Uncertainty quantification for NEON and other biodiversity network data with hierarchical Bayes

workshop background

James S. Clark

2016-10-11

Contents

1 Start from here

What do you need? Minimal background includes some experience with R and basic stats. In addition we'll need the following:

- Bring a laptop with R installed.
- `NEON.rdata`, `bayesReg.R`
- `gjam2.0` package loaded from CRAN for R studio

If you have time, browse the `vignettes` and examples on `help` pages before the workshop.

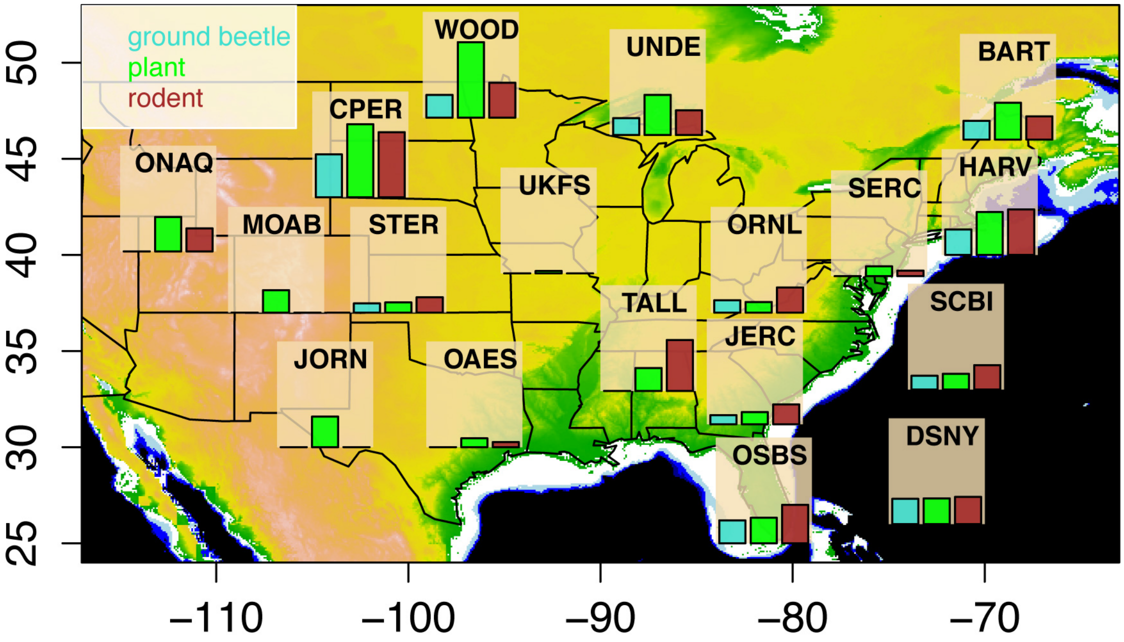

Figure 1: Sampling effort for species groups examined in the NEON examples that follow.

2 Uncertainty in network data

GJAM was motivated by the challenges of modeling distribution and abundance of multiple species, so-called joint species distribution models (JSDMs), where species and other attributes are recorded on different scales. Because species are not independent, they must be modeled together. But how does one combine species recorded on different scales? Data may be continuous, discrete, censored, composition, nominal, and ordinal–combinations of observations are not described by standard distributions. Equally challenging is the fact that most of the values in species-abundance data sets are typically zero. Finally, application of non-linear link functions as used in GLMs make it hard to interpret estimates from a model. Generalized joint attribute modeling (GJAM) provides a common framework for synthesis of ecological attribute and abundance data, both for estimating responses to the environmental and for prediction. I introduce the basic ideas needed for Bayesian analysis, followed by examples showing how GJAM is used integrate biodiversity data from networks.

I begin with the concept of *hierarchical modeling* and why it resides comfortably within the Bayesian paradigm. Uncertainty in parameters, process, and data can all be integrated in a common framework.

I then discuss GJAM for multivariate responses that can be combinations of discrete and continuous variables, where interpretation is needed on the observation scale. To allow transparent interpretation gjam avoids non-linear link functions.

The integration of discrete and continuous data on the observed scales makes use of *censoring*. Censoring extends a model for continuous variables across censored intervals. Continuous observations are uncensored. Discrete observations are censored versions of discrete data.

Censoring is used with the *effort* for an observation to combine continuous and discrete variables with appropriate weight. In count data, effort is determined by the size of the sample plot, search time, or both. It is comparable to the offset in generalized linear models (GLM). In count composition data, e.g. microbiome data, effort is the total count taken over all species. In paleoecological data it is the count for the sample. In `gjam` discrete observations can be viewed as censored versions of an underlying continuous space.

In the next section I use a simple variation on linear regression to illustrate the basic concept of censoring, thus allowing for zeros in continuouos data. I follow with examples showing application of `gjam` to network data.

3 Background Bayes

Hierarchical models are used to combine observations with processes and parameters. They exploit the Bayesian paradigm. Hierarchical models don't have to be Bayesian, but they usually are. I introduce these concepts together.

The basic building blocks are *likelihood* and *prior distribution*. Hierarchical models organize these pieces into graphs having several levels. These are the knowns (data and priors) and the unknowns (latent processes and parameter values). It is natural to think of inference this way:

$$[\text{unknowns} \mid \text{knowns}] = [\text{process, parameters} \mid \text{data, priors}]$$

If notation is unfamiliar: $[A]$ is the probability or density of event A, $[A, B]$ is the joint probability of A *and* B, and $[A|B]$ is the conditional probability of A given B.

This structure comes naturally to a Bayesian. It can be unwieldy for non-Bayesian methods. A hierarchical model typically breaks this down as

$$[\text{process, parameters} \mid \text{data, priors}] \propto$$
$$[\text{data} \mid \text{process, parameters}][\text{process} \mid \text{parameters}][\text{parameters} \mid \text{priors}]$$

The left hand side is the joint distribution of unknowns to be estimated, called the *posterior distribution.* This structure is amenable to simulation, using *Markov Chain Monte Carlo (MCMC). Gibbs sampling* is a common MCMC technique, because the posterior can be factored into smaller pieces that can be dealt with one at a time, as simpler conditional distributions. I begin with examples involving regression.

3.1 Traditional regression

For an example, load some NEON data,

```
library(gjam)
load('NEON.rdata')
ls()
```

```
## [1] "elist"     "formula"   "modelList" "rlist"     "type"      "typeNames"
## [7] "xdata"     "ydata"
```

I'll say more about the objects here later. For now I use this example to illustrate combinations of discrete and continuous data for a univariate case.

A regression model looks like this:

$$y_i \sim N(\mu_i, \sigma^2)$$

$$\mu_i = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_{p+1} x_{p+1,i}$$

The normal distribution on the right-hand side is called a *likelihood*. It is called a *linear model*, because it is a linear function of the parameters β.

Here is a linear regression fitted to the abundance of the first plant species using the standard function `lm` in R. In the `formula` I have specified a model containing main effects and interactions involving temperature and precipitation (`temp*pred`) and a `factor` variable `soil` type. I tell `lm` that the variables are to be found as columns in the `data.frame data`.

```
pindex <- which(type == 'p')                # plant columns
plants <- as.matrix(ydata[,pindex])         # the first plant species
p1     <- plants[,1]
data   <- cbind(p1,xdata)                   # put p1 in the data.frame
fitLM  <- lm( p1 ~ temp*prec + soil, data ) # linear regression
summary(fitLM)
```

```
##
## Call:
## lm(formula = p1 ~ temp * prec + soil, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.96650 -0.18195 -0.08270 -0.00318  1.85818
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -2.074e-01  2.230e-01  -0.930   0.3543
## temp             -8.433e-02  3.709e-02  -2.274   0.0248 *
## prec              4.476e-04  2.258e-04   1.982   0.0498 *
## soilInceptisols   5.221e-01  2.005e-01   2.604   0.0104 *
## soilMollisols    -5.322e-01  2.211e-01  -2.408   0.0176 *
## soilOthers       -2.777e-02  2.065e-01  -0.134   0.8933
## soilSpodHist     2.033e-01  1.885e-01   1.078   0.2830
## soilUltisols    -4.864e-02  2.023e-01  -0.240   0.8104
## temp:prec        4.649e-05  3.182e-05   1.461   0.1466
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5421 on 119 degrees of freedom
##   (37 observations deleted due to missingness)
## Multiple R-squared:  0.3745, Adjusted R-squared:  0.3325
## F-statistic: 8.907 on 8 and 119 DF,  p-value: 1.657e-09
```

The `formula p1 ~ temp*prec + soil` indicates that `p1` is the response variable, predicted by main effects and the interaction between `temp` and `prec` and the `factor soil` type. This `summary` says that main effects of `temp` and `prec` are *significant*. Two soil types are significantly different from the 'reference' soil type for this analysis, which happens to be Entisols.

3.2 What about the zeros?

The problem with this analysis is that regression does not allow zeros. Most of the observed values are zero:

```
hist(p1,nclass=100)
```

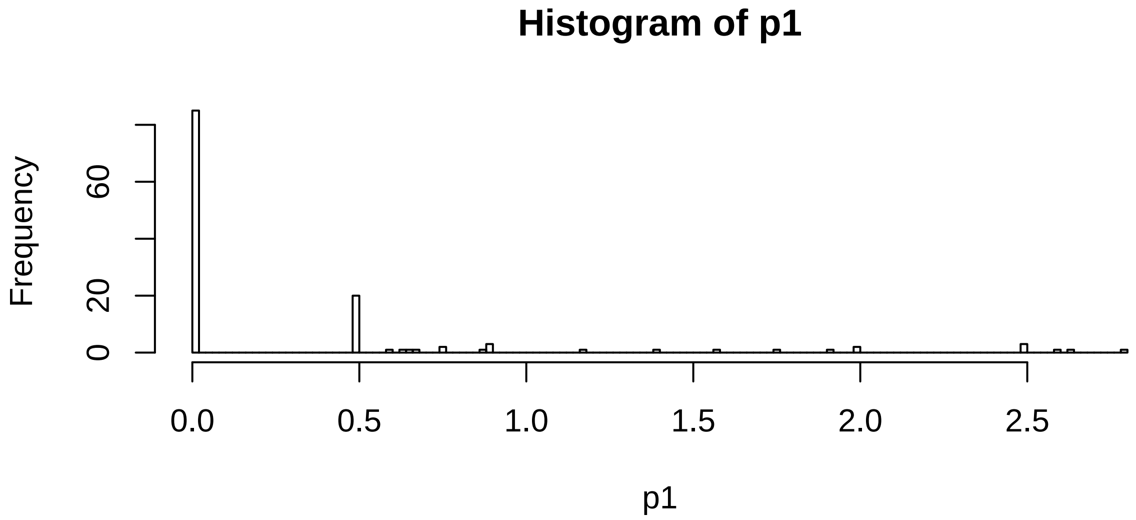

Figure 2: Plant cover values are mostly zero

This is not just a minor problem. If there were a few zeros, with most values bounded away from zero I might argue that it's close enough. That's not the case here.

If these were discrete data, I might turn to a zero-inflated Poisson or negative binomial model. In fact, these models sometimes work ok, provided there are not too many zeros, but here zeros dominate. Regardless, these are not discrete data, so I cannot use either.

I cannot add a small amount to each observation and then transform the data to a log scale. If I do that, every coefficient I estimate depends the arbitrary value I used.

3.3 Tobit regression

Censoring provides an attractive alternative. It allows me to combine continuous and censored data without changing the scale. In a Tobit model the censored value is zero, but it also works with other values. I introduce a latent variable W that is equal to the response Y whenever $Y > 0$. When $Y = 0$, then the latent variable is negative.

$$y_i = \begin{cases} w_i, & w_i > 0 \\ 0, & w_i \leq 0 \end{cases}$$

With this model the regression moves to the latent W,

$$w_i \sim N(\mu_i, \sigma^2) \times [\boldsymbol{\beta}, \sigma^2]$$

The second factor is the prior distribution, which, again, I've taken to be non-informative. Note that I now have w_i, not y_i, on the left-hand side. I have included the Tobit model as an option for a Bayesian regression in the file `bayesReg.R`. To make the code as transparent as possible I have simply assumed a non-informative prior distribution of the coefficients $\boldsymbol{\beta}, \sigma$.

First, to convince myself that a non-informative prior should give about the same answer as the tradition model fitted with `lm` I fit `bayesReg.R` without the Tobit option. In other words, like with the previous fit using `lm`, I am again ignoring the fact that the likelihood does not allow zeros:

```r
source('bayesReg.R')
fitBayes <- bayesReg(p1 ~ temp*prec + soil, data, TOBIT=F)
fitBayes$beta
```

```
##                     median      0.025      0.975
## intercept       -0.2192000 -6.589e-01  0.2579000
## temp            -0.0835300 -1.602e-01 -0.0079960
## prec             0.0004537 -1.992e-05  0.0009129
## soilInceptisols  0.5197000  1.303e-01  0.9145000
## soilMollisols   -0.5222000 -9.923e-01 -0.0968400
## soilOthers      -0.0224900 -4.416e-01  0.3828000
## soilSpodHist     0.1973000 -1.711e-01  0.5898000
## soilUltisols    -0.0463800 -4.481e-01  0.3524000
## temp:prec        0.0000462 -1.893e-05  0.0001107
```

I choose to organize my output a bit differently from `lm`, mine reporting 95% credible intervals, rather than standard errors. However, I see that point estimates for coefficients are nearly the same as I obtained with `lm`. I also see that the coefficients having credible intervals that do not span zero in the Bayesian analysis are the same coefficients that `lm` flagged as 'signficant'. (They need not be the same). More importantly, given the assumption there there is not point mass at zero the traditional and Bayesian methods tell a similar story.

Now allowing for the zeros in Y, the fitted coefficients differ substantially from both previous models:

```
fitTobit <- bayesReg(p1 ~ temp*prec + soil, data, TOBIT=T)
```

```
## fitted as Tobit model
```

```
fitTobit$beta
```

```
##                      median      0.025     0.975
## intercept       -6.789e+00 -1.026e+01 -4.192000
## temp            -1.282e+00 -1.990e+00 -0.816200
## prec             7.807e-03  4.845e-03  0.012340
## soilInceptisols -2.867e+00 -6.135e+00 -0.479500
## soilMollisols   -1.908e+01 -2.948e+01 -8.615000
## soilOthers      -7.454e+00 -2.813e+01  0.471400
## soilSpodHist    -5.181e+00 -9.481e+00 -2.095000
## soilUltisols    -2.286e+00 -4.142e+00 -0.746500
## temp:prec        7.708e-04  4.641e-04  0.001207
```

More coefficients are 'different' than zero, i.e., ones for which lower and upper bounds have the same sign. Moreover, root mean square prediction errors (rmspe) have also declined from 0.514 in the regression to 0.454 for the Tobit. And some that the traditional analysis flag as 'significant' have opposing sign with the Tobit model (inceptisols). Finally, with the Tobit model, the interaction `temp*prec` posterior does not include zero.

In summary, not only is the Tobit defensible as a valid model for continuous data with zeros–it also finds more effects and better predicts data than the traditional regression. `gjam` makes use of censoring to accommodate the many combinations of data types, in a multivariate setting.

4 gjam model summary

The basic model for combining data from many species together is outlined in a series of vignettes for the R package `gjam`.

```
browseVignettes('gjam')
```

To summarize, an observation consists of environmental variables and species attributes, $\{\mathbf{x}_i, \mathbf{y}_i\}$, $i = 1, ..., n$. The vector $\mathbf{x}_i$ contains predictors $x_{iq} : q = 1, ..., Q$. The vector $\mathbf{y}_i$

Figure 3: Ground beetles are collected in NEON pitfall traps, mostly predatory, many flightless. This species, now well-established in the eastern US, was introduced to slow the spread of gypsy moth (from Evans, *Beetles of Eastern North America*).

contains attributes (responses), such as species abundance, presence-absence, and so forth, $y_{is} : s = 1, ..., S$. The effort E_{is} invested to obtain the observation of response s at location i can affect the observation. The vignettes mentioned above provide many examples with real and simulated data. Here I show application to NEON data.

4.1 NEON ground beetles, mammals, plants

The NEON data used for this tutorial come from pitfall traps for ground beetles, 1-m^2 plots for plants, and traps for small mammals. The plant data are continuous (cover abundance) with point mass at zero. The other data types are counts. Each has a different level of effort, including numbers of traps, length of time trapping and plot area. The relative effect devoted to samping is shown in Figure 1.

The objects I loaded from the file `NEON.rdata` are R objects of this type:

object	type	comment
`xdata`	`data.frame`	explanatory variables
`ydata`	`data.frame`	species abundance
`typeNames`	`character vector`	data types
`elist`	`list`	sample effort
`modelList`	`list`	specify model, data
`rlist`	`list`	dimension reduction

If `storage.mode` for R objects is unfamiliar to you, it's worth skimming the `help` files.

As before, I can think about the data as consisting of *predictors* X and *responses* Y. The difference now is that the response is not only multivariate, but also on different scales. The predictors, along with some additional useful information about plots, are contained in a `data.frame xdata`. Here are a few rows,

```
xdata[1:3,]
```

```
##                lon       lat elev     soil         u1         u2         u3
```

The `data.frame ydata` holds species abundance data as plots × species. The columns in `ydata` are species. They are of three types indexed by a `character vector type`,

```
type
```

```
##   [1] "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "c"
##  [18] "c" "c" "c" "c" "c" "c" "c" "c" "c" "c" "p" "p" "p" "p" "p" "p" "p"
##  [35] "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p"
##  [52] "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p"
##  [69] "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p"
##  [86] "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "p" "m" "m" "m" "m"
## [103] "m" "m" "m" "m" "m" "m" "m" "m" "m" "m" "m" "m" "m" "m" "m"
```

There is one value in `type` for each column in `ydata`. The first set of columns, labeled `"c"`, refers to carabids (ground beetles). Then come columns for plants (`type == "p"`). Finally, there are mammal columns (`"m"`). Here are the numbers for each species group,

```
table(type)
```

```
## type
##  c  m  p
## 27 19 71
```

Type `"c"` and `"m"` are counts (numbers in pitfall and mammal traps, respectively). Note that these columns have only integer values. Type `"p"` are continuous. These types are recorded in the `character vector typeNames` as disrete abundance (`"DA"`) and continuous abundance (`"CA"`),

```
typeNames
```

```
##   [1] "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA"
##  [15] "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "CA"
##  [29] "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA"
##  [43] "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA"
##  [57] "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA"
##  [71] "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA"
##  [85] "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA" "CA"
##  [99] "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA" "DA"
## [113] "DA" "DA" "DA" "DA" "DA"
```

Collection of these different data types required different amounts of effort. The object `elist` is a `list` containing the column numbers that hold effort information, the `vector elist$columns` and a $n \times S$ `matrix elist$values` containing the amount of effort per plot and species group. These values can be plot-days, plot area, and so on. I could create `elist` like this: `elist <- list(columns = 1:S, values = effort)`.

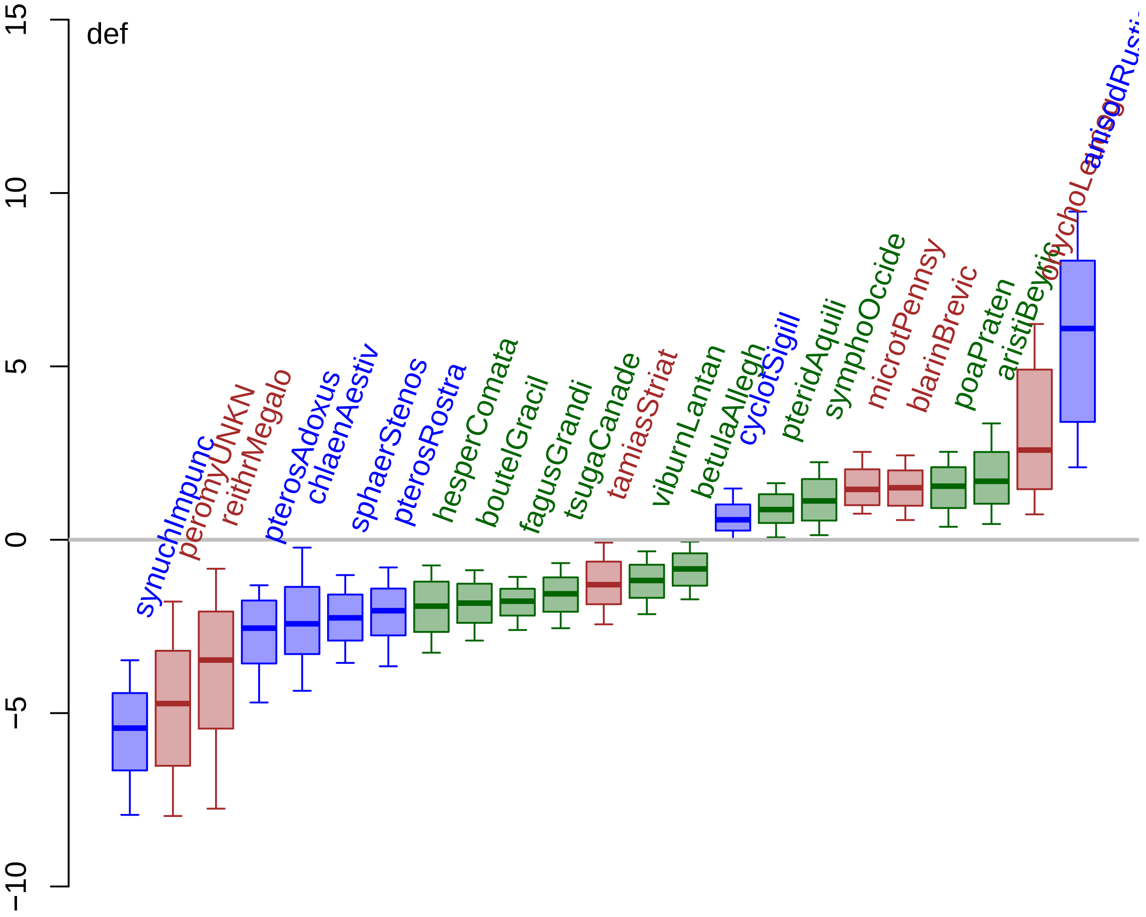

Figure 4: Examples of fitted coefficients. Using the vectors `boxCol` and `boxBorder` passed to `gjamPlot` in `plotPars` to specify colors for species groups I have highlighted carabids in blue, plants in green, and mammals in brown. See the help page for `gjamPlot` and `gjam vignettes`.

4.2 Dimension reduction

The sizes of objects discussed in the previous section are

```
dim(xdata)
```

```
## [1] 165  12
```

```
dim(elist$values)
```

```
## [1] 165 117
```

```
dim(ydata)
```

```
## [1] 165 117
```

All of these objects have the same number of rows (plots). `elist$values` and `ydata` have the same number of columns, the number of species. `xdata` has a different number of columns, including both predictors and other information about plots. There are already some clear challenges here. We have nearly as many species to fit ($S = 117$) as we have samples ($n = 165$). Suppose the number of predictors in the model is Q. The model we fit will have a $Q \times S$ matrix of coefficients $\mathbf{B}$ and a $S \times S$ covariance matrix Σ matrix. The latter has $S(S+1)/2$ unique coefficients to estimate. Thus, for our $S = 117$ species a model with 5 predictors requires $QS + S(S+1)/2 = 7488$ estimates.

Large covariance matrices are a problem, because they have to be inverted. An $S \times S$ covariance matrix requires order S^3 operations. The size of the problem has to be reduced.

The high dimensionality is addressed with dimension reduction, methods for finding a lower dimensional representation of the covariance matrix. The idea here is to summarize the $S \times S$ matrix Σ with a much smaller $N \times r$ matrix. A dimension reduction method has been developed for `gjam`. Here I specify a much reduced version of Σ in a list containing these two values:

```
rlist
```

4.3 Fitting the model

I place the elements of the model into `modelList`,

```
modelList <- list(ng=500, burnin=50, typeNames = typeNames,
                  effort = elist, reductList = rlist)
```

In `modelList` I provide the number of Gibbs steps `ng`, the number of iterations to discard `burnin`, the dimension reduction values `reductList`, and variable names in `xdata` that should not be standardized `notStandard`. Any predictors that are not `factors` and not included in `notStandard` are standardized. However, all coefficients are returned on their original scales. In other words, the coefficients in matrix $\mathbf{B}$ have the expected dimension of Y/X.

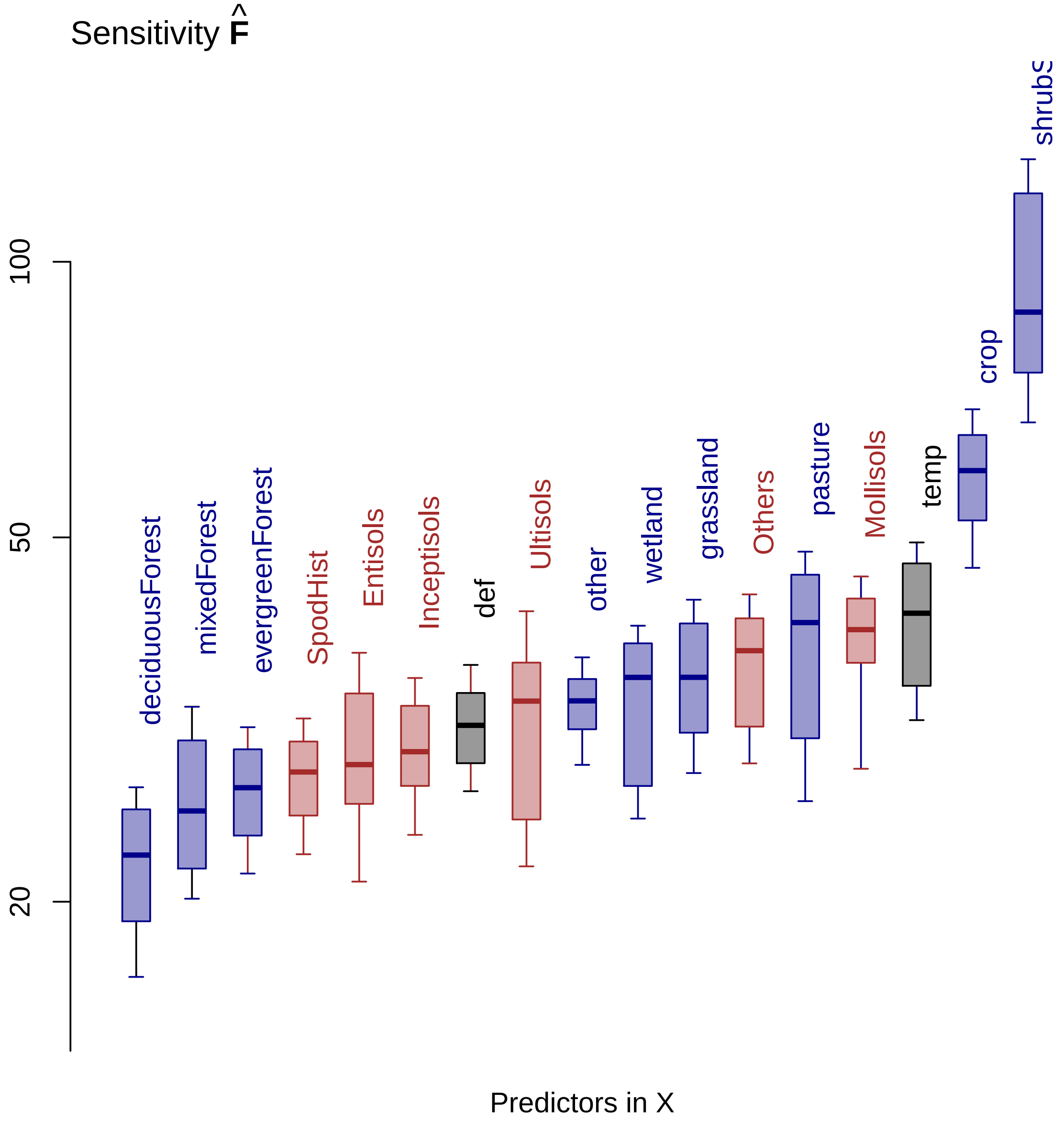

Figure 5: Sensitivity coefficients integrate the effects of predictors across all responses (species) in the model.

I fit the model using the function gjamGibbs standard R syntax for formula,

```
out1    <- gjamGibbs(~ temp + def + soil + nlcd, xdata = xdata, ydata, modelList)
```

I can plot the output here:

```
S <- length(typeNames)
specColor <- rep('brown', S)
specColor[type == 'p'] <- 'darkgreen'
specColor[type == 'c'] <- 'blue'
plotPars  <- list(GRIDPLOTS=T, CLUSTERPLOTS=T, SMALLPLOTS=F, SAVEPLOTS=F,
                  specColor = specColor)
fit       <- gjamPlot(out1, plotPars)
```

The options for plots is illustrated with vignettes at http://sites.nicholas.duke.edu/clarklab/code/.

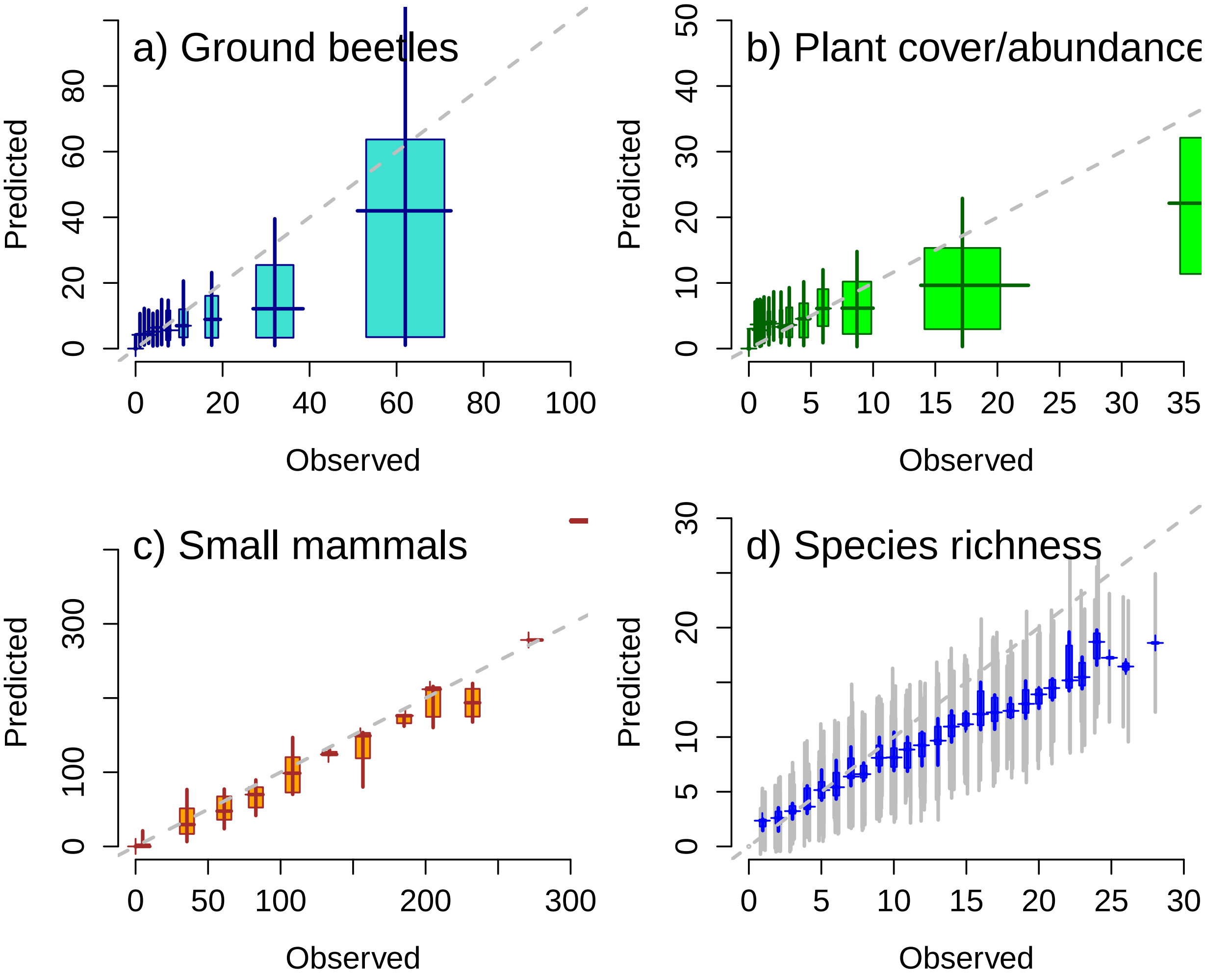

Figure 6: Predictive capacity of the model by species group (a - c) and for species richness (d).

4.4 NEON values are censored

I included Figure 2 without commenting on something that seemed odd to me; beyond all the zeros–there is also point mass at 0.5. We then learned from NEON that values of greater than zero but less than 0.5% cover are assigned 0.5. Examples of this type of censoring implemented to speed data collection are common in ecological data. In many cases I might simply ignore this as a minor detail. I worried about it here, because 0.5 is the second most common value in zero dominated data. In other words, the value 0.5 dominates the non-zero observations. It would be remarkable if piling most of the observations on this one value had no impact on inference. Fortunately, `gjam` readily accommodates censoring (see `gjamCensorY help` page), so I checked. Here is the censoring for the values at `0` and `0.5`:

```
values <- c(0, 0.5)                      # censored values in data
intvls <- rbind( c(-Inf, 0), c(0, .5) )  # partition for those values
censor <- gjamCensorY(values = values, intervals = intvls,
                      y = ydata, whichcol = pindex)$censor
censor <- list('CA' = censor)
modelList$censor <- censor
censor
```

The `modelList` now includes the `censor list`, named for the `data typeNames 'CA'`, and including the `values` that indicate censoring `(0, 0.5)` and the intervals they represent. The `censor list` is appended to the `modelList`. Here is the analysis again with censored data and a few plots showing substantial effects:

```
out2    <- gjamGibbs(~ temp + def + soil + nlcd,
                       xdata = xdata, ydata, modelList = modelList)

# comparisons of censored and non-censored predictions:
wf <- which(is.finite(plants))              #only for non-missing values
par(mfrow=c(2,2),bty='n', mar=c(5,5,1,1))
plot(plants[wf],out1$modelSummary$yMu[,pindex][wf], cex=.2,
     xlab='All plant species', ylab='Predicted')
points(plants[wf],out2$modelSummary$yMu[,pindex][wf], cex=.1, col=2)
abline(0,1,lty=2)
legend('topleft',c('uncensored','censored'), text.col=c(1,2),
       bty='n')

plot(plants[wf],out2$modelSummary$yMu[,pindex][wf], cex=.2,
     xlim=c(0,5), ylim=c(0,5), xlab='Observed',ylab='Censored prediction')

plot(out1$parameterTables$fBetaMu, out2$parameterTables$fBetaMu, cex=.2,
     xlab='Uncensored estimates',ylab='Censored')
abline(0,1,lty=2)
```

The panel at upper left suggests that the predictions for the first model are not bad. However, it's hard to see how predictions might be affected down in the range where censoring occurs. Many species are mostly zero, like species 1 (top right), where the censored model knows to allow predictions over continuous positive values, including the censored interval (vertical axis). The big differences stand out in estimates of coefficients (lower right). Thus, with high numbers of zeros, proper censoring can be critical.

4.5 Interpreting a big model

The many fitted coefficients required for a model with many species presents visualization challenges. Here I provide some interpretation for output available from `gjamPlot`.

Estimates from the model are summarized in several formats. Box and whisker plots of 68% and 95% credible intervals default to include only coefficients having 95% intervals that exclude zero (Fig. 5). These plots will be rendered to the screen or written to `.pdf` files if `SAVEPLOTS = T` is included in `plotPars`. As with all plots that follow the uncertainty integrates parameter, process model, and data.

The overall sensitivity can be evaluted with inverse prediction of input variables. In this example the predictors having greatest impact are climate (temperature `temp`, deficit `def`, and thermal energy `therm`), soil (variable names beginning with `soil`) and nlcd cover types (variable names beginning with `nlcd`) (Fig. 6). The smallest effect in this example are slope-aspect variables (`u1`, `u2`, `u3`).

The capacity to predict data is evaluated both for inputs in X and outputs in Y. The observed vs predicted values for Y are shown in Figure 7a - c. Unlike species distribution models (SDMs), which predict either distribution or abundance, `gjam` predicts both. Species richness (Fig. 7d) can be predicted from the same model as abundance, because each species has an explicit probability of zero.

The inverse prediction of continuous inputs in X is shown in Figure 8 and of factors in Figure 9. These confident predictions say that the community is diagnostic of the site, include discrete factors such as soils and cover type (Fig. 9). Although specific indicator species rarely provide good estimates of environment `gjam` shows that the full community is a fingerprint for the environment. Here again, we have full uncertainty in the inverse predictions.

Commonalities in responses across species help to define communities on the basis of *response*, rather than distribution or abundance. The clustering of responses in Figure 10 can help us rethink community relationships across these diverse species groups.

In remaining figures are examples of prediction out-of-sample using `gjamPredict`. Note these predictions derive from only a handfull of NEON sites (Fig. 1) and include both presence-absence and abundance with full uncertainty.

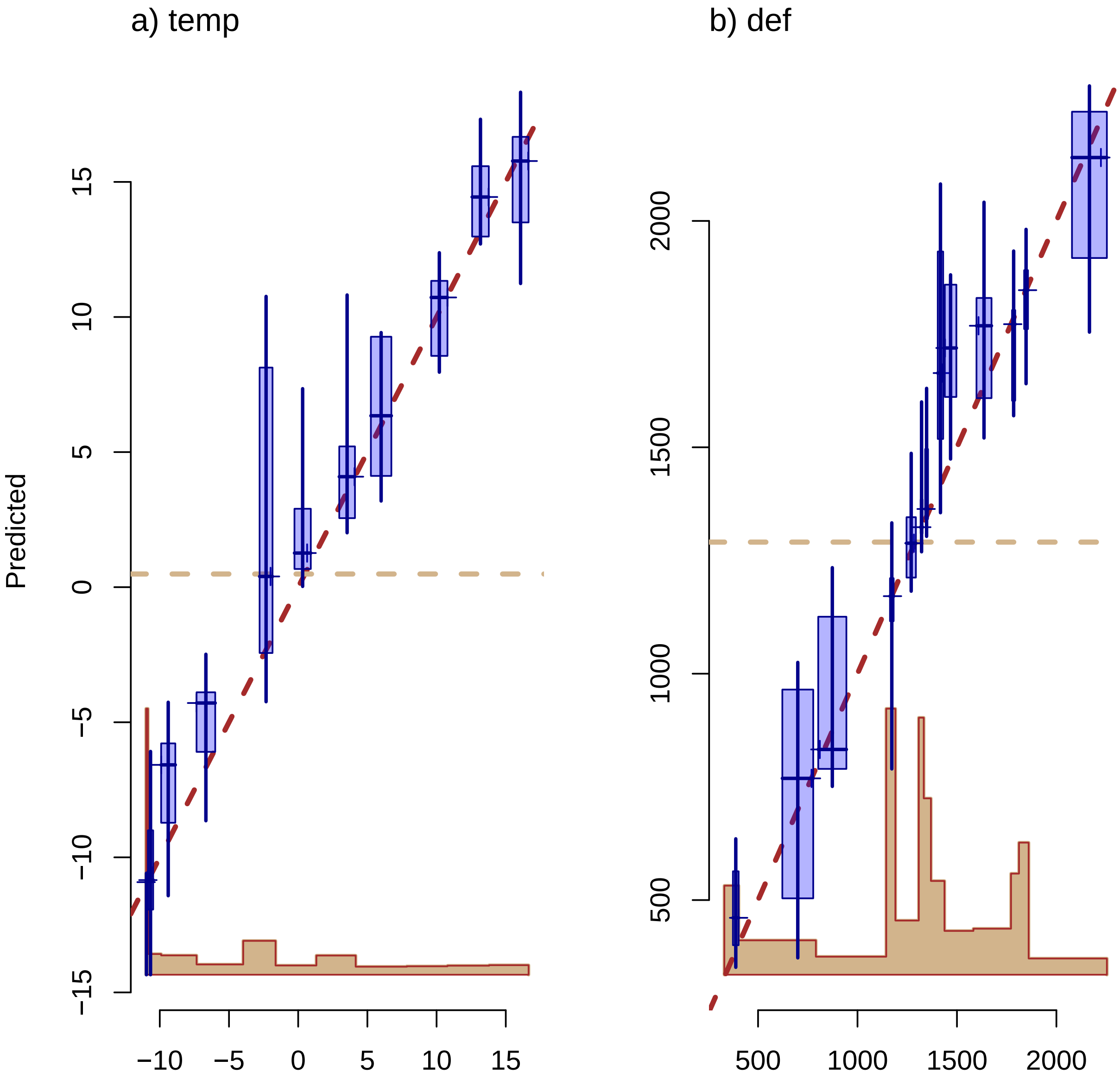

Figure 7: Inverse predictive capacity of the model for the inputs in the model. Inputs that can be accurately predicted by model responses have large impact across the full community of responses.

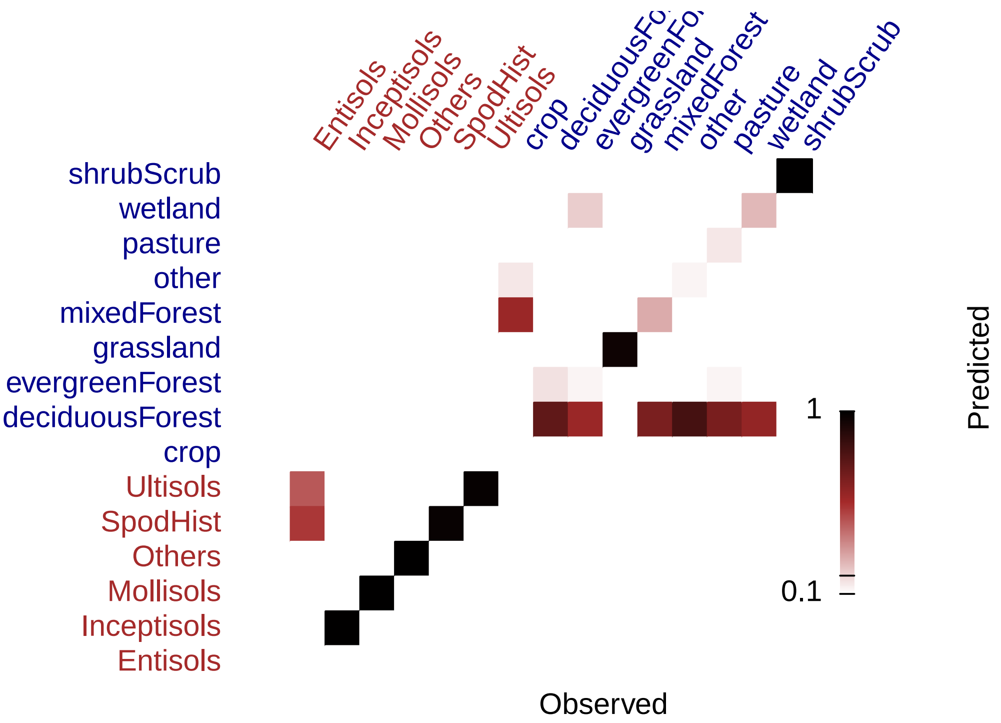

Figure 8: Inverse predictive of two multi-level factors, including soil and cover types, indicate that the community is highly diagnostic of the environment.

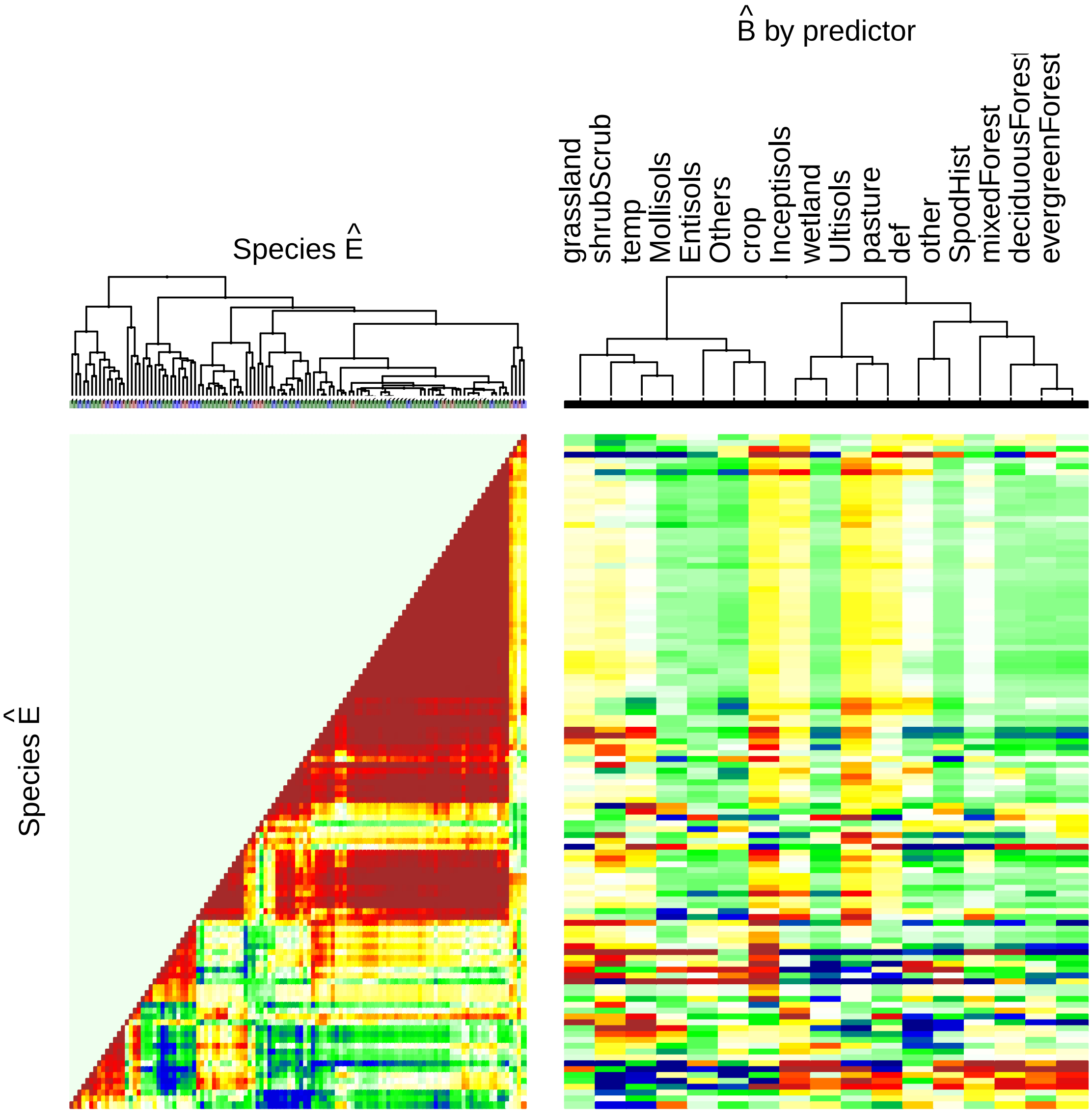

Figure 9: Species similarities in their responses to environmental variables in X. Warm colors are similar species pairs, and vice versa. Species are rows. At left species are also columns. Clusters of warm colors indicate similar responses to X. At right is the fitted coefficient matrix $\mathbf{B}$, with columns as predictors. In this example there are not strong patterns in $\mathbf{B}$ that match similar groups at right.

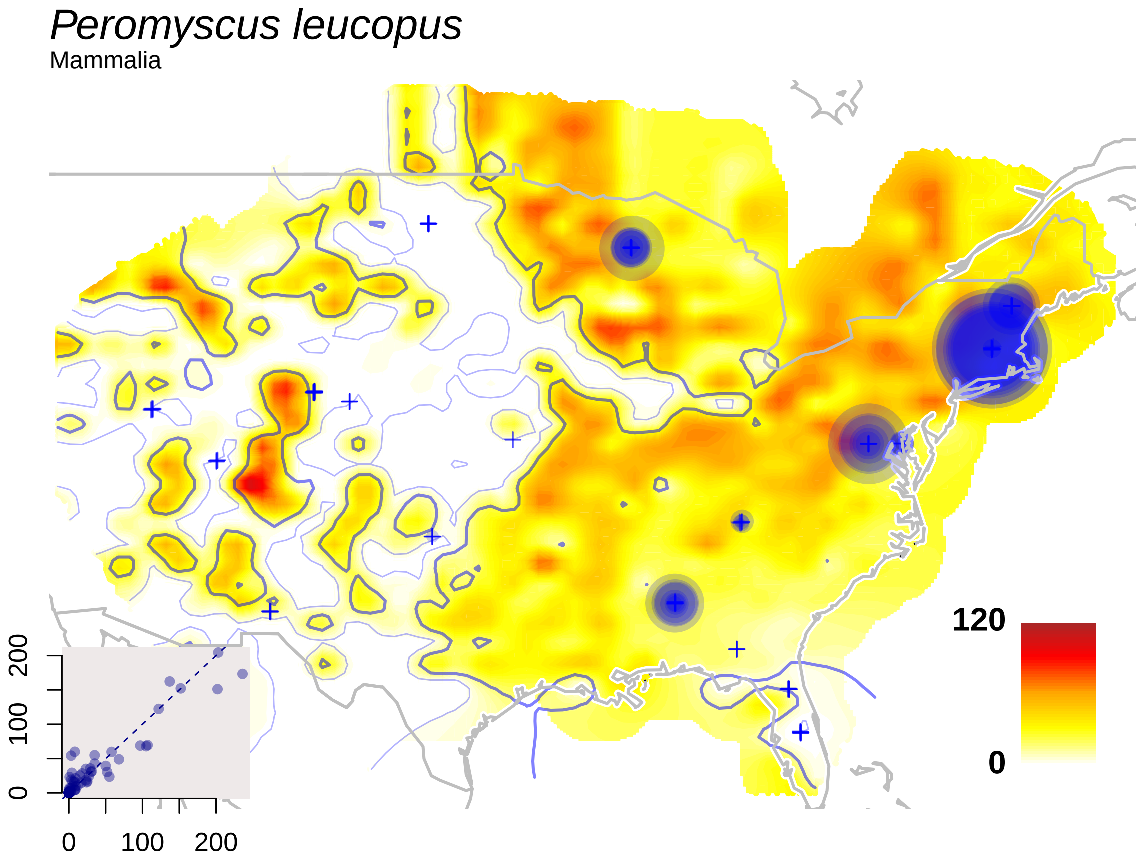

Figure 10: Out-of-sample predictive distributions for the white-footed mouse, including abundance (shaded surface), presence-absence probabilities (0.1, 0.5, in blue), and observations at NEON sites (blue circles).

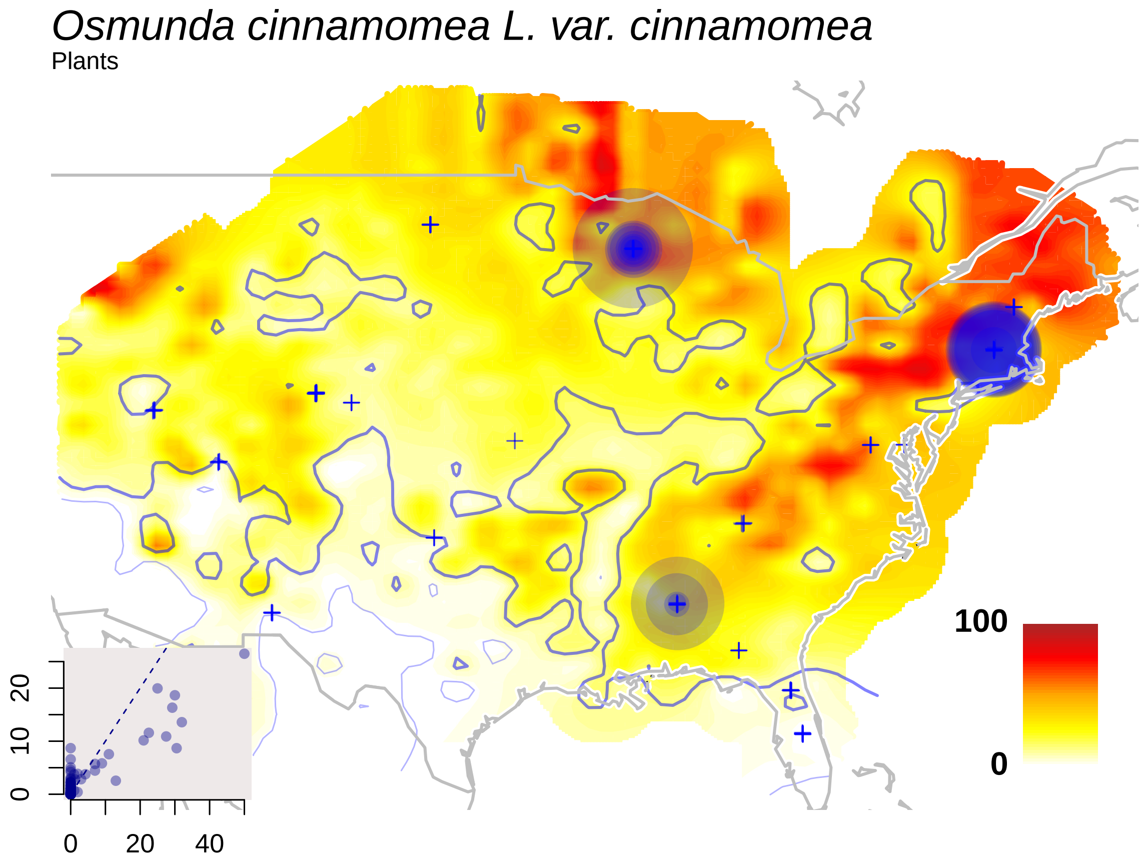

Figure 11: Out-of-sample predictive distributions for cinnamon fern. Symbolism follows Figure 11.

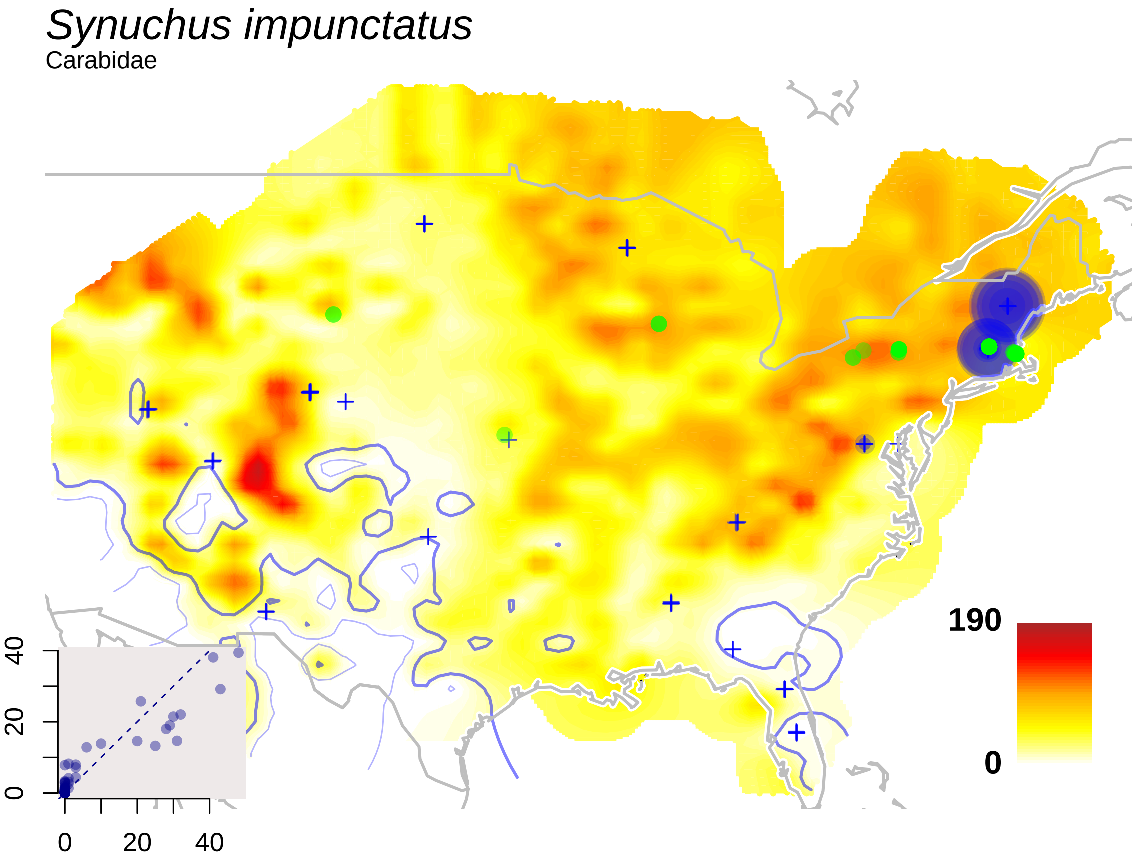

Figure 12: Out-of-sample predictive distributions for a gastropod-feeding ground beetle. Symbolism follows Figure 11.

5 Other resources

`help` files and `vignettes` provide fast, simple examples for many data types

Publications available at http://sites.nicholas.duke.edu/clarklab/ include the following:

Clark, J.S. 2016. Why species tell us more about traits than traits tell us about species: Predictive models. Ecology, in press.

Clark, J.S., D. Nemergut, B. Seyednasrollah, P. Turner, and S. Zhang. 2016. Generalized joint attribute modeling for biodiversity analysis: Median-zero, multivariate, multifarious data, Ecological Monographs, in press.

Taylor-Rodriguez, D., K. Kaufeld, E. Schliep, J. S. Clark, and A. Gelfand, 2016. Joint Species distribution modeling: dimension reduction using Dirichlet processes, Bayesian Analysis, in press.