Generalized joint attribute modeling - gjam

James S. Clark

2016-08-08

Contents

citation:

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, in review.

files are found here

gjam vignettes:

1. *Generalized joint attribute modeling - gjam*: **this overview**
2. *Dimension reduction in gjam*: application to many response variables ('Big-S')
3. *Trait modeling in gjam*: ecological trait analysis

Overview

`gjam` models multivariate responses that can be combinations of discrete and continuous variables, where interpretation is needed on the observation scale. It 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. Some species groups are counted. Some may be continuous cover values or basal area. Some may be recorded in ordinal bins, such as 'rare', 'moderate', and 'abundant'. Others may be presence-absence. Some are composition data, either fractional (continuous on (0, 1)) or counts (e.g., molecular and fossil pollen data). Attributes such as body condition, infection status, and herbivore damage are often included in field data. To allow transparent interpretation `gjam` avoids non-linear link functions.

To combine different types of observations on their respective scales gjam defines three elements: representations in a *continuous space*, in a *discrete space*, and a *partition* of continuous space that joins them.

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. Censored observations are discrete and can depend on sample effort.

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, effort is the total count taken over all species. In PCR, effort is the number of reads for the sample. 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.

Model summary

The basic model is detailed in Clark et al. (2016). 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 $\mathbf{y}_i$ 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 combinations of continuous and discrete measurements in observed $\mathbf{y}_i$ motivate the three elements of `gjam`:

- A length-S vector $\mathbf{w}_i \in \Re^S$ represents response $\mathbf{y}_i$ in continuous space. This continuous space allows for the dependence structure with a covariance matrix. An element w_{is} can be known (e.g., continuous response y_{is}) or unknown (e.g., discrete responses).
- A length-S vector of integers $\mathbf{z}_i$ represents $\mathbf{y}_i$ in discrete space. Each observed y_{is} is assigned to an interval $z_{is} \in \{0, ..., K_{is}\}$. This discrete space allows for error in discrete variables; zero-inflation is a common example, where a sample is assigned $y_{is} = 0$, when in fact $z_{is} \neq 0$. The number of intervals K_{is} can differ between observations and between species, because each species can be observed in different ways.
- The partition of continuous space at points $p_{is,z} \in \mathcal{P}$ defines discrete intervals z_{is}, thus connecting continuous w_{is} and discrete z_{is}. Tow values $(p_{is,k}, p_{is,k+1}]$ bound the k^{th} interval of s in observation i. Intervals are contiguous and provide support over the real line $(-\infty, \infty)$. For discrete observations, k is a censored interval, and w_{is} is a latent variable. The set of censored intervals is $\mathcal{C}$. The partition set $\mathcal{P}$ can include both known (discrete counts, including composition data) and unknown (ordinal, categorical) points.

An observation y maps to both w and z,

$$y_{is} = \begin{cases} w_{is} & \textit{continuous} \\ z_{is}, p_{is,k} < w_{is,k} < p_{is,k+1} & \textit{discrete} \end{cases}$$

Effort E_{is} affects the partition for discrete data. For the simple case where there is no error in the assignment of discrete intervals, $\mathbf{z}_i$ is known, and the model for $\mathbf{w}_i$ is

$$\mathbf{w}_i|\mathbf{x}_i, \mathbf{y}_i, \mathbf{E}_i \sim MVN(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}) \times \prod_{s=1}^{S} \mathcal{I}_{is}$$

$$\boldsymbol{\mu}_i = \boldsymbol{\beta}'\mathbf{x}_i$$

$$\mathcal{I}_{is} = \prod_{k \in \mathcal{C}} I_{is,k}^{I(y_{is}=k)} (1 - I_{is,k})^{I(y_{is} \neq k)}$$

where $I_{is} = I(w_{is}E_{is} \in \mathbf{p}_{is,k})$, $\mathcal{C}$ is the set of discrete intervals, $\boldsymbol{\beta}$ is a $Q \times S$ matrix of coefficients, and $\boldsymbol{\Sigma}$ is a $S \times S$ covariance matrix. There is a correlation matrix associated with $\boldsymbol{\Sigma}$,

$$\mathbf{R}_{s,s'} = \frac{\boldsymbol{\Sigma}_{s,s'}}{\sqrt{\boldsymbol{\Sigma}_{s,s}\boldsymbol{\Sigma}_{s',s'}}}$$

Model interpretation

As a data-generating mechanism the model can be thought of like this: There is a vector of continuous responses $\mathbf{w}_i$ generated from mean vector $\boldsymbol{\mu}_i$ and covariance $\boldsymbol{\Sigma}$ (Fig. 1a). The partition $\mathbf{p}_{is}$ segments the continuous scale into intervals, some of which are censored and others not. Each interval is defined by two values, $\mathbf{p}_{isk} = (p_{is,k}, p_{is,k+1}]$. For a value of w_{is} that falls within a censored interval k the observed y_{is} is assigned to discrete interval $z_{is} = k$. For a value of w_{is} that falls in an uncensored interval y_{is} is assigned w_{is}.

Of course, data present us with the inverse problem: the observed y_{is} are continuous or discrete, with known or unknown partition $(p_{is,k}, p_{is,k+1}]$ (Fig. 1b). Depending on how the data are observed, we must impute at least the elements of $n \times S$ matrix $\mathbf{W}$ that lie within censored intervals. Unknown elements of $\mathbf{Z}$ and $\mathcal{P}$ will also be imputed in order to estimate $\boldsymbol{\beta}$ and $\boldsymbol{\Sigma}$.

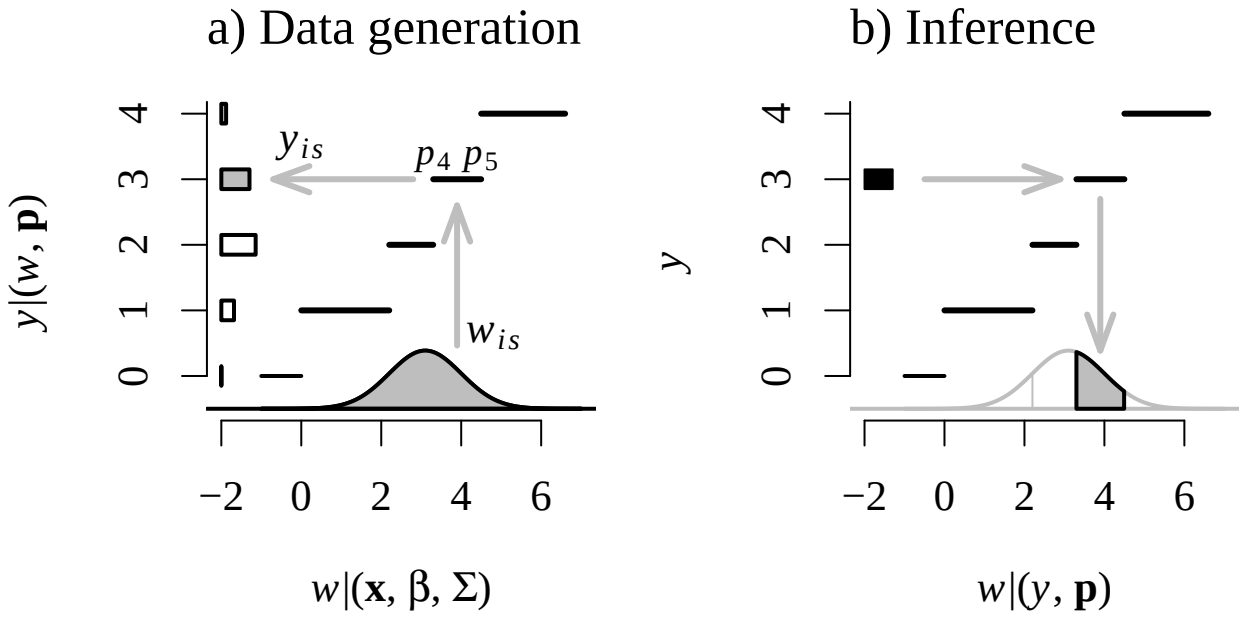

Figure 1. Censoring in gjam. As a data-generating model (a), a realization w_{is} that lies within a censored interval is translated by the partition $\mathbf{p}_{is}$ to discrete y_{is}. The distribution of data (bars at left) is induced by the latent scale and the partition. For inference (b), observed discrete y_{is} takes values on the latent scale from a truncated distibution.

Summary of data types

The different types of data that can be included in the model are summarized here, assigned to the `character` variable `typeNames` that is included in the `modelList` passed to `gjamGibbs`:

Table 1. Partition for each data type

`typeNames`	Type	Obs values	Default partition	Comments
`'CON'`	continuous, uncensored	$(-\infty, \infty)$	none	e.g., centered, standardized
`'CA'`	continuous abundance	$[0, \infty)$	$(-\infty, 0, \infty)$	
`'DA'`	discrete abundance	$[0, 1, 2, \dots]$	$(-\infty, \frac{1}{2E_i}, \frac{3}{2E_i}, \dots \frac{max_s(\mathbf{y}_{is})-1/2}{E_i}, \infty)^1$	e.g., count data
`'PA'`	presence-absence	$[0, 1]$	$(-\infty, 0, \infty)$	unit variance scale
`'OC'`	ordinal counts	$[0, 1, 2, \dots, K]$	$(-\infty, 0, estimates, \infty)$	unit variance scale, imputed partition
`'FC'`	fractional composition	$[0, 1]$	$(-\infty, 0, 1, \infty)$	relative abundance
`'CC'`	count composition	$[0, 1, 2, \dots, E_i]$	$(-\infty, \frac{1}{2E_i}, \frac{3}{2E_i}, \dots \frac{1}{2E_i}, \infty)^1$	relative abundance counts
`'CAT'`	categorical	$[0, 1]$	$(-\infty, max_k(w_{is,k}), \infty)^2$	unit variance, multiple levels

[1] For `'DA'` and `'CC'` data the second element of the partition is not zero, but rather depends on effort. There is thus zero-inflation. The default partition for each data type can be changed with the function `gjamCensorY` (see **Specifying censored intervals**).

[2] For `'CAT'` data species s has $k = 1, \dots, K_s$ total categories. The category with the largest $w_{is,k}$ is the '1', all others are zeros.

For **presence-absence** data with y being discrete zeros and ones, $\mathbf{p}_{is} = (-\infty, 0, \infty)$. This is equivalent to Chib and Greenberg's (2008) model, which could be written $\mathcal{I}_{is} = I(w_{is} > 0)^{y_{is}} I(w_{is} \leqslant 0)^{1-y_{is}}$.

For a continous variable with point mass at zero, **continuous abundance**, this is a multivariate Tobit model, with $\mathcal{I}_{is} = I(w_{is} = y_{is})^{I(y_{is}>0)} I(w_{is} \leqslant 0)^{I(y_{is} \leqslant 0)}$. This is the same partition used for the probit model, the difference being that the positive values in the Tobit are uncensored.

Categorical responses fit within the same framework. Each categorical response occupies as many columns in $\mathbf{Y}$ as there are independent levels in response s, levels being $k = 1, ..., K_s - 1$. For example, if randomly sampled plots are scored by one of five cover types, then there are four columns in $\mathbf{Y}$ for the response s. The four columns can have at most one 1. If all four columns are 0, then the reference level is observed. The observed level has the largest value of $w_{is,k}$ (Table 1). This is similar to Zhang et al.'s (2008) model for categorical data.

For **ordinal counts** gjam is Lawrence et al.'s (2008) model having the partition $\mathbf{p}_{is} = (-\infty, 0, p_{is,2}, p_{is,3}, ..., p_{is,K}, \infty)$, where all but the first two and the last elements must be inferred. The partition must be inferred, because the ordinal scale is only relative.

Like categorical data, **composition** data also have one reference class. For discrete count data the partition for observation i can be defined to account for sample effort (see next section).

Effort and weight of discrete data

The partition for a discrete interval k depends on effort for sample i

$$(p_{i,k}, p_{i,k+1}] = \left(\frac{k - 1/2}{E_i}, \frac{k + 1/2}{E_i} \right]$$

In gjam effort affects the partition, and therefore the censor intervals on the latent variable w_{is} and, finally, the weight of each observation. For *discrete abundance* (`'DA'`) data on plots of a given area, large plots contribute more weight than small plots. Because plots have different areas one might choose to model w_{is} on a 'per-area' scale (density) rather than a 'per-plot' scale. Here is a table of variables for the case where counts represent the same density, but have different effort due to different plot areas:

count $y_{is} = z_{is}$	plot area E_i	density w_{is}	bin k	density $\mathbf{p}_{ik}$
10	0.1 ha	100 ha^{-1}	11	(95, 105)
100	1.0 ha	100 ha^{-1}	101	(99.5, 100.5)

The wide partition on the 0.1-ha plot admits large variance around the observation of 10 trees per 0.1 ha plot. Wide variance on an observation decreases its contribution to the fit. Conversely, the narrow partition on the 1.0-ha plot constrains density to a narrow interval around the observed value.

For *composition count* (`'CC'`) data effort is represented by the total count, and w_{is} lies on the composition scale $(0, 1)$. Using the same partition as previously the table for two observations that represent the fraction 0.10 with different effort (e.g., total reads in PCR data) looks like this:

count $y_{is} = z_{is}$	total count E_i	fraction w_{is}	bin k	fraction $\mathbf{p}_{ik}$
10	100	0.1	11	(0.095, 0.105)
10,000	100,000	0.1	10,001	(0.099995, 0.100005)

Again, on the composition scale (0, 1), weight of the observation is determined by the partition width and, in turn, effort.

Using gjam

It's easiest to start with the examples from `gjam` help pages. This section includes **Simulated examples**, which expands on these examples. The section that follows, **Your data**, discusses some of the issues you might encounter when specifying your own model applied to your data.

Simulated examples

Simulated data are used to check that the algorithm can recover true parameter values and predict data, including underlying latent variables. To illustrate I simulate a sample of size $n = 500$ for $S = 10$ species and $Q = 3$ predictors. To indicate that all species are continuous abundance data I specify `typeNames` as `'CA'`:

```
library(gjam)
sim <- gjamSimData(n = 500, S = 10, Q = 3, typeNames = 'CA')
summary(sim)
```

The object `sim` includes elements needed to analyze the simulated data set. `sim$typeNames` is now a length-S vector. The `formula` follows standard R syntax. It does not start with `y ~`, because gjam is multivariate. The multivariate response is supplied as a $n \times S$ matrix `ydata`. Here is the formula for this example:

```
sim$formula
```

The model can include interactions.

The simulated parameter values are returned from `gjamSimData` in the list `sim$trueValues`, shown below with the corresponding names of estimates from `gjamGibbs`:

Table 2. Variable names and scales in simulation and fitting

model	sim$trueValues[1]	out$modelSummary[2]	out$chains[2]	scale
source	gjamSimData	gjamGibbs	gjamGibbs	modelSummary
$\boldsymbol{\beta}$	beta	betaMu	bgibbs	Y/X
$\boldsymbol{\Sigma}$	sigma	sigMu	sgibbs	$Y_s Y_{s'}$
$\mathbf{R}$	corSpec	corMu	sgibbs	correlation
$\mathcal{P}$ [3]	cuts	cutMu	sgibbs	correlation
K [4]	-	-	kgibbs	dimensionless
σ^2 [4]	-	-	sigErrGibbs	Y^2
$\boldsymbol{\alpha}$ [5]	-	betaTraitMu	agibbs	U/X [6]
$\boldsymbol{\Omega}$ [5]	-	sigmaTraitMu	mgibbs	$U_m U_{m'}$ [6]

[1] `sim` is a fitted object from `gjamSimData`.

[2] `out` is a fitted object from `gjamGibbs`.

[3] Only when `ydata` includes ordinal types.

[4] Only with dimension reduction, `reductList` is included in `modelList` (Dimension reduction vignette).

[5] Only for trait analysis, `traitList` is included in `modelList` (Trait vignette).

[6] U is the response data in the trait vignette.

As is typical in species abundance data the zeros can be overwhelming. The simulator generates many zeros:

```
par(bty = 'n', mfrow = c(1,2), family='')
h <- hist(c(-1,sim$y),nclass = 50,plot = F)
plot(h$counts,h$mids,type = 's')
plot(sim$w,sim$y,cex = .2)
```

Here is a short Gibbs sampler with `ng = 100` iterations to estimate parameters and fit the data. The function `gjamGibbs` needs the `formula` for the model, the `data.frame xdata`, which includes the predictors, the response matrix `ydata`, and a `modelList` specifying number of Gibbs steps (`ng`), the `burnin`, and `typeNames`.

```
# a few iterations
ml  <- list(ng = 100, burnin = 10, typeNames = sim$typeNames)
out <- gjamGibbs(sim$formula, sim$xdata, sim$y, modelList = ml)
summary(out)
```

Among the objects to consider initially are the design matrix `out$x`, response matrix `y`, and the Gibbs sampler `chains` with these names and sizes:

```
summary(out$chains)
```

`chains` is a list of matrices, each with `ng` rows and as many columns as needed to hold parameter estimates. Here are the `chains` and their summaries in `modelSummary`:

Table 3. Variables names in ouput variables and chains

chains	modelSummary	size	comments
sgibbs	sigMu, sigSe	$S \times S$	covariance matrix $\mathbf{\Sigma}$
bgibbs	betaMu, betaSe	$Q \times S$	coefficient matrix $\boldsymbol{\beta}$

Additional summaries are available in the list `modelSummary`:

```
summary(out$modelSummary)
```

The matrix `classBySpec` shows the number of observations in each interval. For this example of continuous data censored at zero, the two bins are $k = 0, 1$ corresponding to the intervals $(p_{s,0}, p_{s,1}] = (-\infty, 0]$ and $\{p_{s,1}, p_{s,2}) = (0, \infty)$. The length-$(K+1)$ partition vector is the same for all species, $\mathbf{p} = (-\infty, 0, \infty)$. Here is `classBySpec` for this example:

```
out$modelSummary$classBySpec
```

The first interval is censored (all values of $y_{is} = 0$). The second interval is not censored ($y_{is} = w_{is}$).

The data are also predicted in `gjamGibbs`, summarized by predictive means and standard errors. These are contained in $n \times Q$ matrices `$modelSummary$xpredMu` and `$modelSummary$xpredSd` and $n \times S$ matrices `$modelSummary$yMu` and `$modelSummary$ySd`. The latent states are included in `$modelSummary$wMu` and `$modelSummary$wSd`.

The output can be viewed with the function `gjamPlot`:

```
sim <- gjamSimData(n = 500, S = 10, typeNames = 'CA')
ml  <- list(ng = 2000, burnin = 500, typeNames = sim$typeNames)
out <- gjamGibbs(sim$formula, sim$xdata, sim$ydata, modelList = ml)
pl  <- list(trueValues = sim$trueValues,width = 3,height = 2, GRIDPLOTS = T, SMALLPLOTS = F)
gjamPlot(output = out, plotPars = pl)
```

`gjamPlot` creates a number of plots comparing true and estimated parameters (for simulated data). Here are some simple biplots:

```
par(bty = 'n', mfrow = c(1,3), family='')
plot(sim$trueValues$beta, out$modelSummary$betaMu)
plot(sim$trueValues$corSpec, out$modelSummary$corMu)
plot(sim$y,out$modelSummary$yMu, cex = .2)
```

To process the output beyond what is provided in `gjamPlot` I can work directly with the `chains`.

My data

`gjam` uses the standard `R` syntax in the `formula` that I would with functions like `lm()` and `glm()`. Because `gjam` uses inverse prediction to summarize large multivariate output, it is important to abide by this syntax.

For example, to analyze a model with quadratic and interaction terms, I might simply construct my own design matrix with these columns included, i.e., side-stepping the standard syntax for these effects that can be specified in `formula`. This would be fine for model fitting. However, without specifying this in the `formula` there is no way for `gjam` to know that these columns are in fact non-linear transformations of other columns. Without this knowledge there is no way to properly predict them. The prediction that `gjam` would return would include silly variable combinations.

To illustrate proper model specification I use a few lines from the `data.frame` of predictors in the `forestTraits` data set:

```
library(gjam)
data(forestTraits)
xdata <- forestTraits$xdata[,c(1,2,8)]
```

```
xdata[1:5,]
```

```
##    temp deficit      soil
## 1  1.22    0.04 reference
## 2  0.18    0.21 reference
## 3 -0.94    0.20  SpodHist
## 4  0.64    0.82 reference
## 5  0.82   -0.18 reference
```

Here is a simple model specification with `as.formula()` that includes only main effects:

```
formula <- as.formula( ~ temp + deficit + soil )
```

The design matrix `x` that is generated in `gjam` has an intercept, two covariates, and four columns for the multilevel factor `soil`:

```
##   (Intercept)  temp deficit soilMol soilreference soilSpodHist soilUltKan
## 1           1  1.22    0.04       0             1            0          0
## 2           1  0.18    0.21       0             1            0          0
## 3           1 -0.94    0.20       0             0            1          0
## 4           1  0.64    0.82       0             1            0          0
## 5           1  0.82   -0.18       0             1            0          0
```

To include interactions between temp and soil I use the symbol '`*`':

```
formula <- as.formula( ~ temp*soil )
```

Here is the design matrix that results from this `formula` with interaction terms indicated by the symbol `':'`

```
##   (Intercept)  temp soilMol soilreference soilSpodHist soilUltKan
## 1           1  1.22       0             1            0          0
## 2           1  0.18       0             1            0          0
## 3           1 -0.94       0             0            1          0
## 4           1  0.64       0             1            0          0
## 5           1  0.82       0             1            0          0
##   temp:soilMol temp:soilreference temp:soilSpodHist temp:soilUltKan
## 1            0               1.22              0.00               0
## 2            0               0.18              0.00               0
## 3            0               0.00             -0.94               0
## 4            0               0.64              0.00               0
## 5            0               0.82              0.00               0
```

For a quadratic term I use the R function I():

```
formula <- as.formula( ~ temp + I(temp^2) + deficit )
```

Here is the design matrix with linear and quadratic terms:

```
##   (Intercept)  temp I(temp^2) deficit
## 1           1  1.22    1.4884    0.04
## 2           1  0.18    0.0324    0.21
## 3           1 -0.94    0.8836    0.20
## 4           1  0.64    0.4096    0.82
## 5           1  0.82    0.6724   -0.18
```

Here is a quadratic response surface for temp and deficit:

```
formula <- as.formula( ~ temp*deficit + I(temp^2) + I(deficit^2) )
```

Here is the design matrix with all combinations:

```
##   (Intercept)  temp deficit I(temp^2) I(deficit^2) temp:deficit
## 1           1  1.22    0.04    1.4884       0.0016       0.0488
## 2           1  0.18    0.21    0.0324       0.0441       0.0378
## 3           1 -0.94    0.20    0.8836       0.0400      -0.1880
## 4           1  0.64    0.82    0.4096       0.6724       0.5248
## 5           1  0.82   -0.18    0.6724       0.0324      -0.1476
```

These are examples of the formula options available in gjam. Using them will allow for proper inverse prediction of x. To optimize MCMC gjam does not predict x for higher order polynomials. For such models set predictX = F in the modelList.

I can use this model to analyze a tree data set. For my data set I use the tree data contained in forestTraits. It is stored in de-zeroed format, so I extract it with the function gjamReZero. Here are dimensions and the upper left corner of the response matrix $\mathbf{Y}$,

```
ydata  <- gjamReZero(forestTraits$treesDeZero)  # extract y
dim(ydata)
```

```
## [1] 1617   98
```

```
ydata[1:5,1:6]
```

```
##      abieBals acerBarb acerNegu acerPens acerRubr acerSac2
## [1,]        0        0        1        0        5        0
## [2,]        0        0       10        0        5        6
## [3,]        3        0        0        0       15        0
## [4,]        0        0        4        0       20        1
## [5,]        0        0        2        0       10        0
```

In code that follows I treat them as discrete counts, typeNames = 'DA'. Because of the large number of columns (98) I speed things up calling for dimension reduction, passed as $N \times r = 20 \times 4$:

```
rl       <- list(r = 4, N = 20)
ml       <- list(ng = 1000, burnin = 500, typeNames = 'DA', reductList = rl)
form     <- as.formula( ~ temp*deficit + I(temp^2) + I(deficit^2) )
out      <- gjamGibbs(form, xdata = xdata, ydata = ydata, modelList = ml)
plotPars <- list(SMALLPLOTS = F, GRIDPLOTS=T, corLines=F, specLabs = F)
gjamPlot(output = out, plotPars = plotPars)
```

Additional information on variable types and their treatment in `gjam` is included later in this document and in the other `gjam vignettes`.

Plotting output

In the foregoing example arguments passed to `gjamPlot` in the `list plotPars` included `SMALLPLOTS = F` (do not compress margins and axes), `GRIDPLOTS = T` (draw grid diagrams as heat maps for parameter values and predictions), `corLines = F` (do not separate parameter values with lines on gridplots), and `specLabs = F` (do not put species labels on plots, because there are too many see clearly). In this section I summarize plots generated by `gjamPlot`.

By default, plots are sent to the screen. I hit return to see the next plot. Faster execution obtains if I write plots directly to pdf files, with `SAVEPLOTS = T`. I can specify a folder this way:

```
plotPars <- list(SMALLPLOTS = F, GRIDPLOTS=T, SAVEPLOTS = T, outfolder = 'stuff')
```

In all plots, posterior distributions and predictions are shown as 68% (boxes) and 95% (whiskers) intervals, respectively. Here are the plots in alphabetical order by file name:

Name	Comments
`betaAll`	CIs for $\boldsymbol{\beta}$
`beta_(variable)`	CIs one file per variable
`betaChains`	Example MCMC chains for $\boldsymbol{\beta}$
`clusterDataE`	Cluster analysis of raw data and $\mathbf{E}$ matrix
`clusterGridB`	Cluster and grid of $\boldsymbol{\beta}$
`clusterGridE`	Cluster and grid of $\mathbf{E}$
`clusterGridR`	Cluster and grid of $\mathbf{R}$
`corChains`	Example MCMC chains for $\mathbf{R}$
`dimRed`	Dimension reduction (see `vignette`) for Σ matrix
`gridB_0`	Grid of $\boldsymbol{\beta}$ and inclusion probability $\boldsymbol{\omega}$

Name	Comments
`gridY_E`	Grid of cor($\mathbf{Y}$) and $\mathbf{E}$, ordered by cor($\mathbf{Y}$)
`gridF_B`	Grid of sensitivity $\mathbf{F}$ and $\boldsymbol{\beta}$, ordered by $\mathbf{F}$
`gridF_X`	Grid of sensitivity $\mathbf{F}$ and cov($\mathbf{X}$), ordered by $\mathbf{F}$
`gridR_E`	Grid of $\mathbf{R}$ and $\mathbf{E}$ ordered by $\mathbf{R}$
`gridR`	Grid of $\mathbf{R}$, ordered by cluster analysis.
`gridTraitB`	If traits are predicted, see `gjam` `vignette` on traits.
`ordination`	Ordination of $\mathbf{E}$ matrix, including eigenvalues (cumulative)
`partition`	If ordinal data in $\mathbf{Y}$, posterior distribution of $\mathcal{P}$
`richness`	Predictive distribution of richness with data (brown histogram)
`sensitivity`	Overall sensitivity $\mathbf{f}$ by predictor variable.
`traits`	If traits are predicted, see `gjam` `vignette` on traits.

Name	Comments
`traitPred`	If traits are predicted, see `gjam vignette` on traits.
`trueVsPars`	If simulated data and `trueValues` in `plotPars`, CIs for parameter values
`xPred`	Inverse predictive distribution of of $\mathbf{X}$
`xPredFactors`	Inverse predictive distribution of factor levels
`yPred`	Predictive distribution of $\mathbf{Y}$, in- (blue bars) and out-of-sample (dots)
`yPredAll`	If `plotAllY = T` predictions of up to 16 species are generated.

Flexibility in gjam

Heterogeneous sample effort

Here is an example with discrete abundance data, now with heterogeneous sample effort. Heterogeneous effort applies wherever plot area or search time varies, such as vegetation plots of varying area, animal survey data with variable search time, or catch returns from fishing vessels with different gear and trawling times. Here I simulate a list containing the columns and the effort that applies to those columns, shown for 50 observations:

```
S   <- 5
n   <- 50
ef  <- list( columns = 1:S, values = round(runif(n,.5,5),1) )
sim <- gjamSimData(n, S, typeNames = 'DA', effort = ef)
ef
```

If `ef$values` consists of a length-n `vector`, then gjam assumes each value applies to all species in the observation for the corresponding element of `vector ef$columns`. This is the case shown above and would apply when effort is plot area, search time, sample volumn, and so forth. Alternatively, `values` can be

supplied as a matrix, which could differ by observation and species. For example, camera trap data detect large animals at greater distances than small animals. For simulation purposes `gjamSimData` only accepts a vector.

Because observations are discrete the continuous latent variables w_{is} are censored. Unlike the previous continuous example, observations y_{is} now assume only discrete values:

```
plot(sim$w,sim$y, cex = .2)
```

The large scatter reflects the variable effort represented by each observation. Incorporating the effort scale gives this plot:

```
plot(sim$w*ef$values, sim$y, cex = .2)
```

The heterogeneous effort affects the weight of each observation in model fitting. The `effort` is entered in `modelList`. Increase the number of iterations and look at plots:

```
S   <- 10
n   <- 1500
ef  <- list( columns = 1:S, values = round(runif(n,.5,5),1) )
sim <- gjamSimData(n, S, typeNames = 'DA',effort = ef)
ml  <- list(ng = 1000, burnin = 250, typeNames = sim$typeNames, effort = ef)
out <- gjamGibbs(sim$formula, sim$xdata, sim$ydata, modelList = ml)
pl  <- list(trueValues = sim$trueValues,SMALLPLOTS=F)
gjamPlot(output = out, plotPars = pl)
```

Sample effort in composition data

Composition count (`'CC'`) data have heterogenous effort due to different numbers of counts for each sample. For example, in microbiome data, the number of reads per sample can range from 10^2 to 10^6. The number of reads does not depend on total abundance. It is generally agreed that only relative differences are important. gjam knows that the effort in `CC` data is the total count for the sample, so `effort` does not need to be specified. Here is an example with simulated data:

```
sim   <- gjamSimData(S = 8, typeNames = 'CC')
types <- sim$typeNames
xdata <- sim$xdata
y     <- sim$y
ml    <- list(ng = 2000, burnin = 500, typeNames = types)
out   <- gjamGibbs(sim$formula, xdata, y, modelList = ml)
pl    <- list(trueValues = sim$trueValues, width = 3, height = 3,
                  GRIDPLOTS = T, SMALLPLOTS = F)
gjamPlot(output = out, plotPars = pl)
```

For comparison, here is an example with fractional composition, where there is no effort:

```
sim   <- gjamSimData(S = 20, typeNames = 'FC')
types <- sim$typeNames
xdata <- sim$xdata
y     <- sim$y
ml    <- list(ng = 1000, burnin = 250, typeNames = types)
out   <- gjamGibbs(sim$formula, xdata, y, modelList = ml)
```

```
pl    <- list(trueValues = sim$trueValues, width = 3, height = 3,
                  GRIDPLOTS = T, SMALLPLOTS = F)
gjamPlot(output = out, plotPars = pl)
```

The default censoring for different data types can be changed. A `gjam vignette` on trait modeling provides an example.

The partition in ordinal data

Ordinal count (`'OC'`) data are collected where abundance must be evaluated rapidly or precise measurements are difficult. Because there is no absolute scale the partition must be inferred. Here is an example with 10 species:

```
sim <- gjamSimData(typeNames = 'OC')
ml  <- list(ng = 2000, burnin = 500, typeNames = sim$typeNames)
out <- gjamGibbs(sim$formula, sim$xdata, sim$ydata, modelList = ml)
```

A simple plot of the posterior mean values of `cutMu` shows the estimates with true values from simulation:

```
keep <- strsplit(colnames(out$modelSummary$cutMu),'C-') #only saved columns
keep <- matrix(as.numeric(unlist(keep)), ncol = 2, byrow = T)[,2]
plot(sim$trueValues$cuts[,keep],out$modelSummary$cutMu)
```

Here are plots:

```
pl  <- list(trueValues = sim$trueValues, SMALLPLOTS = F)
gjamPlot(output = out, plotPars = pl)
```

Categorical data

Categorical data have levels within groups. The levels are unordered. For a given observation the observed level is assigned as a `factor` in `data.frame ydata`. In observation vector $\mathbf{y}_i$ there is an element for each level, one of which is a 1 and remainder are 0. Suppose that observations are obtained on attributes of individual plants, each plant being an observation. The group `leaf` type might have four levels broadleaf decidious `bd`, needleleaf decidious `nd`, broadleaf evergreen `be`, and needleaf evergreen `ne`. A second group `xylem` anatomy might have three levels diffuse porous `dp`, ring porous `rp`, and tracheid `tr`. In both cases I assign the last class to be a reference class, `other`. Ten rows of the response matrix data might look like this:

```
##     leaf xylem
## 1  other    rp
## 2  other    dp
## 3     nd    dp
## 4     bd    dp
## 5  other    dp
## 6  other    dp
## 7  other    dp
## 8     bd    rp
## 9     nd    rp
## 10 other    dp
```

gjam expands these two groups into four and three columns in y, respectively. As for composition data there is one redundant column for each group. Here is an example with simulated data, having two categorical groups and one fractional compostion group:

```
types <- c('CAT', 'CAT','CAT')
f     <- gjamSimData(n=2000, S = length(types), typeNames = types)
ml    <- list(ng = 1500, burnin = 500, typeNames = f$typeNames, PREDICTX = F)
out   <- gjamGibbs( f$formula, xdata = f$xdata, ydata = f$ydata, modelList = ml)
pl  <- list(trueValues = f$trueValues, SMALLPLOTS=F, plotAllY = T)
gjamPlot(out, plotPars = pl)
```

Combinations of data types

One of the advantages of gjam is that it combines data of many types. Here is an example showing joint analysis of 12 species represented by five data types, specified by column:

```
types <- c('OC','OC','OC','OC','CC','CC','CC','CC','CC','CA','CA','PA','PA')
sim   <- gjamSimData(S = length(types), Q = 3, typeNames = types)
ml    <- list(ng = 2000, burnin = 500, typeNames = sim$typeNames)
out   <- gjamGibbs(sim$formula, sim$xdata, sim$ydata, modelList = ml)
tmp   <- data.frame(sim$typeNames, out$modelSummary$classBySpec[,1:10])
print(tmp)
```

I have displayed the first 10 columns of `classBySpec` from the `modelSummary` of `out`, with their `typeNames`. The ordinal count (`'OC'`) data occupy lower intervals. The width of each interval in `OC` data depends on the estimate of the partition in `cutMu`.

The composition count (`'CC'`) data occupy a broader range of intervals. Because `CC` data are only relative, there is information on only $S - 1$ species. One species is selected as `other`. The `other` class can be a collection of rare species (Clark et al. 2016).

Both continuous abundance (`'CA'`) and presence-absence (`'PA'`) data have two intervals. For CA data only the first interval is censored, the zeros (see above). For `PA` data both interval are censored; it is a multivariate probit.

Here are some plots for analysis of this model:

```
pl  <- list(trueValues = sim$trueValues, SMALLPLOTS = F)
gjamPlot(output = out, plotPars = pl)
```

Missing data, out-of-sample prediction

gjam identifies missing values in `xdata` and `y` and models them as part of the posterior distribution. These are identified by the vector `missingIndex` as part of the output from gjamGibbs. The estimates for missing $\mathbf{X}$ are `missingX` and `missingXSd`. The estimates for missing $\mathbf{Y}$ are `yMissMu` and `yMissSd`.

To simulate missing data use `nmiss` to indicate number of missing value. The actual value will be less than `nmiss`:

```
sim <- gjamSimData(S = 5, typeNames = 'OC', nmiss = 20)
which(is.na(sim$xdata), arr.ind = T)
```

Note that missing values are assumed to occur in random rows and columns, but not in column one, which is the intercept. No further action is needed for model fitting, as `gjamGibbs` knows to treat these as missing data.

Out-of-sample prediction of $\mathbf{Y}$ is not part of the posterior distribution. Holdouts can be specified randomly with `holdoutN` (the number of plots to be held out at random) or with `holdoutIndex` (plot numbers). The latter might be useful when a comparison of predictions is desired for different models using the same plots as holdouts. Of course, out-of-sample prediction assumes that $\mathbf{X}$ is known for predicted values, and predicted $\mathbf{Y}$ are unknown.

```
f   <- gjamSimData(typeNames = 'CA', nmiss = 20)
ml  <- list(ng = 2000, burnin = 500, typeNames = f$typeNames, holdoutN = 50)
out <- gjamGibbs(f$formula, f$xdata, f$ydata, modelList = ml)

par(mfrow=c(1,3))
plot(out$x[out$missingIndex], out$modelSummary$xpredMu[out$missingIndex])
title('missing in x'); abline(0,1)
plot(out$x[out$holdoutIndex,-1], out$modelSummary$xpredMu[out$holdoutIndex,-1])
title('holdouts in x'); abline(0,1)
plot(out$y[out$holdoutIndex,], out$modelSummary$yMu[out$holdoutIndex,])
title('holdouts in y'); abline(0,1)
```

Conditional prediction

`gjam` can predict a subset of columns `y` conditional on other columns using the function `gjamPredict`. An example is provided in the `gjam vignette` on dimension reduction.

Grid and cluster plots

If the `plotPars` list passed to gjamPlot specifies `GRIDPLOTS = T`, then grid and clusture plots are generated as gridded values for $\boldsymbol{\beta}$, $\boldsymbol{\Sigma}$ and $\mathbf{R}$. An example for $\boldsymbol{\Sigma}$ and $\mathbf{R}$ are shown here. In the case of $\boldsymbol{\beta}$ the predictors are organized from high to low sensitivity, from $diag(\boldsymbol{\beta}\boldsymbol{\Sigma}\boldsymbol{\beta}')$. Here is an example:

```
pl  <- list(trueValues = f$trueValues, GRIDPLOTS = T,
            SMALLPLOTS = F)
fit <- gjamPlot(output = out, plotPars = pl)
```

Gridplots of matrix $\mathbf{R}$ show conditional and marginal dependence in white and grey. In plots of $\mathbf{E}$ marginal independence is shown in grey, but conditional independence is not shown, as the matrix does not have an inverse (Clark et al. 2016).

The sensitivity matrix $\mathbf{F}$ is shown together in a plot with individual species responses $\boldsymbol{\beta}$.

The plot in which the model residual correlation $\mathbf{R}$ and the response correlation $\boldsymbol{\beta}$ are compared are ordered by their similiarity in the $\mathbf{R}$. If the two contain similar structure, then it will be evident in this comparison. There is no reason to expect them to be similar.

For large S the labels are not shown on the graphs, they would be too small. The order of species and the cluster groups to which they belong are returned in `fit$clusterOrder` and `fit$clusterIndex`.

When a model won't execute

A joint model for data sets with many response variables can be unstable for several reasons. gjam is vulnerable due to the fact that columns in `y` have different scales and, thus, can range over orders of magnitude. If execution fails there are several options.

If you are simulating data, first try it again. The simulator aims to generate data that will actually work, more challenging than would be the case for a univariate simulation of a single data type.

Check to insure that all columns in `y` include at least some non-zero values. One would not expect a univariate model to fit a data set where `y` is all zeros. However, when there are many columns in `y`, the fact that some are never or rarely observed can be overlooked. The functions `hist`, `colSums`, and, for discrete data, `table`, can be used. The function `gjamTrimY` can be used to limit `y` to only those columns with sufficient non-zero observations.

If a simulation fails due to a cholesky error ($\mathbf{\Sigma}$ is not positive definite), consider either reducing the number of columns in `y` or implementing dimension reduction (see the gjam vignette on this subject).

Algorithm summary

Model fitting is done by Gibbs sampling. Parameters $\boldsymbol{\beta}$ and $\mathbf{\Sigma}$ are sampled directly,

1. $\mathbf{\Sigma}|\mathbf{W}, \boldsymbol{\beta}$
2. $\boldsymbol{\beta}|\mathbf{\Sigma}, \mathbf{W}$
3. For unknown partition (ordinal variables) the partition is sampled, $\mathcal{P}|\mathbf{Z}, \mathbf{W}$
4. For ordinal, presence-absence, and categorical data, latent variables are drawn on the correlation scale, $\mathbf{W}|\mathbf{R}, \boldsymbol{\alpha}, \mathbf{P}$, where $\mathbf{R} = \mathbf{D}^{-1/2}\mathbf{\Sigma}\mathbf{D}^{-1/2}$, $\boldsymbol{\alpha} = \mathbf{D}^{-1/2}\boldsymbol{\beta}$, $\mathbf{P} = \mathbf{D}^{-1/2}\mathcal{P}$, and $\mathbf{D} = diag(\mathbf{\Sigma})$.
   For other variables that are discrete or censored, latent variables are drawn on the covariance scale, $\mathbf{W}|\mathbf{\Sigma}, \boldsymbol{\beta}, \mathcal{P}$.

Where discrete intervals can be observed with error, the z_{is} are sampled (e.g., zero-inflation). That option is not currently available, but see Clark et al. (2016).

Inverse prediction of input variables provides sensitivity analysis (Clark et al. 2011, 2014). Columns in $\mathbf{X}$ that are linear (not involved in interactions, polynomial terms, or factors) are sampled directly from the inverted model. Others are sampled by Metropolis. Sampling is described in the Supplement file to Clark et al. (2016).

For additional information see this link

The model is described in Clark et al (2016).

Acknowledgements

For valuable feedback on the model and computation I thank Bene Bachelot, Alan Gelfand, Diana Nemergut, Erin Schliep, Bijan Seyednasrollah, Daniel Taylor-Rodriquez, Brad Tomasek, Phillip Turner, and Stacy Zhang. I thank the members of NSF's *SAMSI* program on *Ecological Statistics* and my class *Bayesian Analysis of Environmental Data* at Duke University.

References

Brynjarsdottir, J. and A.E. Gelfand. 2014. Collective sensitivity analysis for ecological regression models with multivariate response. *Journal of Biological, Environmental, and Agricultural Statistics*, 19, 481-502.

Chib, S. and E. Greenberg. 1998. Analysis of multivariate probit models. *Biometrika* 85, 347-361.

Clark, J.S., D.M. Bell, M.H. Hersh, and L. Nichols. 2011. Climate change vulnerability of forest biodiversity: climate and resource tracking of demographic rates. *Global Change Biology*, 17, 1834-1849.

Clark, J.S., D. M Bell, M. Kwit, A. Powell, And K. Zhu. 2013. Dynamic inverse prediction and sensitivity analysis with high-dimensional responses: application to climate-change vulnerability of biodiversity. *Journal of Biological, Environmental, and Agricultural Statistics*, 18, 376-404.

Clark, J.S., A.E. Gelfand, C.W. Woodall, and K. Zhu. 2014. More than the sum of the parts: forest climate response from joint species distribution models. *Ecological Applications* 24, 990-999

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, in review.

Lawrence, E., D. Bingham, C. Liu and V. N. Nair (2008) Bayesian inference for multivariate ordinal data using parameter expansion. *Technometrics* 50, 182-191.

Zhang, X., W.J. Boscardin, and T.R. Belin. 2008. Bayesian analysis of multivariate nominal measures using multivariate multinomial probit models. *Computational Statistics and Data Analysis* 52, 3697-3708.