Fitting occupancy models with `spOccupancy`

Jeffrey W. Doser

October 09, 2021

Contents

1 Introduction **1**
1.1 Example data set: Foliage-gleaning birds at Hubbard Brook . . . 1

2 Single species occupancy models **2**
2.1 Basic model description . . . 2
2.2 Fitting single species occupancy models with `PGOcc` . . . 3
2.3 Convergence diagnostics . . . 6
2.4 Posterior predictive checks . . . 9
2.5 Model selection using WAIC . . . 11
2.6 k-fold cross-validation . . . 12
2.7 Prediction . . . 15

3 Single species spatial occupancy models **17**
3.1 Basic model description . . . 17
3.2 Fitting single species spatial occupancy models with `spPGOcc` . . . 17
3.3 Convergence diagnostics . . . 22
3.4 Posterior predictive checks . . . 24
3.5 Model selection using WAIC . . . 24
3.6 Prediction . . . 24

4 Multispecies occupancy models **25**
4.1 Basic model description . . . 25
4.2 Fitting multispecies occupancy models with `msPGOcc` . . . 26
4.3 Convergence diagnostics . . . 31
4.4 Posterior predictive checks . . . 33
4.5 Model selection using WAIC . . . 34
4.6 Prediction . . . 34

5 Multispecies spatial occupancy models **35**
5.1 Basic model description . . . 35
5.2 Fitting multispecies spatial occupancy models with `spMsPGOcc` . . . 36
5.3 Convergence diagnostics . . . 44
5.4 Posterior predictive checks . . . 45
5.5 Model selection using WAIC . . . 46
5.6 Prediction . . . 46

6 Single species integrated occupancy models **47**
6.1 Basic model description . . . 47
6.2 Simulating data using `simIntOcc` . . . 48
6.3 Fitting single species integrated occupancy models with `intPGOcc` . . . 49
6.4 Convergence diagnostics . . . 54

6.5 Posterior predictive checks . . . 54
6.6 Model selection using WAIC . . . 55
6.7 Prediction . . . 55

7 Single species spatial integrated occupancy models 56
7.1 Basic model description . . . 56
7.2 Simulating data using `simIntOcc` . . . 56
7.3 Fitting single speces spatial integrated occupancy models using `spIntPGOcc` . . . 57
7.4 Convergence diagnostics . . . 61
7.5 Posterior predictive checks . . . 62
7.6 Model selection using WAIC . . . 63
7.7 Prediction . . . 63

References 65

1 Introduction

This vignette provides worked examples and explanations on fitting single species and multispecies occupancy models available in the `spOccupancy` R package. We will provide step by step examples on how to fit the following models:

1. Occupancy model using `PGOcc`.
2. Spatial occupancy model using `spPGOcc`.
3. Multispecies occupancy model using `msPGOcc`.
4. Spatial multispecies occupancy model using `spMsPGOcc`.
5. Integrated occupancy model using `intPGOcc`.
6. Spatial integrated occupancy model using `spIntPGOcc`.

In this vignette, we will provide a brief description of each model, with full statistical details provided in a separate MCMC sampler vignette. We will also show how `spOccupancy` provides functions for posterior predictive checks as a Goodness of Fit assessment, model comparison and assessment using the Widely Applicable Information Criterion (WAIC) and k-fold cross-validation, and out of sample predictions using standard R helper functions (i.e., `predict`).

To get started, we load the `spOccupancy` package, as well as the `coda` package, which we will use for some MCMC diagnostics. We will also use the `stars` and `ggplot2` packages to create some very basic plots of our results. We then set a seed so we get the same results.

```
library(spOccupancy)
library(coda)
library(stars)
library(ggplot2)
set.seed(101)
```

1.1 Example data set: Foliage-gleaning birds at Hubbard Brook

As an example data set throughout this vignette, we will use data from twelve foliage-gleaning birds collected from point count surveys at Hubbard Brook Experimental Forest (HBEF) in New Hampshire, USA. Specific details on the data set are available on the Hubbard Brook website and Doser et al. (2021). The data are provided in the `spOccupancy` package and are loaded with `data(hbef2015)`. Some brief information on the data collection protocol and the species included in the data set are found via `help(hbef2015)`.

```
data(hbef2015)
str(hbef2015)
```

```
List of 4
 $ y       : num [1:12, 1:373, 1:3] 0 0 0 1 0 1 1 0 0 0 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:12] "AMRE" "BAWW" "BHVI" "BLBW" ...
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "1" "2" "3"
 $ occ.covs: num [1:373, 1:2] -0.889 -0.765 -0.413 -0.14 -0.13 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "Elevation" "Elevation.2"
 $ det.covs:List of 3
  ..$ day  : num [1:373, 1:3] -1.62 -1.62 -1.62 -1.62 -1.62 ...
  ..$ tod  : num [1:373, 1:3] -1.565 -1.378 -1.084 -0.79 -0.549 ...
  ..$ day.2: num [1:373, 1:3] 2.61 2.61 2.61 2.61 2.61 ...
 $ coords  : num [1:373, 1:2] 280000 280000 280000 280001 280000 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:2] "Easting" "Northing"
```

The object `hbef2015` is a list comprised of the detection-nondetection data (`y`), covariates on the occurrence portion of the model (`occ.covs`), covariates on the detection portion of the model (`det.covs`), and the spatial coordinates of each site (`coords`) for use in spatial occupancy models and plotting. This list is in the exact format required for input to `spOccupancy` model functions. `hbef2015` contains data on 12 species in the three-dimensional array `y`, where the dimensions of `y` correspond to species (12), sites (373), and replicates (3). Here we will use data on the charming Ovenbird (OVEN; *Seiurus aurocapilla*) to display single species models, so we next subset the `hbef2015` list to only include data from OVEN in a new object `ovenHBEF`.

```
sp.names <- attr(hbef2015$y, "dimnames")[[1]]
ovenHBEF <- hbef2015
ovenHBEF$y <- ovenHBEF$y[sp.names == "OVEN", , ]
table(ovenHBEF$y)
```

```
  0   1
569 539
```

We see OVEN is detected at around half of all site-replicate combinations.

2 Single species occupancy models

2.1 Basic model description

Let z_j be the true presence (1) or absence (0) of a species at site j, with $j = 1, \ldots, J$. We assume this latent occurrence process arises from a Bernoulli process following

$$\begin{aligned} z_j &\sim \text{Bernoulli}(\psi_j), \\ \text{logit}(\psi_j) &= \boldsymbol{x}_j' \cdot \boldsymbol{\beta}, \end{aligned} \tag{1}$$

where ψ_j is the probability of occurrence at site j, which is a function of site-specific covariates $\boldsymbol{X}$ and a vector of regression coefficients ($\boldsymbol{\beta}$).

We do not directly observe z_j and rather we observe an imperfect representation of the latent occurrence process as a result of imperfect detection (i.e., the failure to detect a species at a site when it is truly present). Let $y_{j,k}$ be the observed detection (1) or nondetection (0) of a species of interest at site j during replicate k

for each of $k = 1, \ldots, K_j$ replicates. We envision the detection-nondetection data as arising from a Bernoulli process conditional on the true latent occurrence process:

$$\begin{aligned} y_{j,k} &\sim \text{Bernoulli}(p_{j,k} \cdot z_j), \\ \text{logit}(p_{j,k}) &= \boldsymbol{v}'_{j,k} \cdot \boldsymbol{\alpha}, \end{aligned} \tag{2}$$

where $p_{j,k}$ is the probability of detecting a species at site j during replicate k (given it is present at site j), which is a function of site and replicate specific covariates $\boldsymbol{V}$ and a vector of regression coefficients ($\boldsymbol{\alpha}$).

To complete the Bayesian specification of the model, we assign multivariate normal priors for the occurrence ($\boldsymbol{\beta}$) and detection ($\boldsymbol{\alpha}$) regression coefficients. To yield an efficient implementation of the occupancy model using a logit link function, we use Pólya-Gamma data augmentation (Polson, Scott, and Windle 2013), which is described in depth in a separate MCMC sampler vignette ($\mathbf{P}$ólya-$\mathbf{G}$ammais where the `PG` comes from in all `spOccupancy` model fitting functions).

2.2 Fitting single species occupancy models with `PGOcc`

The `PGOcc` function fits single species occupancy models using Pólya-Gamma latent variables, which makes it more efficient than standard Bayesian implementations of occupancy models using a logit link function (Clark and Altwegg 2019; Polson, Scott, and Windle 2013). `PGOcc` has the following arguments:

```
PGOcc(occ.formula, det.formula, data, starting, priors, n.samples,
      n.omp.threads = 1, verbose = TRUE, n.report = 100,
      n.burn = round(.10 * n.samples), n.thin = 1,
      k.fold, k.fold.threads = 1, k.fold.seed, ...)
```

The first two arguments, `occ.formula` and `det.formula`, use standard R model syntax to denote the covariates included in the occurrence and detection portions of the model, respectively. Only the right hand side of the formulas are included. Random intercepts can be included in both the occurrence and detection portions of the single-species occupancy model using `lme4` syntax (Bates et al. 2015). The names of variables given in the formulas should correspond to those found in `data`, which is a list consisting of the following tags: `y` (detection-nondetection data), `occ.covs` (occurrence covariates), `det.covs` (detection covariates). `y` should be stored as a sites x replicate matrix, `occ.covs` as a matrix or data frame with site-specific covariate values, and `det.covs` as a list with each list element corresponding to a covariate to include in the detection portion of the model. Covariates on detection can vary by site and/or survey, and so these covariates may be specified as a site by survey matrix for survey-level covariates or as a one-dimensional vector for survey level covariates. The `ovenHBEF` list is already in the required format. Here we will model OVEN occurrence as a function of linear and quadratic elevation and will include three observational covariates (linear and quadratic day of survey, time of day of survey) on the detection portion of the model. We specify the formulas below

```
oven.occ.formula <- ~ Elevation + Elevation.2
oven.det.formula <- ~ day + tod + day.2
# Check out the format of ovenHBEF
str(ovenHBEF)
```

```
List of 4
 $ y       : num [1:373, 1:3] 1 1 0 1 0 0 1 1 1 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "1" "2" "3"
 $ occ.covs: num [1:373, 1:2] -0.889 -0.765 -0.413 -0.14 -0.13 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "Elevation" "Elevation.2"
```

```
$ det.covs:List of 3
 ..$ day  : num [1:373, 1:3] -1.62 -1.62 -1.62 -1.62 -1.62 ...
 ..$ tod  : num [1:373, 1:3] -1.565 -1.378 -1.084 -0.79 -0.549 ...
 ..$ day.2: num [1:373, 1:3] 2.61 2.61 2.61 2.61 2.61 ...
$ coords  : num [1:373, 1:2] 280000 280000 280000 280001 280000 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:373] "1" "2" "3" "4" ...
 .. ..$ : chr [1:2] "Easting" "Northing"
```

Note that the covariates in `hbef2015` are already standardized to have mean 0 and standard deviation 1. If the data were not already standardized, we could do the standardization directly in the formula specification (e.g., `~ scale(Elevation) + scale(Elevation)^2`).

Next, we specify the starting values for the MCMC sampler in `starting`. `PGOcc` (and all other `spOccupancy` model fitting functions) will set starting values by default, but here we will do this explicitly. Starting values are specified in a list with the following tags: `z` (latent occurrence values), `alpha` (detection regression coefficients), and `beta` (occurrence regression coefficients). Below we set all initial values of the regression coefficients to 0, and set starting values for `z` based on the detection-nondetection data matrix. For the occurrence (`beta`) and detection (`alpha`) regression coefficients, the starting values are passed in either a vector of length corresponding to the number of estimated parameters, or as a single value if setting the starting values to be equal for all parameters. Below we take the latter appraoch.

```
oven.starting <- list(alpha = 0,
                      beta = 0,
                      z = apply(ovenHBEF$y, 1, max, na.rm = TRUE))
```

We next specify the priors for the occurrence and detection regression coefficients. The Pólya-Gamma data augmentation algorithm employed by `spOccupancy` assumes normal priors for both the detection and occurrence regression coefficients. These priors are specified in a list with tags `beta.normal` for occurrence and `alpha.normal` for detection parameters. Each list element is then itself a list, with the first element of the list consisting of the hypermeans for each coefficient to be estimated and the second element of the list consisting of the hypervariances for each coefficient. Alternatively, the hypermean and hypervariances can be specified as a single value if the same prior is used for all regression coefficients. By default, `spOccupancy` will set the hypermeans to 0 and the hypervariances to 2.72, which corresponds to a relatively flat prior on the probability scale (0, 1) (Broms, Hooten, and Fitzpatrick 2016). We will use these default priors here, but we specify them explicitly below for clarity

```
oven.priors <- list(alpha.normal = list(mean = 0, var = 2.72),
                    beta.normal = list(mean = 0, var = 2.72))
```

Our last step is to specify the number of samples to run the MCMC (`n.samples`), the amount of burn-in (`n.burn`), and how often we want to thin the posterior samples (`n.thin`). For a simple single species occupancy model, we shouldn't need too many samples and will only need a small amount of burn-in and minimal thinning.

```
n.samples <- 5000
n.burn <- 3000
n.thin <- 2
```

We are now nearly set to run the occupancy model. Single species occupancy models are fast, and so we set `n.omp.threads = 1` to indicate we won't use multiple threads to run the model. For more time consuming models, we can set `n.omp.threads` to a number greater than 1 and smaller than the number of threads on the computer you are using. Note this argument will only use multiple threads if `spOccupancy` was compiled for OpenMP support. The `verbose` argument is a logical value indicating whether or not MCMC sampler progress is reported to the screen. If `verbose = TRUE`, sampler progress is reported after the specified number of iterations in the `n.report` argument. We set `verbose = TRUE` and `n.report = 1000` to report progress after every 1000th MCMC iteration. The last three arguments to `PGOcc` (`k.fold`, `k.fold.threads`,

k.fold.seed) are used for performing k-fold cross-validation for model assessment. We will display how to perform k-fold cross-validation in a subsequent section. For now, we won't specify the arguments, which will tell PGOcc not to perform k-fold cross-validation.

```
out <- PGOcc(occ.formula = oven.occ.formula,
             det.formula = oven.det.formula,
             data = ovenHBEF,
             starting = oven.starting,
             n.samples = n.samples,
             priors = oven.priors,
             n.omp.threads = 1,
             verbose = TRUE,
             n.report = 1000,
             n.burn = n.burn,
             n.thin = n.thin)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Occupancy model with Polya-Gamma latent
variable fit with 373 sites.

Number of MCMC samples: 5000
Burn-in: 3000
Thinning Rate: 2
Total Posterior Samples: 1000


Source compiled with OpenMP support and model fit using 1 thread(s).

Sampling ...
Sampled: 1000 of 5000, 20.00%
--------------------------------------------------
Sampled: 2000 of 5000, 40.00%
--------------------------------------------------
Sampled: 3000 of 5000, 60.00%
--------------------------------------------------
Sampled: 4000 of 5000, 80.00%
--------------------------------------------------
Sampled: 5000 of 5000, 100.00%
```

```
names(out)
```

```
 [1] "beta.samples"  "alpha.samples" "z.samples"     "psi.samples"
 [5] "y.rep.samples" "X"             "X.p"           "y"
 [9] "n.samples"     "call"          "n.post"        "n.thin"
[13] "n.burn"        "pRE"           "psiRE"         "run.time"
```

PGOcc returns a list of class PGOcc with a suite of different objects, many of them being coda::mcmc objects of posterior samples. Notice the "Preparing the data" printed section doesn't have any information shown in it. spOccupancy model fitting functions will present messages when preparing the data for the model in this section, or will print out the default priors or starting values used when they are not specified in the function call. Here we specified everything explicitly so no information was reported.

For a nice summary of the regression parameters we can use `summary` on the resulting `PGOcc` object.

```
summary(out)
```

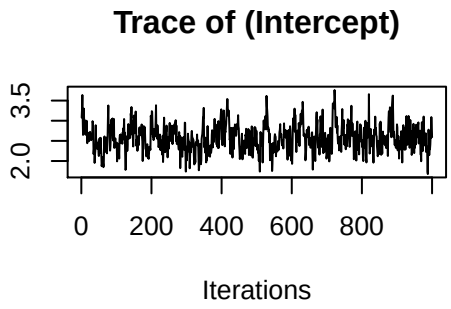
```
Call:
PGOcc(occ.formula = oven.occ.formula, det.formula = oven.det.formula,
    data = ovenHBEF, starting = oven.starting, priors = oven.priors,
    n.samples = n.samples, n.omp.threads = 1, verbose = TRUE,
    n.report = 1000, n.burn = n.burn, n.thin = n.thin)

Chain Information:
Total samples: 5000
Burn-in: 3000
Thin: 2
Total Posterior Samples: 1000

Occurrence:
               2.5%     25%     50%     75%   97.5%
(Intercept)  1.9637  2.3452  2.5726  2.7715  3.3012
Elevation   -1.7179 -1.3325 -1.1895 -1.0660 -0.8521
Elevation.2 -1.0656 -0.8476 -0.7418 -0.6356 -0.3914

Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept)  0.3830  0.5409  0.6317  0.7243  0.9081
day         -0.1560 -0.0730 -0.0278  0.0179  0.1078
tod         -0.2720 -0.1769 -0.1294 -0.0827  0.0032
day.2       -0.3262 -0.2119 -0.1610 -0.1097 -0.0120
```

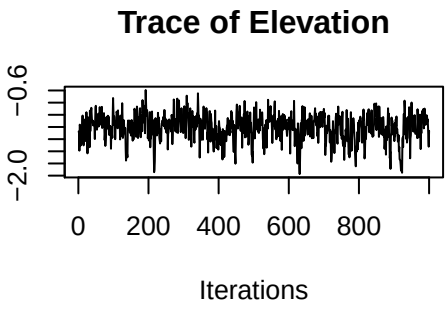
Note that all coefficients are printed on the logit scale. We see OVEN is fairly prominent in the forest given the large intercept value, and the negative linear and quadratic terms for `Elevation` suggest occurrence probability peaks at mid-elevations.

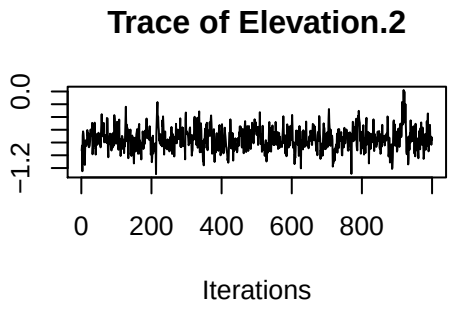
2.3 Convergence diagnostics

The posterior samples in the `PGOcc` object are `coda::mcmc` objects, which we can quickly assess for convergence visually using trace plots.

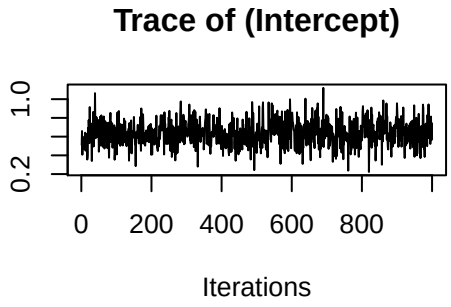
```
plot(out$beta.samples, density = FALSE)
```

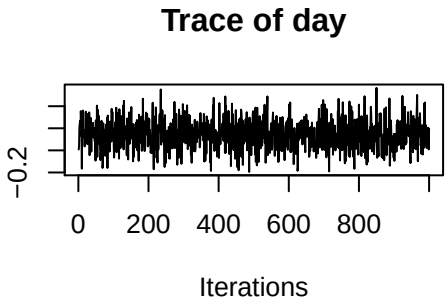


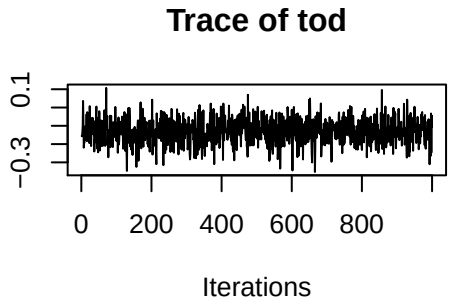


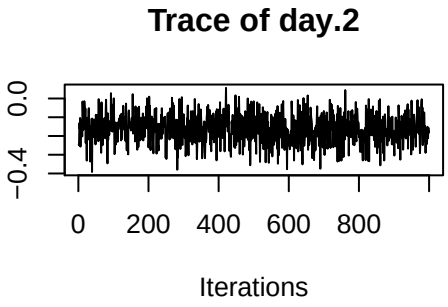



```
plot(out$alpha.samples, density = FALSE)
```









For a complete analysis (i.e., in a peer-reviewed manuscript), we will likely want to more formally check for convergence, perhaps using the Gelman-Rubin R-hat diagnostic (Brooks and Gelman 1998). This requires running multiple chains at largely different starting values for the regression parameters. For a single species

non-spatial occupancy model, we can accomplish this by running multiple chains sequentially (since they run really fast) with different starting values, then combining the output into a `coda::mcmc.list` object for use with the `coda::gelman.diag` function. Notice below we set `verbose = FALSE` to suppress the messages printed by `PGOcc`.

```
oven.starting <- list(alpha = 2,
                      beta = 2,
                      z = apply(ovenHBEF$y, 1, max, na.rm = TRUE))
out.2 <- PGOcc(occ.formula = oven.occ.formula,
               det.formula = oven.det.formula,
               data = ovenHBEF,
               starting = oven.starting,
               n.samples = n.samples,
               priors = oven.priors,
               n.omp.threads = 1,
               verbose = FALSE,
               n.report = 1000,
               n.burn = n.burn,
               n.thin = n.thin)
oven.starting <- list(alpha = -2,
                      beta = -2,
                      z = apply(ovenHBEF$y, 1, max, na.rm = TRUE))
out.3 <- PGOcc(occ.formula = oven.occ.formula,
               det.formula = oven.det.formula,
               data = ovenHBEF,
               starting = oven.starting,
               n.samples = n.samples,
               priors = oven.priors,
               n.omp.threads = 1,
               verbose = FALSE,
               n.report = 1000,
               n.burn = n.burn,
               n.thin = n.thin)
# beta convergence
gelman.diag(mcmc.list(out$beta.samples, out.2$beta.samples,
                      out.3$beta.samples))
```

```
Potential scale reduction factors:

            Point est. Upper C.I.
(Intercept)       1.00       1.01
Elevation         1.03       1.04
Elevation.2       1.00       1.00

Multivariate psrf

1
```

```
# alpha convergence
gelman.diag(mcmc.list(out$alpha.samples, out.2$alpha.samples,
                      out.3$alpha.samples))
```

```
Potential scale reduction factors:

            Point est. Upper C.I.
```

```
(Intercept)       1.00       1.01
day               1.00       1.00
tod               1.01       1.02
day.2             1.00       1.01

Multivariate psrf

1
```

All R-hat values are less than 1.1, indicating the chains have converged and we are in good shape to proceed.

2.4 Posterior predictive checks

The function `ppcOcc` performs a posterior predictive check on all `spOccupancy` model objects as a Goodness of Fit (GoF) assessment. The fundamental idea of a posterior predictive check is as follows: our model should generate data that closely align with the observed data. If there are drastic differences in the true data from the model generated data, our model likely is not very useful (Hobbs and Hooten 2015). GoF assessments are more complicated using binary data, like detection-nondetection used in occupancy models, as standard approaches are not valid assessments for binary data (Broms, Hooten, and Fitzpatrick 2016; McCullagh 2019). Thus, any approach to assess model fit for detection-nondetection data must bin the raw values in some manner, and then perform a model fit assessment on the binned values. There are numerous ways we could envision binning the raw detection-nondetection values (Kéry and Royle 2015).

The resulting `PGOcc` model object is sent as input to the `ppcOcc` function, along with a fit statistic (`fit.stat`) and numeric value indicating how to group the data (`group`). Currently supported fit statistics include the Freeman-Tukey statistic and the Chi-Square statistic (`freeman-tukey` or `chi-square`, respectively, Kéry and Royle (2015)). Currently, `ppcOcc` allows the user to group the data by row (site; `group = 1`) or column (replicate; `group = 2`). `ppcOcc` will then return a set of posterior samples for the fit statistic (or discrepancy measure) using the observed data (`fit.y`) and model generated data set (`fit.y.rep`), summed across all data points. These values can be used with the `summary` function to generate a Bayesian p-value. Bayesian p-values are sensitive to individual values, so we should also explore the discrepancy measures for each "grouped" data point. `ppcOcc` returns a matrix of posterior quantiles for the fit statistic for both the observed (`fit.y.group.quants`) and model generated data (`fit.y.rep.group.quants`) for each "grouped" data point.

We next perform a posterior predictive check using the Freeman-Tukey statistic grouping the data by sites. We summarize the posterior predictive check with the `summary` function, which reports a Bayesian p-value. A Bayesian p-value that hovers around 0.5 indicates adequate model fit, while values less than 0.1 or greater than 0.9 suggest our model does not fit the data well (Hobbs and Hooten 2015).

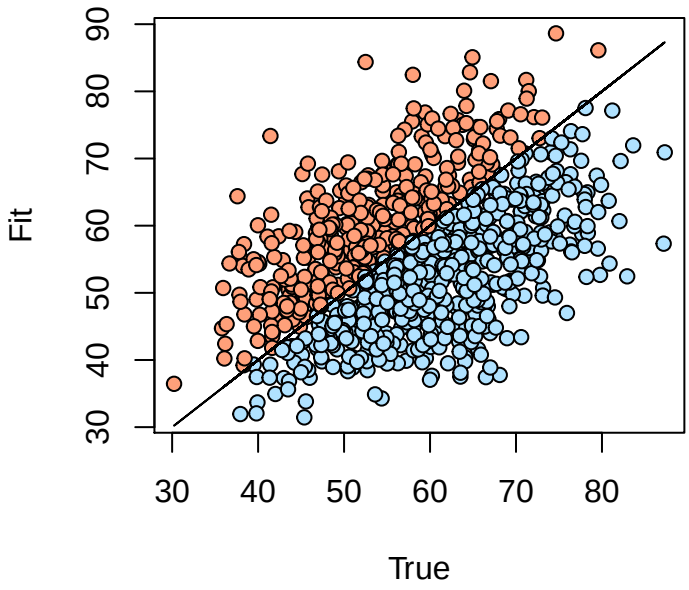
```
ppc.out <- ppcOcc(out, fit.stat = 'freeman-tukey', group = 1)
summary(ppc.out)
```

```
Call:
ppcOcc(object = out, fit.stat = "freeman-tukey", group = 1)

Chain Information:
Total samples: 5000
Burn-in: 3000
Thin: 2
Total Posterior Samples: 1000

Bayesian p-value:  0.383
Fit statistic:  freeman-tukey
```

The Bayesian p-value is the proportion of posterior samples of the fit statistic of the model generated data that are greater than the corresponding fit statistic of the true data, summed across all "grouped" data points.

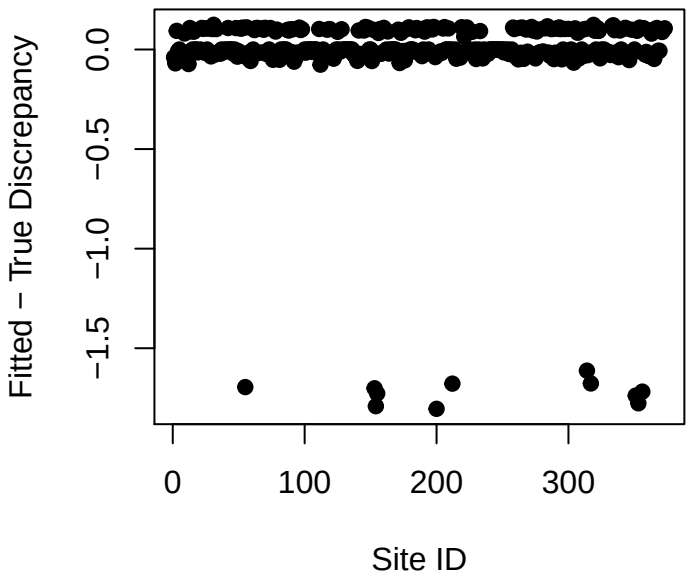
We can create a visual representation of the Bayesian p-value as follows, which is highly motivated by Kéry and Royle (2015).

```
ppc.df <- data.frame(fit = ppc.out$fit.y,
                     fit.rep = ppc.out$fit.y.rep,
                     color = 'lightskyblue1')
ppc.df$color[ppc.df$fit.rep > ppc.df$fit] <- 'lightsalmon'
plot(ppc.df$fit, ppc.df$fit.rep, bg = ppc.df$color, pch = 21,
     ylab = 'Fit', xlab = 'True')
lines(ppc.df$fit, ppc.df$fit, col = 'black')
```



Our Bayesian p-value indicates no lack of fit for the OVEN model. However, relying solely on the Bayesian p-value as an assessment of model fit is not always a great option, as individual data points can have an overbearing influence on the resulting summary value. Instead of summing across all data points for a single discrepancy measure, `ppcOcc` also allows us to explore discrepancy measures on a "grouped" point by point basis. The resulting `ppcOcc` object will contain the objects `fit.y.group.quants` and `fit.y.rep.group.quants`, which contain quantiles of the posterior distributions for the discrepancy measures of each grouped data point. Below we plot the difference in the discrepancy measure between the fitted and true data across each of the sites.

```
diff.fit <- ppc.out$fit.y.rep.group.quants[3, ] - ppc.out$fit.y.group.quants[3, ]
plot(diff.fit, pch = 19, xlab = 'Site ID', ylab = 'Fitted - True Discrepancy')
```



We see there are a few sites where the true discrepancy is much larger than the discrepancy under the fitted data. Here we will ignore this, but in a real analysis we would explore these sites further to see what could explain this pattern (e.g., are the sites close together in space?).

2.5 Model selection using WAIC

Posterior predictive checks allow us to assess how well our model fits the data, but they are not very useful if we want to compare multiple competing models and ultimately select a final model based on some criterion. Bayesian model selection is very much a constantly changing field, especially in the ecological and environmental sciences. See Hooten and Hobbs (2015) for an accessible overview of Bayesian model selection for ecologists.

For Bayesian hierarchical models like occupancy models, the most common Bayesian model selection criterion, DIC, is not applicable (Hooten and Hobbs 2015). Instead, we can use the Widely Applicable Information Criterion (Watanabe 2010) to compare a set of models and select the best performing model according to the WAIC for final analysis.

The WAIC is calculated for all `spOccupancy` model objects using the function `waicOcc`. We calculate the WAIC as

$$\text{WAIC} = -2 \times (\text{elpd} - \text{pD}),$$

where elpd is the expected log pointwise predictive density and PD is the effective number of parameters. We calculate elpd by calculating the likelihood for each posterior sample, taking the mean of these likelihoods, taking the log of the mean of the likelihoods, and summing these values across all sites. We calculate the effective number of parameters by calculating the variance of the log likelihood for each site taken over all posterior samples, and then summing these values across all sites. See Appendix S1 from Broms, Hooten, and Fitzpatrick (2016) for more details.

We calculate the WAIC using `waicOcc` for our OVEN model below.

```
waicOcc(out)
```

```
       elpd          pD        WAIC
-685.239088    7.069378 1384.616932
```

Next we rerun the OVEN model, but this time we assume occurrence is constant across the HBEF, and subsequently compare the WAIC value to the full model

```
out.small <- PGOcc(occ.formula = ~ 1,
           det.formula = oven.det.formula,
           data = ovenHBEF,
           starting = oven.starting,
           n.samples = n.samples,
           priors = oven.priors,
           n.omp.threads = 1,
           verbose = FALSE,
           n.burn = n.burn,
           n.thin = n.thin)
waicOcc(out.small)
```

```
       elpd          pD        WAIC
-729.216688    4.876786 1468.186949
```

Smaller values of WAIC indicate models with better performance. We see the WAIC for the model with elevation is smaller than the intercept only model, indicating elevation is an important predictor for OVEN occurrence in HBEF.

2.6 k-fold cross-validation

When focused primarily on predictive performance, a k-fold cross-validation approach is another attractive (but more computationally intensive) alternative to compare a series of models, especially since WAIC may not be a reliable metric for all models (Link, Sauer, and Niven 2020). In `spOccupancy`, k-fold cross-validation is accomplished using the arguments `k.fold`, `k.fold.threads`, and `k.fold.seed` in the model fitting function. A k-fold cross validation approach requires fitting a model k times, where each time the model is fit using J/k data points, where J is the total number of sites surveyed at least once in the data set. Each time the model is fit, it uses a different portion of the data and then predicts the remaining $J - J/k$ hold out values. Because the data are not used to fit the model, this yields true samples from the posterior predictive distribution that we can use to assess the predictive capacity of the model.

As a measure of out-of-sample predictive performance, we use the deviance as a cross-validation score following Hooten and Hobbs (2015). For K-fold cross-validation, our scoring function is computed as

$$-2\sum_{k=1}^{K}\log\left(\frac{\sum_{q=1}^{Q}\text{Bernoulli}(\boldsymbol{y}_k \mid \boldsymbol{p}^{(q)}\boldsymbol{z}_k^{(q)})}{Q}\right), \tag{3}$$

where $\boldsymbol{p}^{(q)}$ and $\boldsymbol{z}_k^{(q)}$ are MCMC samples of detection probability and latent occurrence, respectively, arising from a model that is fit without the observations $\boldsymbol{y}_k$. Q is the total number of posterior samples from the MCMC sampler. The -2 is used so that smaller values indicate better model fit, which aligns with most information criteria used for model fit (like the WAIC implemented using `waicOcc`).

The final three arguments (`k.fold`, `k.fold.threads`, `k.fold.seed`) in `PGOcc` control whether or not k-fold cross validation is performed following the complete fit of the model using the entire data set. The `k.fold` argument indicates the number of k folds to use for cross-validation. If `k.fold` is not specified, cross-validation is not performed and `k.fold.threads` and `k.fold.seed` are ignored. The `k.fold.threads`

argument indicates the number of threads to use for running the k models in parallel across multiple threads. Parallel processing is accomplished using the R packages `foreach` and `doParallel`. Specifying `k.fold.threads > 1` can substantially increase run time since it allows for models to be fit simultaneously on different threads rather than sequentially. The `k.fold.seed` indicates the seed used to randomly split the data into k groups. This is by default set to 100.

Below we refit the occupancy model with elevation (linear and quadratic) as an occurrence predictor this time performing 4-fold cross-validation. We set `k.fold = 4` to perform 4-fold cross-validation and `k.fold.threads = 1` to run the model using 1 thread. Normally we would set `k.fold.threads = 4`, but using multiple threads leads to complications when compiling this vignette, so we leave that to you to explore the computational improvements of performing cross-validation across multiple cores. We subsequently refit the intercept only occupancy model, and compare the deviance metrics from the 4-fold cross-validation.

```
out.k.fold <- PGOcc(occ.formula = ~ Elevation,
                    det.formula = oven.det.formula,
                    data = ovenHBEF,
                    starting = oven.starting,
                    n.samples = n.samples,
                    priors = oven.priors,
                    n.omp.threads = 1,
                    verbose = TRUE,
                    n.report = 1000,
                    n.burn = n.burn,
                    n.thin = n.thin,
                    k.fold = 4,
                    k.fold.threads = 1)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Occupancy model with Polya-Gamma latent
variable fit with 373 sites.

Number of MCMC samples: 5000
Burn-in: 3000
Thinning Rate: 2
Total Posterior Samples: 1000


Source compiled with OpenMP support and model fit using 1 thread(s).

Sampling ...
Sampled: 1000 of 5000, 20.00%
--------------------------------------------------
Sampled: 2000 of 5000, 40.00%
--------------------------------------------------
Sampled: 3000 of 5000, 60.00%
--------------------------------------------------
Sampled: 4000 of 5000, 80.00%
--------------------------------------------------
Sampled: 5000 of 5000, 100.00%
----------------------------------------
```

```
    Cross-validation
----------------------------------------

Performing 4-fold cross-validation using 1 thread(s).
```

```
# Model fitting information is surpressed for space.
out.int.k.fold <- PGOcc(occ.formula = ~ 1,
                        det.formula = oven.det.formula,
                data = ovenHBEF,
                starting = oven.starting,
                n.samples = n.samples,
                priors = oven.priors,
                n.omp.threads = 1,
                verbose = TRUE,
                n.report = 1000,
                n.burn = n.burn,
                n.thin = n.thin,
                        k.fold = 4,
                        k.fold.threads = 1)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Occupancy model with Polya-Gamma latent
variable fit with 373 sites.

Number of MCMC samples: 5000
Burn-in: 3000
Thinning Rate: 2
Total Posterior Samples: 1000


Source compiled with OpenMP support and model fit using 1 thread(s).

Sampling ...
Sampled: 1000 of 5000, 20.00%
--------------------------------------------------
Sampled: 2000 of 5000, 40.00%
--------------------------------------------------
Sampled: 3000 of 5000, 60.00%
--------------------------------------------------
Sampled: 4000 of 5000, 80.00%
--------------------------------------------------
Sampled: 5000 of 5000, 100.00%
----------------------------------------
    Cross-validation
----------------------------------------

Performing 4-fold cross-validation using 1 thread(s).
```

The cross-validation metric (model deviance) is stored in the `k.fold.deviance` tag of the resulting model object.

```
out.k.fold$k.fold.deviance
```

```
[1] 1518.631
```

```
out.int.k.fold$k.fold.deviance
```

```
[1] 1537.448
```

Similar to the results from the WAIC, we see the model including elevation with a predictor outperforms the intercept only model.

2.7 Prediction

All resulting model objects from `spOccupancy` model functions can be used with `predict` to generate a series of posterior predictive samples at non-sampled locations, given the values of all covariates used in the model fitting process. The object `hbefElev` (provided in the `spOccupancy` package) contains elevation values at a 30x30m resolution from the National Elevation Dataset across the entire HBEF. The values are standardized using the mean and standard deviation of the elevation values used to fit the model. We load the data below

```
data(hbefElev)
str(hbefElev)
```

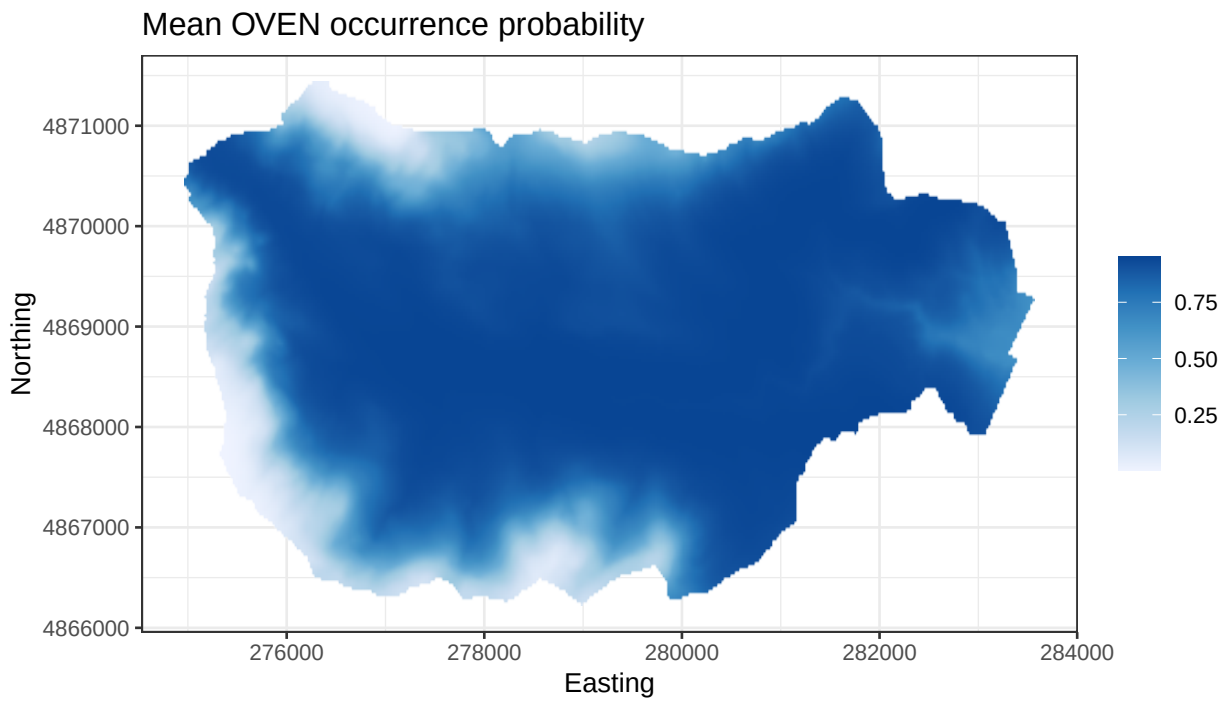
```
'data.frame':   46091 obs. of  3 variables:
 $ val     : num  2.07 2.08 2.09 2.1 2.12 ...
 $ Easting : num  276269 276291 276314 276336 276358 ...
 $ Northing: num  4871425 4871425 4871425 4871425 4871425 ...
```

The column `val` contains the standardized elevation values, while `Easting` and `Northing` contain the spatial coordinates that we will use for plotting. We can obtain posterior predictive samples for the occurrence probabilities at these sites by using the `predict` function and our `PGOcc` model object.

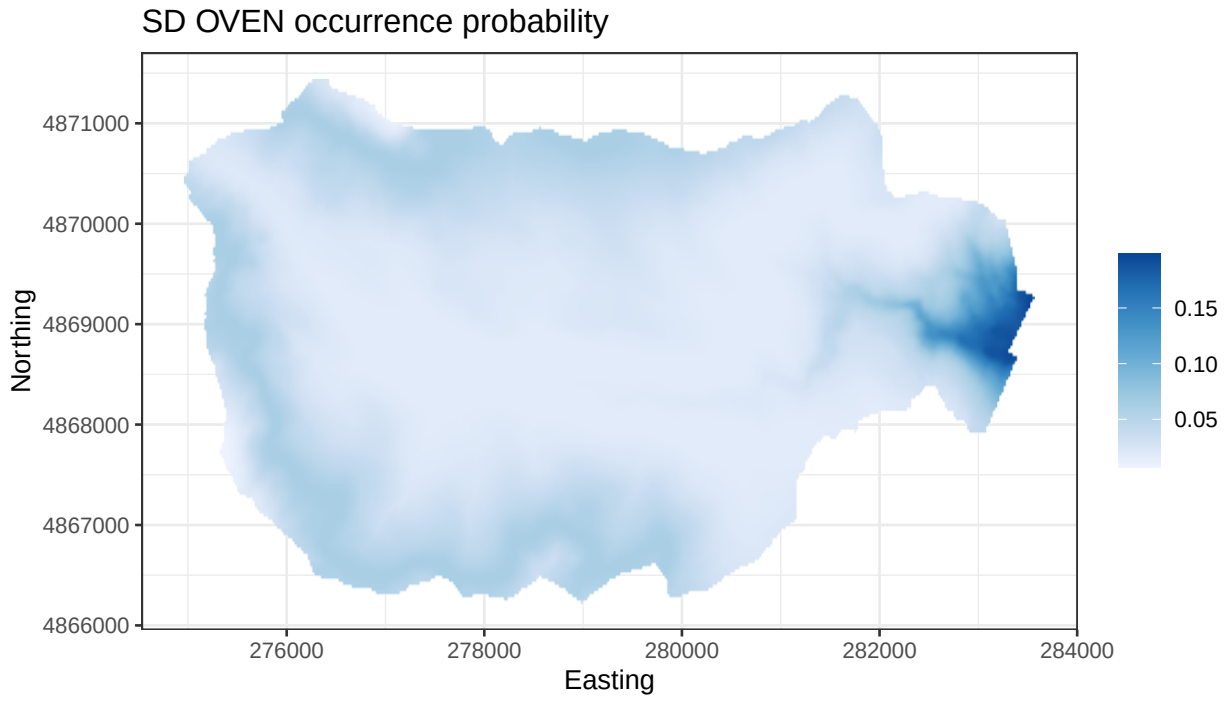
```
X.0 <- cbind(1, hbefElev$val, hbefElev$val^2)
out.pred <- predict(out, X.0)
```

For `PGOcc` objects, the `predict` function takes two arguments: (1) the `PGOcc` model object; and (2) a matrix or data frame consisting of the design matrix for the prediction locations (including an intercept). The resulting object consists of posterior predictive samples for the latent occurrence probabilities (`psi.0.samples`) and latent occurrence values (`z.0.samples`). The beauty of the Bayesian paradigm is that these predictions all have fully propagated uncertainty. We can use these values to create plots of the predicted mean occurrence values, as well as their standard deviation.

```
plot.dat <- data.frame(x = hbefElev$Easting,
                       y = hbefElev$Northing,
                       mean.psi = apply(out.pred$psi.0.samples, 2, mean),
                       sd.psi = apply(out.pred$psi.0.samples, 2, sd))

dat.stars <- st_as_stars(plot.dat, dims = c('x', 'y'))
ggplot() +
  geom_stars(data = dat.stars, aes(x = x, y = y, fill = mean.psi)) +
  scale_fill_distiller(palette = 'Blues', direction = 1, na.value = 'transparent') +
  labs(x = 'Easting', y = 'Northing', fill = '',
       title = 'Mean OVEN occurrence probability') +
  theme_bw()
```




```
ggplot() +
  geom_stars(data = dat.stars, aes(x = x, y = y, fill = sd.psi)) +
  scale_fill_distiller(palette = 'Blues', direction = 1, na.value = 'transparent') +
  labs(x = 'Easting', y = 'Northing', fill = '',
       title = 'SD OVEN occurrence probability') +
  theme_bw()
```



3 Single species spatial occupancy models

3.1 Basic model description

When working across large spatial domains, accounting for residual spatial autocorrelation in species distributions can often improve predictive performance, leading to more accurate species distribution maps (Guélat and Kéry 2018; Lany et al. 2020). We extend the basic single species occupancy model to incorporate a spatial Gaussian Process that accounts for unexplained spatial variation in species occurrence across a region of interest. The species-specific occurrence probability at site j, ψ_j, now takes the form

$$\text{logit}(\psi_j) = \boldsymbol{x}_j' \cdot \boldsymbol{\beta} + \text{w}_j, \tag{4}$$

where w_j is a realization from a zero-mean spatial Gaussian Process, i.e.,

$$\mathbf{w} \sim N(\mathbf{0}, \boldsymbol{\Sigma}(\boldsymbol{s}, \boldsymbol{s'}, \boldsymbol{\theta})). \tag{5}$$

We define $\boldsymbol{\Sigma}(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{\theta})$ as a $J \times J$ covariance matrix that is a function of the distances between any pair of site coordinates $\boldsymbol{s}$ and $\boldsymbol{s}'$ and a set of parameters ($\boldsymbol{\theta}$) that govern the spatial process. The vector $\boldsymbol{\theta}$ is equal to $\boldsymbol{\theta} = \{\sigma^2, \phi, \nu\}$, where σ^2 is a spatial variance parameter, ϕ is a spatial decay parameter, and ν is a spatial smoothness parameter. ν is only specified when using a Matern correlation function.

The detection portion of the occupancy model remains unchanged from the non-spatial occupancy model and follows Equation (2). Single species spatial occupancy models, like all models in `spOccupancy` are fit using Pólya-Gamma data augmentation (see MCMC sampler vignette for details).

When the number of sites is moderately large, say 1000, the above described spatial Gaussian process model can be drastically slow as a result of needing to take the inverse of the spatial covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{\theta})$ at each MCMC iteration. Numerous approximation methods exist to reduce this computational cost (Heaton et al. 2019). One attractive approach is the Nearest Neighbor Gaussian Process (NNGP; Datta et al. (2016)). Instead of modeling the spatial process using a full Gaussian Process as shown in Equation (5), we replace the Gaussian Process prior specification with a NNGP, which leads to drastic increases in run time with nearly identicial inference and prediction as the full Gaussian Process specification. See Datta et al. (2016), Finley et al. (2019), and the MCMC sampler vignette for additional statistical details on NNGPs and their implementation in spatial occupancy models.

3.2 Fitting single species spatial occupancy models with `spPGOcc`

The function `spPGOcc` fits single species spatial occupancy models using Pólya-Gamma latent variables, where spatial autocorrelation is accounted for using a spatial Gaussian Process. `spPGOcc` fits saptial occupancy models using either a full Gaussian process or an NNGP. See Finley, Datta, and Banerjee (2020) for details on using NNGPs with Polya-Gamma latent variables.

We will fit the same occupancy model for OVEN that we fit previously using `PGOcc`, but we will now make the model spatially explicit by incorporating a spatial process with `spPGOcc`. First, let's take a look at the arguments for `spPGOcc`:

```
spPGOcc(occ.formula, det.formula, data, starting, n.batch,
        batch.length, accept.rate = 0.43, priors,
        cov.model = "exponential", tuning, n.omp.threads = 1,
        verbose = TRUE, NNGP = FALSE, n.neighbors = 15,
        search.type = "cb", n.report = 100,
        n.burn = round(.10 * n.batch * batch.length),
        n.thin = 1, k.fold, k.fold.threads = 1,
    k.fold.seed = 100, ...)
```

We will walk through each of the arguments to `spPGOcc` in the context of our Ovenbird example. The occurrence (`occ.formula`) and detection (`det.formula`) formulas, as well as the list of data (`data`), take the same form as we saw in `PGOcc`, with the exception that random intercepts can only be specified in `det.formula`. Notice the `coords` matrix in the `ovenHBEF` list of data. We did not use this for `PGOcc` but specifying the spatial coordinates in `data` is required for all spatially explicit models in `spOccupancy`.

```
oven.occ.formula <- ~ Elevation + Elevation.2
oven.det.formula <- ~ day + tod + day.2
str(ovenHBEF) # coords is required for spPGOcc.
```

```
List of 4
 $ y       : num [1:373, 1:3] 1 1 0 1 0 0 1 1 1 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "1" "2" "3"
 $ occ.covs: num [1:373, 1:2] -0.889 -0.765 -0.413 -0.14 -0.13 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "Elevation" "Elevation.2"
 $ det.covs:List of 3
  ..$ day  : num [1:373, 1:3] -1.62 -1.62 -1.62 -1.62 -1.62 ...
  ..$ tod  : num [1:373, 1:3] -1.565 -1.378 -1.084 -0.79 -0.549 ...
  ..$ day.2: num [1:373, 1:3] 2.61 2.61 2.61 2.61 2.61 ...
 $ coords  : num [1:373, 1:2] 280000 280000 280000 280001 280000 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:2] "Easting" "Northing"
```

The starting values (`starting`) are again specified in a list. Valid tags for starting values now additionally include the parameters associated with the spatial random effects. These include: `sigma.sq` (spatial variance parameter), `phi` (spatial range parameter), `w` (the latent spatial random effects at each site), and `nu` (spatial smoothness parameter). `nu` is only specified if using a Matern covariance function (i.e., `cov.model = 'matern'`). `spOccupancy` supports four spatial covariance models (`exponential`, `spherical`, `gaussian`, and `matern`), which are specified in the `cov.model` argument. Here we will use an exponential covariance model. As a starting value for the spatial range parameter `phi`, we compute the mean distance between points in HBEF and then set it equal to 3 divided by this mean distance. When using an exponential covariance function, $\frac{3}{\phi}$ is the effective range, or the distance at which the residual spatial correlation between two sites is 0.05 (Banerjee, Carlin, and Gelfand 2003). Thus our initial guess for this effective range is the average distance betweeen sites across HBEF.

```
# Distances between sites
dist.hbef <- dist(ovenHBEF$coords)
# Exponential covariance model
cov.model <- "exponential"
oven.starting <- list(alpha = 0,
                      beta = 0,
                      z = apply(ovenHBEF$y, 1, max, na.rm = TRUE),
                      sigma.sq = 2,
                      phi = 3 / mean(dist.hbef),
                      w = rep(0, nrow(ovenHBEF$y)))
```

The next three arguments (`n.batch`, `batch.length`, and `accept.rate`) are all related to the Adaptive MCMC sampler we use to fit the model. Updates for the spatial range parameter (and smoothness parameter if `cov.model = 'matern'`) require the use of a Metropolis Hastings algorithm. We implement an adaptive Metropois-Hastings algorithm discussed in Roberts and Rosenthal (2009). This algorithm adjusts the tuning values for each parameter that requires a Metropolis-Hastings update within the sampler itself. This process

results in a more efficient sampler than if we were to fix the tuning parameters prior to fitting the model. The parameter `accept.rate` is the target acceptance rate for each parameter, and the algorithm will adjust the tuning parameters to hover around this value. The default value is 0.43, which we suggest leaving as is unless you have a good reason to change it. The tuning parameters are updated after a single "batch". We must specify the total `n.batch` batches, where each "batch" consists of `batch.length` MCMC samples. Thus, the total number of MCMC samples is `n.batch * batch.length`. Typically, we set `batch.length = 25` and then play around with `n.batch` until convergence is reached. Here we set `n.batch = 400` for a total of 10000 MCMC samples. We will additionally specify a burn-in period of 2000 samples and a thinning rate of 8. We also need to specify an initial value for the tuning parameters for the spatial decay and smoothness parameters (if applicable). These values are sent as input in the form of a list with tags `phi` and `nu`. The initial tuning value can be any value greater than 0, but we recommend starting the value out around 0.5. After some initial runs of the model, if you notice the final acceptance rate of a parameter is much larger or smaller than the target acceptance rate (`accept.rate`), you can then change the initial tuning value to get closer to the target rate. Here we set the initial tuning value for `phi` to 1 after some initial runs of the model.

```
batch.length <- 25
n.batch <- 400
n.burn <- 2000
n.thin <- 8
oven.tuning <- list(phi = 1)
```

Priors are again specified in a list in the argument `priors`. We assume an inverse gamma prior for the spatial variance parameter `sigma.sq` (tag is `sigma.sq.ig`), and uniform priors for the spatial decay parameter `phi` and smoothness parameter `nu` (if Matern), with the associated tags `phi.unif` and `nu.unif`. The hyperparameters of the inverse Gamma are passed as a vector of length two, with the first and second elements corresponding to the shape and scale, respectively. The lower and upper bounds of the uniform distribution are passed in as a two-element vector for the uniform priors.

The priors for the spatial parameters in a spatially-explicit model must be at least weakly informative for the model to converge (Banerjee, Carlin, and Gelfand 2003). For the inverse-Gammma prior on the spatial variance, we typically set the shape parameter to 2 and the scale parameter equal to our best guess of the spatial variance. Based on our previous work with these data, we expect the residual spatial variation to be minimal, and so we set the scale parameter below to 1. For the spatial decay parameter, we determine the bounds of the uniform distribution by computing the smallest distance between sites and the largest distance between sites. We then set the lower bound of the uniform to `3/max` and the upper bound to `3/min`, where min and max correspond to the predetermined distances between sites.

```
min.dist <- min(dist.hbef)
max.dist <- max(dist.hbef)
oven.priors <- list(beta.normal = list(mean = 0,
                                       var = 2.72),
                    alpha.normal = list(mean = 0,
                                        var = 2.72),
                    sigma.sq.ig = c(2, 1),
                    phi.unif = c(3/max.dist, 3/min.dist))
```

The argument `n.omp.threads` specifies the number of threads to use for parallelization, while `verbose` specifies whether or not to print the progress of the sampler. We *highly* recommend setting `verbose = TRUE` for all spatial models to ensure the adaptive MCMC is working as you want. The argument `n.report` specifies the interval to report the Metropolis sampler acceptance. Note that `n.report` is specified in terms of batches, not the overall number of samples. Below we set `n.report = 100`, which will result in information on the acceptance rate and tuning parameters every 100th batch.

```
n.omp.threads <- 1
verbose <- TRUE
n.report <- 100
```

The remaining parameters (`NNGP`, `n.neighbors` and `search.type`) relate to whether or not you want to fit the model with a Gaussian Process or NNGP. The argument `NNGP` is a logical value indicating whether to fit the model with an NNGP (`TRUE`) or a regular Gaussian Process (`FALSE`). For data sets that have more than 1000 locations, using an NNGP will have substantial increases in run time. Even for more modest size data sets (like the HBEF data set), using an NNGP will be quite a bit faster. Unless you are concerned about the NNGP approximation for some reason, we recommend setting `NNGP = TRUE`. The argument `n.neighbors` and `search.type` specify the number of neighbors used in the NNGP and the nearest neighbor search algorithm, respectively, to use for the NNGP model. Generally, the default values of these arguments will be adequate. Datta et al. (2016) showed that setting `n.neighbors = 15` is usually sufficient, although for certain data sets a good approximation can be achieved with as small as five neighbors, which could substantially decrease run time. We generally recommend leaving `search.type = "cb"`, as this results in a fast code book nearest neighbor search algorithm. However, details on when you may want to change this are described in Finley, Datta, and Banerjee (2020). We will run an NNGP model using the default value for `search.type` and setting `n.neighbors = 5`.

We now fit the model and summarize the results using `summary`.

```
out.sp <- spPGOcc(occ.formula = oven.occ.formula,
                  det.formula = oven.det.formula,
                  data = ovenHBEF,
                  starting = oven.starting,
                  n.batch = n.batch,
                  batch.length = batch.length,
                  priors = oven.priors,
                  cov.model = cov.model,
                  NNGP = TRUE,
                  n.neighbors = 5,
                  tuning = oven.tuning,
                  n.report = n.report,
                  n.burn = n.burn,
                  n.thin = n.thin)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Building the neighbor list
----------------------------------------
----------------------------------------
Building the neighbors of neighbors list
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
NNGP Occupancy model with Polya-Gamma latent
variable fit with 373 sites.

Number of MCMC samples: 10000 (400 batches of length 25)
Burn-in: 2000
Thinning Rate: 8
Total Posterior Samples: 1000

Using the exponential spatial correlation model.

Using 5 nearest neighbors.
```

```
Source compiled with OpenMP support and model fit using 1 thread(s).

Adaptive Metropolis with target acceptance rate: 43.0
Sampling ...
Batch: 100 of 400, 25.00%
    parameter   acceptance  tuning
    phi     32.0        0.65051
-------------------------------------------------
Batch: 200 of 400, 50.00%
    parameter   acceptance  tuning
    phi     80.0        0.62500
-------------------------------------------------
Batch: 300 of 400, 75.00%
    parameter   acceptance  tuning
    phi     32.0        0.35701
-------------------------------------------------
Batch: 400 of 400, 100.00%
```

```
class(out.sp)
```

```
[1] "spPGOcc"
```

```
names(out.sp)
```

```
 [1] "beta.samples"   "alpha.samples"  "z.samples"      "psi.samples"
 [5] "y.rep.samples"  "theta.samples"  "w.samples"      "tune"
 [9] "accept"         "coords"         "X"              "X.p"
[13] "y"              "call"           "n.samples"      "n.neighbors"
[17] "cov.model.indx" "type"           "n.post"         "n.thin"
[21] "n.burn"         "pRE"            "run.time"
```

```
summary(out.sp)
```

```
Call:
spPGOcc(occ.formula = oven.occ.formula, det.formula = oven.det.formula,
    data = ovenHBEF, starting = oven.starting, priors = oven.priors,
    tuning = oven.tuning, cov.model = cov.model, NNGP = TRUE,
    n.neighbors = 5, n.batch = n.batch, batch.length = batch.length,
    n.report = n.report, n.burn = n.burn, n.thin = n.thin)

Chain Information:
Total samples: 10000
Burn-in: 2000
Thin: 8
Total Posterior Samples: 1000

Occurrence:
               2.5%     25%     50%     75%   97.5%
(Intercept)  2.1521  2.5752  2.8503  3.1683  3.9386
Elevation   -2.0631 -1.5855 -1.3891 -1.2169 -0.9097
Elevation.2 -1.2635 -0.9910 -0.8512 -0.7204 -0.4264

Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept)  0.4063  0.5597  0.6404  0.7283  0.8920
```

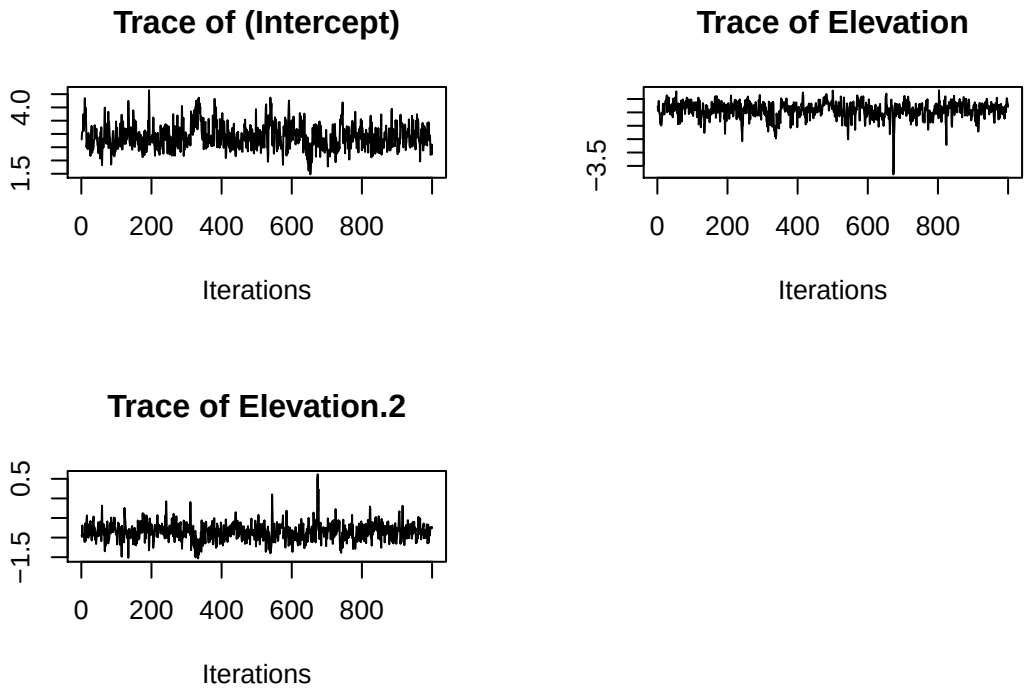
```
day          -0.1613 -0.0776 -0.0338  0.0099  0.0863
tod          -0.2736 -0.1813 -0.1379 -0.0875  0.0089
day.2        -0.3226 -0.2166 -0.1623 -0.1138 -0.0053

Covariance:
           2.5%    25%    50%    75%  97.5%
sigma.sq 0.2120 0.4957 0.7318 1.1179 2.9546
phi      0.0009 0.0018 0.0033 0.0115 0.0264
```

We see `spPGOcc` returns a list of class `spPGOcc` and consists of posterior samples for all parameters. Note that posterior samples for spatial parameters are stored in the list element `theta.samples`. The `summary` function reveals model results generally align with those found using the non-spatial model.

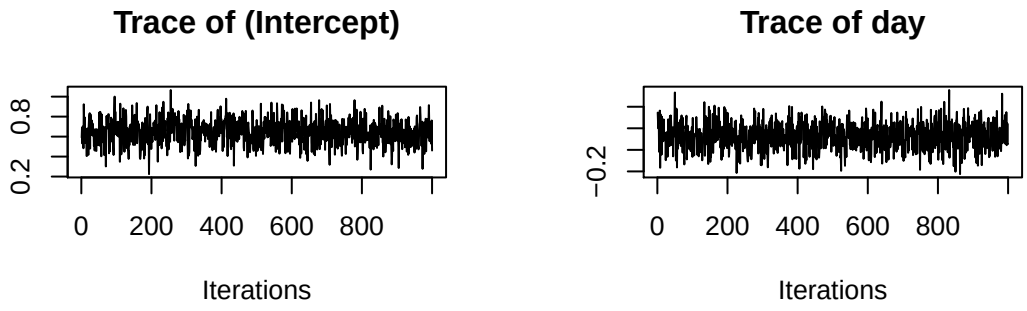
3.3 Convergence diagnostics

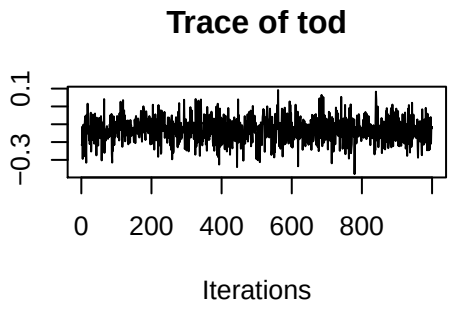
Convergence diagnostics, posterior predictive checks, model selection, and out-of-sample prediction all proceed analogously to what we saw with the non-spatial occupancy model using `PGOcc`.

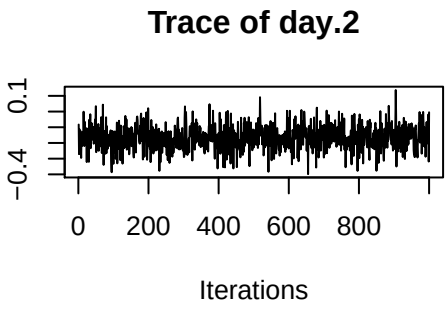
```
plot(out.sp$beta.samples, density = FALSE)
```



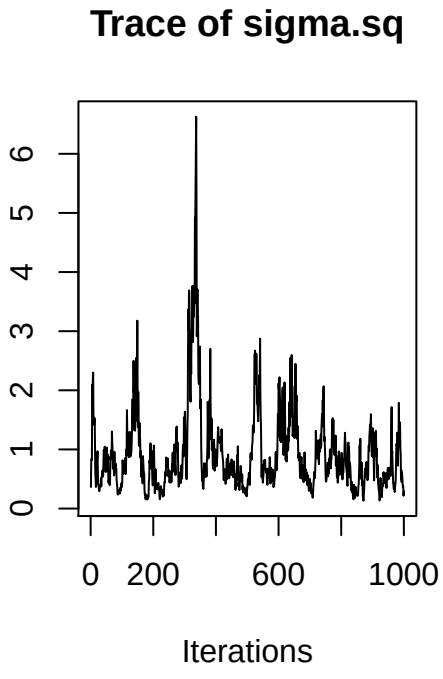

```
plot(out.sp$alpha.samples, density = FALSE)
```

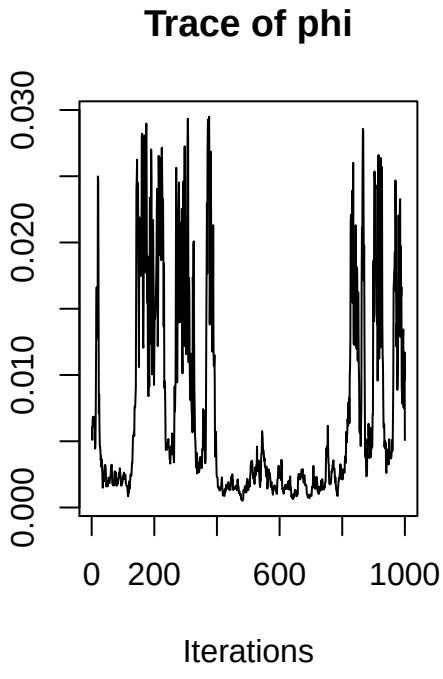







```
plot(out.sp$theta.samples, density = FALSE)
```





We should run the chain for a bit longer to ensure convergence of the spatial parameters, but we'll resist doing so for now. Convergence can be more formally assessed using the Gelman-Rubin diagnostic as done for the nonspatial model.

3.4 Posterior predictive checks

For our posterior predictive check, we send the `spPGOcc` model object to the `ppcOcc` function, this time grouping by replicate (`group = 2`) instead of by site (`group = 1`).

```
ppc.sp.out <- ppcOcc(out.sp, fit.stat = 'freeman-tukey', group = 2)
summary(ppc.sp.out)
```

```
Call:
ppcOcc(object = out.sp, fit.stat = "freeman-tukey", group = 2)

Chain Information:
Total samples: 10000
Burn-in: 2000
Thin: 8
Total Posterior Samples: 1000

Bayesian p-value:  0.89
Fit statistic:  freeman-tukey
```

The Bayesian p-value is hovering right around 0.9 potentially suggesting a lack of fit, but we would want to run the model longer to ensure convergence before determining if there is lack of fit.

3.5 Model selection using WAIC

We next use the `waicOcc` function to compute the WAIC, which we can compare to the non-spatial model to assess the benefit of incorporating the spatial random effects.

```
waicOcc(out.sp)
```

```
      elpd         pD       WAIC
-665.43616   23.96199 1378.79629
```

```
# Compare to non-spatial model
waicOcc(out)
```

```
       elpd          pD        WAIC
-685.239088    7.069378 1384.616932
```

We see the WAIC value for the spatial model is slightly smaller to that of the nonspatial model, indicating that incorporation of the spatial random effects may yield improvement in predictive performance, although this improvement would likely be quite small. This is not all that surprising, as we expect elevation to soak up most of the spatial variation in OVEN occurrence across the forest.

3.6 Prediction

Finally, we can perform out of sample prediction using the `predict` function just as before. Out of sample prediction for spatial models is more computationally intensive than non-spatial models, and so the `predict` function for `spPGOcc` class objects also has options for parallelization (`n.omp.threads`) and reporting sampler progress (`verbose` and `n.report`). Note that for `spPGOcc`, you also need to supply the coordinates of the out of sample prediction locations in addition to the covariate values. We do not execute the below code, but it can be used to compare prediction results from the nonspatial to the spatial model (which are extremely similar, except the standard deviations for the spatial predictions are larger).

```
coords.0 <- as.matrix(hbefElev[, c('Easting', 'Northing')])
out.sp.pred <- predict(out.sp, X.0, coords.0, verbose = FALSE)
```

```
plot.dat <- data.frame(x = hbefElev$Easting,
                       y = hbefElev$Northing,
                       mean.psi = apply(out.sp.pred$psi.0.samples, 2, mean),
                       sd.psi = apply(out.sp.pred$psi.0.samples, 2, sd))
dat.stars <- st_as_stars(plot.dat, dims = c('x', 'y'))
ggplot() +
  geom_stars(data = dat.stars, aes(x = x, y = y, fill = mean.psi)) +
  scale_fill_distiller(palette = 'Blues', direction = 1, na.value = 'transparent') +
  labs(x = 'Easting', y = 'Northing', fill = '',
       title = 'Mean OVEN occurrence probability') +
  theme_bw()
ggplot() +
  geom_stars(data = dat.stars, aes(x = x, y = y, fill = sd.psi)) +
  scale_fill_distiller(palette = 'Blues', direction = 1, na.value = 'transparent') +
  labs(x = 'Easting', y = 'Northing', fill = '',
       title = 'SD OVEN occurrence probability') +
  theme_bw()
```

4 Multispecies occupancy models

4.1 Basic model description

Let $z_{i,j}$ be the true presence (1) or absence (0) of a species i at site j, with $j = 1, \ldots, J$ and $i = 1, \ldots, N$. We assume the latent occurrence process arises from a Bernoulli process following

$$\begin{aligned} z_{i,j} &\sim \text{Bernoulli}(\psi_{i,j}), \\ \text{logit}(\psi_{i,j}) &= \boldsymbol{x}'_j \cdot \boldsymbol{\beta}_i, \end{aligned} \tag{6}$$

where $\psi_{i,j}$ is the probability of occurrence of species i at site j, which is a function of site-specific covariates $\boldsymbol{X}$ and a vector of species-specific regression coefficients ($\boldsymbol{\beta}_i$). The regression coefficients in multispecies occupancy models are envisioned as random effects arising from a common community level distribution:

$$\boldsymbol{\beta}_i \sim \text{Normal}(\boldsymbol{\mu}_\beta, \boldsymbol{T}_\beta), \tag{7}$$

where $\boldsymbol{\mu}_\beta$ is a vector of community level mean effects for each occurrence covariate effect (including the intercept) and $\boldsymbol{T}_\beta$ is a diagonal matrix with diagonal elements $\boldsymbol{\tau}^2_\beta$ that represent the variability of each occurrence covariate effect among species in the community.

We do not directly observe $z_{i,j}$ and rather we observe an imperfect representation of the latent occurrence process. Let $y_{i,j,k}$ be the observed detection (1) or nondetection (0) of a species i at site j during replicate k for each of $k = 1, \ldots, K_j$ replicates at each site j. We envision the detection-nondetection data as arising from a Bernoulli process conditional on the true latent occurrence process:

$$\begin{aligned} y_{i,j,k} &\sim \text{Bernoulli}(p_{i,j,k} \cdot z_{i,j}), \\ \text{logit}(p_{i,j,k}) &= \boldsymbol{v}'_{i,j,k} \cdot \boldsymbol{\alpha}_i, \end{aligned} \tag{8}$$

where $p_{i,j,k}$ is the probability of detecting species i at site j during replicate k (given it is present at site j), which is a function of site and replicate specific covariates $\boldsymbol{V}$ and a vector of species-specific regression

coefficients ($\boldsymbol{\alpha}_i$). Similarly to the occurrence regression coefficients, the species specific detection coefficients are envisioned as random effects arising from a common community level distribution:

$$\boldsymbol{\alpha}_i \sim \text{Normal}(\boldsymbol{\mu}_\alpha, \boldsymbol{T}_\alpha), \tag{9}$$

where $\boldsymbol{\mu}_\alpha$ is a vector of community level mean effects for each detection covariate effect (including the intercept) and $\boldsymbol{T}_\alpha$ is a diagonal matrix with diagonal elements $\boldsymbol{\tau}^2_\alpha$ that represent the variability of each detection covariate effect among species in the community.

To complete the Bayesian specification of the model, we assign multivariate normal priors for the occurrence ($\boldsymbol{\mu}_\beta$) and detection ($\boldsymbol{\mu}_\alpha$) community-level regression coefficient means and independent inverse-Gamma priors for each element of $\boldsymbol{\tau}^2_\beta$ and $\boldsymbol{\tau}^2_\alpha$. We again use Polya-Gamma data augmentation to yield an efficient implementation of the multispecies occupancy model, which is described in depth in the MCMC sampler vignette.

4.2 Fitting multispecies occupancy models with `msPGOcc`

`spOccupancy` uses nearly identical syntax for fitting multispecies models as it does for single species models and provides the same functionality for posterior predictive checks, model assessment and selection using WAIC (and k-fold cross-validation), and out of sample prediction. The `msPGOcc` function fits nonspatial multispecies occupancy models using Polya-Gamma latent variables, which results in substantial increases in run time compared to standard implementations of logit link multispecies occupancy models. `msPGOcc` has exactly the same arguments as `PGOcc`:

```
msPGOcc(occ.formula, det.formula, data, starting, n.samples, priors,
        n.omp.threads = 1, verbose = TRUE, n.report = 100,
        n.burn = round(.10 * n.samples), n.thin = 1,
    k.fold, k.fold.threads = 1, k.fold.seed, ...)
```

We will again use the Hubbard Brook data in `hbef2015` as an example data set, but we will now model occurrence for all 12 species in the community. Below we reload the `hbef2015` data set to get a fresh copy.

```
data(hbef2015)
```

We will model occurrence for all species as a function of linear and quadratic elevation, and detection as a function of linear and quadratic day of survey as well as the time of day the survey occurred. These models are specified in `occ.formula` and `det.formula` as before, which reference variables stored in the `data` list. Random intercepts can be included in both the occurrence and detection portions of the occupancy model using `lme4` syntax (Bates et al. 2015). For multispecies models, the multispecies detection-nondetection data `y` is now a three-dimensional array with dimensions corresponding to species, sites, and replicates. This is how the data are provided in the `hbef2015` object, so we don't need to do any additional prep.

```
occ.ms.formula <- ~ Elevation + Elevation.2
det.ms.formula <- ~ day + tod + day.2
str(hbef2015)
```

```
List of 4
 $ y       : num [1:12, 1:373, 1:3] 0 0 0 1 0 1 1 0 0 0 ...
  ..- attr(*, "dimnames")=List of 3
  .. ..$ : chr [1:12] "AMRE" "BAWW" "BHVI" "BLBW" ...
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "1" "2" "3"
 $ occ.covs: num [1:373, 1:2] -0.889 -0.765 -0.413 -0.14 -0.13 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "Elevation" "Elevation.2"
```

```
 $ det.covs:List of 3
  ..$ day  : num [1:373, 1:3] -1.62 -1.62 -1.62 -1.62 -1.62 ...
  ..$ tod  : num [1:373, 1:3] -1.565 -1.378 -1.084 -0.79 -0.549 ...
  ..$ day.2: num [1:373, 1:3] 2.61 2.61 2.61 2.61 2.61 ...
 $ coords  : num [1:373, 1:2] 280000 280000 280000 280001 280000 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:2] "Easting" "Northing"
```

Next we specify the starting values in `starting`. For multispecies occupancy models, we supply starting values for community-level and species-level parameters. In `msPGOcc`, we will supply starting values for the following parameters: `alpha.comm` (community level detection coefficients), `beta.comm` (community level occurrence coefficients), `alpha` (species level detection coefficients), `beta` (species level occurrence coefficients), `tau.sq.beta` (community level occurrence variance parameters), `tau.sq.alpha` (community level detection variance parameters, `z` (latent occurrence values for all species). These are all specified in a single list. Starting values for community level parameters are vectors of length corresponding to the number of community-level detection or occurrence parameters in the model (including the intercepts), while starting values for species level parameters are matrices with the number of rows indicating the number of species, and each column corresponding to a different regression parameter. The starting values for the latent occurrence matrix are specified as a matrix with N rows corresponding to the number of species and J columns corresponding to the number of sites.

```
N <- dim(hbef2015$y)[1]
# Number of community level detection parameters (including intercept)
p.det <- 4
# Number of community level occurrence parameters (including intercept)
p.occ <- 3
ms.starting <- list(alpha.comm = rep(0, p.det),
                    beta.comm = rep(0, p.occ),
                    beta = matrix(0, N, p.occ),
                    alpha = matrix(0, N, p.det),
                    tau.sq.beta = rep(1, p.occ),
                    tau.sq.alpha = rep(1, p.det),
                    z = apply(hbef2015$y, c(1, 2), max, na.rm = TRUE))
```

In multispecies models, we specify priors on the community-level coefficients rather than the species-level effects. For nonspatial models, these priors are specified with the following tags: `beta.comm.normal` (normal prior on the community level occurrence mean effects), `alpha.comm.normal` (normal prior on the community level detection mean effects), `tau.sq.beta.ig` (inverse-Gamma prior on the community level occurrence variance parameters), `tau.sq.alpha.ig` (inverse-Gamma prior on the community level detection variance parameters). Each tag consists of a list with elements corresponding to the mean and variance for normal priors and scale and shape for inverse-Gamma priors.

Below we specify normal priors to be relatively non-informative on the probability scale with a mean of 0 and variance of 2.72, and specify vague inverse gamma priors on the community level variance parameters setting both the shape and scale parameters to 0.1.

```
ms.priors <- list(beta.comm.normal = list(mean = rep(0, p.occ),
                                          var = rep(2.72, p.occ)),
                  alpha.comm.normal = list(mean = rep(0, p.det),
                                           var = rep(2.72, p.det)),
                  tau.sq.beta.ig = list(a = rep(0.1, p.occ),
                                        b = rep(0.1, p.occ)),
                  tau.sq.alpha.ig = list(a = rep(0.1, p.det),
                                         b = rep(0.1, p.det)))
```

All that's left to do is specify the number of threads to use (`n.omp.threads`), the number of MCMC samples (`n.samples`), the amount of samples to discard as burn-in (`n.burn`), the the thinning rate (`n.thin`), and arguments to control the display of sampler progress (`verbose`, `n.report`).

```
out.ms <- msPGOcc(occ.formula = occ.ms.formula,
                  det.formula = det.ms.formula,
                  data = hbef2015,
                  starting = ms.starting,
                  n.samples = 20000,
                  priors = ms.priors,
                  n.omp.threads = 1,
                  verbose = TRUE,
                  n.report = 5000,
                  n.burn = 10000,
                  n.thin = 10)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Multi-species Occupancy Model with Polya-Gamma latent
variable fit with 373 sites and 12 species.

Number of MCMC samples: 20000
Burn-in: 10000
Thinning Rate: 10
Total Posterior Samples: 1000


Source compiled with OpenMP support and model fit using 1 thread(s).

Sampling ...
Sampled: 5000 of 20000, 25.00%
--------------------------------------------------
Sampled: 10000 of 20000, 50.00%
--------------------------------------------------
Sampled: 15000 of 20000, 75.00%
--------------------------------------------------
Sampled: 20000 of 20000, 100.00%
```

```
out.ms$run.time
```

```
   user  system elapsed
148.857   0.121 148.993
```

We see `msPGOcc` took less than 3 minutes to run the multispecies occupancy model with 373 sites and 12 species for a total of 20,000 iterations. The resulting object `out.ms` is a list of class `msPGOcc` consisting primarily of posterior samples of all community and species level parameters, as well as some additional objects that are used for summaries, prediction, and model fit evaluation. We can display a nice summary of these results using the `summary` function. For multispecies objects, when using summary we need to specify the level of parameters we want to summarize. We do this using the argument `level`, which takes values `community`, `species`, or `both` to print results for community-level parameters, species-level parameters, or all parameters. `level` is the second argument, so we can also avoid typing it out explicitly every time we want to call it

```
summary(out.ms, level = 'both')
```

```
Call:
msPGOcc(occ.formula = occ.ms.formula, det.formula = det.ms.formula,
    data = hbef2015, starting = ms.starting, priors = ms.priors,
    n.samples = 20000, n.omp.threads = 1, verbose = TRUE, n.report = 5000,
    n.burn = 10000, n.thin = 10)

Chain Information:
Total samples: 20000
Burn-in: 10000
Thin: 10
Total Posterior Samples: 1000

----------------------------------------
    Community Level
----------------------------------------
Occurrence Means:
              2.5%     25%     50%    75%  97.5%
(Intercept) -1.2354 -0.2112  0.3358 0.8470 1.9803
Elevation   -1.0462 -0.3172  0.0083 0.3232 1.0229
Elevation.2 -0.9383 -0.4539 -0.2326 0.0001 0.6298

Occurrence Variances:
              2.5%    25%     50%     75%   97.5%
(Intercept) 4.4171 7.5795 10.1751 13.9330 27.5018
Elevation   1.0386 1.9336  2.7169  3.8611  7.7105
Elevation.2 0.4093 0.8353  1.2238  1.8974  4.7904

Detection Means:
              2.5%     25%     50%     75%  97.5%
(Intercept) -1.3665 -0.7489 -0.4767 -0.2015 0.3583
day         -0.3050 -0.1676 -0.1138 -0.0560 0.0587
tod         -0.2833 -0.1628 -0.1122 -0.0590 0.0382
day.2       -0.2586 -0.1505 -0.1016 -0.0495 0.0394

Detection Variances:
              2.5%    25%    50%    75%  97.5%
(Intercept) 0.7952 1.3631 1.8919 2.7090 5.6737
day         0.0232 0.0470 0.0674 0.0984 0.2065
tod         0.0165 0.0310 0.0434 0.0627 0.1440
day.2       0.0167 0.0295 0.0413 0.0604 0.1262

----------------------------------------
    Species Level
----------------------------------------
Occurrence:
                     2.5%     25%     50%     75%   97.5%
(Intercept)-AMRE -3.6928 -3.0415 -2.7165 -2.3406 -1.5214
(Intercept)-BAWW -1.1537 -0.0083  0.9718  2.0335  4.4319
(Intercept)-BHVI -0.7197 -0.1458  0.2257  0.8672  2.9177
(Intercept)-BLBW  2.5144  3.0796  3.4475  4.0287  6.2277
(Intercept)-BLPW -5.6779 -4.9048 -4.5014 -4.1389 -3.5390
(Intercept)-BTBW  3.5535  4.1893  4.5990  5.0416  6.0896
```

```
(Intercept)-BTNW  2.1671  2.7069  3.0553  3.7092  6.8361
(Intercept)-CAWA -1.9246 -1.4331 -1.1843 -0.8887 -0.0927
(Intercept)-MAWA -2.3659 -2.0684 -1.9050 -1.7405 -1.4787
(Intercept)-NAWA -5.8052 -4.1403 -3.4426 -2.7402 -1.4063
(Intercept)-OVEN  2.0833  2.4482  2.6599  2.9214  3.5256
(Intercept)-REVI  1.9285  2.2829  2.4914  2.7399  3.3267
Elevation-AMRE   -3.3397 -2.2143 -1.7046 -1.2531 -0.4748
Elevation-BAWW   -1.2156 -0.4737 -0.1647  0.1831  1.4104
Elevation-BHVI   -1.3449 -0.4738 -0.1242  0.2712  1.3235
Elevation-BLBW   -3.5268 -1.2043 -0.9310 -0.7426 -0.4441
Elevation-BLPW    1.5856  2.0066  2.3234  2.6860  3.7688
Elevation-BTBW   -1.3724 -1.1111 -0.9785 -0.8488 -0.6118
Elevation-BTNW   -1.5704 -0.1292  0.6136  1.5121  2.8081
Elevation-CAWA    0.7695  1.2779  1.5943  1.9227  2.6780
Elevation-MAWA    1.5088  1.7676  1.9371  2.0941  2.4584
Elevation-NAWA   -0.0418  0.4558  0.8160  1.1760  2.3462
Elevation-OVEN   -1.7187 -1.3780 -1.2298 -1.0932 -0.8785
Elevation-REVI   -3.4053 -2.3035 -1.9891 -1.7108 -1.3267
Elevation.2-AMRE -2.2251 -1.4104 -1.0827 -0.8009 -0.2924
Elevation.2-BAWW -3.2456 -2.0371 -1.5288 -1.1089 -0.5177
Elevation.2-BHVI -0.1150  0.6054  1.1907  1.8688  3.4375
Elevation.2-BLBW -1.1498 -0.8239 -0.6908 -0.5510  0.0579
Elevation.2-BLPW  0.0438  0.5665  0.7915  1.0031  1.3472
Elevation.2-BTBW -1.7215 -1.3785 -1.2346 -1.1043 -0.8748
Elevation.2-BTNW -1.0031 -0.2781  0.2180  0.7765  1.7738
Elevation.2-CAWA -2.0276 -1.4057 -1.1343 -0.8861 -0.4421
Elevation.2-MAWA  0.1563  0.4242  0.5560  0.6915  0.9135
Elevation.2-NAWA  0.3996  0.8108  1.1106  1.4482  2.2684
Elevation.2-OVEN -1.1304 -0.8907 -0.7715 -0.6772 -0.4212
Elevation.2-REVI -0.7151 -0.4257 -0.2571 -0.0486  0.4990

Detection:
                    2.5%     25%     50%     75%   97.5%
(Intercept)-AMRE -2.1957 -1.3379 -0.9451 -0.5766  0.1024
(Intercept)-BAWW -3.3391 -2.8662 -2.5035 -2.0485 -1.2656
(Intercept)-BHVI -2.4316 -2.1447 -1.9839 -1.8089 -1.3567
(Intercept)-BLBW -0.0110  0.1394  0.2222  0.3003  0.4612
(Intercept)-BLPW -0.2895  0.0266  0.2036  0.3963  0.7662
(Intercept)-BTBW  0.6426  0.7873  0.8627  0.9413  1.1008
(Intercept)-BTNW  0.1451  0.2703  0.3472  0.4255  0.5800
(Intercept)-CAWA -2.1182 -1.5120 -1.2324 -0.9459 -0.4517
(Intercept)-MAWA -0.0662  0.1760  0.3027  0.4362  0.6760
(Intercept)-NAWA -4.0654 -3.3058 -2.8123 -2.2947 -1.1002
(Intercept)-OVEN  0.3706  0.5252  0.6061  0.6891  0.8597
(Intercept)-REVI  0.4420  0.5886  0.6713  0.7410  0.8944
day-AMRE         -0.8445 -0.4689 -0.3186 -0.1748  0.0948
day-BAWW         -0.4804 -0.2617 -0.1618 -0.0587  0.1200
day-BHVI          0.0335  0.1678  0.2453  0.3187  0.4696
day-BLBW         -0.2378 -0.1603 -0.1195 -0.0757 -0.0025
day-BLPW         -0.4904 -0.2701 -0.1803 -0.0761  0.1079
day-BTBW         -0.0636  0.0230  0.0663  0.1101  0.1914
day-BTNW         -0.0165  0.0537  0.0933  0.1299  0.2076
day-CAWA         -0.5721 -0.3795 -0.2696 -0.1780  0.0075
day-MAWA         -0.4494 -0.3182 -0.2490 -0.1748 -0.0484
```

```
day-NAWA         -0.7338 -0.4158 -0.2620 -0.1139  0.1105
day-OVEN         -0.1561 -0.0773 -0.0295  0.0142  0.0957
day-REVI         -0.3069 -0.2175 -0.1731 -0.1260 -0.0477
tod-AMRE         -0.7388 -0.3972 -0.2712 -0.1501  0.1035
tod-BAWW         -0.4074 -0.2107 -0.1108 -0.0078  0.1797
tod-BHVI         -0.1812 -0.0605  0.0110  0.0744  0.2095
tod-BLBW         -0.1984 -0.1163 -0.0743 -0.0298  0.0521
tod-BLPW         -0.5431 -0.3468 -0.2524 -0.1579  0.0071
tod-BTBW         -0.1912 -0.1003 -0.0520 -0.0056  0.0781
tod-BTNW         -0.0758  0.0041  0.0444  0.0821  0.1591
tod-CAWA         -0.4856 -0.2689 -0.1618 -0.0588  0.1230
tod-MAWA         -0.4425 -0.2975 -0.2232 -0.1542 -0.0214
tod-NAWA         -0.5086 -0.2504 -0.1243  0.0053  0.2263
tod-OVEN         -0.2525 -0.1730 -0.1277 -0.0799  0.0062
tod-REVI         -0.1903 -0.0985 -0.0572 -0.0100  0.0777
day.2-AMRE       -0.6082 -0.3204 -0.2046 -0.0842  0.1284
day.2-BAWW       -0.5312 -0.3168 -0.2044 -0.1154  0.0793
day.2-BHVI       -0.3731 -0.2138 -0.1412 -0.0704  0.0889
day.2-BLBW       -0.1134 -0.0254  0.0236  0.0708  0.1677
day.2-BLPW       -0.4563 -0.2105 -0.1049 -0.0005  0.2204
day.2-BTBW       -0.1162 -0.0197  0.0289  0.0776  0.1763
day.2-BTNW       -0.1810 -0.0970 -0.0509 -0.0045  0.0733
day.2-CAWA       -0.4849 -0.2584 -0.1622 -0.0558  0.1475
day.2-MAWA       -0.2704 -0.1145 -0.0339  0.0474  0.1973
day.2-NAWA       -0.6047 -0.2962 -0.1533 -0.0309  0.2105
day.2-OVEN       -0.2875 -0.1975 -0.1501 -0.1015 -0.0021
day.2-REVI       -0.1945 -0.1019 -0.0488 -0.0002  0.0949
```

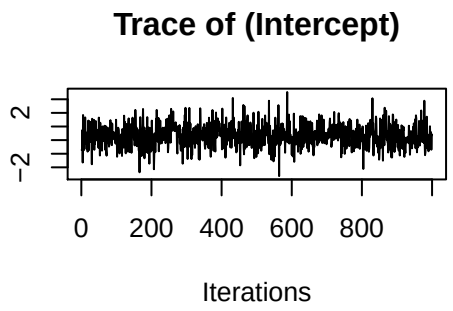
```
# Or
# summary(out.ms, 'both')
```

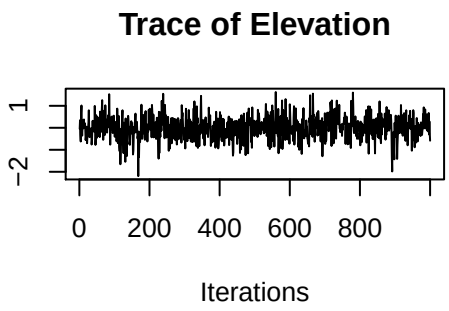
Looking at the community level variance parameters, we see large variability in the average occurrence (the intercept) for the twelve species, as well as substantial variability in the effect of elevation across the community. There appears to be less variability across species in the detection portion of the model. We can look directly at the species-specific effects to confirm this.

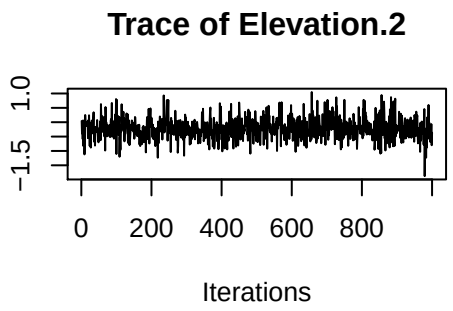
4.3 Convergence diagnostics

The resulting posterior samples in the `msPGOcc` object are `coda::mcmc` samples, and so convergence diagnostics can proceed as we saw with single species models.

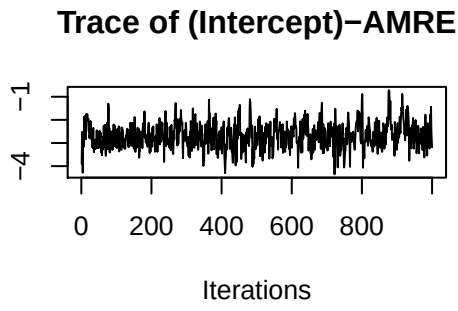
```
plot(out.ms$beta.comm.samples, density = FALSE)
```

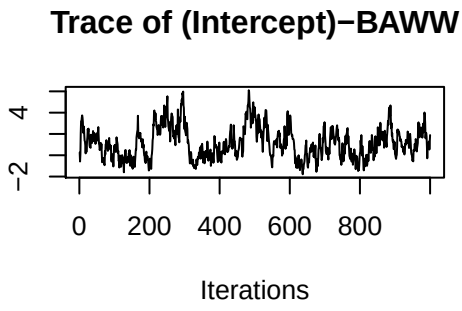


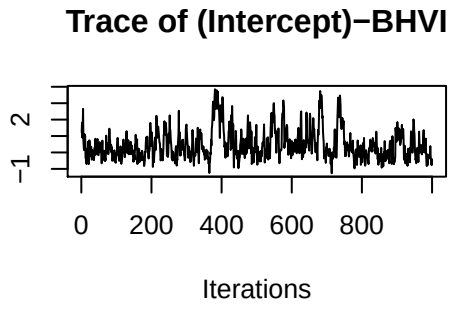


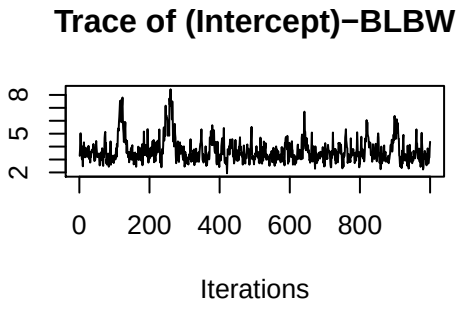



```
# Look at the first few species-specific occurrence intercepts
plot(out.ms$beta.samples[, 1:4], density = FALSE)
```









Looking at the species-specific intercepts, we would probably want to run the model a bit longer. Formal assessments of convergence using the Gelman-Rubin diagnostic can be accomplished following the steps shown

for PGOcc using the gelman.diag function.

4.4 Posterior predictive checks

We can use the ppcOcc function to perform a posterior predictive check, and summarize the check with a Bayesian p-value using the summary function. The summary function again requires the level argument to specify if you want an overall Bayesian p-value for the entire community (level = 'community'), each individual species (level = 'species'), or both (level = 'both').

```
ppc.ms.out <- ppcOcc(out.ms, 'chi-square', group = 1)
```

```
[1] "Currently on species 1 out of 12"
[1] "Currently on species 2 out of 12"
[1] "Currently on species 3 out of 12"
[1] "Currently on species 4 out of 12"
[1] "Currently on species 5 out of 12"
[1] "Currently on species 6 out of 12"
[1] "Currently on species 7 out of 12"
[1] "Currently on species 8 out of 12"
[1] "Currently on species 9 out of 12"
[1] "Currently on species 10 out of 12"
[1] "Currently on species 11 out of 12"
[1] "Currently on species 12 out of 12"
```

```
summary(ppc.ms.out, level = 'both')
```

```
Call:
ppcOcc(object = out.ms, fit.stat = "chi-square", group = 1)

Chain Information:
Total samples: 20000
Burn-in: 10000
Thin: 10
Total Posterior Samples: 1000

----------------------------------------
    Community Level
----------------------------------------
Bayesian p-value:  0.3029167

----------------------------------------
    Species Level
----------------------------------------
AMRE Bayesian p-value: 0.305
BAWW Bayesian p-value: 0.558
BHVI Bayesian p-value: 0.415
BLBW Bayesian p-value: 0.104
BLPW Bayesian p-value: 0.323
BTBW Bayesian p-value: 0.116
BTNW Bayesian p-value: 0.113
CAWA Bayesian p-value: 0.457
MAWA Bayesian p-value: 0.387
NAWA Bayesian p-value: 0.683
OVEN Bayesian p-value: 0.161
REVI Bayesian p-value: 0.013
```

```
Fit statistic:  chi-square
```

The Bayesian p-value for the overall community suggests an adequate model fit, but looking closer at each individual species reveals certain species the model may not be fitting well for all species. We should explore this further in a complete analysis (and also of course run the model longer to ensure convergence, as this is likely contributing to many of the extreme values).

4.5 Model selection using WAIC

We can compute the WAIC for comparison with alternative models using the `waicOCC` function.

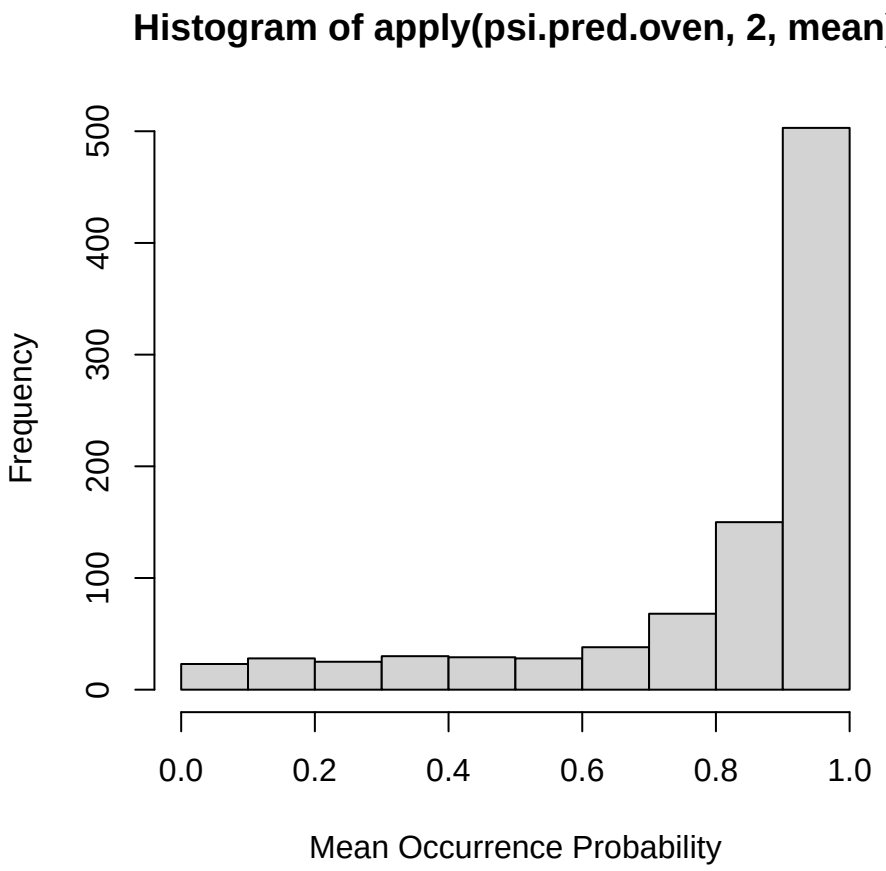
```
waicOcc(out.ms)
```

```
       elpd          pD        WAIC
-4638.61419    70.97646  9419.18130
```

4.6 Prediction

Out-of-sample prediction with `msPGOcc` objects is exactly analogous to what we saw with `PGOcc`. We can use the `predict` function along with a data frame of covariates at new locations. We predict across the entire HBEF for all twelve species using the elevation data stored in `hbefElev`. Instead of predicting for every 30 x 30 m cell across the HBEF as we did with the single species model, below we predict values at every 50th cell. We predict only for OVEN for comparison to predictions using single species occupancy models.

```
X.0.ms <- cbind(1, hbefElev$val, hbefElev$val^2)
X.0.ms <- X.0.ms[seq(1, nrow(X.0.ms), by = 50), ]
out.ms.pred <- predict(out.ms, X.0.ms)
psi.pred.oven <- out.ms.pred$psi.0.samples[, which(sp.names == 'OVEN'), ]
hist(apply(psi.pred.oven, 2, mean), xlab = 'Mean Occurrence Probability')
```



The histogram supports our results from the single species model that indicated OVEN was common throughout most of the HBEF.

5 Multispecies spatial occupancy models

5.1 Basic model description

Residual spatial autocorrelation may perhaps be more prominent in multispecies occupancy models compared to single species models, as a single set of covariates is used to explain occurrence probability across a region of interest for all species. Given the large variety individual species show in habitat requirements, this may result in important drivers of occurrence probability not being included for certain species, resulting in many species having high residual spatial autocorrelation. We extend the previous multispecies occupancy model to incorporate a distinct spatial Gaussian Process (GP) for each species that accounts for unexplained spatial variation in each individual species occurrence across a spatial region. Occurrence probability for species i at site j, $\psi_{i,j}$, now takes the form

$$\text{logit}(\psi_{i,j}) = \boldsymbol{x}_j'\boldsymbol{\beta}_i + \text{w}_{i,j}, \tag{10}$$

where the species-specific regression coefficients $\boldsymbol{\beta}_i$ follow the community level distribution in Equation (7), and $\text{w}_{i,j}$ is a realization from a zero-mean spatial GP, i.e.,

$$\mathbf{w}_i \sim \text{Normal}(\mathbf{0}, \mathbf{\Sigma}_i(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{\theta}_i)). \tag{11}$$

We define $\mathbf{\Sigma}_i(\boldsymbol{s}, \boldsymbol{s}', \boldsymbol{\theta}_i)$ as a $J \times J$ covariance matrix that is a function of the distances between any pair of site coordinates $\boldsymbol{s}$ and $\boldsymbol{s}'$ and a set of parameters ($\boldsymbol{\theta}_i$) that govern the spatial process. The vector $\boldsymbol{\theta}_i$ is equal to $\boldsymbol{\theta}_i = \{\sigma_i^2, \phi_i, \nu_i\}$, where σ_i^2 is a spatial variance parameter for species i, ϕ_i is a spatial decay parameter for species i, and ν_i is a spatial smoothness parameter for species i. ν_i is only specified when using a Matern correlation function.

The detection portion of the multispecies spatial occupancy model remains unchanged from the non-spatial multispecies occupancy model and follows Equations (8) and (9). We fit the model again using Polya-Gamma data augmentation to enable an efficient Gibbs sampler (see MCMC sampler vignette for details). Similar to our discussion on the single species spatial occupancy model, we also allow for specification of the spatial process using an NNGP instead fo a full GP. This leads to even larger computational gains over the full GP given that a separate covariance matrix is specified for each species in the model. See Datta et al. (2016), Finley, Datta, and Banerjee (2020), and the MCMC sampler vignette for additional details on NNGPs and their implementation in multispecies spatial occupancy models.

5.2 Fitting multispecies spatial occupancy models with `spMsPGOcc`

The function `spMsPGOcc` fits spatially explicit multispecies occupancy models. Similar to single species models using `spPGOcc`, models can be fit using either a full Gaussian Process (GP) or a Nearest Neighbor Gaussian Process (NNGP). `spMsPGOcc` fits a separate spatial process for each species. The syntax for `spMsPGOcc` is analogous to the syntax for single species spatially-explicit models using `spPGOcc`.

```
spMsPGOcc(occ.formula, det.formula, data, starting, n.batch,
          batch.length, accept.rate = 0.43, priors,
          cov.model = "exponential", tuning, n.omp.threads = 1,
          verbose = TRUE, NNGP = TRUE, n.neighbors = 15,
          search.type = "cb", n.report = 100,
          n.burn = round(.10 * n.batch * batch.length), n.thin = 1, ...)
```

We will again display the model using the HBEF foliage-gleaning bird data set, with the same predictors in our occurrence and detection models

```
occ.ms.sp.formula <- ~ Elevation + Elevation.2
det.ms.sp.formula <- ~ day + tod + day.2
```

Our starting values in the `starting` argument will look analagous to what we specified for the nonspatial multispecies occupancy model using `msPGOcc`, but we will also include additional starting values for the parameters controlling the spatial processes: `sigma.sq` is the species-specific spatial variance parameter, `phi` is the species specific spatial decay parameter, and `w` is the latent spatial proccess for each species at each site. We will use an exponential covariance model, but when using a Matern covariance model we must also specify starting values for `nu`, the species-specific spatial smoothness parameter. Note that all species-specific spatial parameters are independent of each other. We currently do not leverage any correlation between spatial processes of different species, although this is something we plan to incorporate for future `spOccupancy` development. Starting values for `phi`, `sigma.sq`, and `nu` (if applicable) are specified as vectors with N elements (the number of species being modeled), while the starting values for the latent spatial processes are specified as a matrix with N rows (i.e., species) and J columns (i.e., sites). Here we set the starting value for the spatial variances equal to 2 for all species and set the starting values for the spatial decay parameter to yield an effective range of the average distance between sites across the HBEF.

```
# Number of species
N <- dim(hbef2015$y)[1]
# Distances between sites
dist.hbef <- dist(hbef2015$coords)
```

```
# Number of detection and occupancy regression parameters
p.det <- length(hbef2015$det.covs) + 1
p.occ <- ncol(hbef2015$occ.covs) + 1
# Exponential covariance model
cov.model <- "exponential"
ms.starting <- list(alpha.comm = rep(0, p.det),
                    beta.comm = rep(0, p.occ),
                    beta = matrix(0, N, p.occ),
                    alpha = matrix(0, N, p.det),
                    tau.sq.beta = rep(1, p.occ),
                    tau.sq.alpha = rep(1, p.det),
                    z = apply(hbef2015$y, c(1, 2), max, na.rm = TRUE),
                    sigma.sq = rep(2, N),
                    phi = rep(3 / mean(dist.hbef), N),
                    w = matrix(0, N, dim(hbef2015$y)[2]))
```

We next specify the priors in the `priors` argument. The priors are the same as those we specified for the non-spatial multispecies model, with the addition of priors for the parameters controlling the species-specific spatial processes. We assume independent priors for all spatial parameters across the different species. For each species, we assign an inverse gamma prior for the spatial varaince parameter `sigma.sq` (tag is `sigma.sq.ig`) and uniform priors for the spatial decay parameter `phi` and smoothness parameter `nu` (if `cov.model = 'matern'`), with the associated tags `phi.unif` and `nu.unif`. All priors are specified as lists with two elements. For the inverse-Gamma prior, the first element is a length N vector of shape parameters for each species, and the second element is a length N vector of scale parameters for each species. For the uniform priors, the first element is a length N vector of the lower bounds for each species, and the second element is a length N vector of upper bounds for each species. For the inverse-Gamma prior on the spatial variances, here we set the shape parameter to 2 and the scale parameter equal to 2. For a more formal analysis, we would likely want to do some exploratory data analysis to obtain a better guess for the spatial variance for each species, and then replace the scale parameter with this estimated guess for each species. For the spatial decay parameter, we determine the bounds of the uniform distribution by computing the smallest distance between sites and the largest distance between sites. We then set the lower bound of the uniform to `3/max` and the upper bound to `3/min`, where `min` and `max` correspond to the predetermined distances between sites.

```
# Minimum value is 0, so need to grab second element.
min.dist <- sort(unique(dist.hbef))[2]
max.dist <- max(dist.hbef)
ms.priors <- list(beta.comm.normal = list(mean = rep(0, p.occ),
                                          var = rep(2.72, p.occ)),
                  alpha.comm.normal = list(mean = rep(0, p.det),
                                           var = rep(2.72, p.det)),
                  tau.sq.beta.ig = list(a = rep(0.1, p.occ),
                                        b = rep(0.1, p.occ)),
                  tau.sq.alpha.ig = list(a = rep(0.1, p.det),
                                         b = rep(0.1, p.det)),
                  sigma.sq.ig = list(a = rep(2, N),
                                     b = rep(2, N)),
                  phi.unif = list(a = rep(3/max.dist, N),
                                  b = rep(3/min.dist, N)))
```

We next set the parameters controlling the Adaptive MCMC algorithm (see `spPGOcc` section for details). Notice our specification of the starting tuning values is exactly the same as for `spPGOcc`. We assume the same initial tuning value for all species. However, the adaptive algorithm will allow for species specific tuning parameters, so these will be adjusted in the algorithm as needed (and reported to the R console if `verbose = TRUE`).

```
batch.length <- 25
n.batch <- 400
n.burn <- 2000
n.thin <- 8
ms.tuning <- list(phi = 0.5)
n.omp.threads <- 1
# Values for reporting
verbose <- TRUE
n.report <- 50
```

Spatially explicit multispecies occupancy models are currently the most computationally intensive models fit by `spOccupancy`. Even for modest sized data sets, we encourage the use of NNGPs instead of full GPs when fitting spatially-explicit models to ease the computational burden of fitting these models. We fit the model with an NNGP below using 5 neighbors and summarize it using the `summary` function, where we specify that we want to summarize both species and community level parameters.

```
out.sp.ms <- spMsPGOcc(occ.formula = occ.ms.sp.formula,
                       det.formula = det.ms.sp.formula,
                       data = hbef2015,
                       starting = ms.starting,
                       n.batch = n.batch,
                       batch.length = batch.length,
                       accept.rate = 0.43,
                       priors = ms.priors,
                       cov.model = cov.model,
                       tuning = ms.tuning,
                       n.omp.threads = n.omp.threads,
                       verbose = TRUE,
                       NNGP = TRUE,
                       n.neighbors = 5,
                       n.report = n.report,
                       n.burn = n.burn,
                       n.thin = n.thin)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Building the neighbor list
----------------------------------------
----------------------------------------
Building the neighbors of neighbors list
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
NNGP Multi-species Occupancy Model with Polya-Gamma latent
variable fit with 373 sites and 12 species.

Number of MCMC samples 10000 (400 batches of length 25)
Burn-in: 2000
Thinning Rate: 8
Total Posterior Samples: 1000

Using the exponential spatial correlation model.
```

```
Using 5 nearest neighbors.

Source compiled with OpenMP support and model fit using 1 thread(s).

Adaptive Metropolis with target acceptance rate: 43.0
Sampling ...
Batch: 50 of 400, 12.50%
    parameter   acceptance  tuning
    phi[0]      40.0        0.41348
    phi[1]      72.0        0.49502
    phi[2]      48.0        0.39727
    phi[3]      12.0        0.53625
    phi[4]      48.0        0.37413
    phi[5]      28.0        0.41348
    phi[6]      88.0        0.50503
    phi[7]      64.0        0.53625
    phi[8]      40.0        0.38169
    phi[9]      36.0        0.42183
    phi[10]     52.0        0.38940
    phi[11]     32.0        0.38169
Batch: 100 of 400, 25.00%
    parameter   acceptance  tuning
    phi[0]      32.0        0.31250
    phi[1]      44.0        0.54709
    phi[2]      24.0        0.39727
    phi[3]      56.0        0.58092
    phi[4]      36.0        0.38940
    phi[5]      72.0        0.52564
    phi[6]      64.0        0.68171
    phi[7]      32.0        0.49502
    phi[8]      40.0        0.31250
    phi[9]      80.0        0.49502
    phi[10]     36.0        0.33853
    phi[11]     44.0        0.33853
Batch: 150 of 400, 37.50%
    parameter   acceptance  tuning
    phi[0]      40.0        0.27716
    phi[1]      32.0        0.72387
    phi[2]      72.0        0.37413
    phi[3]      44.0        0.75341
    phi[4]      44.0        0.32525
    phi[5]      16.0        0.50503
    phi[6]      8.0     0.70953
    phi[7]      32.0        0.37413
    phi[8]      56.0        0.28847
    phi[9]      68.0        0.55814
    phi[10]     24.0        0.28847
    phi[11]     32.0        0.30025
Batch: 200 of 400, 50.00%
    parameter   acceptance  tuning
    phi[0]      80.0        0.35946
    phi[1]      16.0        0.81616
    phi[2]      84.0        0.58092
```

```
    phi[3]      16.0        0.64201
    phi[4]      52.0        0.34537
    phi[5]      80.0        0.66821
    phi[6]      16.0        0.83265
    phi[7]      96.0        0.52564
    phi[8]      20.0        0.30631
    phi[9]      40.0        0.48522
    phi[10]     48.0        0.33183
    phi[11]     52.0        0.29430
Batch: 250 of 400, 62.50%
    parameter   acceptance  tuning
    phi[0]      36.0        0.35234
    phi[1]      36.0        0.51523
    phi[2]      24.0        0.69548
    phi[3]      32.0        0.61684
    phi[4]      32.0        0.38169
    phi[5]      72.0        0.81616
    phi[6]      24.0        0.70953
    phi[7]      36.0        0.49502
    phi[8]      56.0        0.31881
    phi[9]      80.0        0.56941
    phi[10]     56.0        0.35946
    phi[11]     28.0        0.28276
Batch: 300 of 400, 75.00%
    parameter   acceptance  tuning
    phi[0]      68.0        0.40529
    phi[1]      24.0        0.61684
    phi[2]      4.0     0.81616
    phi[3]      80.0        0.43905
    phi[4]      40.0        0.33853
    phi[5]      60.0        0.90199
    phi[6]      12.0        0.59265
    phi[7]      40.0        0.43035
    phi[8]      48.0        0.38940
    phi[9]      12.0        0.56941
    phi[10]     52.0        0.37413
    phi[11]     36.0        0.28276
Batch: 350 of 400, 87.50%
    parameter   acceptance  tuning
    phi[0]      64.0        0.48522
    phi[1]      64.0        0.86663
    phi[2]      76.0        0.84947
    phi[3]      32.0        0.49502
    phi[4]      60.0        0.41348
    phi[5]      56.0        0.88413
    phi[6]      36.0        0.48522
    phi[7]      44.0        0.46620
    phi[8]      32.0        0.34537
    phi[9]      72.0        0.62930
    phi[10]     28.0        0.31250
    phi[11]     28.0        0.30631
Batch: 400 of 400, 100.00%
```

```
summary(out.sp.ms, level = 'both')
```

```
Call:
spMsPGOcc(occ.formula = occ.ms.sp.formula, det.formula = det.ms.sp.formula,
    data = hbef2015, starting = ms.starting, priors = ms.priors,
    tuning = ms.tuning, cov.model = cov.model, NNGP = TRUE, n.neighbors = 5,
    n.batch = n.batch, batch.length = batch.length, accept.rate = 0.43,
    n.omp.threads = n.omp.threads, verbose = TRUE, n.report = n.report,
    n.burn = n.burn, n.thin = n.thin)

Chain Information:
Total samples: 10000
Burn-in: 2000
Thin: 8
Total Posterior Samples: 1000

----------------------------------------
    Community Level
----------------------------------------
Occurrence Means:
               2.5%     25%     50%     75%  97.5%
(Intercept) -1.8210 -0.2965  0.3976  1.1080 2.6081
Elevation   -1.3558 -0.3681  0.1332  0.6126 1.6894
Elevation.2 -1.4124 -0.7593 -0.4122 -0.0436 0.7664

Occurrence Variances:
              2.5%     25%     50%     75%   97.5%
(Intercept) 9.5514 17.1106 24.0927 33.7462 70.7688
Elevation   2.5331  4.5579  6.4931  9.2636 21.8728
Elevation.2 0.6843  1.6044  2.5164  4.0328 10.0179

Detection Means:
               2.5%     25%     50%     75%  97.5%
(Intercept) -1.1939 -0.6986 -0.4477 -0.1762 0.3492
day         -0.3164 -0.1718 -0.1157 -0.0572 0.0648
tod         -0.2877 -0.1694 -0.1181 -0.0665 0.0301
day.2       -0.2656 -0.1544 -0.1005 -0.0500 0.0623

Detection Variances:
              2.5%    25%    50%    75%  97.5%
(Intercept) 0.7874 1.3431 1.8531 2.5650 6.0726
day         0.0251 0.0471 0.0660 0.0971 0.2102
tod         0.0179 0.0315 0.0429 0.0657 0.1291
day.2       0.0165 0.0298 0.0429 0.0599 0.1399

----------------------------------------
    Species Level
----------------------------------------
Occurrence:
                     2.5%     25%     50%     75%   97.5%
(Intercept)-AMRE -9.4882 -4.4886 -3.6452 -3.0511 -1.9317
(Intercept)-BAWW -1.2747  0.5969  1.9123  3.7585  7.8112
(Intercept)-BHVI -1.5456 -0.1903  0.7652  8.3986 13.5169
(Intercept)-BLBW  2.9717  3.8087  4.4498  5.2944  7.3095
```

```
(Intercept)-BLPW -9.6591 -7.0805 -6.1446 -5.3435 -4.2531
(Intercept)-BTBW  4.2936  5.2090  5.8537  6.5900  8.5969
(Intercept)-BTNW  2.8246  3.6588  4.3065  5.2480  8.7364
(Intercept)-CAWA -2.9269 -2.0263 -1.6681 -1.2600 -0.5400
(Intercept)-MAWA -7.0007 -4.8584 -4.0161 -3.3282 -2.2939
(Intercept)-NAWA -7.4514 -5.5198 -4.6053 -3.8053 -2.1384
(Intercept)-OVEN  2.2505  3.0262  3.4360  4.1251  6.6101
(Intercept)-REVI  2.4923  4.1097  4.9664  6.1233  8.6708
Elevation-AMRE   -4.0159 -2.5642 -1.8706 -1.2637 -0.1895
Elevation-BAWW   -2.4281 -0.7277 -0.2769  0.1705  1.2013
Elevation-BHVI   -3.8260 -0.9083 -0.2421  0.4502  3.0349
Elevation-BLBW   -3.4278 -1.4256 -1.1046 -0.8549 -0.5194
Elevation-BLPW    1.9282  2.6957  3.2193  4.0184  6.7026
Elevation-BTBW   -1.9765 -1.4683 -1.2488 -1.0579 -0.7184
Elevation-BTNW   -2.5570 -0.1249  0.8028  2.9144  5.3833
Elevation-CAWA    0.8483  1.6369  2.0356  2.4940  3.4413
Elevation-MAWA    2.5295  3.4531  4.0028  4.7367  6.5201
Elevation-NAWA   -0.2770  0.4235  0.8274  1.3689  2.9399
Elevation-OVEN   -3.8438 -2.0085 -1.6703 -1.4280 -1.0448
Elevation-REVI   -6.4439 -3.8481 -3.1389 -2.5999 -1.7491
Elevation.2-AMRE -3.5727 -1.9785 -1.4005 -0.9926 -0.4094
Elevation.2-BAWW -5.6328 -3.0635 -2.3796 -1.6464 -0.8823
Elevation.2-BHVI -1.3376  0.1376  1.0688  2.0622  6.2902
Elevation.2-BLBW -1.5884 -1.1009 -0.9029 -0.6977 -0.0918
Elevation.2-BLPW -0.3910  0.6083  0.9755  1.3125  2.0840
Elevation.2-BTBW -2.4033 -1.8211 -1.5874 -1.3826 -1.0847
Elevation.2-BTNW -1.1241 -0.4115  0.2714  1.3569  2.6272
Elevation.2-CAWA -2.7121 -1.9485 -1.5431 -1.1837 -0.5668
Elevation.2-MAWA  0.4554  1.0775  1.4878  1.9259  2.7730
Elevation.2-NAWA  0.4228  0.8952  1.2616  1.7068  2.8442
Elevation.2-OVEN -1.9312 -1.2878 -1.0885 -0.8939 -0.5424
Elevation.2-REVI -2.0385 -1.2787 -0.8813 -0.4433  0.4276

Detection:
                    2.5%     25%     50%     75%   97.5%
(Intercept)-AMRE -1.8482 -1.1600 -0.7826 -0.4479  0.1795
(Intercept)-BAWW -3.4221 -2.9225 -2.6653 -2.3091 -1.4243
(Intercept)-BHVI -2.5566 -2.2775 -2.0746 -1.8511 -1.2913
(Intercept)-BLBW -0.0158  0.1523  0.2214  0.3054  0.4704
(Intercept)-BLPW -0.2676  0.0856  0.2564  0.4235  0.7524
(Intercept)-BTBW  0.6308  0.7785  0.8546  0.9397  1.0736
(Intercept)-BTNW  0.1522  0.2828  0.3630  0.4427  0.5910
(Intercept)-CAWA -1.8773 -1.4166 -1.1492 -0.8751 -0.4349
(Intercept)-MAWA -0.0355  0.2275  0.3476  0.4750  0.7059
(Intercept)-NAWA -3.7311 -3.0713 -2.5772 -1.9494 -0.7851
(Intercept)-OVEN  0.3830  0.5241  0.6121  0.6935  0.8475
(Intercept)-REVI  0.4754  0.6273  0.6997  0.7819  0.9401
day-AMRE         -0.8035 -0.4619 -0.3240 -0.1726  0.0614
day-BAWW         -0.4679 -0.2593 -0.1638 -0.0640  0.1189
day-BHVI          0.0365  0.1628  0.2352  0.3185  0.4612
day-BLBW         -0.2281 -0.1601 -0.1191 -0.0777 -0.0028
day-BLPW         -0.4903 -0.2770 -0.1897 -0.0832  0.1172
day-BTBW         -0.0568  0.0236  0.0665  0.1106  0.1849
day-BTNW         -0.0264  0.0524  0.0909  0.1309  0.1993
```

```
day-CAWA         -0.6156 -0.3940 -0.2835 -0.1852  0.0188
day-MAWA         -0.4664 -0.3279 -0.2570 -0.1867 -0.0474
day-NAWA         -0.7684 -0.4210 -0.2659 -0.1164  0.1434
day-OVEN         -0.1577 -0.0709 -0.0298  0.0160  0.0908
day-REVI         -0.2974 -0.2069 -0.1641 -0.1190 -0.0422
tod-AMRE         -0.7374 -0.4129 -0.2661 -0.1413  0.0788
tod-BAWW         -0.4371 -0.2186 -0.1172 -0.0163  0.1810
tod-BHVI         -0.2081 -0.0615  0.0079  0.0797  0.2154
tod-BLBW         -0.1975 -0.1144 -0.0720 -0.0306  0.0536
tod-BLPW         -0.5699 -0.3659 -0.2658 -0.1673  0.0094
tod-BTBW         -0.1819 -0.0987 -0.0465 -0.0030  0.0765
tod-BTNW         -0.0758  0.0137  0.0504  0.0941  0.1732
tod-CAWA         -0.4972 -0.2767 -0.1688 -0.0620  0.1478
tod-MAWA         -0.4379 -0.2897 -0.2184 -0.1488 -0.0240
tod-NAWA         -0.5118 -0.2447 -0.1303 -0.0121  0.2129
tod-OVEN         -0.2643 -0.1761 -0.1295 -0.0828 -0.0003
tod-REVI         -0.1817 -0.1005 -0.0488 -0.0028  0.0869
day.2-AMRE       -0.6106 -0.3367 -0.2173 -0.0839  0.1227
day.2-BAWW       -0.5399 -0.3134 -0.2135 -0.1182  0.0800
day.2-BHVI       -0.3689 -0.2221 -0.1445 -0.0625  0.0874
day.2-BLBW       -0.1153 -0.0180  0.0258  0.0774  0.1675
day.2-BLPW       -0.4197 -0.2147 -0.1192 -0.0207  0.1916
day.2-BTBW       -0.1074 -0.0205  0.0328  0.0871  0.1816
day.2-BTNW       -0.1865 -0.0985 -0.0516 -0.0060  0.0825
day.2-CAWA       -0.4662 -0.2480 -0.1534 -0.0588  0.1435
day.2-MAWA       -0.2656 -0.1270 -0.0459  0.0382  0.1878
day.2-NAWA       -0.5917 -0.2958 -0.1530 -0.0302  0.2253
day.2-OVEN       -0.2919 -0.2008 -0.1547 -0.1027 -0.0118
day.2-REVI       -0.1945 -0.0989 -0.0472  0.0018  0.0933

Covariance:
                2.5%    25%     50%     75%   97.5%
sigma.sq-AMRE 0.4922 1.1979  2.1702  5.3039 36.6280
sigma.sq-BAWW 0.3471 0.7837  1.2375  2.0052  6.0820
sigma.sq-BHVI 0.3184 0.7169  1.1762  2.6461 37.0082
sigma.sq-BLBW 0.3189 0.9875  1.7303  2.9429  7.0670
sigma.sq-BLPW 0.4977 1.7839  3.2691  5.4670 12.3537
sigma.sq-BTBW 0.4492 0.9019  1.3557  2.3663  7.6491
sigma.sq-BTNW 0.3979 0.9549  1.8865  3.9607  9.0038
sigma.sq-CAWA 0.4647 0.9888  1.6889  2.9824  9.4544
sigma.sq-MAWA 4.7927 8.3925 12.8266 18.2887 37.2778
sigma.sq-NAWA 0.4053 0.7804  1.1717  2.1731  8.9690
sigma.sq-OVEN 0.6363 1.2375  1.8360  3.1471  9.9464
sigma.sq-REVI 3.1592 8.0627 11.6753 17.6438 40.3710
phi-AMRE      0.0005 0.0013  0.0026  0.0057  0.0258
phi-BAWW      0.0012 0.0053  0.0129  0.0219  0.0290
phi-BHVI      0.0024 0.0057  0.0128  0.0201  0.0288
phi-BLBW      0.0017 0.0050  0.0093  0.0187  0.0284
phi-BLPW      0.0005 0.0012  0.0020  0.0036  0.0183
phi-BTBW      0.0017 0.0088  0.0155  0.0233  0.0295
phi-BTNW      0.0017 0.0045  0.0099  0.0194  0.0286
phi-CAWA      0.0012 0.0034  0.0066  0.0145  0.0296
phi-MAWA      0.0004 0.0009  0.0013  0.0018  0.0029
phi-NAWA      0.0005 0.0029  0.0095  0.0184  0.0285
```

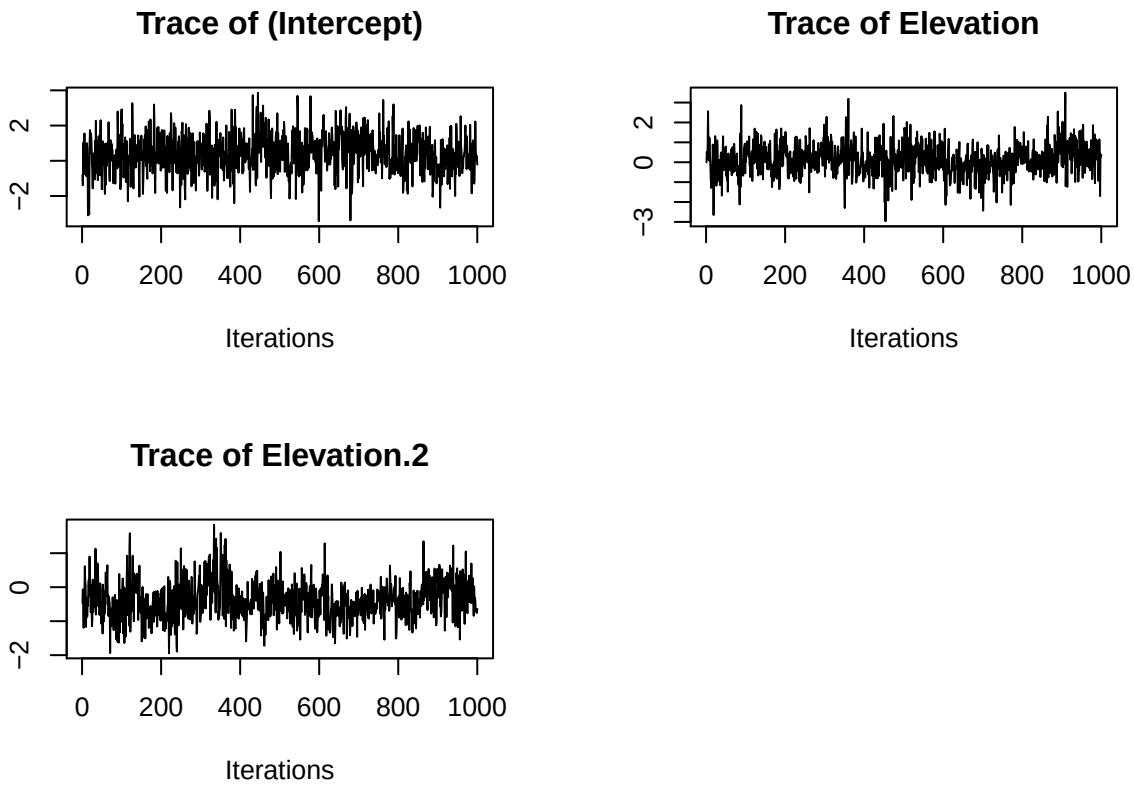
```
phi-OVEN       0.0005 0.0010  0.0017  0.0027  0.0214
phi-REVI       0.0007 0.0016  0.0021  0.0026  0.0039
```

The resulting object `out.sp.ms` is a list of class `spMsPGOcc` consisting primarily of posterior samples of all community and species-level parameters, as well as some additional objects that are used for summaries, predictions, and model fit evaluation.

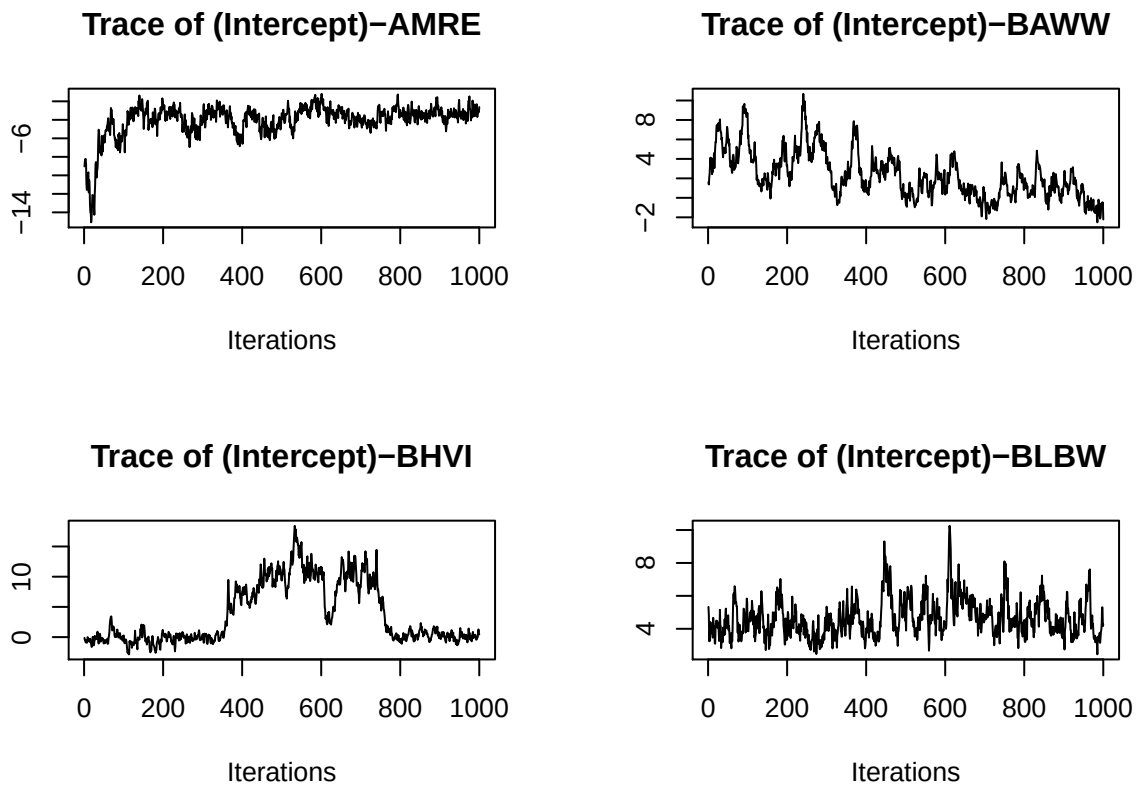
5.3 Convergence diagnostics

Convergence diagnostics proceed as we have seen with all previous `spOccupancy` model objects. Posterior samples are returned as `coda::mcmc` objects, so we can use functions like `plot` and `gelman.diag` to assess convergence.

```
plot(out.sp.ms$beta.comm.samples, density = FALSE)
```




```
# Species-specific effects have yet to converge
plot(out.sp.ms$beta.samples[, 1:4], density = FALSE)
```



5.4 Posterior predictive checks

We perform posterior predictive checks to assess Goodness of Fit using `ppcOcc` just as we have previously seen.

```
ppc.sp.ms.out <- ppcOcc(out.sp.ms, 'freeman-tukey', group = 2)
```

```
[1] "Currently on species 1 out of 12"
[1] "Currently on species 2 out of 12"
[1] "Currently on species 3 out of 12"
[1] "Currently on species 4 out of 12"
[1] "Currently on species 5 out of 12"
[1] "Currently on species 6 out of 12"
[1] "Currently on species 7 out of 12"
[1] "Currently on species 8 out of 12"
[1] "Currently on species 9 out of 12"
[1] "Currently on species 10 out of 12"
[1] "Currently on species 11 out of 12"
[1] "Currently on species 12 out of 12"
```

```
summary(ppc.sp.ms.out, level = 'both')
```

```
Call:
ppcOcc(object = out.sp.ms, fit.stat = "freeman-tukey", group = 2)

Chain Information:
```

```
Total samples: 10000
Burn-in: 2000
Thin: 8
Total Posterior Samples: 1000

----------------------------------------
    Community Level
----------------------------------------
Bayesian p-value:  0.48475

----------------------------------------
    Species Level
----------------------------------------
AMRE Bayesian p-value: 0.371
BAWW Bayesian p-value: 0.428
BHVI Bayesian p-value: 0.513
BLBW Bayesian p-value: 0.503
BLPW Bayesian p-value: 0.577
BTBW Bayesian p-value: 0.496
BTNW Bayesian p-value: 0.516
CAWA Bayesian p-value: 0.273
MAWA Bayesian p-value: 0.508
NAWA Bayesian p-value: 0.598
OVEN Bayesian p-value: 0.531
REVI Bayesian p-value: 0.503
Fit statistic:  freeman-tukey
```

We see all Bayesian p-values are quite large, which is probably at least partly due to the fact that the chains have yet to converge.

5.5 Model selection using WAIC

Below we compute the WAIC using `waicOcc` and compare it to the WAIC for the non-spatial multispecies occupancy model.

```
waicOcc(out.sp.ms)
```

```
      elpd         pD       WAIC
-4300.0960   300.5876  9201.3672
```

```
waicOcc(out.ms)
```

```
       elpd          pD        WAIC
-4638.61419    70.97646  9419.18130
```

The WAIC for the spatial model is much larger than that for the nonspatial model, potentially indicating we don't need the additional complexities brought in by the species-specific spatial processes. However, we should not place a large emphasis on this since neither of the models has completely converged.

5.6 Prediction

Out-of-sample prediction with `spMsPGOcc` objects again uses the `predict` function given a set of covariates and spatial coordinates of unobserved locations. Here we predict values for all 12 species at every 50th cell of the total cells. Results are very similar to the nonspatial multispecies model, so we do not execute the following code.

```
X.0.ms <- cbind(1, hbefElev$val, hbefElev$val^2)
X.0.ms <- X.0.ms[seq(1, nrow(X.0.ms), by = 50), ]
coords.0 <- hbefElev[seq(1, nrow(hbefElev), by = 50), 2:3]
out.sp.ms.pred <- predict(out.sp.ms, X.0.ms, coords.0)
psi.pred.oven <- out.sp.ms.pred$psi.0.samples[, which(sp.names == 'OVEN'), ]
hist(apply(psi.pred.oven, 2, mean), xlab = 'Mean Occurrence Probability')
```

6 Single species integrated occupancy models

Data integration is a model-based approach that leverages multiple data sources to provide inference and prediction on some latent process of interest. Data integration is particularly relevant in ecology as many data sources are often collected to study a single ecological phenomenon, with each data source having pros and cons. Often, multiple detection-nondetection data sources are available to study the occurrence and distribution of some species of interest. For example, both human point count surveys and autonomous recording units could be used to monitor a bird species of conservation concern. Different types of data have different sources of observation error, which should be explicitly incorporated into a model to avoid attributing any variation in detection probability to the true ecological process. Here we describe single species integrated occupancy models, which combine multiple sources of detection-nondetection data (which may or may not be replicated) in a single hierarchical modeling framework.

6.1 Basic model description

The integrated occupancy model has an identical process model to the single species occupancy model, and has a distinct detection model for each data source that are all conditional on the same shared ecological process (species occurrence).

Let z_j be the presence or absence of a species at site j, with $j = 1, \ldots, J$. We assume this latent occurrence process arises from a Bernoulli process following

$$\begin{aligned} z_j &\sim \text{Bernoulli}(\psi_j), \\ \text{logit}(\psi_j) &= \boldsymbol{x}_j'\boldsymbol{\beta}, \end{aligned} \tag{12}$$

where ψ_j is the probability of occurrence at site j, which is a function of site-specific covariates $\boldsymbol{X}$ and a vector of regression coefficients ($\boldsymbol{\beta}$).

We do not directly observe z_j and rather we observe an imperfect representation of the latent occurrence process. In integrated models, we have $r = 1, \ldots, R$ distinct sources of data that are all imperfect representations of a single, shared occurrence process. Let $y_{r,a,k}$ be the observed detection (1) or nondetection (0) of a species of interest in data set r at site a during replicate k. Because different data sources have different variables influencing the observation process, we envision a separate detection model for each data source that is conditional on a single, shared ecological process described by Equation (12). We envision the detection-nondetection data from source r as arising from a Bernoulli process conditional on the true latent occurrence process:

$$\begin{aligned} y_{r,a,k} &\sim \text{Bernoulli}(p_{r,a,k} z_{j[a]}), \\ \text{logit}(p_{r,a,k}) &= \boldsymbol{v}_{r,a,k}'\boldsymbol{\alpha}_r, \end{aligned} \tag{13}$$

where $p_{r,a,k}$ is the probability of detecting a species at site a during replicate k (given it is present at site a) for data source r, which is a function of site, replicate, and data source specific covariates $\boldsymbol{V}_r$ and a vector of

regression coefficients specific to each data source ($\boldsymbol{\alpha}_r$). Note that $z_{j[a]}$ is the true occurrence status at site j corresponding to the ath data source site in the given data set r. Each data source may be available at all J sites in the region of interest or at a subset of the J sites. Additionally, data sources can overlap in the sites they sample, or they can be obtained at distinct sites within all J sites of interest in the overall region.

We assume multivariate normal priors for the occurrence ($\boldsymbol{\beta}$) and data-set specific detection ($\boldsymbol{\alpha}$) regression coefficients to complete the Bayesian specification of a single species occupancy model. Polya-Gamma data augmentation is implemented analgoous to previous models to yield an efficient implementation of integrated occupancy models.

6.2 Simulating data using `simIntOcc`

Here we will simulate data to fit with `intPGOcc` using the function `simIntOcc`. Given input values for certain parameters, `simIntOcc` simulates single-species occupancy data from multiple data sources. `spOccupancy` also includes functions for simulation of single species (`simOcc`) and multispecies (`simMsOcc`) data sets that we do not describe in this vignette. `simIntOcc` takes the following form:

```
simIntOcc(n.data, J.x, J.y, J.obs, n.rep, beta, alpha, sigma.sq = 2,
      phi = 3/0.5, sp = FALSE)
```

`simIntOcc` simulates data assumed to come from a set of sites distributed in a rectangular sampling design. We next briefly detail the purpose of each of the arguments to `simIntOcc`.

- `n.data`: the number of detection-nondetection data sources to simulate.
- `J.x`: number of sites along the horizontal axis.
- `J.y`: number of sites along the vertical axis. The total number of sites across the simulated region of interest is J.x x J.y.
- `J.obs`: a numeric vector with `n.data` values indicating the number of sites to simulate each data source from.
- `n.rep`: a list of `n.rep` elements, where each element is a numeric vector indicating the number of replicates of the detection-nondetection data at each site of a given data source.
- `beta`: a numeric vector of the simulated regression coefficients for the occurrence portion of the model (including an intercept parameter).
- `alpha`: a list of `n.data` elements, where each element is a numeric vector containing the intercept and regression parameters simulated for each data source.

The remaining parameters are used for single species spatial integrated occupancy models, and so we will leave them as their default values for now (which simulates nonspatial data). We simulate data across a region of 225 sites with four data sources.

```
set.seed(101)
J.x <- 15
J.y <- 15
J.all <- J.x * J.y
# Number of data sources.
n.data <- 4
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
# Replicates for each data source.
n.rep <- list()
for (i in 1:n.data) {
  n.rep[[i]] <- sample(1:4, size = J.obs[i], replace = TRUE)
}
# Occupancy covariates
beta <- c(0.5, 1, -2)
p.occ <- length(beta)
```

```
# Detection covariates
alpha <- list()
for (i in 1:n.data) {
  alpha[[i]] <- runif(sample(1:4, 1), -1, 1)
}
p.det.long <- sapply(alpha, length)
p.det <- sum(p.det.long)

# Simulate occupancy data.
dat <- simIntOcc(n.data = n.data, J.x = J.x, J.y = J.y, J.obs = J.obs,
         n.rep = n.rep, beta = beta, alpha = alpha)
```

The return object `dat` is a list comprised of a series of objects related to the simulated data that we can use for fitting an integrated occupancy model, verifying the model works as we would expect, and assessing predictive performance of the model at non-sampled locations. For initial model fitting, we are primarily concerned with the objects `X.obs`, `X.p`, `sites`, and `y`. `X.obs` is a numeric matrix for the occurrence portion of the occupancy model. `X.p` is a list of design matrices for the detection portion of the occupancy model, where each element of the list corresponds to a different data source. `sites` is a list where each element is a vector of site indices for each data source. In an integrated model, data sources can be obtained at the same locations, completely different locations within a single region of interest, or a mix of the two. A key task when working with integrated models is ensuring each data point in a given data source is linked to the correct site in the occurrence portion of the model. For example, consider the simulated `sites` list below

```
str(dat$sites)
```

```
List of 4
 $ : num [1:101] 1 3 5 6 9 10 11 13 15 16 ...
 $ : num [1:90] 3 6 8 12 17 18 26 28 29 30 ...
 $ : num [1:102] 1 2 3 4 7 9 12 14 21 24 ...
 $ : num [1:105] 1 6 7 8 9 10 13 14 18 19 ...
```

The indices in the first vector indicate that the first site in the first data source corresponds to overall site 1, the second site in the first data source corresponds to the overall site 3, the third element in the first data source corresponds to the overall site 5, and so on. Finally, the element `y` consists of the detection-nondetection data matrices for each data source, stored in a list.

```
y <- dat$y
X.p <- dat$X.p
X <- dat$X.obs
sites <- dat$sites
```

6.3 Fitting single species integrated occupancy models with `intPGOcc`

The function `intPGOcc` fits single species integrated occupancy models in `spOccupancy`. Syntax is very similar to single data source models, and specifically takes the following form:

```
intPGOcc(occ.formula, det.formula, data, starting, n.samples, priors,
     n.omp.threads = 1, verbose = TRUE, n.report = 1000,
     n.burn = round(.10 * n.samples), n.thin = 1, ...)
```

The `data` argument contains the list of data elements necessary for fitting an integrated occupancy model. For nonspatial integrated occupancy model, `data` should be a list comprised of the following objects: `y` (list of detection-nondetection data matrices for each data source), `occ.covs` (data frame or matrix of covariates for occurrence model), `det.covs` (a list of lists where each element of the list corresponds to the detection-nondetection data for the given dat asource), `sites` (a list where each element consists of the site

indices for the given data source. To get started, we'll first package up the simulated data into a a list in the necessary format for the `data` argument

```
# Occurrence covariates
occ.covs <- X[, -1, drop = FALSE]
colnames(occ.covs) <- c('occ.cov.1', 'occ.cov.2')
# Detection covariates
det.covs <- list()
# Number of covariates on detection for each data source
lapply(alpha, function(a) length(a) - 1)
```

```
[[1]]
[1] 2

[[2]]
[1] 3

[[3]]
[1] 1

[[4]]
[1] 1
```

```
# Add detection covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2],
                      det.cov.1.2 = X.p[[1]][, , 3])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2],
                      det.cov.2.2 = X.p[[2]][, , 3],
                      det.cov.2.3 = X.p[[2]][, , 4])
det.covs[[3]] <- list(det.cov.3.1 = X.p[[3]][, , 2])
det.covs[[4]] <- list(det.cov.4.1 = X.p[[4]][, , 2])
data.list <- list(y = y,
                  occ.covs = occ.covs,
                  det.covs = det.covs,
                  sites = sites)
str(data.list)
```

```
List of 4
 $ y       :List of 4
  ..$ : int [1:101, 1:4] 0 0 0 1 0 0 0 1 1 1 ...
  ..$ : int [1:90, 1:4] 0 1 0 0 1 1 0 0 0 1 ...
  ..$ : int [1:102, 1:4] 0 1 0 1 0 0 0 0 0 0 ...
  ..$ : int [1:105, 1:4] 0 0 1 1 0 0 1 0 1 1 ...
 $ occ.covs: num [1:202, 1:2] -2.1677 0.5984 0.0431 1.295 0.7063 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "occ.cov.1" "occ.cov.2"
 $ det.covs:List of 4
  ..$ :List of 2
  .. ..$ det.cov.1.1: num [1:101, 1:4] 2.1575 -0.0408 -1.8213 -0.5695 0.4083 ...
  .. ..$ det.cov.1.2: num [1:101, 1:4] -0.163 -1.549 -1.703 0.68 -1.078 ...
  ..$ :List of 3
  .. ..$ det.cov.2.1: num [1:90, 1:4] 0.519 -1.884 -0.333 0.591 0.468 ...
  .. ..$ det.cov.2.2: num [1:90, 1:4] -2.033049 0.669742 -2.371423 -0.000267 1.435769 ...
  .. ..$ det.cov.2.3: num [1:90, 1:4] 0.39 -0.309 0.073 -1.692 -0.218 ...
  ..$ :List of 1
```

```
  .. ..$ det.cov.3.1: num [1:102, 1:4] -0.702 -1.253 0.91 -1.371 1.348 ...
  ..$ :List of 1
  .. ..$ det.cov.4.1: num [1:105, 1:4] 0.1934 -0.1415 2.8139 1.0457 -0.0637 ...
 $ sites   :List of 4
  ..$ : num [1:101] 1 3 5 6 9 10 11 13 15 16 ...
  ..$ : num [1:90] 3 6 8 12 17 18 26 28 29 30 ...
  ..$ : num [1:102] 1 2 3 4 7 9 12 14 21 24 ...
  ..$ : num [1:105] 1 6 7 8 9 10 13 14 18 19 ...
```

An important thing to note for integrated occupancy models in `spOccupancy` is that the order of sites should be based on the ordering of the covariates specified in `occ.covs`. That is, for the site indices for each data source stored in `sites`, site 1 should correspond to the first row in `occ.covs`, site 2 should correspond to the second row, and so on.

We specify the occurrence and detection model formulas using the `occ.formula`, and `det.formula` arguments. The `occ.formula` remains unchanged from previous models. For our simulation here, we model occurrence as a function of a two covariates.

```
sim.occ.formula <- ~ occ.cov.1 + occ.cov.2
```

For the detection models, we need to specify a different detection model for each data source. We do this by sending in a list to the `det.formula` argument, where each element of the list is the model formula for that given data set. Here we specify the four formulas in a list, where the data sources are modeled with 2, 3, 1, and 1 covariates, respectively. The name of the individual list elements is not important. Rather, the position of the model formula should be consistent with the position of the data source in the detection-nondetection data list (`y`) specified in the `data` argument.

```
sim.det.formula = list(f.1 = ~ det.cov.1.1 + det.cov.1.2,
                       f.2 = ~ det.cov.2.1 + det.cov.2.2 + det.cov.2.3,
                       f.3 = ~ det.cov.3.1,
                       f.4 = ~ det.cov.4.1)
```

Next we specify the starting values. Starting values are specified in a list with the following tabs: `z` (latent occurrence values), `alpha` (detection regression coefficients), and `beta` (occurrence regression coefficients. This aligns with fitting single species occupancy models using `PGOcc`. However, since we now have multiple detection models with different coefficients for each data source, starting values for `alpha` are now sent passed to `intPGOcc` as a list, with each element of the list corresponding to the starting detection parameter values for a given data source.

```
J <- nrow(occ.covs)
alpha.start <- list()
for (q in 1:n.data) {
  alpha.start[[q]] <- rep(0, p.det.long[q])
}
starting.list <- list(alpha = alpha.start,
              beta = rep(0, p.occ),
              z = rep(1, J))
```

We next specify the priors for all parameters in the integrated occupancy model in a list that is passed into the `priors` argument. We specify normal priors for both the occurrence and detection regression coefficients, using tags `beta.normal` and `alpha.normal`, respectively.

```
alpha.mu <- list()
alpha.var <- list()
for (q in 1:n.data) {
  alpha.mu[[q]] <- rep(0, p.det.long[q])
  alpha.var[[q]] <- rep(2.72, p.det.long[q])
}
```

```
prior.list <- list(beta.normal = list(mean = rep(0, p.occ),
                                      var = rep(2.72, p.occ)),
                   alpha.normal = list(mean = alpha.mu,
                                       var = alpha.var))
```

Priors for the occurrence regression coefficients are specified as we have seen in previous models. Because we have multiple detection-nondetection data sets each with distinct detection parameters, we specify the hypermeans and hypervariances in individual lists, where each element of the list corresponds to a specific data source. Again, the ordering of the data sources in the lists must align with the order the data sources are saved in the detection-nondetection data supplied to the `data` argument.

Finally, we specify the number of samples, burn-in, and thinning rate using the same approach we have used for previous models.

```
n.samples <- 5000
n.burn <- 1000
n.thin <- 1
```

We can now run the integrated occupancy model. Below we set the number of threads used to 1 and print out sampler progress after every 1000th iteration.

```
out.int <- intPGOcc(occ.formula = sim.occ.formula,
                    det.formula = sim.det.formula,
             data = data.list,
             starting = starting.list,
             n.samples = n.samples,
             priors = prior.list,
             n.omp.threads = 1,
             verbose = TRUE,
             n.report = 1000,
             n.burn = n.burn,
             n.thin = n.thin)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Integrated Occupancy Model with Polya-Gamma latent
variable fit with 202 sites.

Integrating 4 occupancy data sets.

Number of MCMC samples: 5000
Burn-in: 1000
Thinning Rate: 1
Total Posterior Samples: 4000


Source compiled with OpenMP support and model fit using 1 thread(s).

Sampling ...
Sampled: 1000 of 5000, 20.00%
-------------------------------------------------
Sampled: 2000 of 5000, 40.00%
```

```
-------------------------------------------------
Sampled: 3000 of 5000, 60.00%
-------------------------------------------------
Sampled: 4000 of 5000, 80.00%
-------------------------------------------------
Sampled: 5000 of 5000, 100.00%
```

We again consult the `summary` function for a concise description of the model results.

```
summary(out.int)
```

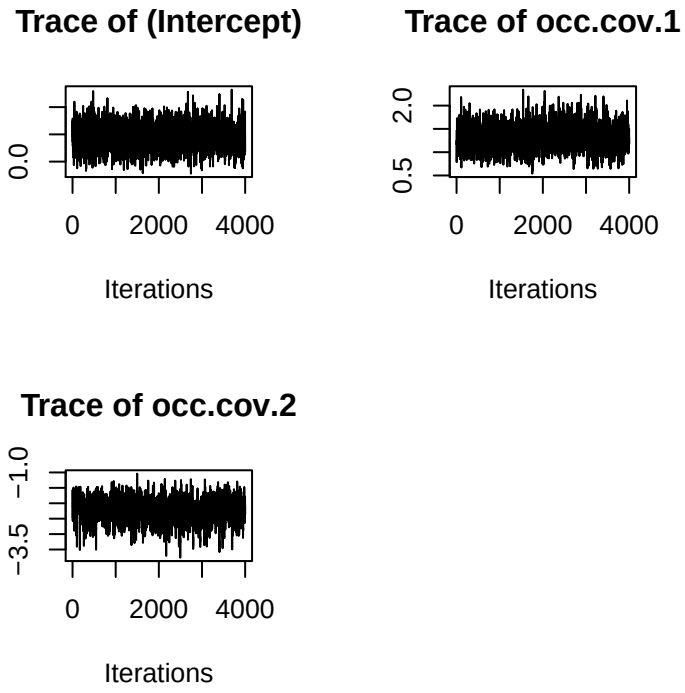
```
Call:
intPGOcc(occ.formula = sim.occ.formula, det.formula = sim.det.formula,
    data = data.list, starting = starting.list, priors = prior.list,
    n.samples = n.samples, n.omp.threads = 1, verbose = TRUE,
    n.report = 1000, n.burn = n.burn, n.thin = n.thin)

Chain Information:
Total samples: 5000
Burn-in: 1000
Thin: 1
Total Posterior Samples: 4000

Occurrence:
               2.5%     25%     50%     75%   97.5%
(Intercept)  0.0343  0.3064  0.4493  0.5901  0.8888
occ.cov.1    0.8479  1.1392  1.3106  1.4877  1.8560
occ.cov.2   -3.0204 -2.4609 -2.2121 -1.9675 -1.5602

Data source 1 Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept)  0.5440  0.7985  0.9328  1.0790  1.3781
det.cov.1.1 -1.1590 -0.8383 -0.6811 -0.5239 -0.2546
det.cov.1.2  0.2711  0.5341  0.6770  0.8232  1.1258

Data source 2 Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept) -0.6212 -0.3479 -0.2100 -0.0647  0.2142
det.cov.2.1 -1.4103 -1.0377 -0.8673 -0.6996 -0.3865
det.cov.2.2 -0.1513  0.1007  0.2265  0.3560  0.6152
det.cov.2.3  0.6429  0.9767  1.1623  1.3654  1.7686

Data source 3 Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept) -0.7224 -0.5019 -0.3849 -0.2737 -0.0641
det.cov.3.1 -0.7578 -0.5272 -0.4095 -0.2934 -0.0607

Data source 4 Detection:
              2.5%    25%    50%    75%  97.5%
(Intercept) 0.3333 0.5508 0.6666 0.7809 1.0062
det.cov.4.1 0.4130 0.6602 0.7889 0.9239 1.1944
```

The `summary` function for integrated models returns the detection parameters separately for each detection covariate. We can compare these results to the values we used to simulate the data. It looks like our model does a pretty good job of recovering the true parameter values.

6.4 Convergence diagnostics

Posterior samples are returned as `coda::mcmc` objects, so as with all `spOccupancy` model objects, we use standard `coda` functions like `plot` and `gelman.diag` to assess convergence.

```
# Occurrence effects
plot(out.int$beta.samples, density = FALSE)
```



6.5 Posterior predictive checks

We perform posterior predictive checks using `ppcocc` as before. GoF assessment for integrated models is an active area of research. In `spOccupancy`, we compute posterior predictive checks separately for each dataset in the integrated model.

```
ppc.int.out <- ppcOcc(out.int, 'freeman-tukey', group = 2)
summary(ppc.int.out)
```

```
Call:
ppcOcc(object = out.int, fit.stat = "freeman-tukey", group = 2)

Chain Information:
Total samples: 5000
Burn-in: 1000
Thin: 1
Total Posterior Samples: 4000

Data Source 1

Bayesian p-value: 0.525
```

```
Fit statistic: freeman-tukey

Data Source 2

Bayesian p-value: 0.79225
Fit statistic: freeman-tukey

Data Source 3

Bayesian p-value: 0.37075
Fit statistic: freeman-tukey

Data Source 4

Bayesian p-value: 0.91125
Fit statistic: freeman-tukey
```

6.6 Model selection using WAIC

We use `waicOcc` to compute the WAIC for integrated occupancy models. Similar to the posterior predictive check, individual WAIC values are reported for each data set. These can be summed across all data sources for an overall WAIC value if desired.

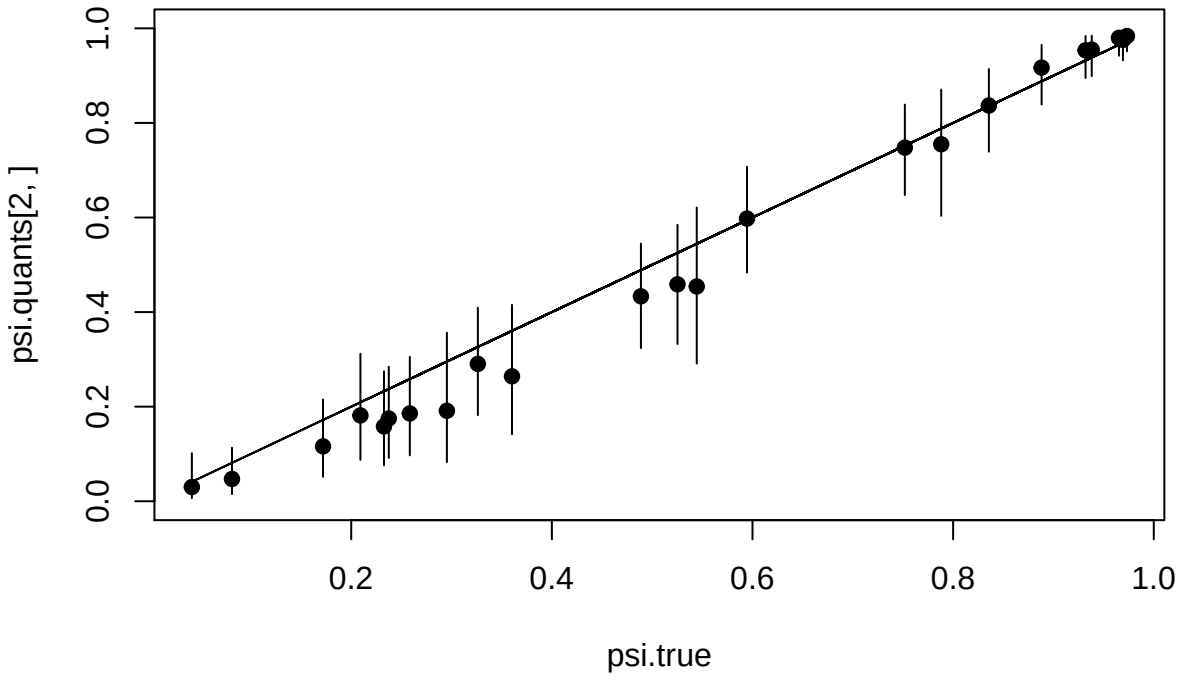
```
waicOcc(out.int)
```

```
       elpd       pD      WAIC
1 -86.99108 6.090087 186.16234
2 -35.32665 1.521254  73.69580
3 -48.26210 1.340008  99.20422
4 -74.27592 2.475897 153.50363
```

6.7 Prediction

Prediction for integrated occupancy models proceeds exactly as before using `predict`. The `simIntOcc` function automatically splits the data into a set of sampled sites and nonsampled sites. We can use the simulated data from the non-sampled sites to assess the ability of our model to make accurrate out of sample predictions.

```
X.0 <- dat$X.pred
out.int.pred <- predict(out.int, X.0)
# Compare predictions to true simulated values
psi.true <- dat$psi.pred
psi.quants <- apply(out.int.pred$psi.0.samples, 2, quantile,
            probs = c(0.025, 0.5, 0.975))
plot(psi.true, psi.quants[2, ], pch = 19, ylim = c(0, 1))
segments(psi.true, psi.quants[1, ], y1 = psi.quants[3, ])
lines(psi.true, psi.true)
```



7 Single species spatial integrated occupancy models

7.1 Basic model description

Single species spatial integrated occupancy models are identical to integrated occupancy models except the ecological process model now incorporates a spatially-structured random effect following the discussion in Section 3. All details for the single species integrated spatial occupancy model have already been presented in previous model descriptions.

7.2 Simulating data using `simIntOcc`

Here we again use `simIntOcc` to simulate a data set to display a single species spatial integrated occupancy model. We use the `simIntOcc` function just as we did in Section **??**, except we now specify the data to be simulated with spatially structured random effects following an exponential correlation function. Notice we now specify the spatial variance (`sigma.sq`) and spatial range (`phi`) parameters that we need to simulate the spatial random effects. We also set `sp = TRUE` to include the spatial random effects in the data simulation.

```
set.seed(405)
J.x <- 15
J.y <- 15
J.all <- J.x * J.y
# Number of data sources.
n.data <- 4
```

```
# Sites for each data source.
J.obs <- sample(ceiling(0.2 * J.all):ceiling(0.5 * J.all), n.data, replace = TRUE)
# Replicates for each data source.
n.rep <- list()
for (i in 1:n.data) {
  n.rep[[i]] <- sample(1:4, size = J.obs[i], replace = TRUE)
}
# Occupancy covariates
beta <- c(0.5, 0.5)
p.occ <- length(beta)
# Detection covariates
alpha <- list()
for (i in 1:n.data) {
  # A single detection covariate effect for each data set.
  alpha[[i]] <- runif(2, -1, 1)
}
p.det.long <- sapply(alpha, length)
p.det <- sum(p.det.long)
sigma.sq <- 3
phi <- 3 / .5
sp <- TRUE

# Simulate occupancy data.
dat <- simIntOcc(n.data = n.data, J.x = J.x, J.y = J.y, J.obs = J.obs,
         n.rep = n.rep, beta = beta, alpha = alpha, sigma.sq = sigma.sq,
         phi = phi, sp = sp)
y <- dat$y
X.p <- dat$X.p
X <- dat$X.obs
sites <- dat$sites
coords <- as.matrix(dat$coords.obs)
```

7.3 Fitting single speces spatial integrated occupancy models using `spIntPGOcc`

The function `spIntPGOcc` fits single species spatial integrated occupancy models in `spOccupancy`. Syntax is very similar to single data source models and specifically takes the following form:

```
spIntPGOcc(occ.formula, det.formula, data, starting, n.batch,
           batch.length, accept.rate = 0.43, priors,
           cov.model = "exponential", tuning, n.omp.threads = 1,
           verbose = TRUE, NNGP = TRUE, n.neighbors = 15,
           search.type = 'cb', n.report = 100,
           n.burn = round(.10 * n.batch * batch.length),
           n.thin = 1, ...)
```

The `occ.formula`, `det.formula`, and `data` arguments are analogous to what we saw with the nonspatial integrated occupancy model. However, as for all spatial models in `spOccupancy`, the `data` list must also contain the spatial coordinates.

```
# Occurrence covariates
occ.covs <- X[, -1, drop = FALSE]
colnames(occ.covs) <- c('occ.cov.1')
# Detection covariates
det.covs <- list()
```

```r
# Number of covariates on detection for each data source
lapply(alpha, function(a) length(a) - 1)
```

```
[[1]]
[1] 1

[[2]]
[1] 1

[[3]]
[1] 1

[[4]]
[1] 1
```

```r
# Add detection covariates one by one
det.covs[[1]] <- list(det.cov.1.1 = X.p[[1]][, , 2])
det.covs[[2]] <- list(det.cov.2.1 = X.p[[2]][, , 2])
det.covs[[3]] <- list(det.cov.3.1 = X.p[[3]][, , 2])
det.covs[[4]] <- list(det.cov.4.1 = X.p[[4]][, , 2])
data.list <- list(y = y,
          occ.covs = occ.covs,
          det.covs = det.covs,
          sites = sites,
          coords = coords)
str(data.list)
```

```
List of 5
 $ y       :List of 4
  ..$ : int [1:101, 1:4] 0 0 0 1 0 0 0 0 1 1 ...
  ..$ : int [1:93, 1:4] 0 1 0 0 0 0 0 0 0 0 ...
  ..$ : int [1:74, 1:4] 1 1 0 1 0 1 1 1 0 1 ...
  ..$ : int [1:109, 1:4] 0 1 0 0 0 0 1 1 1 1 ...
 $ occ.covs: num [1:200, 1] -0.9567 0.0462 -1.1998 -1.7878 0.6352 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "occ.cov.1"
 $ det.covs:List of 4
  ..$ :List of 1
  .. ..$ det.cov.1.1: num [1:101, 1:4] 0.6613 -0.217 -0.0912 1.6326 -0.0668 ...
  ..$ :List of 1
  .. ..$ det.cov.2.1: num [1:93, 1:4] -0.318 -0.849 -1.132 -0.457 0.742 ...
  ..$ :List of 1
  .. ..$ det.cov.3.1: num [1:74, 1:4] -0.673 1.258 -0.511 -0.587 -1.143 ...
  ..$ :List of 1
  .. ..$ det.cov.4.1: num [1:109, 1:4] 1.152 -0.948 1.657 1.154 0.789 ...
 $ sites   :List of 4
  ..$ : num [1:101] 1 3 5 7 8 9 10 13 14 15 ...
  ..$ : num [1:93] 3 6 7 10 11 15 16 17 20 21 ...
  ..$ : num [1:74] 1 2 5 10 11 12 13 15 16 20 ...
  ..$ : num [1:109] 1 2 4 6 7 9 13 14 15 17 ...
 $ coords  : num [1:200, 1:2] 0 0.0714 0.2143 0.3571 0.4286 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:200] "1" "2" "4" "6" ...
  .. ..$ : chr [1:2] "Var1" "Var2"
```

```
sim.occ.formula <- ~ occ.cov.1
sim.det.formula = list(f.1 = ~ det.cov.1.1,
                       f.2 = ~ det.cov.2.1,
                       f.3 = ~ det.cov.3.1,
                       f.4 = ~ det.cov.4.1)
```

Starting values specified in `starting` and priors in `priors` are specified in the same form as for `intPGOcc` with the additional values for spatial parameters. Analogous to all other spatial models in `spOccupancy`, the spatial variance parameter takes an inverse-Gamma prior and the spatial range parameter (and the spatial smoothness parameter if `cov.model = 'matern'`) takes a uniform prior.

```
J <- nrow(occ.covs)
alpha.start <- list()
for (q in 1:n.data) {
  alpha.start[[q]] <- rep(0, p.det.long[q])
}
starting.list <- list(alpha = alpha.start,
              beta = rep(0, p.occ),
              z = rep(1, J),
              phi = 3 / .4,
              sigma.sq = 1,
              w = rep(0, J))
alpha.mu <- list()
alpha.var <- list()
for (q in 1:n.data) {
  alpha.mu[[q]] <- rep(0, p.det.long[q])
  alpha.var[[q]] <- rep(2.72, p.det.long[q])
}
prior.list <- list(beta.normal = list(mean = rep(0, p.occ),
                        var = rep(2.72, p.occ)),
           alpha.normal = list(mean = alpha.mu,
                               var = alpha.var),
           sigma.sq.ig = c(2, 2),
           phi.unif = c(3/1, 3/.1))
```

Finally, we specify the remaining parameters regarding the NNGP specifications, tuning parameters, and the length of the MCMC sampler we will run. We are then all set to run the model.

```
tuning <- list(phi = 1)
batch.length <- 25
n.batch <- 400
cov.model <- "exponential"
out.sp.int <- spIntPGOcc(occ.formula = sim.occ.formula,
             det.formula = sim.det.formula,
             data = data.list,
             starting = starting.list,
             n.batch = n.batch,
             batch.length = batch.length,
             priors = prior.list,
             cov.model = cov.model,
             NNGP = TRUE,
             n.neighbors = 5,
             n.report = 100,
             n.burn = 1000,
             n.thin = 5)
```

```
-------------------------------------
    Preparing the data
-------------------------------------
-------------------------------------
    Building the neighbor list
-------------------------------------
-------------------------------------
Building the neighbors of neighbors list
-------------------------------------
-------------------------------------
    Model description
-------------------------------------
NNGP Integrated Occupancy Model with Polya-Gamma latent
variable fit with 200 sites.

Integrating 4 occupancy data sets.

Number of MCMC samples: 10000 (400 batches of length 25)
Burn-in: 1000
Thinning Rate: 5
Total Posterior Samples: 1800

Using the exponential spatial correlation model.

Using 5 nearest neighbors.


Source compiled with OpenMP support and model fit using 1 thread(s).

Adaptive Metropolis with target acceptance rate: 43.0
Sampling ...
Batch: 100 of 400, 25.00%
    parameter   acceptance  tuning
    phi     44.0        0.74826
-------------------------------------------------
Batch: 200 of 400, 50.00%
    parameter   acceptance  tuning
    phi     24.0        0.67706
-------------------------------------------------
Batch: 300 of 400, 75.00%
    parameter   acceptance  tuning
    phi     40.0        0.71892
-------------------------------------------------
Batch: 400 of 400, 100.00%
    parameter   acceptance  tuning
    phi     0.0     0.68386
-------------------------------------------------
```

```
summary(out.sp.int)
```

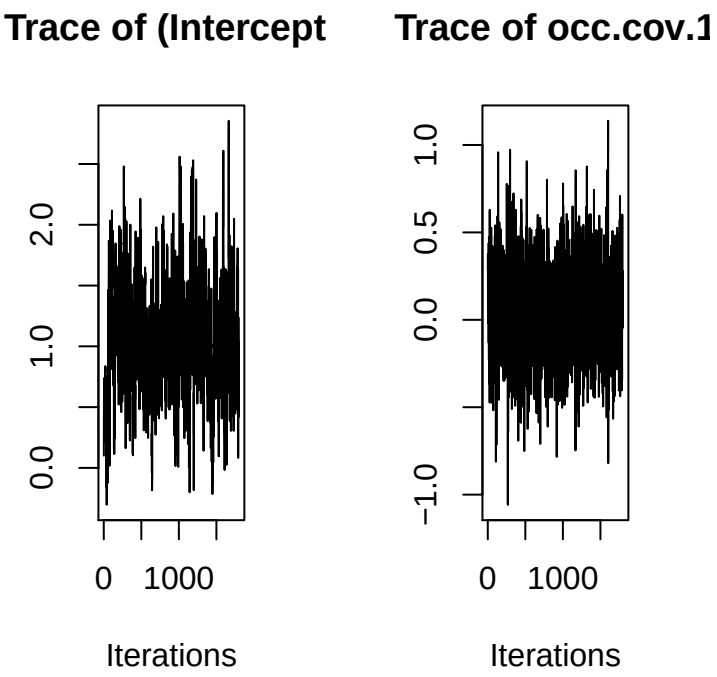
```
Call:
spIntPGOcc(occ.formula = sim.occ.formula, det.formula = sim.det.formula,
    data = data.list, starting = starting.list, priors = prior.list,
    cov.model = cov.model, NNGP = TRUE, n.neighbors = 5, n.batch = n.batch,
    batch.length = batch.length, n.report = 100, n.burn = 1000,
```

```
    n.thin = 5)

Chain Information:
Total samples: 10000
Burn-in: 1000
Thin: 5
Total Posterior Samples: 1800

Occurrence:
               2.5%     25%    50%    75%  97.5%
(Intercept)  0.1174  0.7585 1.0475 1.3466 2.0158
occ.cov.1   -0.4460 -0.1188 0.0541 0.2304 0.5661

Data source 1 Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept) -0.2524 -0.0195  0.1036  0.2191  0.4503
det.cov.1.1 -1.4504 -1.1158 -0.9610 -0.8064 -0.5436

Data source 2 Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept) -0.9643 -0.7533 -0.6205 -0.5015 -0.2697
det.cov.2.1 -0.9099 -0.6427 -0.5101 -0.3882 -0.1686

Data source 3 Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept)  0.4124  0.6540  0.7974  0.9414  1.2011
det.cov.3.1 -1.1749 -0.8796 -0.7293 -0.5787 -0.2873

Data source 4 Detection:
               2.5%     25%     50%     75%   97.5%
(Intercept)  0.1436  0.3496  0.4503  0.5497  0.7514
det.cov.4.1 -0.7823 -0.5815 -0.4672 -0.3598 -0.1586

Covariance:
           2.5%    25%    50%     75%   97.5%
sigma.sq 0.6977 1.4697 2.4686  3.9759  9.6093
phi      4.0311 6.7859 9.0214 11.8073 21.1508
```

7.4 Convergence diagnostics

We use the `coda` package to explore the trace plots. The trace plot below suggests we may want to run the model for longer to ensure convergence.

```
plot(out.sp.int$beta.samples, density = FALSE)
```



7.5 Posterior predictive checks

Below we perform a poseterior predictive check for each of the data sets included in the occupancy model using ppcOcc.

```
ppc.sp.int.out <- ppcOcc(out.sp.int, 'freeman-tukey', group = 2)
summary(ppc.sp.int.out)
```

```
Call:
ppcOcc(object = out.sp.int, fit.stat = "freeman-tukey", group = 2)

Chain Information:
Total samples: 10000
Burn-in: 1000
Thin: 5
Total Posterior Samples: 1800

Data Source 1

Bayesian p-value: 0.8116667
Fit statistic: freeman-tukey

Data Source 2

Bayesian p-value: 0.3738889
Fit statistic: freeman-tukey

Data Source 3
```

```
Bayesian p-value: 0.7272222
Fit statistic: freeman-tukey

Data Source 4

Bayesian p-value: 0.8416667
Fit statistic: freeman-tukey
```

According to the Bayesian p-values, there is no lack of fit for any of the three data sets.

7.6 Model selection using WAIC

We can perform model selection using WAIC with the `waicOcc` function as we have seen previously. Below, we reanalyze the simulated data using the `intPGOcc` function and compare the WAIC for the spatial and nonspatial integrated occupancy models

```
out.no.sp.int <- intPGOcc(occ.formula = sim.occ.formula,
                          det.formul = sim.det.formula,
                          data = data.list,
                          starting = starting.list,
                          priors = prior.list,
                          n.samples = n.batch * batch.length,
                          n.report = 2000,
                          n.burn = 1000,
                          n.thin = 5,
                          verbose = FALSE)
waicOcc(out.sp.int)
```

```
       elpd        pD      WAIC
1 -67.11086 13.229567 160.68085
2 -35.15136  5.810812  81.92433
3 -37.14281  7.470870  89.22736
4 -77.90818 12.550452 180.91726
```

```
waicOcc(out.no.sp.int)
```

```
       elpd        pD      WAIC
1 -71.00332 1.8907086 145.78805
2 -38.50209 0.6652268  78.33463
3 -39.97945 1.1022062  82.16331
4 -85.14092 1.5929944 173.46783
```

We see for this specific set of simulated data the WAIC is smaller for all four data sources in the spatial model compared to the nonspatial model.

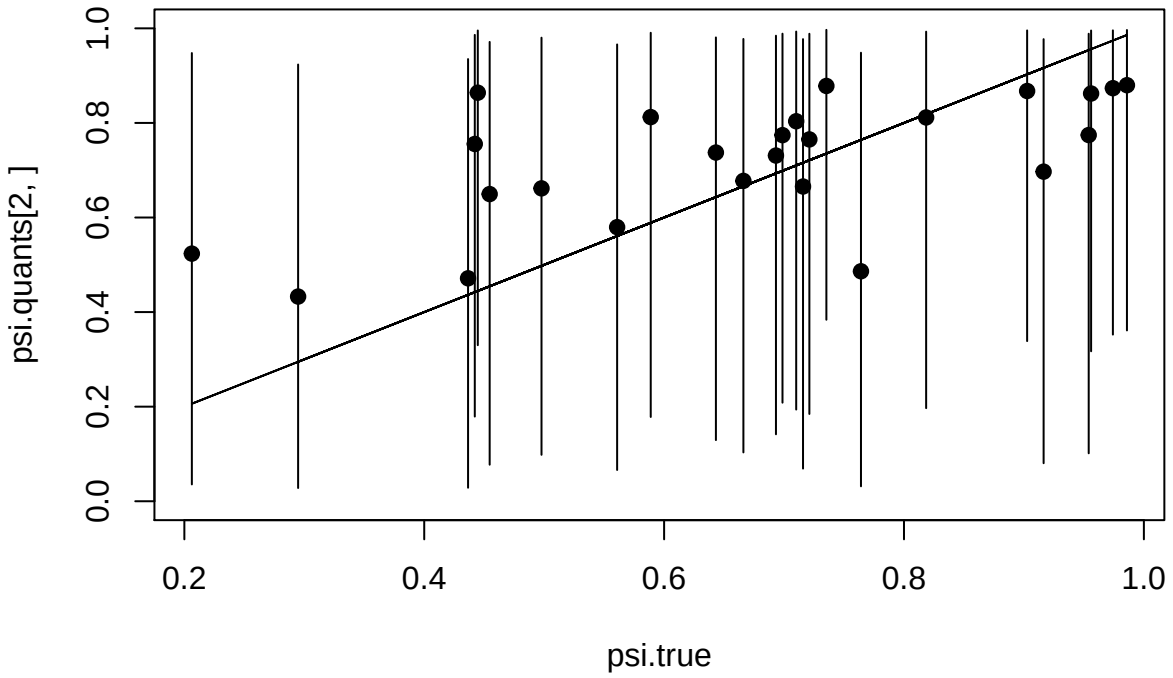
7.7 Prediction

Prediction for spatial integrated occupancy models proceeds exactly analogous to our approach using nonspatial integrated occupancy models. The only difference is that now we must also provide the coordinates of the nonsampled locations.

```
X.0 <- dat$X.pred
coords.0 <- dat$coords.pred
out.sp.int.pred <- predict(out.sp.int, X.0, coords.0)
```

```
----------------------------------------
    Prediction description
----------------------------------------
NNGP Occupancy model with Polya-Gamma latent
variable fit with 200 observations.

Number of covariates 2 (including intercept if specified).

Using the exponential spatial correlation model.

Using 5 nearest neighbors.

Number of MCMC samples 1800.

Predicting at 25 locations.


Source compiled with OpenMP support and model fit using 1 threads.
----------------------------------------------------
        Predicting
----------------------------------------------------
Location: 25 of 25, 100.00%
```

```r
# Compare predictions to true simulated values
psi.true <- dat$psi.pred
psi.quants <- apply(out.sp.int.pred$psi.0.samples, 2, quantile,
            probs = c(0.025, 0.5, 0.975))
plot(psi.true, psi.quants[2, ], pch = 19, ylim = c(0, 1))
segments(psi.true, psi.quants[1, ], y1 = psi.quants[3, ])
lines(psi.true, psi.true)
```



For this data set, the predictions seem to capture the trend in the non-sampled data, but the predictions have large amounts of uncertainty.

References

Banerjee, Sudipto, Bradley P Carlin, and Alan E Gelfand. 2003. *Hierarchical Modeling and Analysis for Spatial Data*. Chapman; Hall/CRC.

Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. "Fitting Linear Mixed-Effects Models Using lme4." *Journal of Statistical Software* 67 (1): 1–48. https://doi.org/10.18637/jss.v067.i01.

Broms, Kristin M, Mevin B Hooten, and Ryan M Fitzpatrick. 2016. "Model Selection and Assessment for Multi-Species Occupancy Models." *Ecology* 97 (7): 1759–70.

Brooks, Stephen P., and Andrew Gelman. 1998. "General Methods for Monitoring Convergence of Iterative Simulations." *Journal of Computational and Graphical Statistics* 7 (4): 434–55.

Clark, Allan E, and Res Altwegg. 2019. "Efficient Bayesian Analysis of Occupancy Models with Logit Link Functions." *Ecology and Evolution* 9 (2): 756–68.

Datta, Abhirup, Sudipto Banerjee, Andrew O Finley, and Alan E Gelfand. 2016. "Hierarchical Nearest-Neighbor Gaussian Process Models for Large Geostatistical Datasets." *Journal of the American Statistical Association* 111 (514): 800–812.

Doser, Jeffrey W., Wendy Leuenberger, T. Scott Sillett, Michael T. Hallworth, and Elise F. Zipkin. 2021. "Integrated Community Occupancy Models: A Framework to Assess Occurrence and Biodiversity Dynamics Using Multiple Data Sources." *arXiv Preprint arXiv:2109.01894*.

Finley, Andrew O, Abhirup Datta, and Sudipto Banerjee. 2020. "SpNNGP R Package for Nearest Neighbor Gaussian Process Models." *arXiv Preprint arXiv:2001.09111*.

Finley, Andrew O, Abhirup Datta, Bruce D Cook, Douglas C Morton, Hans E Andersen, and Sudipto Banerjee. 2019. "Efficient Algorithms for Bayesian Nearest Neighbor Gaussian Processes." *Journal of Computational and Graphical Statistics* 28 (2): 401–14.

Guélat, Jérôme, and Marc Kéry. 2018. "Effects of Spatial Autocorrelation and Imperfect Detection on Species Distribution Models." *Methods in Ecology and Evolution* 9 (6): 1614–25.

Heaton, Matthew J, Abhirup Datta, Andrew O Finley, Reinhard Furrer, Joseph Guinness, Rajarshi Guhaniyogi, Florian Gerber, et al. 2019. "A Case Study Competition Among Methods for Analyzing Large Spatial Data." *Journal of Agricultural, Biological and Environmental Statistics* 24 (3): 398–425.

Hobbs, N Thompson, and Mevin B Hooten. 2015. *Bayesian Models*. Princeton University Press.

Hooten, Mevin B, and N Thompson Hobbs. 2015. "A Guide to Bayesian Model Selection for Ecologists." *Ecological Monographs* 85 (1): 3–28.

Kéry, Marc, and J Andrew Royle. 2015. *Applied Hierarchical Modeling in Ecology: Volume 1: Prelude and Static Models*. Elsevier Science.

Lany, Nina K, Phoebe L Zarnetske, Andrew O Finley, and Deborah G McCullough. 2020. "Complementary Strengths of Spatially-Explicit and Multi-Species Distribution Models." *Ecography* 43 (3): 456–66.

Link, William A, John R Sauer, and Daniel K Niven. 2020. "Model Selection for the North American Breeding Bird Survey." *Ecological Applications* 30 (6): e02137.

McCullagh, Peter. 2019. "Generalized Linear Models."

Polson, Nicholas G, James G Scott, and Jesse Windle. 2013. "Bayesian Inference for Logistic Models Using Pólya–Gamma Latent Variables." *Journal of the American Statistical Association* 108 (504): 1339–49.

Roberts, Gareth O, and Jeffrey S Rosenthal. 2009. "Examples of Adaptive Mcmc." *Journal of Computational and Graphical Statistics* 18 (2): 349–67.

Watanabe, Sumio. 2010. "Asymptotic Equivalence of Bayes Cross Validation and Widely Applicable Information Criterion in Singular Learning Theory." *Journal of Machine Learning Research* 11 (12).