605    12.605
##                      Sr        Cx        Pp        Ac        Gg
## 2*log-likelihood:     -6.605    -6.605     0.000     0.000    -6.605
## Residual Deviance:     4.605     4.605     0.000     0.000     4.605
## AIC:                  12.605    12.605     6.000     6.000    12.605
```

Printing the manyglm result for a negative binomial distribution provides:

- A list of the estimated overdispersion parameters for each species.

- a table with fit statistics for each species

manyglm() defines overdispersion as variance: mu + phi * mu^2, so phi = 0 is a Poisson distribution, no overdispersion.

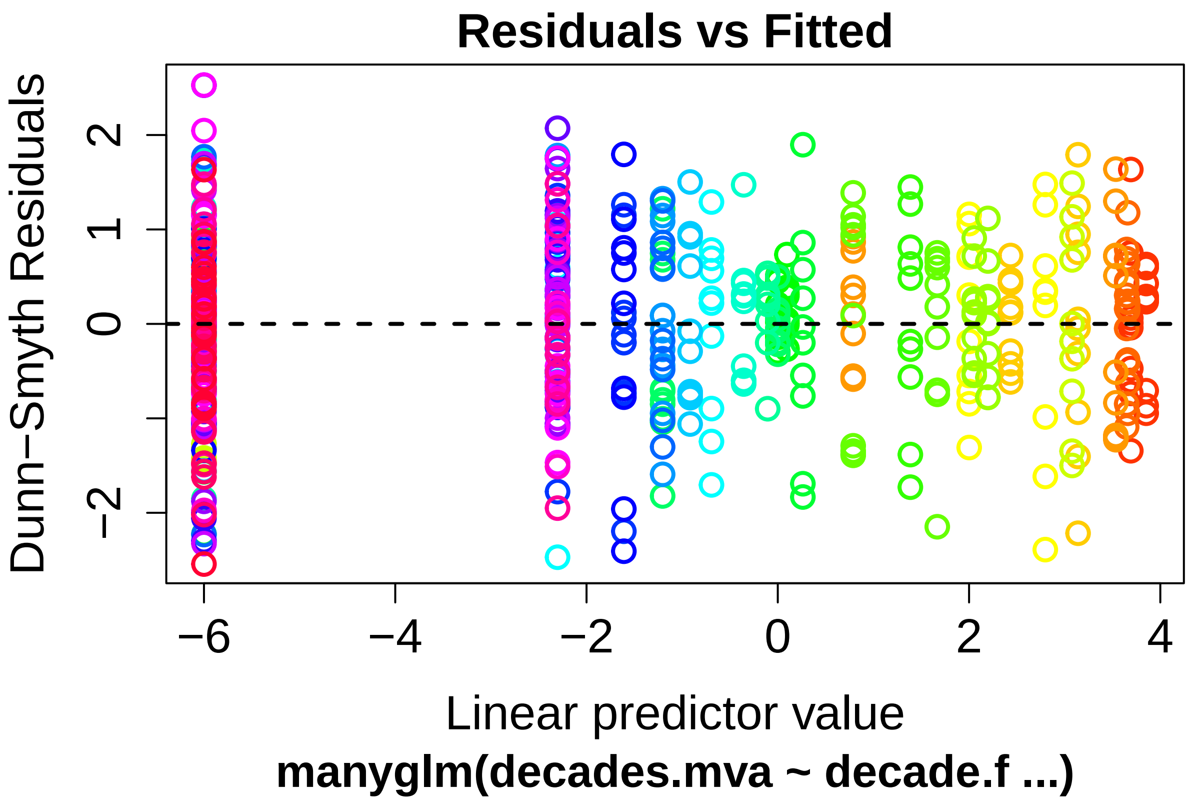
The reported fit statistics are 2*the log likelihood, the residual deviance, and the AIC statistic.

There are many of the usual helper functions:

- logLik(): extracts the log likelihood for each species
- AIC(): extracts the AIC statistic for each species
- predict(): predicts the mean for each species and observation. Default is on the link scale, i.e. log(mean) for a negative binomial. type='response' provides backtransformed predictions of mean counts.
- resid(): provides Dunn-Smyth = PIT residuals for each species and observation.

plotting the fitted model produces a plot of residuals vs predicted values. These residuals should look like a flat band. If the model is appropriate, the residuals should have a standard normal distribution. That means 95% of the residuals should be between -2 and 2.

```r
plot(decades.glm)
```



plot-1.pdf

Randomization based inference on a manyglm() result Note: These functions are often slow (seconds to minutes) because of doing 1000 randomizations.

- summary() will test each regression coefficient
- anova() will do sequential tests of model terms

For both, the default is to only provide the "all species" result. Adding p.uni='adjusted' or p.uni='unadjusted' will provide tests for each species. unadjusted provides results without any adjustment for testing multiple species. adjusted does that adjustment.

```
anova(decades.glm)
```

```
## Time elapsed: 0 hr 0 min 10 sec
```

```
## Analysis of Deviance Table
##
## Model: decades.mva ~ decade.f
##
## Multivariate test:
##             Res.Df Df.diff   Dev Pr(>Dev)
## (Intercept)     19
## decade.f        18       1 203.4    0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Arguments:
##  Test statistics calculated assuming uncorrelated response (for faster computation)
##  P-value calculated using 999 iterations via PIT-trap resampling.
```

which species changed? It is often interesting to look at individual species. To do this, you can examine the individual univariate p-values or look at species-specific changes in AIC between two models. The univariate test approach allows adjusting for multiple testing.

```
temp <- anova(decades.glm, p.uni='adjusted')
```

```
## Time elapsed: 0 hr 0 min 10 sec
```

```
# print out just a bit of the result, specifically the first 5 species:
temp$uni.p[,1:5]
```

```
##                Ho    Ap    Oo    Aa    Sv
## (Intercept)    NA    NA    NA    NA    NA
## decade.f    0.063 0.922 0.001 0.001 0.001
```

The output is a matrix with one row for each test and one column for each species. For this analysis, the first test is intercept = 0 (not very interesting); the second row is change between decades = 0 (what we care about).

Printing the output gives a lot of results. If we're only interested in which species show evidence of a change, we only want to find those with p-values for the decade effect that are < 0.05 (or 0.10).

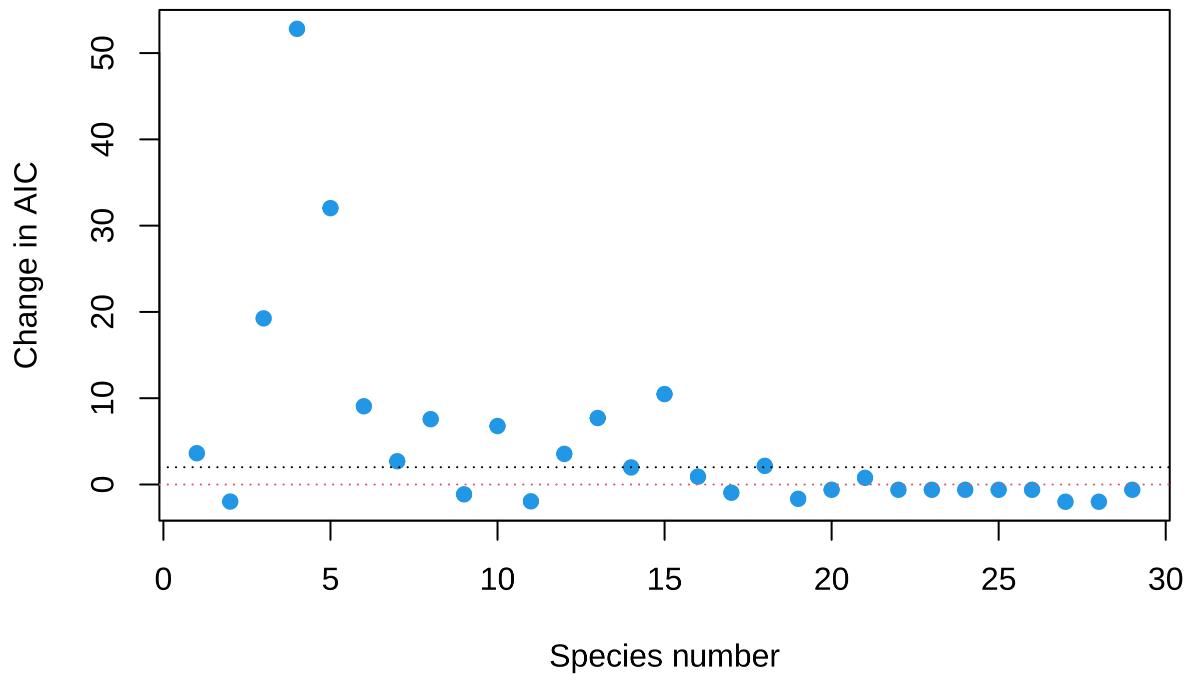
```
which(temp$uni.p[2,] < 0.05)
```

```
## Oo Aa Sv Vv Cm Pm Tm Bb
##  3  4  5  6  8 10 13 15
```

An alternative is to use a model selection approach. That is compare AIC statistics between the model that allows a change between decades and a model that forces both decades to have the same mean. As always with manyglm(), each species is fit separately. This approach doesn't adjust for multiple testing.

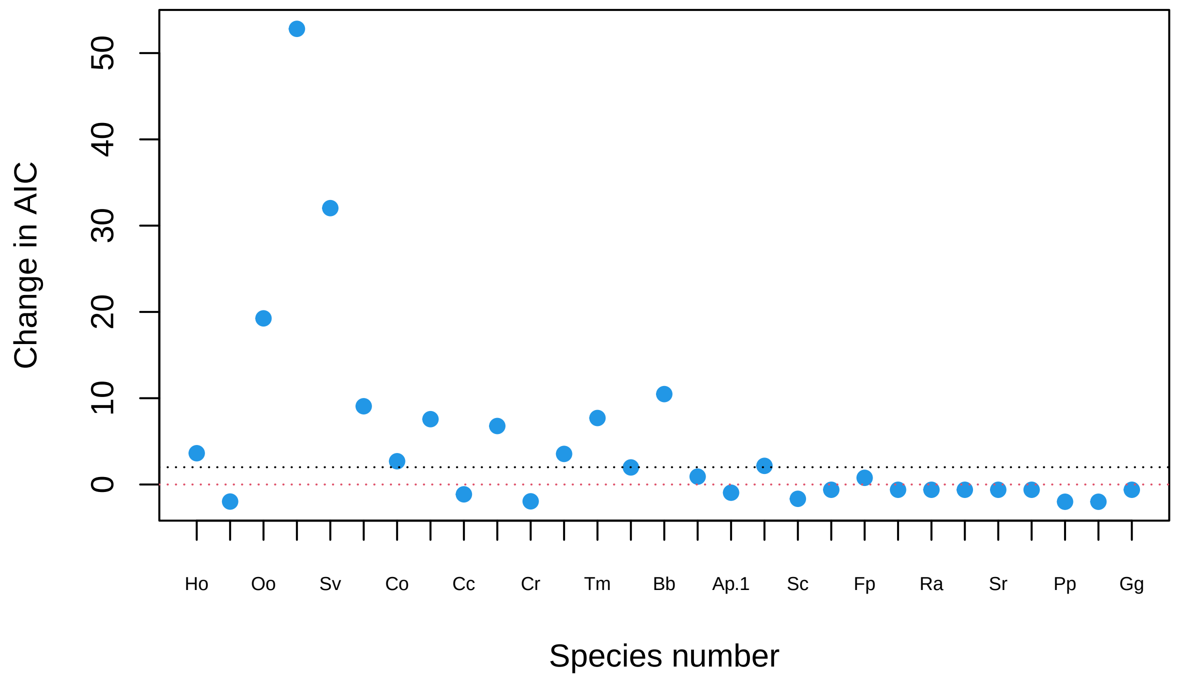
```
# fit the null model of no change
decades.glm0 <- manyglm(decades.mva ~ 1, data=decades)

# then comparing AIC statistics for each species between the two models
plot(AIC(decades.glm0) - AIC(decades.glm), pch=19, col=4,
  xlab='Species number', ylab='Change in AIC'  )
abline(h=2, lty=3)
abline(h=0, lty=3, col=2)
```



spp aic-1.pdf

```
# can get fancy and label X axis with the 2 letter species codes (reduced so all fit)
plot(AIC(decades.glm0) - AIC(decades.glm), pch=19, col=4,
  xlab='Species number', ylab='Change in AIC',
  xaxt='n')
axis(1, at=1:29, dimnames(decades.mva)[[2]], cex.axis=0.6)
abline(h=2, lty=3)
abline(h=0, lty=3, col=2)
```



spp aic-2.pdf

```
# which species have more than a 2 unit change in AIC?
# if you want the species numbers:
which(AIC(decades.glm0) - AIC(decades.glm) > 2)
```

```
##  [1]  1  3  4  5  6  7  8 10 12 13 15 18
```

```
# or get the species codes by subscripting the vector of names
dimnames(decades.mva)[[2]][AIC(decades.glm0) - AIC(decades.glm) > 2]
```

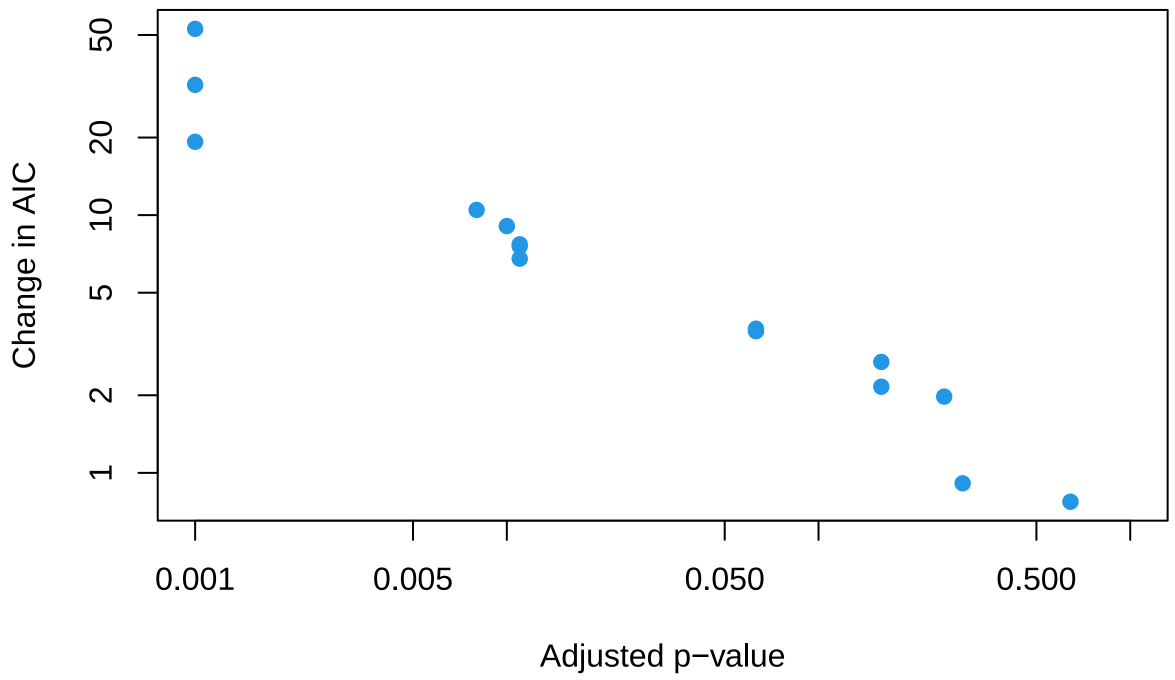
```
##  [1] "Ho" "Oo" "Aa" "Sv" "Vv" "Co" "Cm" "Pm" "Es" "Tm" "Bb" "Hr"
```

```

plot(temp$uni.p[2,], AIC(decades.glm0) - AIC(decades.glm), pch=19, col=4,
  xlab='Adjusted p-value', ylab='Change in AIC',
  log='xy')
```

and (out of curiosity), how does the adjusted p-value compare to the change in AIC?

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 14 y values <= 0 omitted from
## logarithmic plot
```



aic-1.pdf

The test and the AIC approach are very similar, they differ only in the cutoffs for 'signficant change'.

Fitting a regression model

We now consider all years of data and fit a regression model where each species's count follows a log-linear regression on year. This requires working with the full skokholm data set. We also subtract 1928 from all year values so the intercept (year 0) is at the start of the data set (1928), not very, very far in the past (0 CE).

```
# omit year from the species composition matrix
skokholm.mva <- mvabund(
  skokholm %>% select(-year) )
dimnames(skokholm.mva)[[2]]
```

```
##  [1] "Ho"   "Ap"   "Oo"   "Aa"   "Sv"   "Vv"   "Co"   "Cm"   "Cc"   "Pm"
## [11] "Cr"   "Es"   "Tm"   "Gc"   "Bb"   "Ma"   "Ap.1" "Hr"   "Sc"   "As"
## [21] "Fp"   "Cu"   "Ra"   "Er"   "Sr"   "Cx"   "Pp"   "Ac"   "Gg"
```

```
# subtract 1928 from year values, so intercept is abundance in 1928, not 0 CE.
skokholm <- skokholm %>%
  mutate(yearc = year - 1928)
```

This is a regression model, with a continuous covariate

```
skokholm.glm <- manyglm(skokholm.mva ~ yearc, data=skokholm)
anova(skokholm.glm)
```

```
## Time elapsed: 0 hr 0 min 27 sec
```

```
## Analysis of Deviance Table
```

```
##
## Model: skokholm.mva ~ yearc
##
## Multivariate test:
##             Res.Df Df.diff   Dev Pr(>Dev)
## (Intercept)     46
## yearc           45       1 442.6    0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Arguments:
##  Test statistics calculated assuming uncorrelated response (for faster computation)
##  P-value calculated using 999 iterations via PIT-trap resampling.
```

Analysis of species composition

All the previous analyses have examined counts of individuals. That is not the only way to define species composition. Consider 2 sites with 3 species each. Here are the counts:

- Site 1: 15, 5, 30
- Site 2: 75, 25, 150

Site 2 clearly has more total individuals than site 1, so assessing counts will tell you that the two sites have different abundances.

Species composition can also be defined as relative count, i.e. the proportion of each species. For these two sites, the proportions are:

- Site 1: 30%, 10%, 60%
- Site 2: 30%, 10%, 60%

The two sites have identical species composition. There are two ways to convert a glm for counts into an analysis of relative proportion.

- include an offset, here log(total), in the model. An offset variable has a fixed regression coefficient of 1. Since this is on the right-hand side of the model equation, it is equivalent to modeling log(count/effort). This assumes that the total is a fixed constant. Hence, it is best when you want to adjust for different sampling efforts, e.g. 1 hour sampling for site 1 and 5 hours for site 2.
- Include a different intercept for each site. This estimates the log total for each site and adjusts other inferences for estimating the site effects. This does not assume the log totals for each site are known.

The following interpretation assumes that you have used a log link. This is the default for Poisson and Negative Binomial distributions.

The model that includes site effects could be written as log(mean count) = row(= site) + column(=species) + row:column. The row effect tests whether all sites have the same mean abundance. The column effect tests whether all species have the same mean abundance (rarely interesting). The really interesting test is the row:column interaction, which tests whether the species proportions are the same at all sites.

Here is the code and anova results for the 1930's and 1960's Skokholm data:

```
decades.spcomp <- manyglm(decades.mva ~ decade.f,
    data=decades, composition=T)

anova(decades.spcomp)
```

```
## Time elapsed: 0 hr 5 min 56 sec
```

```
## Analysis of Deviance Table
##
## Model: decades.mva ~ cols + decade.f + rows + cols:(decade.f)
```

```
##
## Multivariate test:
##                Res.Df Df.diff   Dev Pr(>Dev)
## (Intercept)       579
## cols              551      28 695.0    0.001 ***
## decade.f          550       1  26.7    0.001 ***
## rows              531      19  44.5    0.004 **
## cols:decade.f     504      28 354.2    0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Arguments: P-value calculated using 999 iterations via PIT-trap resampling.
```

You see that there are significant differences in abundance between the two periods (decade.f test), between years within each period (rows, within period because its after decade.f in the model), and in relative species composition between periods (cols:decade.f).

The composition model works by turning the data from wide format (one column per species) into long format, one row for each combination of species and site. Fitting amodel with composition=T is very slow, because the model has more terms and has to deal with all observations at once (not just those for one species at a time). The unfortunate side effect of this is that the model fits only one overdispersion parameter, which is assumed to be the same for all species.

Fitting an offset to each row (site) is much faster. Here's how to do that:

Note: Fitting this model is VERY slow.

```r
# determine row totals = total # breeding pairs each year
#   easiest to work with the mvabund object because it's only got the counts
#   dimension 1 = site (2 = species)
total <- apply(decades.mva, 1, sum)

# and add log(total) to the data frame.
# NB: the offset MUST be the log total
decades$logtotal <- log(total)

# and fit the model with an offset for each site
#  a quirk: need to specifically indicate where to find the offset value
decades.offset <- manyglm(decades.mva ~ decade.f,
    data=decades, offset=decades$logtotal)

anova(decades.offset)
```

```
## Time elapsed: 0 hr 0 min 12 sec

## Analysis of Deviance Table
##
## Model: decades.mva ~ decade.f
##
## Multivariate test:
##             Res.Df Df.diff   Dev Pr(>Dev)
## (Intercept)     19
## decade.f        18       1 264.7    0.001 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## Arguments:
##  Test statistics calculated assuming uncorrelated response (for faster computation)
##  P-value calculated using 999 iterations via PIT-trap resampling.
```

The results have exactly the same format as those for counts. Because log total is included as an offset, the response variable is the % in the sample.

Blue-sky extensions

It is possible to fit a wide range of models using the manyany() function. This can interface with any R modeling function

For example: what about a smooth, possibly non-linear model for year? For a single species, that would be gam(count ~ s(year), family=nb) from the mgcv library.

The code below worked 2 years ago. It doesn't work in recent versions of mvabund. I have a bug request in to the folks who maintain mvabund. If you need to do something like this, see me to get the older version that will fit spline models.

```
library(mgcv)
```

```
## Loading required package: nlme

##
## Attaching package: 'nlme'

## The following object is masked from 'package:dplyr':
##
##     collapse

## This is mgcv 1.8-42. For overview type 'help("mgcv-package")'.
```

```
source('manyanyOld.r')
skokholm.spline <- manyanyOld('gam', skokholm.mva, skokholm.mva ~ s(yearc),
  family='nb', data=skokholm, get.what='models')
```

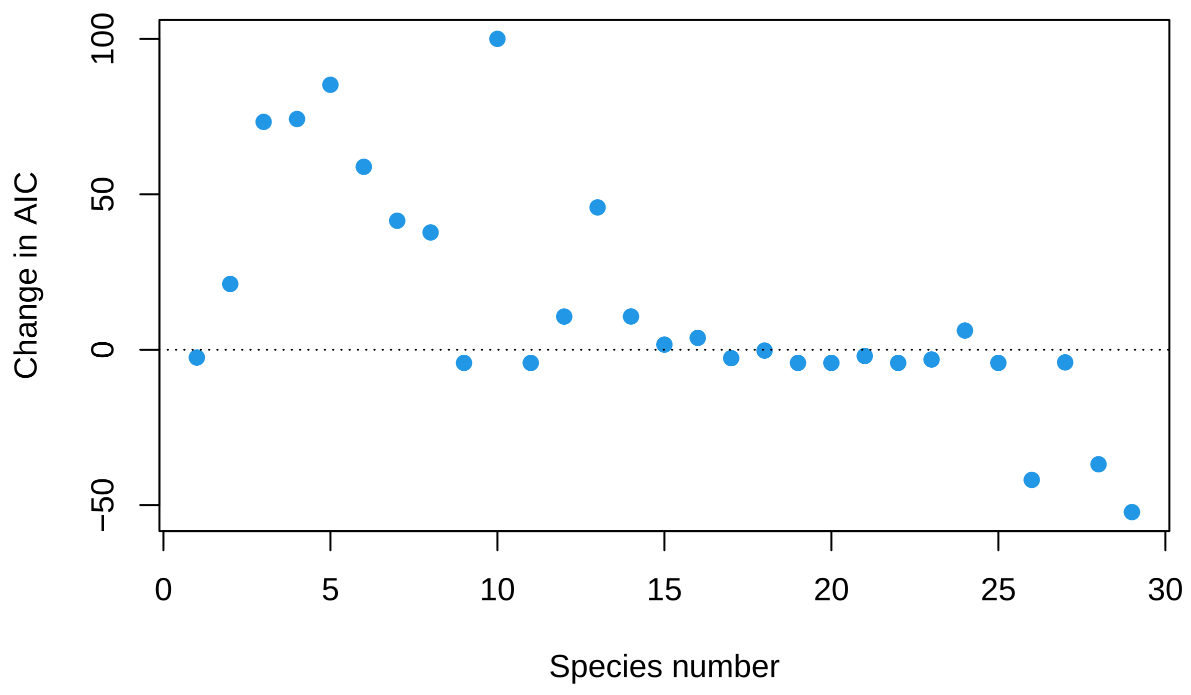
```
## Warning in newton(lsp = lsp, X = G$X, y = G$y, Eb = G$Eb, UrS = G$UrS, L = G$L,
## : Iteration limit reached without full convergence - check carefully

## Warning in newton(lsp = lsp, X = G$X, y = G$y, Eb = G$Eb, UrS = G$UrS, L = G$L,
## : Iteration limit reached without full convergence - check carefully

## Warning in newton(lsp = lsp, X = G$X, y = G$y, Eb = G$Eb, UrS = G$UrS, L = G$L,
## : Fitting terminated with step failure - check results carefully
```

```
# the warnings are for erratic species nesting in very few years.

# change in AIC between the linear and the spline model
#   shows which species have non-linear trends over time
plot(AIC(skokholm.glm) - AIC(skokholm.spline), pch=19, col=4,
  xlab='Species number', ylab='Change in AIC')
abline(h=0, lty=3)
```



If you want to do variable selection, glm1path() uses a LASSO algorithm to fit a sequence of species-specific models with increasing numbers of variables. It also returns the species-specific models that minimize the BIC criterion. See the help file for glm1path for more information.