Fitting occupancy models with `spOccupancy`

Jeffrey W. Doser, Andrew O. Finley, Elise F. Zipkin

October 29, 2021

Contents

1 Introduction

This vignette provides worked examples and explanations for fitting single species, multispecies, and integrated 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 can reproduce 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 0 0 1 0 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] 475 494 546 587 588 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "Elevation"
 $ det.covs:List of 2
  ..$ day: num [1:373, 1:3] 156 156 156 156 156 156 156 156 156 156 ...
  ..$ tod: num [1:373, 1:3] 330 346 369 386 409 425 447 463 482 499 ...
 $ 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] "X" "Y"
```

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

```
  0   1 
518 588 
```

We see OVEN is detected at a few more than 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 $j = 1, \dots, J$. 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 $k = 1, \dots, K_j$ 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 (**P**ólya-**G**amma is where the `PG` comes from in all `spOccupancy` model fitting function names).

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). For example, to include a random intercept for different observers in the detection portion of the model, we would include `(1 | observer)` in the `det.formula`, where `observer` indicates the specific observer for each data point. 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 standardize all covariates by using the `scale` function in our model specification, and use the `I` function to specify quadratic effects:

```
oven.occ.formula <- ~ scale(Elevation) + I(scale(Elevation)^2)
oven.det.formula <- ~ scale(day) + scale(tod) + I(scale(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 0 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] 475 494 546 587 588 ...
  ..- attr(*, "dimnames")=List of 2
```

```
  .. ..$ : NULL
  .. ..$ : chr "Elevation"
 $ det.covs:List of 2
  ..$ day: num [1:373, 1:3] 156 156 156 156 156 156 156 156 156 156 ...
  ..$ tod: num [1:373, 1:3] 330 346 369 386 409 425 447 463 482 499 ...
 $ 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] "X" "Y"
```

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 as a vector of length corresponding to the number of estimated parameters, or as a single value if setting the same starting value for all parameters. Below we take the latter approach.

```
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 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) (Lunn et al. 2013). 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 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, which we will display in a subsequent section. For now, we won't specify the arguments, which will tell `PGOcc` not to perform k-fold cross-validation.

```r
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%
```

```r
names(out)
```

```
 [1] "beta.samples"  "alpha.samples" "z.samples"     "psi.samples"
 [5] "X"             "X.p"           "y"             "n.samples"
 [9] "call"          "n.post"        "n.thin"        "n.burn"
[13] "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, which returns multiple quantiles of the posterior samples of each parameter.

```r
summary(out)
```

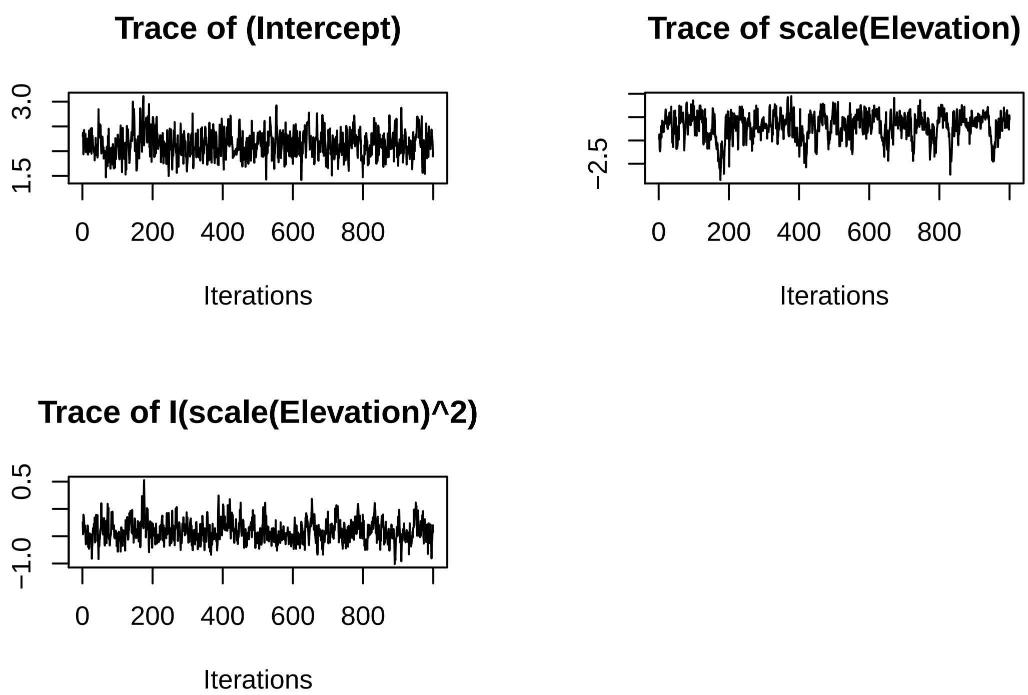
```
Call:
PGOcc(occ.formula = oven.occ.formula, det.formula = oven.det.formula,
    data = ovenHBEF, starting = oven.starting, priors = oven.priors,
    n.samples = n.samples, n.omp.threads = 1, verbose = TRUE,
    n.report = 1000, n.burn = n.burn, n.thin = n.thin)

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

Occurrence:
                        2.5%     25%     50%     75%   97.5%
(Intercept)           1.6543  1.9580  2.1307  2.3087  2.6855
scale(Elevation)      -2.3495 -1.8610 -1.6494 -1.4971 -1.2511
I(scale(Elevation)^2) -0.7537 -0.5443 -0.4280 -0.2993  0.0475

Detection:
                  2.5%     25%     50%     75%  97.5%
(Intercept)      0.5420  0.7069  0.7926  0.8794 1.0373
scale(day)      -0.2416 -0.1430 -0.0925 -0.0377 0.0605
scale(tod)      -0.2067 -0.0997 -0.0422  0.0078 0.1066
I(scale(day)^2) -0.1672 -0.0331  0.0291  0.0903 0.2226
```

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.

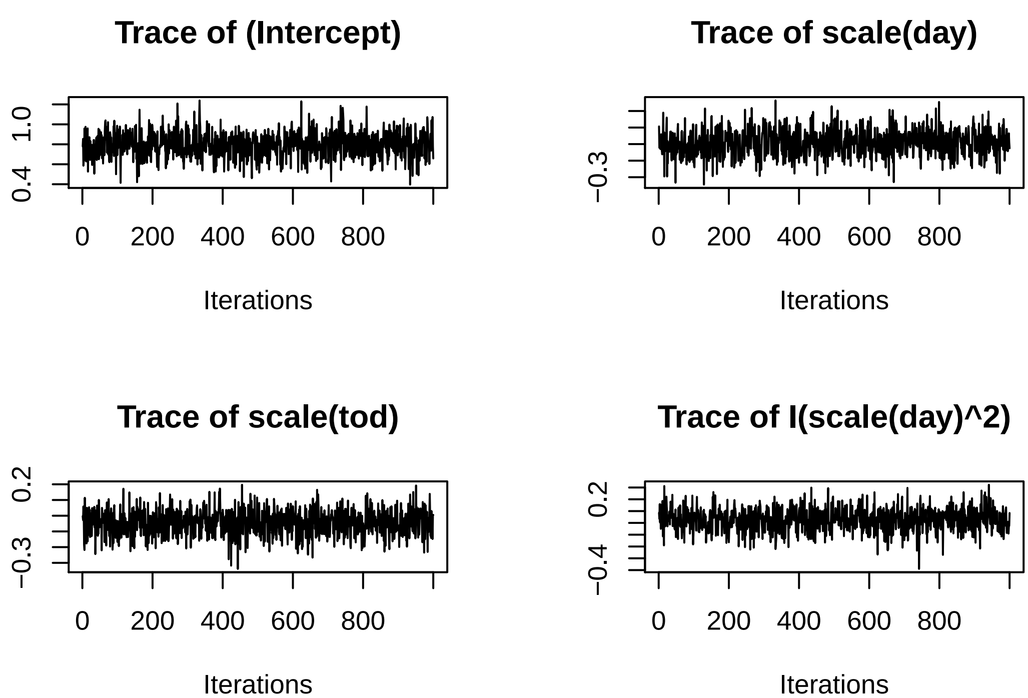
2.3 Convergence diagnostics

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

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




```r
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)                 1.01       1.02
scale(Elevation)            1.00       1.01
I(scale(Elevation)^2)       1.01       1.03

Multivariate psrf

1.01
```

```
# 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)          1.000          1
scale(day)           1.000          1
scale(tod)           0.999          1
I(scale(day)^2)      0.999          1

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. A good 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 is likely 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). In `spOccupancy`, a posterior predictive check broadly takes the following steps:

1. Fit the model using a model-fitting function (in this case `PGOcc`), which generates fitted values for all detection-nondetection data points.
2. Bin the detection-nondetection data in some manner.
3. Compute a fit statistic on the true data and the model generated fitted data.
4. Compare the fit statistics for the true data and model generated data. If they are widely different, this suggests a lack of fit. If they are reasonably close, this suggsests no lack of fit.

To peform a posterior predictive check, we send the resulting `PGOcc` model object 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).

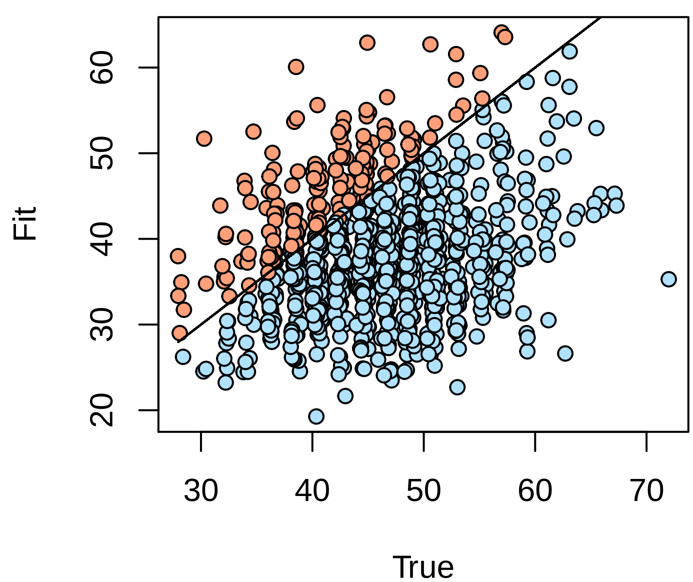
```r
ppc.out <- ppcOcc(out, fit.stat = 'freeman-tukey', group = 1)
summary(ppc.out)
```

```
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:  0.18
Fit statistic:  freeman-tukey
```

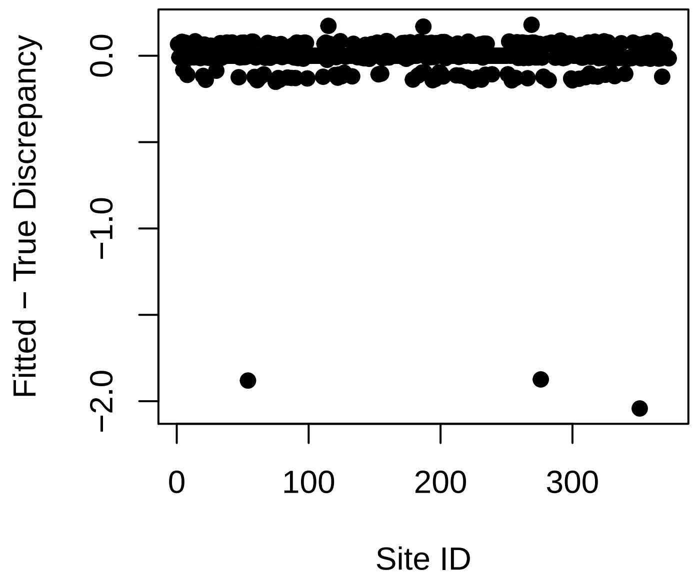
The Bayesian p-value is the proportion of posterior samples of the fit statistic of the model generated data that are greater than the corresponding fit statistic of the true data, summed across all "grouped" data points. We can create a visual representation of the Bayesian p-value as follows, which 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 is above 0.1 indicating no lack of fit, although the above plot indicates most of the fit statistics are smaller for the simulated data than the true data. 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 and k-fold cross-validation

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. 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.

```r
waicOcc(out)
```

```
       elpd          pD        WAIC 
-632.943683    6.231819 1278.351004 
```

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

```r
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 
-692.050575    4.877808 1393.856765 
```

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.

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 always be reliable for occupancy models (Broms, Hooten, and Fitzpatrick 2016). 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 $\boldsymbol{y}_k$, and 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 assessment (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 decrease 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 = 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,
                    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).
```

```r
# 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 = FALSE,
                        n.report = 1000,
                        n.burn = n.burn,
                        n.thin = n.thin,
                        k.fold = 4,
                        k.fold.threads = 1)
```

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

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

```
[1] 1518.765
```

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

```
[1] 1532.764
```

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

2.6 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. We load the data below

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

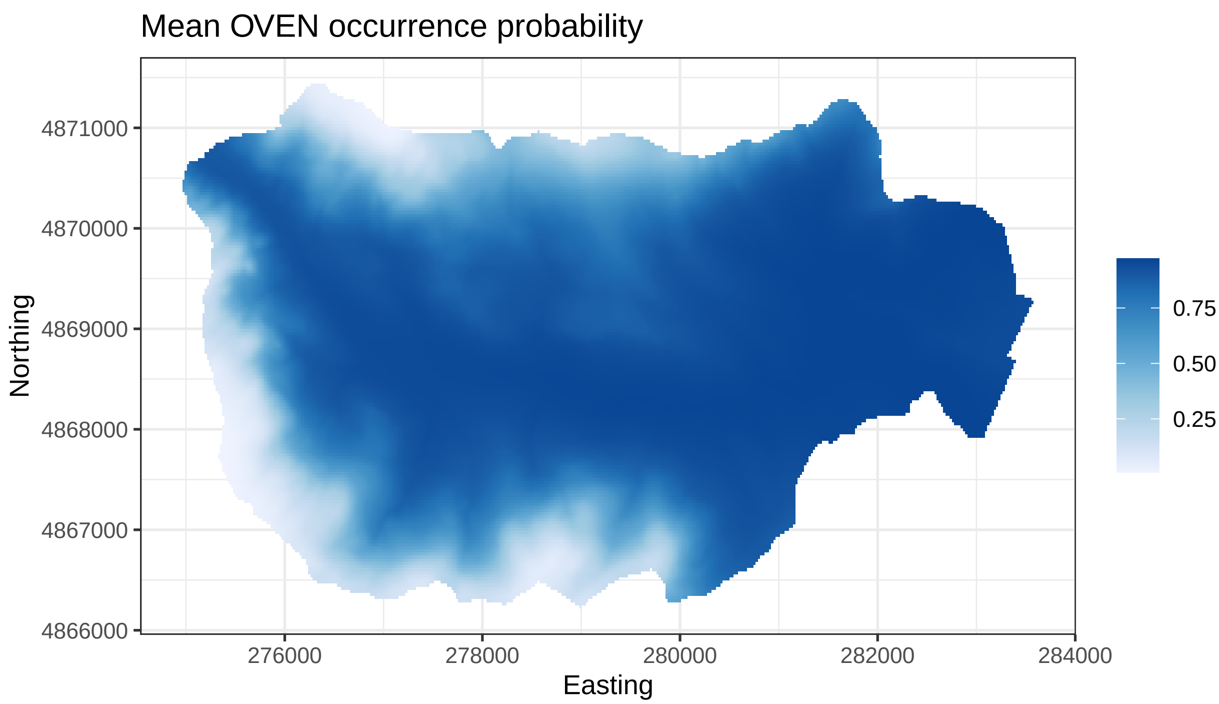
```
'data.frame':   46090 obs. of  3 variables:
 $ val     : num  914 916 918 920 922 ...
 $ Easting : num  276273 276296 276318 276340 276363 ...
 $ Northing: num  4871424 4871424 4871424 4871424 4871424 ...
```

The column `val` contains the 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. Given that we standardized the elevation values when we fit the model, we need to standardize the elevation values for prediction using the mean and standard deviation of the values used to fit the data.

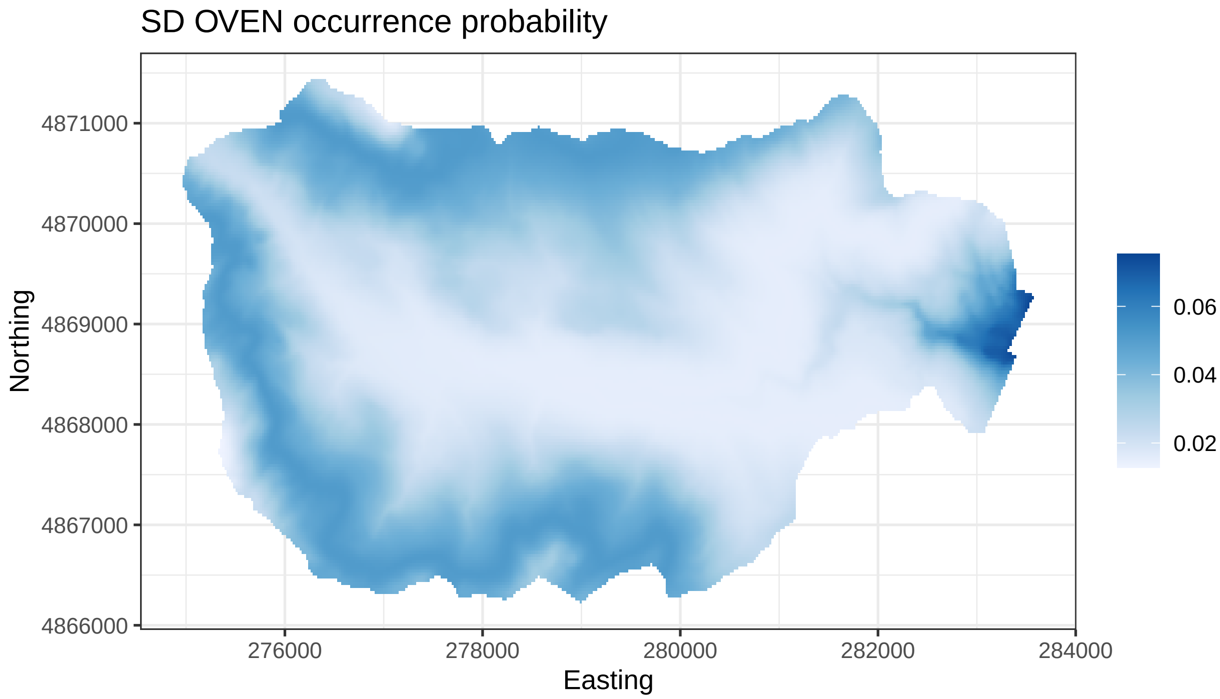
```r
elev.pred <- (hbefElev$val - mean(ovenHBEF$occ.covs[, 1])) / sd(ovenHBEF$occ.covs[, 1])
X.0 <- cbind(1, elev.pred, elev.pred^2)
out.pred <- predict(out, X.0)
```

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 identical 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 Pólya-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 <- ~ scale(Elevation) + I(scale(Elevation)^2)
oven.det.formula <- ~ scale(day) + scale(tod) + I(scale(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 0 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] 475 494 546 587 588 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "Elevation"
 $ det.covs:List of 2
  ..$ day: num [1:373, 1:3] 156 156 156 156 156 156 156 156 156 156 ...
  ..$ tod: num [1:373, 1:3] 330 346 369 386 409 425 447 463 482 499 ...
 $ 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] "X" "Y"
```

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 relatively small, 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 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 (especially for multispecies models). Unless you are concerned about the NNGP approximation for some reason, we recommend setting `NNGP = TRUE`, which is the default. 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`, which we have found in exploratory analysis to closely approximate a full Gaussian Process.

We now fit the model (without k-fold cross-validation) 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     52.0        0.41895
-------------------------------------------------
Batch: 200 of 400, 50.00%
    parameter   acceptance  tuning
    phi     44.0        0.29820
-------------------------------------------------
Batch: 300 of 400, 75.00%
    parameter   acceptance  tuning
    phi     36.0        0.29820
-------------------------------------------------
Batch: 400 of 400, 100.00%
```

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

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

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

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

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

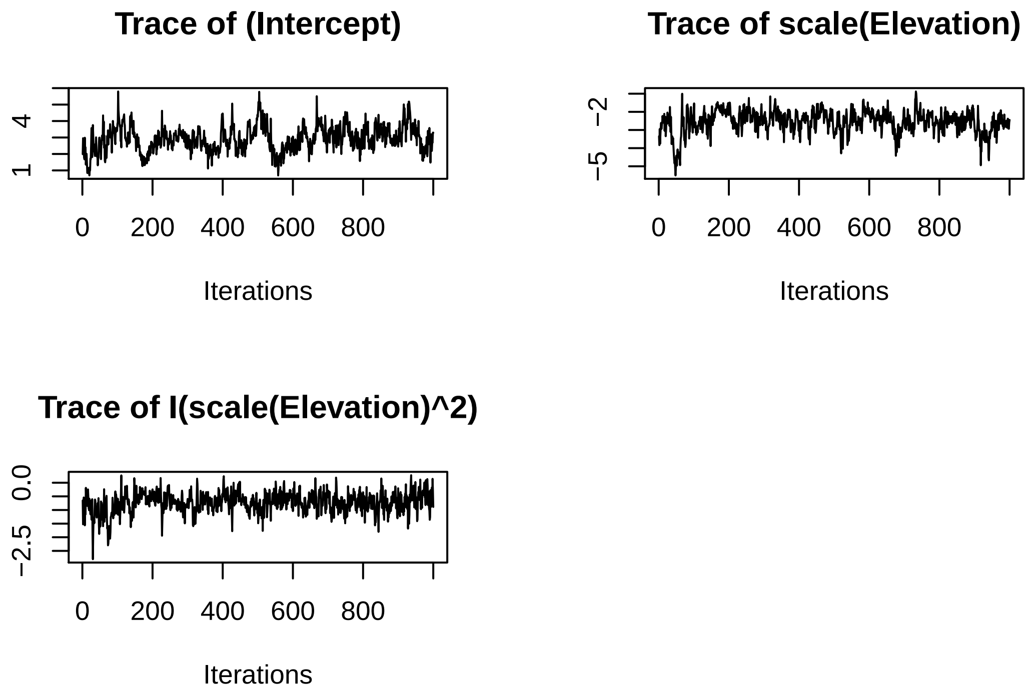
```
Call:
spPGOcc(occ.formula = oven.occ.formula, det.formula = oven.det.formula,
    data = ovenHBEF, starting = oven.starting, priors = oven.priors,
    tuning = oven.tuning, cov.model = cov.model, NNGP = TRUE,
    n.neighbors = 5, n.batch = n.batch, batch.length = batch.length,
    n.report = n.report, 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)            1.4307  2.4236  2.8833  3.4168  4.5185
scale(Elevation)      -4.1290 -2.8985 -2.5029 -2.1398 -1.5582
I(scale(Elevation)^2) -1.5601 -0.9407 -0.7115 -0.4739 -0.0111

Detection:
                   2.5%     25%     50%     75%  97.5%
(Intercept)      0.5928  0.7402  0.8185  0.8990 1.0540
scale(day)      -0.2439 -0.1429 -0.0942 -0.0361 0.0633
scale(tod)      -0.1900 -0.1003 -0.0516  0.0044 0.1151
I(scale(day)^2) -0.1657 -0.0381  0.0246  0.0837 0.2131

Covariance:
           2.5%    25%    50%    75%   97.5%
sigma.sq 0.8406 3.1496 4.8206 6.8445 28.2873
phi      0.0006 0.0013 0.0019 0.0025  0.0044
```

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`.

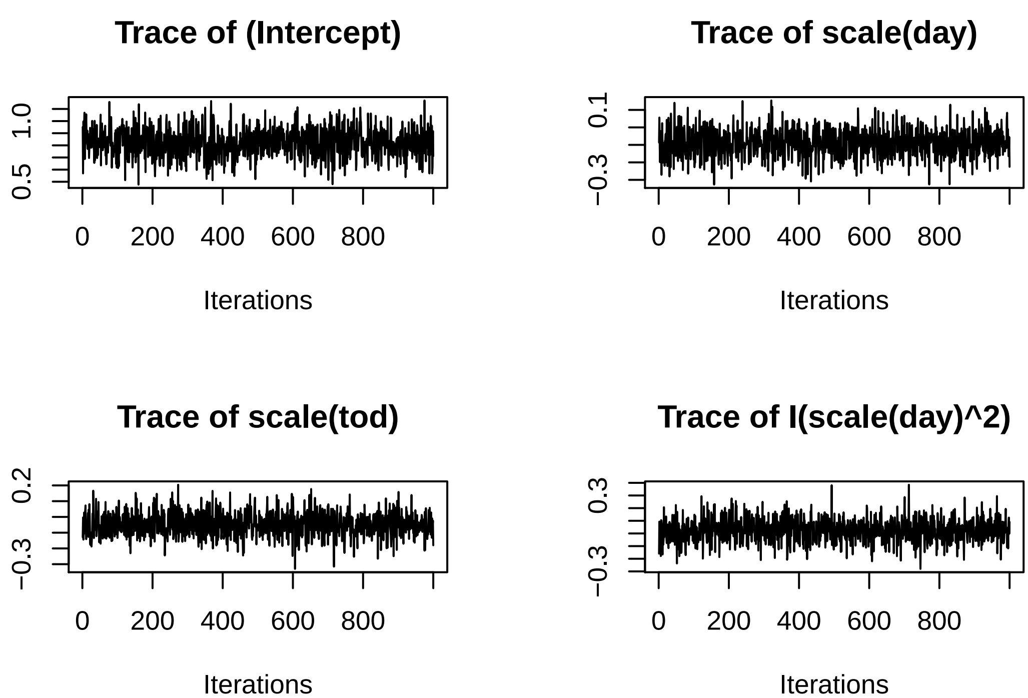
3.3 Convergence diagnostics

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`.

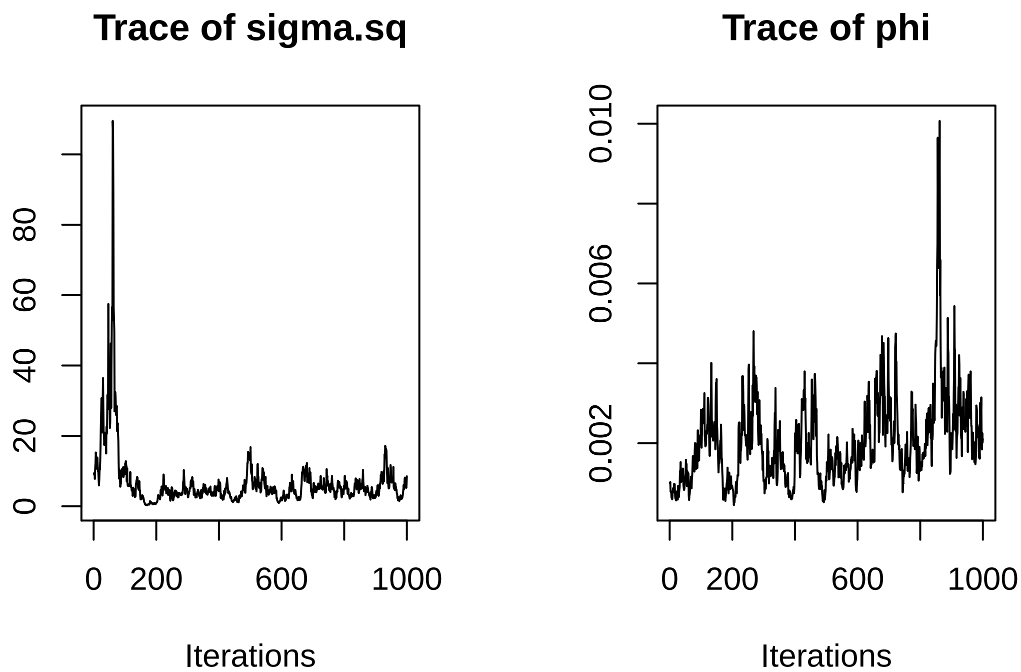
```r
plot(out.sp$beta.samples, density = FALSE)
```




```
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:  0.796
Fit statistic:  freeman-tukey
```

The Bayesian p-value indicates adequate model fit of the spatial occupancy model.

3.5 Model selection using WAIC and k-fold cross-validation

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
-568.0748   46.7733 1229.6962
```

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

```
       elpd          pD        WAIC
-632.943683    6.231819 1278.351004
```

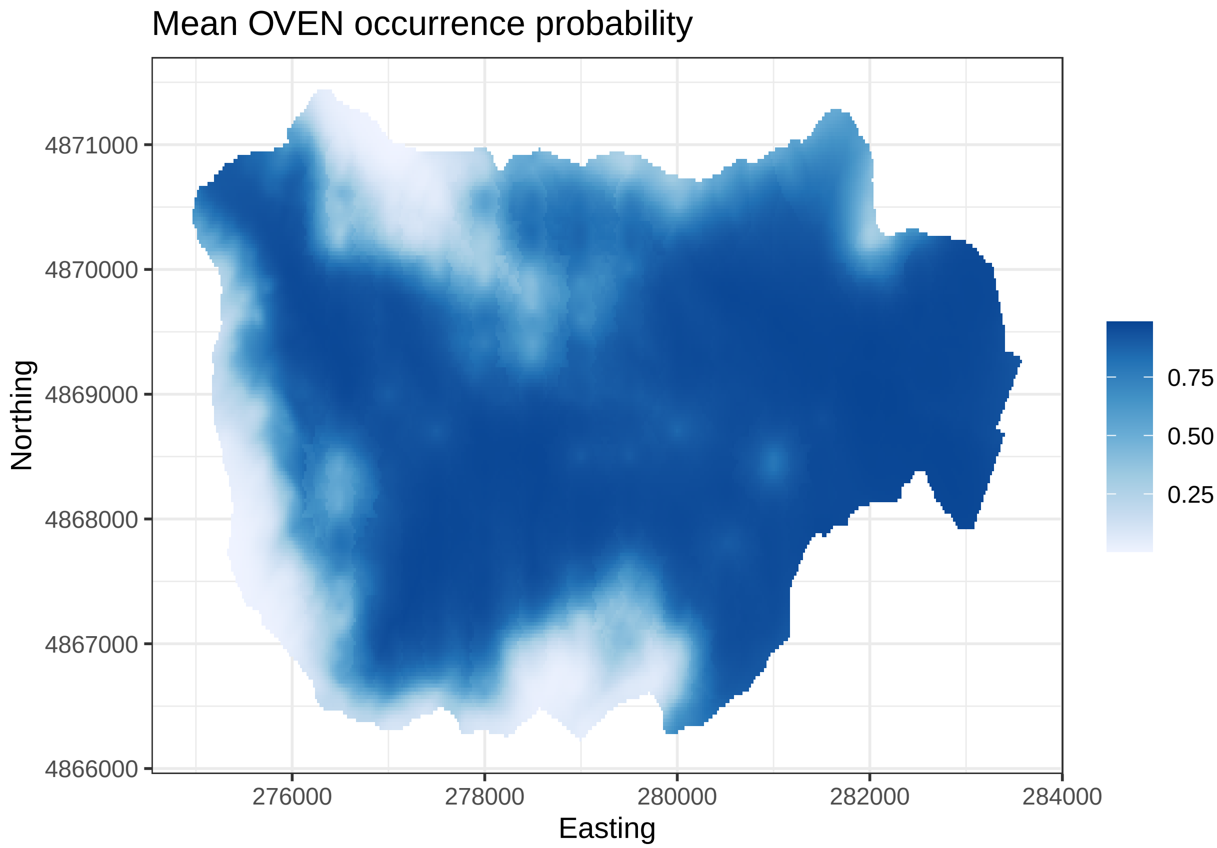
We see the WAIC value for the spatial model is smaller than that of the nonspatial model, indicating that incorporation of the spatial random effects yields improvement in predictive performance.

k-fold cross-validation is accomplished by specifying the `k.fold` argument in `spPGOcc` just as we saw in `PGOcc`. See Section 2.5.

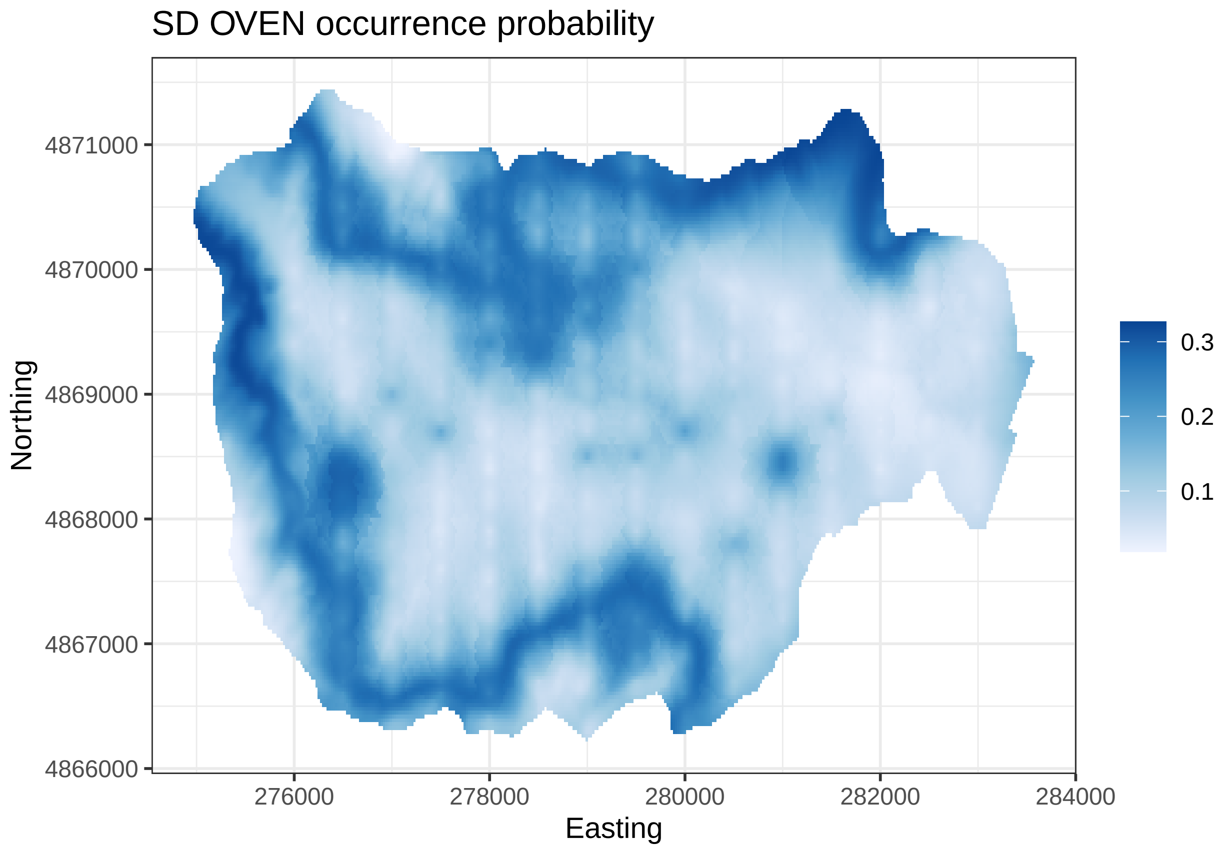
3.6 Prediction

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.

```
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()
```



Comparing this to the non-spatial occupancy model, the spatial model appears to identify areas in HBEF with low OVEN occurrence that are not captured in the non-spatial model. We will resist trying to hypothesize what environmental factors could lead to these patterns.

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 $j = 1, \dots, J$ and $i = 1, \dots, N$. 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 $k = 1, \dots, K_j$ 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 Pólya-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 Pólya-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.

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

```
List of 4
 $ y       : num [1:12, 1:373, 1:3] 0 0 0 0 0 1 0 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] 475 494 546 587 588 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "Elevation"
 $ det.covs:List of 2
  ..$ day: num [1:373, 1:3] 156 156 156 156 156 156 156 156 156 156 ...
  ..$ tod: num [1:373, 1:3] 330 346 369 386 409 425 447 463 482 499 ...
 $ 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] "X" "Y"
```

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 either vectors of length corresponding to the number of community-level detection or occurrence parameters in the model (including the intercepts) or a single value if all parameters are assigned the same starting values. Starting values for species level parameters are either matrices with the number of rows indicating the number of species, and each column corresponding to a different regression parameter, or a single value if the same starting value is used for all species and parameters. 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.

```r
N <- dim(hbef2015$y)[1]
ms.starting <- list(alpha.comm = 0,
                    beta.comm = 0,
                    beta = 0,
                    alpha = 0,
                    tau.sq.beta = 1,
                    tau.sq.alpha = 1,
                    z = apply(hbef2015$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. Values can be specified individually for each parameter or a single value if the same prior is assigned to all parameters of a given type.

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 = 0, var = 2.72),
                  alpha.comm.normal = list(mean = 0, var = 2.72),
                  tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                  tau.sq.alpha.ig = list(a = 0.1, b = 0.1))
```

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 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
160.764   0.071 160.847
```

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

```r
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)          -1.1263 -0.0667  0.4755  1.0671 2.0858
scale(Elevation)     -0.8690 -0.0349  0.2680  0.6040 1.2788
I(scale(Elevation)^2) -0.7148 -0.3676 -0.2230 -0.0784 0.2583

Occurrence Variances:
                       2.5%    25%    50%     75%   97.5%
(Intercept)          4.2615 6.9157 9.3935 12.9425 27.3294
scale(Elevation)     1.0813 1.9142 2.6694  3.8713  7.8775
I(scale(Elevation)^2) 0.1459 0.3075 0.4489  0.6894  1.5922

Detection Means:
                  2.5%     25%     50%     75%  97.5%
(Intercept)    -1.6763 -1.1007 -0.8068 -0.5210 0.1346
scale(day)     -0.1267 -0.0025  0.0524  0.1144 0.2238
scale(tod)     -0.1824 -0.0863 -0.0365  0.0126 0.1169
I(scale(day)^2) -0.1891 -0.0789 -0.0188  0.0342 0.1464

Detection Variances:
                 2.5%    25%    50%    75%  97.5%
(Intercept)    0.9754 1.7719 2.4418 3.5571 7.2754
scale(day)     0.0248 0.0422 0.0573 0.0838 0.1665
scale(tod)     0.0163 0.0304 0.0431 0.0632 0.1431
I(scale(day)^2) 0.0178 0.0333 0.0458 0.0665 0.1452

----------------------------------------
    Species Level
----------------------------------------
Occurrence:
```

	2.5%	25%	50%	75%	97.5%
(Intercept)-AMRE	-4.7286	-2.7747	-1.5670	0.2872	4.1895
(Intercept)-BAWW	0.0470	1.0352	1.8514	2.7824	5.1735
(Intercept)-BHVI	0.3511	2.2727	3.9925	5.4993	8.4476
(Intercept)-BLBW	1.7098	2.1586	2.4449	2.8522	4.0194
(Intercept)-BLPW	-6.0497	-4.9780	-4.5872	-4.2066	-3.5614
(Intercept)-BTBW	2.8162	3.3160	3.5868	3.9138	4.6769
(Intercept)-BTNW	1.8698	2.2053	2.3849	2.5896	3.0953
(Intercept)-CAWA	-3.2454	-2.6069	-2.2800	-1.9385	-1.1790
(Intercept)-MAWA	-2.2655	-1.8946	-1.7060	-1.5225	-1.1851
(Intercept)-NAWA	-3.5633	-2.8933	-2.5448	-2.2197	-1.4491
(Intercept)-OVEN	1.6751	1.9758	2.1545	2.3426	2.7554
(Intercept)-REVI	2.4096	2.8390	3.1409	3.4950	4.2584
scale(Elevation)-AMRE	-0.5501	0.4786	1.1077	1.7476	4.2415
scale(Elevation)-BAWW	-3.6093	-1.1517	-0.5927	-0.2164	0.5407
scale(Elevation)-BHVI	-1.9934	-0.4802	0.2174	0.9159	2.6255
scale(Elevation)-BLBW	-0.9006	-0.5391	-0.4067	-0.2838	-0.0686
scale(Elevation)-BLPW	1.7167	2.2353	2.5966	3.0522	4.1842
scale(Elevation)-BTBW	-0.8470	-0.6091	-0.4871	-0.3829	-0.1777
scale(Elevation)-BTNW	0.2367	0.4761	0.6236	0.8041	1.3459
scale(Elevation)-CAWA	0.8242	1.4663	1.8226	2.2157	3.1849
scale(Elevation)-MAWA	1.1028	1.3799	1.5478	1.7435	2.1106
scale(Elevation)-NAWA	0.5825	0.8778	1.1237	1.3827	1.9542
scale(Elevation)-OVEN	-2.2822	-1.8444	-1.6315	-1.4674	-1.2395
scale(Elevation)-REVI	-3.9134	-2.7133	-2.1746	-1.6932	-1.1409
I(scale(Elevation)^2)-AMRE	-1.9715	-1.0384	-0.6321	-0.2580	0.5229
I(scale(Elevation)^2)-BAWW	-1.9249	-1.2858	-0.9920	-0.7405	-0.0225
I(scale(Elevation)^2)-BHVI	-0.7948	-0.1216	0.2669	0.6822	1.7166
I(scale(Elevation)^2)-BLBW	-0.9806	-0.6894	-0.5596	-0.4641	-0.2397
I(scale(Elevation)^2)-BLPW	0.0007	0.4950	0.7441	0.9888	1.4161
I(scale(Elevation)^2)-BTBW	-1.3782	-1.1372	-1.0219	-0.9172	-0.7398
I(scale(Elevation)^2)-BTNW	-0.4213	-0.2428	-0.1399	-0.0286	0.1919
I(scale(Elevation)^2)-CAWA	-1.3283	-0.7764	-0.5286	-0.2597	0.2354
I(scale(Elevation)^2)-MAWA	-0.1864	0.1393	0.2943	0.4490	0.7608
I(scale(Elevation)^2)-NAWA	-0.2202	0.1204	0.2875	0.4438	0.7655
I(scale(Elevation)^2)-OVEN	-0.7734	-0.5577	-0.4456	-0.3111	-0.0514
I(scale(Elevation)^2)-REVI	-0.6350	-0.2569	0.0131	0.2759	0.8651

Detection:

	2.5%	25%	50%	75%	97.5%
(Intercept)-AMRE	-5.6013	-4.5364	-3.5757	-2.3013	-0.5733
(Intercept)-BAWW	-3.3176	-2.9405	-2.7546	-2.5162	-2.0163
(Intercept)-BHVI	-3.1927	-2.9111	-2.7753	-2.6345	-2.3454
(Intercept)-BLBW	-0.2827	-0.1227	-0.0414	0.0398	0.2052
(Intercept)-BLPW	-0.9611	-0.6276	-0.4422	-0.2750	0.0236
(Intercept)-BTBW	0.4240	0.5574	0.6316	0.7090	0.8727
(Intercept)-BTNW	0.3763	0.5096	0.5820	0.6599	0.8029
(Intercept)-CAWA	-2.4903	-1.8287	-1.4651	-1.1003	-0.5760
(Intercept)-MAWA	-1.2520	-0.9366	-0.7853	-0.6152	-0.3127
(Intercept)-NAWA	-2.4411	-1.8319	-1.4751	-1.1536	-0.5355
(Intercept)-OVEN	0.5641	0.7113	0.7997	0.8722	1.0118
(Intercept)-REVI	0.3366	0.4863	0.5467	0.6131	0.7477
scale(day)-AMRE	-0.4358	-0.1072	0.0383	0.1784	0.5177
scale(day)-BAWW	-0.0463	0.1182	0.2110	0.2985	0.4879

```
scale(day)-BHVI      -0.0180  0.1282  0.2100  0.2870  0.4447
scale(day)-BLBW      -0.3641 -0.2762 -0.2287 -0.1794 -0.0969
scale(day)-BLPW      -0.2452 -0.0420  0.0733  0.1874  0.3861
scale(day)-BTBW       0.1360  0.2241  0.2745  0.3182  0.4062
scale(day)-BTNW       0.0176  0.1010  0.1509  0.1913  0.2703
scale(day)-CAWA      -0.3883 -0.1456 -0.0285  0.0949  0.3049
scale(day)-MAWA      -0.1343  0.0295  0.1114  0.1925  0.3565
scale(day)-NAWA      -0.3373 -0.0962  0.0149  0.1350  0.3621
scale(day)-OVEN      -0.2207 -0.1240 -0.0732 -0.0254  0.0682
scale(day)-REVI      -0.2118 -0.1200 -0.0744 -0.0291  0.0666
scale(tod)-AMRE      -0.4327 -0.1570 -0.0183  0.1202  0.3911
scale(tod)-BAWW      -0.4646 -0.2705 -0.1723 -0.0850  0.0740
scale(tod)-BHVI      -0.2702 -0.1179 -0.0412  0.0246  0.1597
scale(tod)-BLBW      -0.0654  0.0114  0.0561  0.1004  0.1874
scale(tod)-BLPW      -0.1716  0.0167  0.1115  0.2115  0.3941
scale(tod)-BTBW      -0.1698 -0.0821 -0.0377  0.0054  0.0830
scale(tod)-BTNW      -0.0996 -0.0080  0.0372  0.0809  0.1622
scale(tod)-CAWA      -0.5704 -0.3188 -0.2089 -0.0896  0.0959
scale(tod)-MAWA      -0.2010 -0.0544  0.0253  0.1013  0.2512
scale(tod)-NAWA      -0.3920 -0.1720 -0.0729  0.0384  0.2537
scale(tod)-OVEN      -0.1855 -0.0968 -0.0505  0.0025  0.0937
scale(tod)-REVI      -0.2002 -0.1204 -0.0766 -0.0329  0.0567
I(scale(day)^2)-AMRE -0.5820 -0.2519 -0.1018  0.0306  0.2697
I(scale(day)^2)-BAWW -0.3341 -0.1336 -0.0292  0.0657  0.2286
I(scale(day)^2)-BHVI -0.1837 -0.0308  0.0577  0.1381  0.3079
I(scale(day)^2)-BLBW -0.3271 -0.2236 -0.1708 -0.1192 -0.0153
I(scale(day)^2)-BLPW -0.1479  0.0723  0.1878  0.3150  0.5919
I(scale(day)^2)-BTBW -0.2237 -0.1110 -0.0661 -0.0082  0.0953
I(scale(day)^2)-BTNW -0.2104 -0.1104 -0.0585 -0.0076  0.0861
I(scale(day)^2)-CAWA -0.4030 -0.1406 -0.0309  0.0930  0.3015
I(scale(day)^2)-MAWA -0.2563 -0.0846  0.0080  0.0990  0.2815
I(scale(day)^2)-NAWA -0.5113 -0.2402 -0.1364 -0.0030  0.1860
I(scale(day)^2)-OVEN -0.1426 -0.0389  0.0201  0.0811  0.1925
I(scale(day)^2)-REVI -0.1069 -0.0172  0.0349  0.0891  0.1863
```

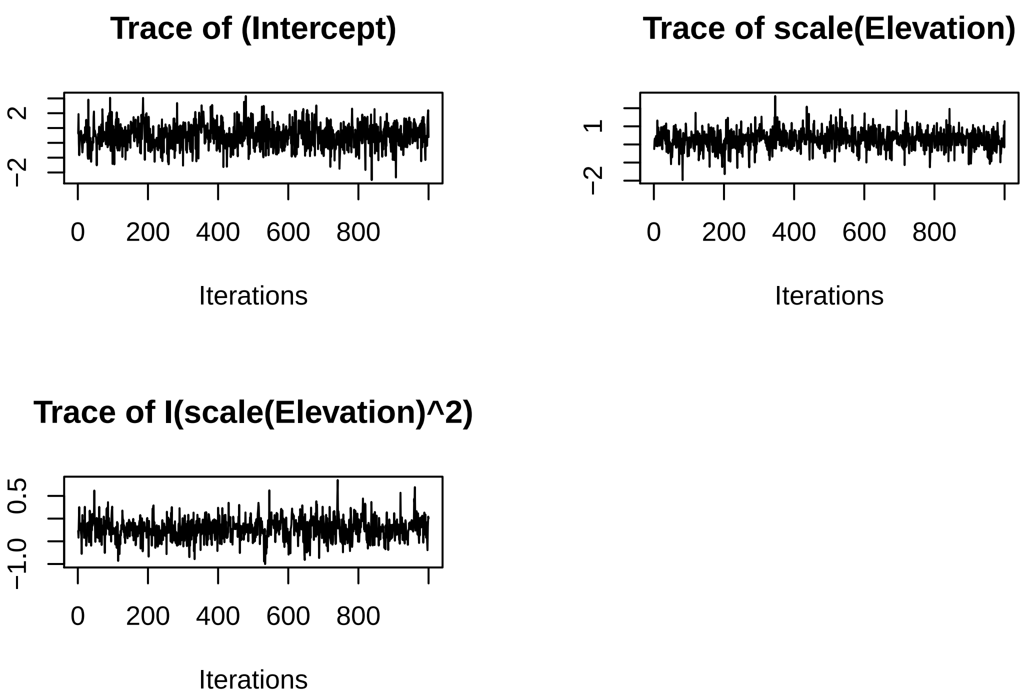
```r
# Or
# summary(out.ms, 'both')
```

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.

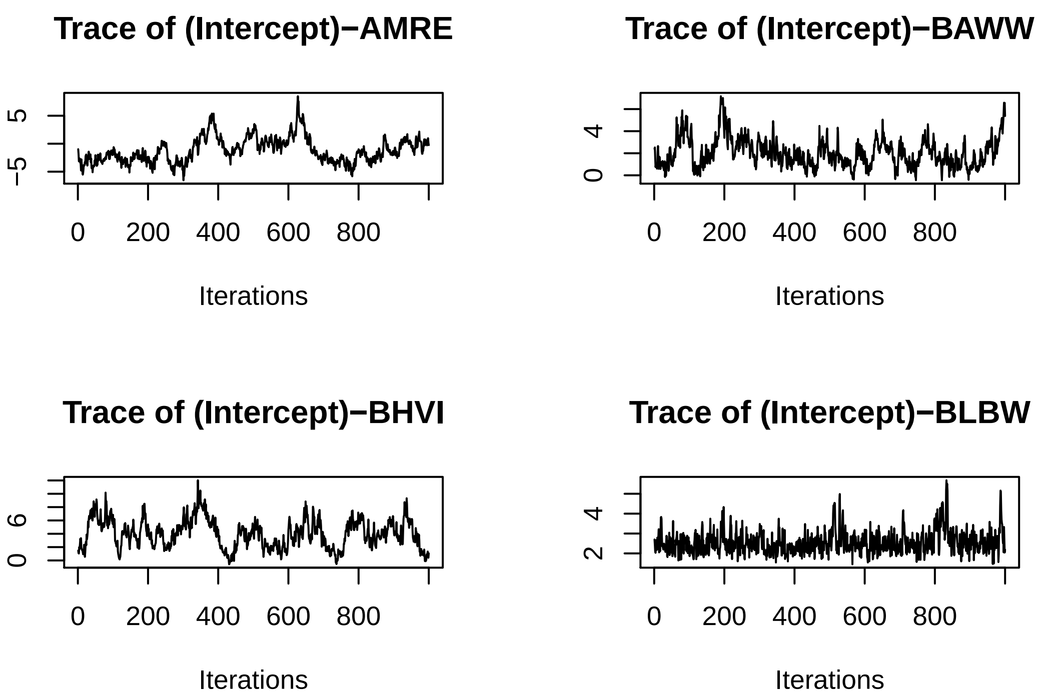
4.3 Convergence diagnostics

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.

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




```r
# 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 should 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'`).

```r
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"
```

```r
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:  0.3685

----------------------------------------
    Species Level
----------------------------------------
AMRE Bayesian p-value: 0.602
BAWW Bayesian p-value: 0.653
BHVI Bayesian p-value: 0.573
BLBW Bayesian p-value: 0.097
BLPW Bayesian p-value: 0.615
BTBW Bayesian p-value: 0.218
BTNW Bayesian p-value: 0.04
CAWA Bayesian p-value: 0.48
MAWA Bayesian p-value: 0.447
NAWA Bayesian p-value: 0.567
OVEN Bayesian p-value: 0.045
REVI Bayesian p-value: 0.085
```

```
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 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 and k-fold cross-validation

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

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

```
       elpd          pD        WAIC
-4531.39371    65.12906  9193.04555
```

k-fold cross-validation is again accomplished using the `k.fold` argument as shown in Section 2.5. For multispecies occupancy models, using multiple threads can greatly reduce the time needed for k-fold cross-validation, so we encourage the use of multiple threads if such computing power is readily available. Using up to k threads will generally involve substnatial decreases in run time. For multispecies models, a separate deviance measure is reported for each species as a measure of predictive capacity, allowing for comparisons across multiple models for individual species, as well as for the entire community (by summing all species-specific values).

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 can predict across the entire HBEF for all twelve species using the elevation data stored in `hbefElev`.

```
elev.pred <- (hbefElev$val - mean(ovenHBEF$occ.covs[, 1])) / sd(ovenHBEF$occ.covs[, 1])
X.0 <- cbind(1, elev.pred, elev.pred^2)
out.ms.pred <- predict(out.ms, X.0)
```

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 $\boldsymbol{\theta}_i = \{\sigma^2_i, \phi_i, \nu_i\}$, where σ^2_i 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 Pólya-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 of 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,
          k.fold, k.fold.threads = 1, k.fold.seed, ...)
```

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 <- ~ scale(Elevation) + I(scale(Elevation)^2)
det.ms.sp.formula <- ~ scale(day) + scale(tod) + I(scale(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) or as a single value that is used for all species, 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$coords)
# Exponential covariance model
cov.model <- "exponential"
ms.starting <- list(alpha.comm = 0,
                    beta.comm = 0,
                    beta = 0,
                    alpha = 0,
                    tau.sq.beta = 1,
                    tau.sq.alpha = 1,
                    z = apply(hbef2015$y, c(1, 2), max, na.rm = TRUE),
                    sigma.sq = 2,
                    phi = 3 / mean(dist.hbef),
                    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. If the same prior is used for all species, both elements can be specified as single values. 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. If the same prior is used for all species, both the lower and upper bounds can be specified as single values. 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 = 0, var = 2.72),
                  alpha.comm.normal = list(mean = 0, var = 2.72),
                  tau.sq.beta.ig = list(a = 0.1, b = 0.1),
                  tau.sq.alpha.ig = list(a = 0.1, b = 0.1),
                  sigma.sq.ig = list(a = 2, b = 2),
                  phi.unif = list(a = 3 / max.dist, b = 3 / min.dist))
```

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)
```

```r
n.omp.threads <- 1
# Values for reporting
verbose <- TRUE
n.report <- 100
```

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.

```r
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: 100 of 400, 25.00%
	parameter	acceptance	tuning
	phi[0]		56.0		0.51523
	phi[1]		20.0		0.58092
	phi[2]		48.0		0.75341
	phi[3]		8.0		0.64201
	phi[4]		20.0		0.39727
	phi[5]		44.0		0.37413
	phi[6]		32.0		0.49502
	phi[7]		32.0		0.56941
	phi[8]		44.0		0.31881
	phi[9]		24.0		0.69548
	phi[10]		36.0		0.30631
	phi[11]		44.0		0.39727
Batch: 200 of 400, 50.00%
	parameter	acceptance	tuning
	phi[0]		56.0		0.86663
	phi[1]		52.0		0.88413
	phi[2]		68.0		0.81616
	phi[3]		20.0		0.69548
	phi[4]		64.0		0.38169
	phi[5]		48.0		0.31881
	phi[6]		36.0		0.46620
	phi[7]		20.0		0.65498
	phi[8]		44.0		0.30025
	phi[9]		64.0		0.93881
	phi[10]		36.0		0.30025
	phi[11]		68.0		0.41348
Batch: 300 of 400, 75.00%
	parameter	acceptance	tuning
	phi[0]		80.0		0.65498
	phi[1]		24.0		1.31897
	phi[2]		24.0		1.05850
	phi[3]		56.0		0.75341
	phi[4]		60.0		0.35234
	phi[5]		48.0		0.39727
	phi[6]		44.0		0.44792
	phi[7]		28.0		0.72387
	phi[8]		48.0		0.30025
	phi[9]		40.0		1.37280
	phi[10]		40.0		0.29430
	phi[11]		56.0		0.40529
Batch: 400 of 400, 100.00%
```

```r
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)          -1.6733 -0.2344  0.4865  1.2638 2.6583
scale(Elevation)     -1.1646 -0.1560  0.3153  0.8156 1.6802
I(scale(Elevation)^2) -1.0023 -0.4962 -0.2723 -0.0157 0.5474

Occurrence Variances:
                      2.5%     25%     50%     75%   97.5%
(Intercept)          7.5702 13.9157 19.4473 26.6172 56.1001
scale(Elevation)     2.0226  3.8410  5.6262  8.5428 19.3087
I(scale(Elevation)^2) 0.3096  0.7036  1.0579  1.6145  3.7104

Detection Means:
                 2.5%     25%     50%     75%  97.5%
(Intercept)     -1.7548 -1.0407 -0.7334 -0.4475 0.1588
scale(day)      -0.1089  0.0040  0.0588  0.1173 0.2327
scale(tod)      -0.1884 -0.0844 -0.0353  0.0067 0.1032
I(scale(day)^2) -0.1992 -0.0765 -0.0197  0.0312 0.1560

Detection Variances:
                 2.5%    25%    50%    75%  97.5%
(Intercept)     0.8952 1.6682 2.4532 3.5272 7.4632
scale(day)      0.0220 0.0404 0.0550 0.0785 0.1709
scale(tod)      0.0166 0.0292 0.0410 0.0573 0.1311
I(scale(day)^2) 0.0180 0.0326 0.0476 0.0698 0.1377

----------------------------------------
    Species Level
----------------------------------------
Occurrence:
                          2.5%     25%     50%     75%   97.5%
(Intercept)-AMRE        -6.1197 -3.7032 -2.4008 -0.1150  8.3935
(Intercept)-BAWW        -0.0002  1.6050  2.8636  4.5890  7.6568
(Intercept)-BHVI        -0.2752  1.0960  2.8800  6.1983 11.9750
(Intercept)-BLBW         2.0392  2.6039  3.0065  3.5657  6.0712
(Intercept)-BLPW        -8.6032 -6.5617 -5.7830 -5.2256 -4.3050
(Intercept)-BTBW         3.4582  4.4134  5.2446  6.4234  8.7225
(Intercept)-BTNW         2.2861  2.8946  3.4289  4.1013  6.1120
(Intercept)-CAWA        -4.5784 -3.5158 -2.9814 -2.4212 -0.7652
(Intercept)-MAWA        -6.3492 -4.3260 -3.5774 -2.8449 -1.0716
(Intercept)-NAWA        -5.8599 -3.8830 -3.3131 -2.8367 -1.8631
(Intercept)-OVEN         2.2341  3.2699  3.9554  4.7404  6.6231
(Intercept)-REVI         2.9265  3.7445  4.4265  5.1804  6.7901
scale(Elevation)-AMRE   -0.4958  0.6061  1.3165  2.2462  5.2247
```

```
scale(Elevation)-BAWW      -5.8302 -1.7921 -0.9639 -0.4251  1.0172
scale(Elevation)-BHVI      -2.8265 -0.4203  0.2686  1.1859  3.3456
scale(Elevation)-BLBW      -1.1512 -0.6326 -0.4508 -0.2753  0.0589
scale(Elevation)-BLPW       2.0103  2.5924  3.0184  3.6484  5.1627
scale(Elevation)-BTBW      -1.5545 -0.8387 -0.6230 -0.4343 -0.0525
scale(Elevation)-BTNW       0.2238  0.5893  0.8480  1.1406  1.9730
scale(Elevation)-CAWA       1.0175  1.8225  2.3467  2.9019  4.4863
scale(Elevation)-MAWA       1.6146  2.3037  2.8054  3.4042  4.7683
scale(Elevation)-NAWA       0.6195  1.0740  1.3981  1.7979  2.8274
scale(Elevation)-OVEN      -6.1641 -3.9158 -3.0726 -2.5388 -1.7977
scale(Elevation)-REVI      -5.9773 -3.6356 -2.8755 -2.2529 -1.3698
I(scale(Elevation)^2)-AMRE -2.8322 -1.3782 -0.7394 -0.1381  1.9305
I(scale(Elevation)^2)-BAWW -2.8554 -1.6735 -1.1999 -0.8022  0.7201
I(scale(Elevation)^2)-BHVI -0.9209  0.1106  0.5975  1.1762  2.8657
I(scale(Elevation)^2)-BLBW -1.2965 -0.8369 -0.6789 -0.5409 -0.2734
I(scale(Elevation)^2)-BLPW  0.0461  0.7454  1.0607  1.3700  2.0051
I(scale(Elevation)^2)-BTBW -2.6370 -1.8452 -1.4918 -1.2477 -0.9259
I(scale(Elevation)^2)-BTNW -0.7359 -0.3881 -0.2196 -0.0622  0.3524
I(scale(Elevation)^2)-CAWA -1.9428 -1.1335 -0.7145 -0.3880  0.1792
I(scale(Elevation)^2)-MAWA -0.2038  0.2854  0.6027  0.9293  1.5778
I(scale(Elevation)^2)-NAWA -0.2392  0.1565  0.3915  0.6105  1.1628
I(scale(Elevation)^2)-OVEN -1.9982 -1.1626 -0.8582 -0.5680  0.2435
I(scale(Elevation)^2)-REVI -0.9289 -0.3452  0.0208  0.4338  1.4111

Detection:
                       2.5%     25%     50%     75%   97.5%
(Intercept)-AMRE     -5.7510 -4.5214 -3.0779 -1.9729 -0.5325
(Intercept)-BAWW     -3.4287 -3.0482 -2.8332 -2.5257 -2.0468
(Intercept)-BHVI     -3.1382 -2.8510 -2.6848 -2.4945 -2.0595
(Intercept)-BLBW     -0.2461 -0.1054 -0.0284  0.0569  0.2045
(Intercept)-BLPW     -0.9257 -0.5711 -0.3996 -0.2272  0.1016
(Intercept)-BTBW      0.4343  0.5612  0.6341  0.7019  0.8556
(Intercept)-BTNW      0.3805  0.5203  0.6006  0.6777  0.8197
(Intercept)-CAWA     -2.6791 -1.7596 -1.3905 -1.0750 -0.5056
(Intercept)-MAWA     -1.0817 -0.8387 -0.6846 -0.5245 -0.2206
(Intercept)-NAWA     -2.4300 -1.8257 -1.4918 -1.1546 -0.5715
(Intercept)-OVEN      0.5926  0.7449  0.8201  0.8923  1.0500
(Intercept)-REVI      0.3166  0.4656  0.5453  0.6183  0.7630
scale(day)-AMRE      -0.4143 -0.1048  0.0507  0.2094  0.5474
scale(day)-BAWW      -0.0535  0.1216  0.2231  0.3201  0.4901
scale(day)-BHVI      -0.0325  0.1122  0.2032  0.2809  0.4414
scale(day)-BLBW      -0.3618 -0.2666 -0.2280 -0.1811 -0.0926
scale(day)-BLPW      -0.2373 -0.0347  0.0621  0.1641  0.3470
scale(day)-BTBW       0.1403  0.2213  0.2664  0.3162  0.4033
scale(day)-BTNW       0.0192  0.1039  0.1492  0.1930  0.2889
scale(day)-CAWA      -0.3842 -0.1357 -0.0263  0.0900  0.3198
scale(day)-MAWA      -0.1345  0.0300  0.1107  0.1914  0.3577
scale(day)-NAWA      -0.3292 -0.0973  0.0187  0.1329  0.3657
scale(day)-OVEN      -0.2150 -0.1244 -0.0777 -0.0295  0.0694
scale(day)-REVI      -0.2089 -0.1210 -0.0735 -0.0284  0.0639
scale(tod)-AMRE      -0.4420 -0.1661 -0.0285  0.1074  0.3945
scale(tod)-BAWW      -0.4533 -0.2604 -0.1689 -0.0823  0.0862
scale(tod)-BHVI      -0.2811 -0.1284 -0.0557  0.0193  0.1645
scale(tod)-BLBW      -0.0724  0.0118  0.0567  0.1048  0.1874
```

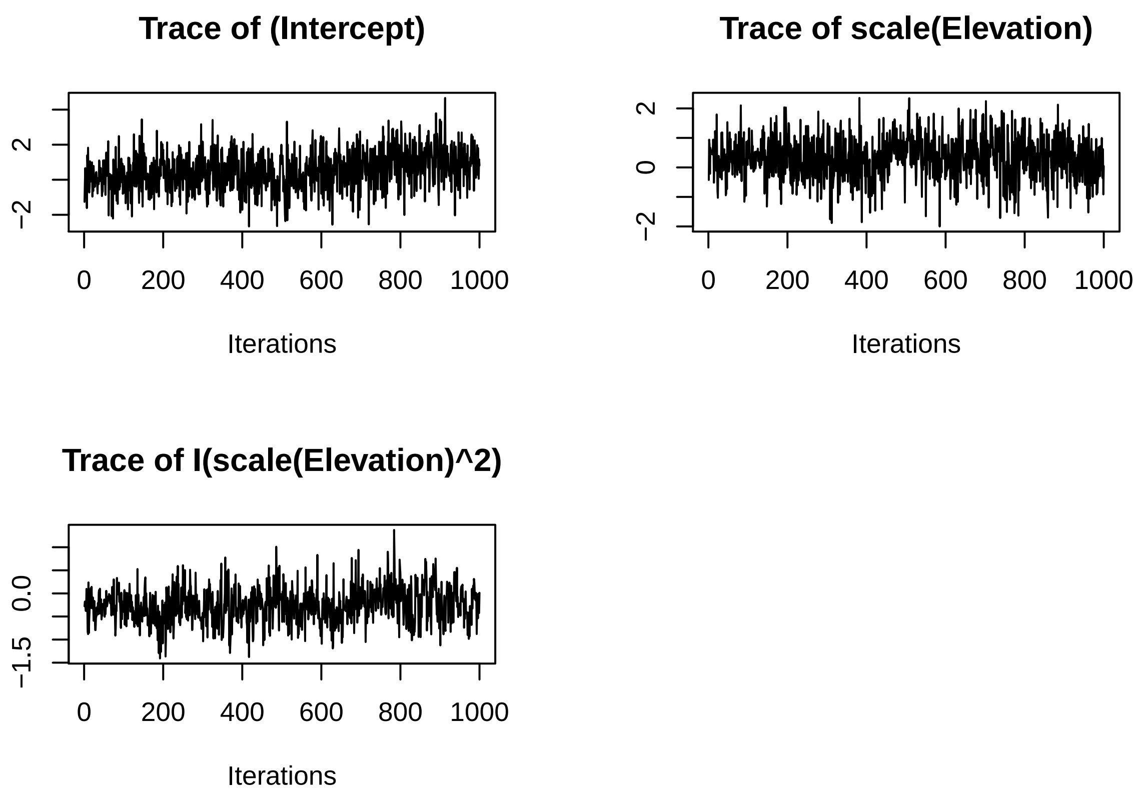
```
scale(tod)-BLPW      -0.1790  0.0027  0.1020  0.2027  0.4021
scale(tod)-BTBW      -0.1698 -0.0809 -0.0387  0.0070  0.0896
scale(tod)-BTNW      -0.0860 -0.0047  0.0379  0.0795  0.1548
scale(tod)-CAWA      -0.5855 -0.3120 -0.1865 -0.0886  0.0976
scale(tod)-MAWA      -0.1988 -0.0625  0.0190  0.0941  0.2346
scale(tod)-NAWA      -0.3969 -0.1710 -0.0751  0.0316  0.2337
scale(tod)-OVEN      -0.1771 -0.0920 -0.0423  0.0028  0.0884
scale(tod)-REVI      -0.1954 -0.1239 -0.0818 -0.0370  0.0527
I(scale(day)^2)-AMRE -0.6054 -0.2350 -0.0906  0.0538  0.3345
I(scale(day)^2)-BAWW -0.3297 -0.1319 -0.0306  0.0668  0.2529
I(scale(day)^2)-BHVI -0.2131 -0.0457  0.0449  0.1357  0.3344
I(scale(day)^2)-BLBW -0.3261 -0.2224 -0.1724 -0.1159 -0.0218
I(scale(day)^2)-BLPW -0.1573  0.0806  0.1945  0.3266  0.5715
I(scale(day)^2)-BTBW -0.2224 -0.1118 -0.0628 -0.0095  0.0910
I(scale(day)^2)-BTNW -0.2054 -0.1165 -0.0656 -0.0103  0.0801
I(scale(day)^2)-CAWA -0.3967 -0.1479 -0.0455  0.0831  0.3249
I(scale(day)^2)-MAWA -0.2586 -0.0925  0.0082  0.0925  0.2878
I(scale(day)^2)-NAWA -0.4857 -0.2529 -0.1239  0.0050  0.2150
I(scale(day)^2)-OVEN -0.1504 -0.0448  0.0165  0.0742  0.1858
I(scale(day)^2)-REVI -0.1131 -0.0172  0.0383  0.0877  0.1931

Covariance:
                2.5%    25%    50%     75%   97.5%
sigma.sq-AMRE 0.3214 0.7472 1.2352  2.3460  8.7653
sigma.sq-BAWW 0.3360 0.7042 1.0903  1.6242  7.6974
sigma.sq-BHVI 0.4171 0.8606 1.3927  2.3581  4.5243
sigma.sq-BLBW 0.4189 0.9443 1.5081  2.6153 17.0094
sigma.sq-BLPW 0.4570 1.0567 1.8973  3.4607  9.0676
sigma.sq-BTBW 0.5176 1.5105 2.9746  6.6109 15.1913
sigma.sq-BTNW 0.5065 1.6592 3.4425  5.7928 15.5268
sigma.sq-CAWA 0.3948 0.8926 1.6511  3.1361  8.0138
sigma.sq-MAWA 2.1921 5.4458 8.5273 13.0186 30.3788
sigma.sq-NAWA 0.3730 0.8900 1.5831  3.3953 15.0251
sigma.sq-OVEN 2.3748 5.8456 8.5304 12.8892 36.4719
sigma.sq-REVI 0.5083 1.2591 2.3115  4.6832  9.4874
phi-AMRE      0.0021 0.0077 0.0141  0.0218  0.0293
phi-BAWW      0.0041 0.0107 0.0175  0.0241  0.0292
phi-BHVI      0.0045 0.0092 0.0146  0.0214  0.0289
phi-BLBW      0.0022 0.0058 0.0113  0.0204  0.0286
phi-BLPW      0.0011 0.0023 0.0039  0.0076  0.0241
phi-BTBW      0.0007 0.0022 0.0049  0.0092  0.0252
phi-BTNW      0.0019 0.0040 0.0069  0.0139  0.0276
phi-CAWA      0.0009 0.0031 0.0085  0.0156  0.0284
phi-MAWA      0.0006 0.0014 0.0018  0.0024  0.0038
phi-NAWA      0.0048 0.0112 0.0176  0.0240  0.0291
phi-OVEN      0.0007 0.0015 0.0019  0.0026  0.0044
phi-REVI      0.0005 0.0019 0.0041  0.0081  0.0222
```

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.

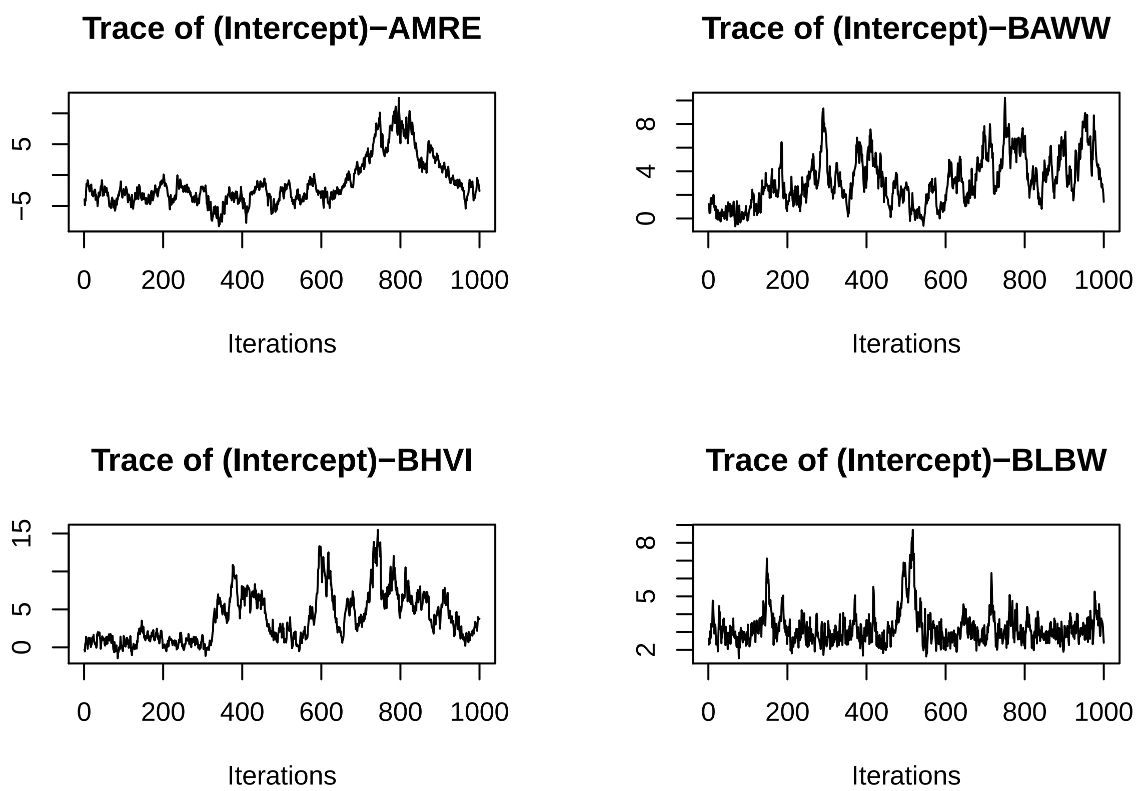
5.3 Convergence diagnostics

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:  0.5085833

----------------------------------------
    Species Level
----------------------------------------
AMRE Bayesian p-value: 0.715
BAWW Bayesian p-value: 0.526
BHVI Bayesian p-value: 0.58
BLBW Bayesian p-value: 0.287
BLPW Bayesian p-value: 0.387
BTBW Bayesian p-value: 0.489
BTNW Bayesian p-value: 0.487
CAWA Bayesian p-value: 0.556
MAWA Bayesian p-value: 0.614
NAWA Bayesian p-value: 0.607
OVEN Bayesian p-value: 0.378
REVI Bayesian p-value: 0.477
Fit statistic:  freeman-tukey
```

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.

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

```
      elpd         pD       WAIC 
-4191.3538   318.5708  9019.8491 
```

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

```
       elpd          pD        WAIC 
-4531.39371    65.12906  9193.04555 
```

The WAIC for the spatial model is smaller than that for the nonspatial model, indicating the species-specific spatial processes improve prediction across the entire community. However, in a complete analysis we should ensure the models fully converge before performing any model selection or comparison.

k-fold cross-validation proceeds using the `k.fold` argument as discussed in Section 2.5, returning a spearate scoring rule (deviance) for each species.

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.

```
elev.pred <- (hbefElev$val - mean(ovenHBEF$occ.covs[, 1])) / sd(ovenHBEF$occ.covs[, 1])
X.0 <- cbind(1, elev.pred, elev.pred^2)
coords.0 <- as.matrix(hbefElev[, c('Easting', 'Northing')])
out.sp.ms.pred <- predict(out.sp.ms, X.0, coords.0)
```

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 (Miller et al. 2019). 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 we should explicitly incorporate 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 $j = 1, \dots, J$. 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 $r = 1, \dots, R$ 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. Pólya-Gamma data augmentation is implemented analgous to previous models to yield an efficient implementation of integrated occupancy models.

6.2 Example data sources: Ovenbird occurrence in the White Mountain National Forest

To illustrate an integrated occupancy model, we will use two data sets that come from the White Mountain National Forest (WMNF) in New Hampshire, Maine, USA. Our goal is to model the occurrence of OVEN in the WMNF in 2015. Our first data source is the HBEF data set we have used to display all single data source models. Our second data source comes from the National Ecological Observatory Network (NEON) at Bartlett Experimental Forest (Barnett et al. 2019; National Ecological Observatory Network (NEON) 2021). The Barlett Forest and HBEF are both within the larger WMNF. Suppose we are interested in OVEN occurrence across the entire WMNF. By leveraging both data sources in a single integrated model, we will expand the range of covariates across which we can make reliable predictions, and may obtain results that are more indicative across the entire region of interest and not just a single data source location (Doser et al. 2021). In this particular case, there is no overlap between the two data sources (i.e., Bartlett Forest and HBEF do not overlap spatially). However, the integrated occupancy models fit by `spOccupancy` can integrate data sources with no overlap, partial overlap, or complete overlap.

The NEON data are provided along with `spOccupancy` in the `neon2015` list. We load the NEON data along with the HBEF data below

```
data(hbef2015)
data(neon2015)
str(neon2015)
```

```
List of 4
 $ y       : num [1:12, 1:80, 1:3] 0 0 0 0 0 0 1 0 0 0 ...
 $ occ.covs: num [1:80, 1] 390 425 443 382 441 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "Elevation"
 $ det.covs:List of 2
  ..$ day: num [1:80] 169 169 169 169 169 169 169 169 169 170 ...
  ..$ tod: int [1:80] 8 8 6 7 8 6 7 7 7 8 ...
 $ coords  : num [1:80, 1:2] 318472 318722 318972 318472 318722 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:80] "1" "2" "3" "4" ...
  .. ..$ : chr [1:2] "X" "Y"
```

Details on the NEON data set are provided in the package documentation as well as Doser et al. (2021). The NEON data are collected at 80 point count sites in Bartlett Forest using a removal protocol with three time periods, resulting in replicated detection-nondetection data that can be used in an occupancy modeling framework. The `neon2015` list, like the `hbef2015` object, contains the detection-nondetection data for 12 foliage-gleaning bird species (`y`), occurrence covariates stored in `occ.covs`, detection covariates stored in `det.covs`, and the coordinates of the 80 point count locations stored in `coords`. Below we subset the detection-nondetection data in both data sources to solely work with OVEN.

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

```r
ovenNEON <- neon2015
ovenNEON$y <- ovenNEON$y[sp.names == "OVEN", , ]
table(ovenHBEF$y)
```

```
  0   1
518 588
```

```r
table(ovenNEON$y)
```

```
 0  1
61 62
```

OVEN is observed in a little over half of the possible site/replicate combinations in both of the data sources.

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:

```r
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,
         k.fold, k.fold.threads = 1, k.fold.seed, k.fold.data, ...)
```

The `data` argument contains the list of data elements necessary for fitting an integrated occupancy model. For nonspatial integrated occupancy models, `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 data source), `sites` (a list where each element consists of the site indices for the given data source.

The `ovenHBEF` and `ovenNEON` lists are currently formatted for use in single data source models and so we need to combine these data sources together. Perhaps the trickiest part of data integration is ensuring each point count location in each data source lines up with the correct geographical location where you want to determine the true presence/absence of the species of interest. In `spOccupancy`, most of this bookkeeping is done under the hood, but we will need to combine the two data sources together into a single list in which we are consistent about how the data sources are sorted. To accomplish this, we recommend first creating the occurrence covariates matrix for all data sources. Because our two data sources do not overlap spatially, this is relatively simple here as we can just use `rbind`.

```r
occ.covs.int <- rbind(ovenHBEF$occ.covs, ovenNEON$occ.covs)
str(occ.covs.int)
```

```
 num [1:453, 1] 475 494 546 587 588 ...
 - attr(*, "dimnames")=List of 2
  ..$ : NULL
  ..$ : chr "Elevation"
```

Notice the order in which we placed these covariates: all covariate values for HBEF come first, followed by all covariates for NEON. We need to ensure we use this ordering for all objects in the `data` list. Next, we create the site indices stored in `sites`. `sites` should be a list with two elements (one for each data source), where each element consists of a vector that indicates the rows in `occ.covs` that correspond with the specific row of the detection-nondetection data for that data source. When the data sources sample distinct points (like in our current case), this is relatively straightforward as the indices simply correspond to how we ordered the points in `occ.covs`.

```r
sites.int <- list(hbef = 1:nrow(ovenHBEF$occ.covs),
                  neon = 1:nrow(ovenNEON$occ.covs) + nrow(ovenHBEF$occ.covs))
str(sites.int)
```

```
List of 2
 $ hbef: int [1:373] 1 2 3 4 5 6 7 8 9 10 ...
 $ neon: int [1:80] 374 375 376 377 378 379 380 381 382 383 ...
```

Next we create the detection-nondetection data `y`. For integrated models in `spOccupancy`, `y` is a list of matrices, with each element containing the detection-nondetection matrix for the specific data source. Again, we must ensure that we place the data sources in the correct order.

```r
y.int <- list(hbef = ovenHBEF$y,
              neon = ovenNEON$y)
str(y.int)
```

```
List of 2
 $ hbef: num [1:373, 1:3] 1 1 0 1 0 0 1 0 1 1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "1" "2" "3"
 $ neon: num [1:80, 1:3] 1 1 0 1 1 0 1 1 0 1 ...
```

Lastly, we create the detection covariates `det.covs`. `det.covs` should be a list of the detection covariates from each individual data source. Because individual data source detection covariates are stored as lists for single data source models in `spOccupancy`, `det.covs` is now a list of lists for integrated occupancy models.

```r
det.covs.int <- list(hbef = ovenHBEF$det.covs,
             neon = ovenNEON$det.covs)
```

Finally, we package everything together into a single list, which we call `data.int`.

```r
data.int <- list(y = y.int,
                 occ.covs = occ.covs.int,
                 det.covs = det.covs.int,
                 sites = sites.int)
str(data.int)
```

```
List of 4
 $ y       :List of 2
  ..$ hbef: num [1:373, 1:3] 1 1 0 1 0 0 1 0 1 1 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:373] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:3] "1" "2" "3"
  ..$ neon: num [1:80, 1:3] 1 1 0 1 1 0 1 1 0 1 ...
 $ occ.covs: num [1:453, 1] 475 494 546 587 588 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr "Elevation"
 $ det.covs:List of 2
  ..$ hbef:List of 2
  .. ..$ day: num [1:373, 1:3] 156 156 156 156 156 156 156 156 156 156 ...
  .. ..$ tod: num [1:373, 1:3] 330 346 369 386 409 425 447 463 482 499 ...
  ..$ neon:List of 2
  .. ..$ day: num [1:80] 169 169 169 169 169 169 169 169 169 170 ...
  .. ..$ tod: int [1:80] 8 8 6 7 8 6 7 7 7 8 ...
 $ sites   :List of 2
```

```
  ..$ hbef: int [1:373] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ neon: int [1:80] 374 375 376 377 378 379 380 381 382 383 ...
```

We specify the occurrence and detection model formulas using the `occ.formula`, and `det.formula` arguments. The `occ.formula` remains unchanged from previous models, and we will specify occurrence of OVEN as a function of linear and quadratic elevation.

```
occ.formula.int <- ~ scale(Elevation) + I(scale(Elevation)^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 detection model for HBEF as a function of linear and quadratic day of survey as well as linear time of survey. In this case, we include the same covariates for the NEON model (although different coefficients are estimated for the two data sources). However, there is no requirement for the data sources to be a function of the same covariates.

```
det.formula.int = list(hbef = ~ scale(day) + scale(tod) + I(scale(day)^2),
                       neon = ~ scale(day) + scale(tod) + I(scale(day)^2))
```

Next we specify the starting values. Starting values are specified in a list with the following tags: `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 passed to `intPGOcc` as a list, with each element of the list corresponding to the starting detection parameter values for a given data source (which are specified either as a vector with a value for each parameter or a single value for all parameters).

```
# Total number of sites
J <- nrow(data.int$occ.covs)
starting.list <- list(alpha = list(0, 0),
                      beta = 0,
                      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.

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

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 <- 8000
n.burn <- 3000
n.thin <- 5
```

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 2000th iteration.

```
out.int <- intPGOcc(occ.formula = occ.formula.int,
                    det.formula = det.formula.int,
                    data = data.int,
```

```r
                    starting = starting.list,
                    n.samples = n.samples,
                    priors = prior.list,
                    n.omp.threads = 1,
                    verbose = TRUE,
                    n.report = 2000,
                    n.burn = n.burn,
                    n.thin = n.thin)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Integrated Occupancy Model with Polya-Gamma latent
variable fit with 453 sites.

Integrating 2 occupancy data sets.

Number of MCMC samples: 8000
Burn-in: 3000
Thinning Rate: 5
Total Posterior Samples: 1000


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

Sampling ...
Sampled: 2000 of 8000, 25.00%
-------------------------------------------------
Sampled: 4000 of 8000, 50.00%
-------------------------------------------------
Sampled: 6000 of 8000, 75.00%
-------------------------------------------------
Sampled: 8000 of 8000, 100.00%
```

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

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

```
Call:
intPGOcc(occ.formula = occ.formula.int, det.formula = det.formula.int,
    data = data.int, starting = starting.list, priors = prior.list,
    n.samples = n.samples, n.omp.threads = 1, verbose = TRUE,
    n.report = 2000, n.burn = n.burn, n.thin = n.thin)

Chain Information:
Total samples: 8000
Burn-in: 3000
Thin: 5
Total Posterior Samples: 1000

Occurrence:
                         2.5%     25%     50%     75%   97.5%
```

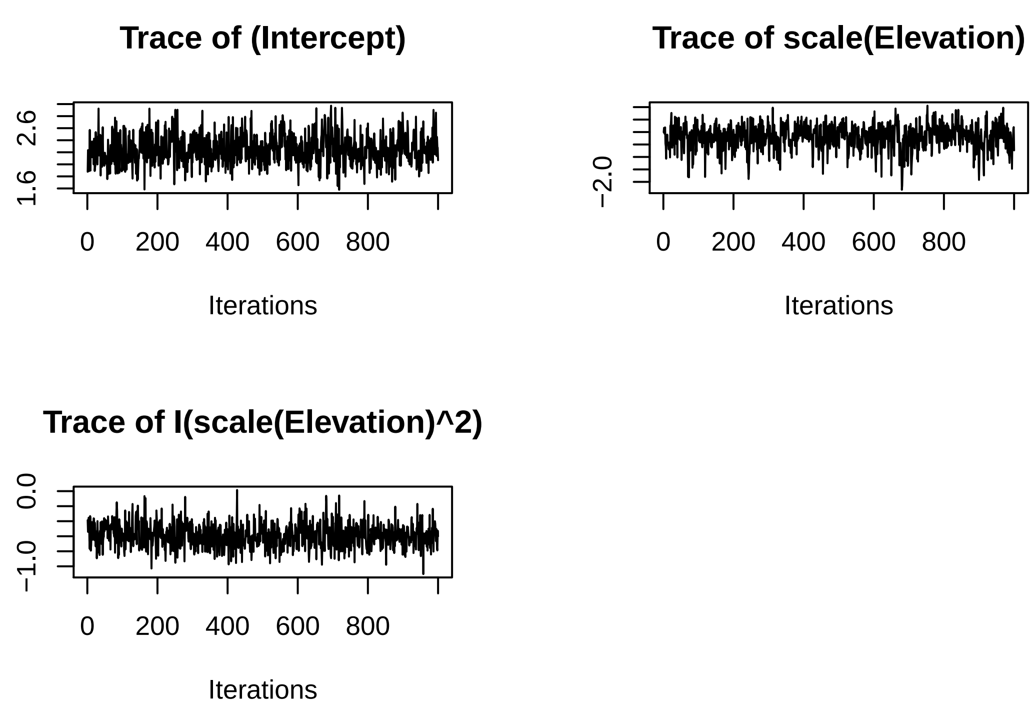
```
(Intercept)             1.8144  2.0733  2.2426  2.4063  2.7667
scale(Elevation)       -1.7597 -1.4024 -1.2564 -1.1478 -0.9461
I(scale(Elevation)^2) -0.8908 -0.6994 -0.5926 -0.4841 -0.2635

Data source 1 Detection:
                   2.5%     25%     50%     75%  97.5%
(Intercept)      0.5828  0.7258  0.8013  0.8853 1.0452
scale(day)      -0.2473 -0.1429 -0.0899 -0.0377 0.0564
scale(tod)      -0.1883 -0.0985 -0.0457  0.0083 0.1195
I(scale(day)^2) -0.1542 -0.0393  0.0310  0.0926 0.2121

Data source 2 Detection:
                   2.5%     25%     50%     75%   97.5%
(Intercept)      1.0405  1.6476  2.0408  2.4520  3.3220
scale(day)       0.2745  0.6775  0.9027  1.1388  1.6755
scale(tod)      -0.2841  0.5171  0.9137  1.2274  1.7552
I(scale(day)^2) -1.5204 -0.9924 -0.7686 -0.5651 -0.1567
```

The `summary` function for integrated models returns the detection parameters separately for each detection covariate. Looking at the occurrence parameters, we see fairly similar estimates to those from the single data source model using HBEF data only.

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.

```r
# 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: 8000
Burn-in: 3000
Thin: 5
Total Posterior Samples: 1000

Data Source 1

Bayesian p-value: 0.771
Fit statistic: freeman-tukey

Data Source 2

Bayesian p-value: 0.072
Fit statistic: freeman-tukey
```

The low Bayesian p-value for NEON suggests a potential lack of fit which we would explore in a complete analysis.

6.6 Model selection using WAIC and k-fold cross-validation

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 -633.37923 5.78042059 1278.319297
2   -4.55661 0.05771227    9.228644
```

k-fold cross-validation is implemented using the `k.fold` argument as we have seen in previous `spoccupancy` model functions. Cross-validation for models without multiple data sources is not as straightforward as single data source models, and we could envision splitting the data in multiple different ways to assess predictive performance, depending on the purpose of our comparison. In `spOccupancy`, we implement two approaches for cross-validation of integrated occupancy models. In our first approach, we hold out locations irrespective of what data source they came from. This results in a scoring rule (deviance) for each individual data source based on the hold out sites where that data source is sampled. More specifically, our first algorithm for K-fold cross-validation is:

1. Randomly split the total number of sites with at least one data source into K groups.
2. For each $k = 1, \ldots, K$, fit the model without the data at the sites in the kth group of hold-out locations.
3. Predict the detection-nondetection data at the locations in the kth hold out set.
4. Compute the deviance for each hold out data point.

5. Sum the deviance values separately for each data source to yield a scoring rule for each data source separately.

This form of k-fold cross-validation is applicable for model-selection between different integrated occupancy models. In other words, this approach can be used to compare models that integrate the same data sources but include different covariates in the occurrence and/or detection portion of the occupancy model. Using our example, we implement 4-fold cross-validation to compare the full integrated model with covariates to an intercept only integrated occupancy model. We do this using the `k.fold`, `k.fold.threads`, and `k.fold.seed` arguments as with previous `spOccupancy` models. Below we use the default values for `k.fold.threads` and `k.fold.seed`.

```
out.int.k.fold <- intPGOcc(occ.formula = occ.formula.int,
                           det.formula = det.formula.int,
                           data = data.int,
                           starting = starting.list,
                           n.samples = n.samples,
                           priors = prior.list,
                           n.omp.threads = 1,
                           verbose = TRUE,
                           n.report = 2000,
                           n.burn = n.burn,
                           n.thin = n.thin,
                           k.fold = 4)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Integrated Occupancy Model with Polya-Gamma latent
variable fit with 453 sites.

Integrating 2 occupancy data sets.

Number of MCMC samples: 8000
Burn-in: 3000
Thinning Rate: 5
Total Posterior Samples: 1000


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

Sampling ...
Sampled: 2000 of 8000, 25.00%
-------------------------------------------------
Sampled: 4000 of 8000, 50.00%
-------------------------------------------------
Sampled: 6000 of 8000, 75.00%
-------------------------------------------------
Sampled: 8000 of 8000, 100.00%
----------------------------------------
    Cross-validation
----------------------------------------

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

```r
out.int.k.fold.small <- intPGOcc(occ.formula = ~ 1,
                                 det.formula = list(hbef = ~ 1, neon = ~ 1),
                                 data = data.int,
                                 starting = starting.list,
                                 n.samples = n.samples,
                                 priors = prior.list,
                                 n.omp.threads = 1,
                                 verbose = TRUE,
                                 n.report = 2000,
                                 n.burn = n.burn,
                                 n.thin = n.thin,
                                 k.fold = 4)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Integrated Occupancy Model with Polya-Gamma latent
variable fit with 453 sites.

Integrating 2 occupancy data sets.

Number of MCMC samples: 8000
Burn-in: 3000
Thinning Rate: 5
Total Posterior Samples: 1000


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

Sampling ...
Sampled: 2000 of 8000, 25.00%
-------------------------------------------------
Sampled: 4000 of 8000, 50.00%
-------------------------------------------------
Sampled: 6000 of 8000, 75.00%
-------------------------------------------------
Sampled: 8000 of 8000, 100.00%
----------------------------------------
    Cross-validation
----------------------------------------

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

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

```
[1] 1508.9607  157.0867
```

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

```
[1] 1533.8793  205.5286
```

We see the deviance for the full model is lower for both data sources compared to the intercept only model. Alternatively, we may not wish to compare different integrated occupancy models together, but rather wish to assess whether or not data integration is necessary compared to using a single data source occupancy model. To accomplish this task, we can perform cross-validation with the integrated occupancy model using only a single data source as the hold out set, and then compare the deviance scoring rule to a scoring rule obtained from cross-validation with a single data source occupancy model. We accomplish this by using the argument `k.fold.data`. If `k.fold.data` is specified as an integer between 1 and the number of data sources integrated (in this case 2), only the data source corresponding to that integer will be used in the hold out and k-fold cross-validation process. If `k.fold.data` is not specified, k-fold cross-validation holds out data irrespective of the data sources at the given location. Here, we set `k.fold.data = 1` and compare the cross-validation results of the integrated model to the occupancy model using only HBEF we fit with `PGOcc` (which is stored in the `out.k.fold` object).

```
out.int.k.fold.hbef <- intPGOcc(occ.formula = occ.formula.int,
                                det.formula = det.formula.int,
                                data = data.int,
                                starting = starting.list,
                                n.samples = n.samples,
                                priors = prior.list,
                                n.omp.threads = 1,
                                verbose = TRUE,
                                n.report = 2000,
                                n.burn = n.burn,
                                n.thin = n.thin,
                                k.fold = 4,
                                k.fold.data = 1)
```

```
----------------------------------------
    Preparing the data
----------------------------------------
----------------------------------------
    Model description
----------------------------------------
Integrated Occupancy Model with Polya-Gamma latent
variable fit with 453 sites.

Integrating 2 occupancy data sets.

Number of MCMC samples: 8000
Burn-in: 3000
Thinning Rate: 5
Total Posterior Samples: 1000


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

Sampling ...
Sampled: 2000 of 8000, 25.00%
-------------------------------------------------
Sampled: 4000 of 8000, 50.00%
-------------------------------------------------
Sampled: 6000 of 8000, 75.00%
-------------------------------------------------
Sampled: 8000 of 8000, 100.00%
----------------------------------------
    Cross-validation
----------------------------------------
```

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

```
Only holding out data from data source 1.
```

```r
# Single data source model
out.k.fold$k.fold.deviance
```

```
[1] 1518.765
```

```r
# Integrated model
out.int.k.fold.hbef$k.fold.deviance
```

```
[1] 1521.238
```

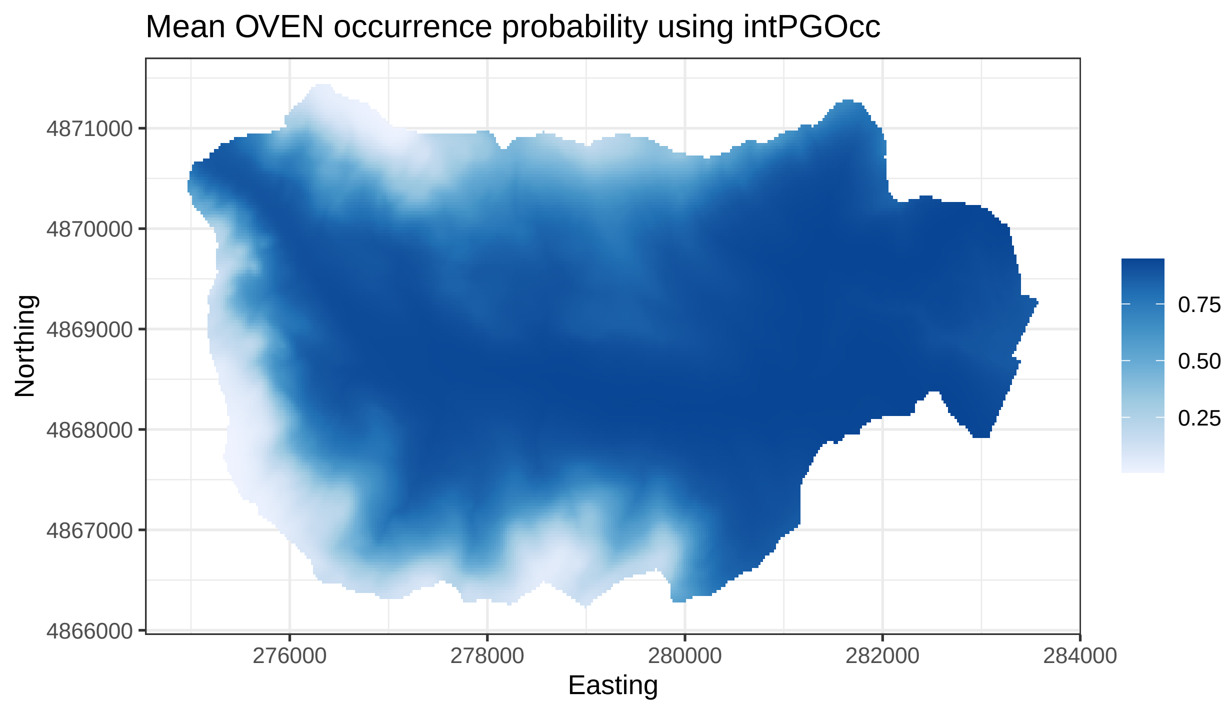
Here we see that integration of the two data sources does not improve predictive performance at HBEF. We should also do the same thing with the NEON data. We close this section by emphasizing that there are potentially numerous other benefits to data integration than predictive performance that must be carefully considered when trying to determine if data integration is necessary or not. See Simmonds et al. (2020) and discussion in Doser et al. (2021) for more on this topic.

6.7 Prediction

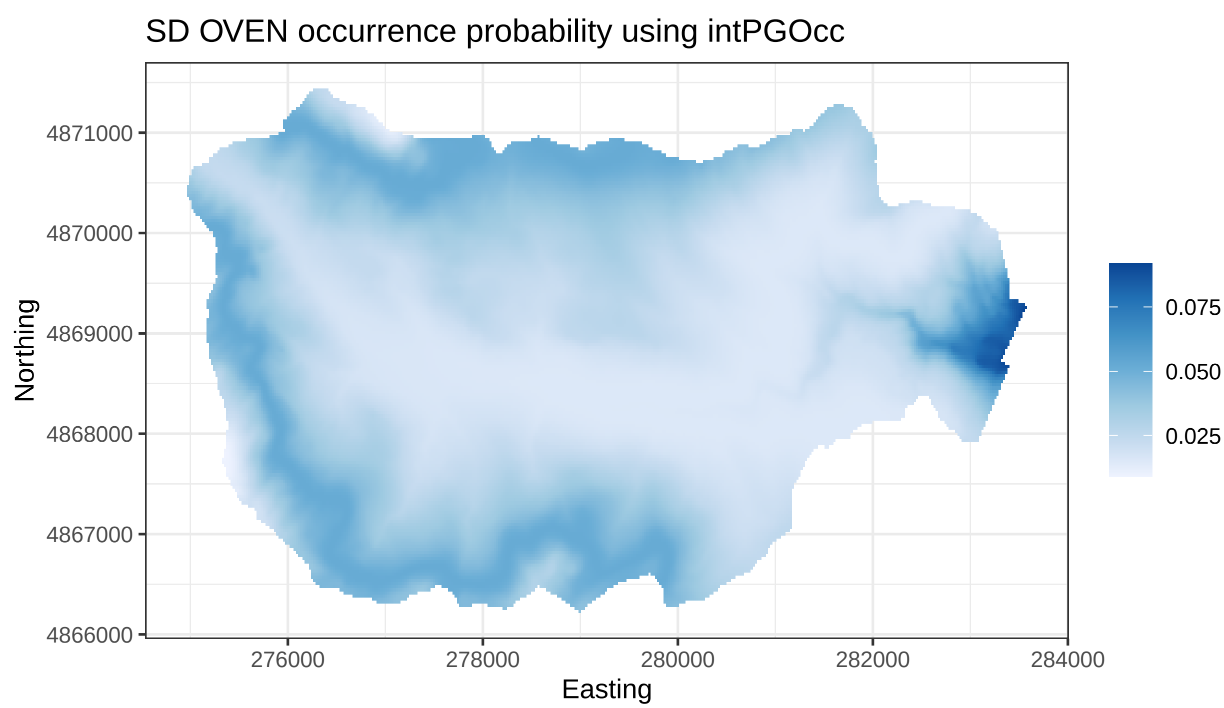
Prediction for integrated occupancy models proceeds exactly as before using `predict`. Here we predict occurrence across HBEF for comparison with the single data source models.

```r
# Make sure to standardize using mean and sd from fitted model
elev.pred <- (hbefElev$val - mean(data.int$occ.covs[, 1])) / sd(data.int$occ.covs[, 1])
X.0 <- cbind(1, elev.pred, elev.pred^2)
out.int.pred <- predict(out.int, X.0)
```

```r
plot.dat <- data.frame(x = hbefElev$Easting,
                       y = hbefElev$Northing,
                       mean.psi = apply(out.int.pred$psi.0.samples, 2, mean),
                       sd.psi = apply(out.int.pred$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 using intPGOcc') +
  theme_bw()
```




```r
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 using intPGOcc') +
  theme_bw()
```



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 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, k.fold, k.fold.threads = 1,
           k.fold.seed, ...)
```

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 in the `coords` tag, which we add below.

```
data.int$coords <- rbind(hbef2015$coords, neon2015$coords)
occ.formula.int <- ~ scale(Elevation) + I(scale(Elevation)^2)
det.formula.int <- list(hbef = ~ scale(day) + scale(tod) + I(scale(day)^2),
                        neon = ~ scale(day) + scale(tod) + I(scale(day)^2))
```

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.

```
dist.int <- dist(data.int$coords)
min.dist <- min(dist.int)
max.dist <- max(dist.int)
J <- nrow(data.int$occ.covs)
# Exponential covariance model
cov.model <- "exponential"
starting.list <- list(alpha = list(0, 0),
                      beta = 0,
                      z = rep(1, J),
                      sigma.sq = 2,
                      phi = 3 / mean(dist.int),
                      w = rep(0, J))
prior.list <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = list(0, 0),
                                       var = list(2.72, 2.72)),
                   sigma.sq.ig = c(2, 1),
                   phi.unif = c(3 / max.dist, 3 / min.dist))
```

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. Remember that spatially-explicit models in `spOccupancy` are implemented using an efficient adaptive MCMC sampler that requires us to specify the number of MCMC batches (`n.batch`) and the length of each MCMC batch (`batch.length`), which together determine the number of MCMC samples (i.e., `n.samples = n.batch * batch.length`). We run the model and summarize the results with `summary`.

```
batch.length <- 25
n.batch <- 400
n.burn <- 5000
n.thin <- 5
tuning <- list(phi = 1)
out.sp.int <- spIntPGOcc(occ.formula = occ.formula.int,
                         det.formula = det.formula.int,
                         data = data.int,
                         starting = starting.list,
                         priors = prior.list,
                         tuning = tuning,
                         cov.model = cov.model,
                         NNGP = TRUE,
                         n.neighbors = 5,
                         n.batch = n.batch,
                         n.burn = 5000,
                         batch.length = batch.length,
                         n.report = 100)
```

```
----------------------------------------
    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 453 sites.

Integrating 2 occupancy data sets.

Number of MCMC samples: 10000 (400 batches of length 25)
Burn-in: 5000
Thinning Rate: 1
Total Posterior Samples: 5000

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		24.0		0.36422
-------------------------------------------------
Batch: 200 of 400, 50.00%
	parameter	acceptance	tuning
	phi		24.0		0.22537
-------------------------------------------------
Batch: 300 of 400, 75.00%
	parameter	acceptance	tuning
	phi		36.0		0.23457
-------------------------------------------------
Batch: 400 of 400, 100.00%
```

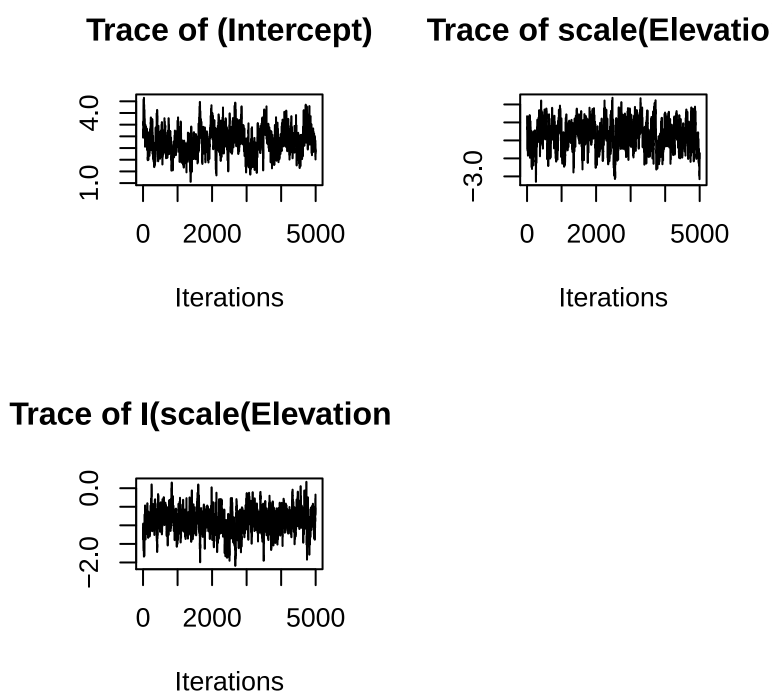
```
summary(out.sp.int)
```

```
Call:
spIntPGOcc(occ.formula = occ.formula.int, det.formula = det.formula.int,
    data = data.int, starting = starting.list, priors = prior.list,
    tuning = tuning, cov.model = cov.model, NNGP = TRUE, n.neighbors = 5,
    n.batch = n.batch, batch.length = batch.length, n.report = 100,
    n.burn = 5000)

Chain Information:
Total samples: 10000
Burn-in: 5000
Thin: 1
Total Posterior Samples: 5000

Occurrence:
                        2.5%     25%     50%     75%   97.5%
(Intercept)           1.8260  2.4190  2.7816  3.1594  3.9147
scale(Elevation)     -2.5751 -2.0901 -1.8290 -1.5862 -1.1917
I(scale(Elevation)^2) -1.5683 -1.0680 -0.8759 -0.6837 -0.2972

Data source 1 Detection:
                  2.5%     25%     50%     75%  97.5%
(Intercept)      0.5877  0.7438  0.8253  0.9087 1.0644
scale(day)      -0.2418 -0.1423 -0.0916 -0.0377 0.0669
scale(tod)      -0.1965 -0.1000 -0.0461  0.0072 0.1083
I(scale(day)^2) -0.1545 -0.0358  0.0277  0.0946 0.2184

Data source 2 Detection:
                  2.5%     25%     50%     75%  97.5%
(Intercept)      0.9661  1.7505  2.1558  2.5588 3.3932
scale(day)      -0.4601  0.4258  0.7517  1.0541 1.6697
scale(tod)      -0.7027  0.0759  0.5120  0.9604 1.5962
I(scale(day)^2) -1.4288 -0.9428 -0.6536 -0.2707 1.0554

Covariance:
           2.5%    25%    50%    75%   97.5%
sigma.sq 1.9359 3.0296 3.9630 5.2130 10.1046
phi      0.0009 0.0013 0.0017 0.0021  0.0030
```

7.3 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 and adequate mixing of the MCMC chains.

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



7.4 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: 5000
Thin: 1
Total Posterior Samples: 5000

Data Source 1

Bayesian p-value: 0.7778
Fit statistic: freeman-tukey
```

```
Data Source 2

Bayesian p-value: 0.157
Fit statistic: freeman-tukey
```

According to the Bayesian p-values, there is no lack of fit for either the HBEF or NEON data.

7.5 Model selection using WAIC and k-fold cross-validation

We can perform model selection using WAIC with the `waicOcc` function as we have seen previously. Here we compare the WAIC from the spatial integrated model to the non-spatial integrated model.

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

```
        elpd         pD        WAIC
1 -633.37923 5.78042059 1278.319297
2   -4.55661 0.05771227    9.228644
```

```r
waicOcc(out.sp.int)
```

```
         elpd       pD      WAIC
1 -572.396955 42.76514 1230.3242
2   -5.732929  1.95077   15.3674
```

Interestingly, the spatial model performs better for the HBEF data but worse for the NEON data. However, we should ensure convergence of the model prior to assigning any weight to these results.

Two forms of k-fold cross-validation are implemented for `spIntPGOcc`, analogous to those discussed for non-spatial integrated occupancy models. We first use cross-validation to compare the predictive performance of the spatial integrated model to the nonspatial integrated model across all sites.

```r
out.sp.int.k.fold <- spIntPGOcc(occ.formula = occ.formula.int,
                    det.formula = det.formula.int,
                    data = data.int,
                    starting = starting.list,
                    priors = prior.list,
                    tuning = tuning,
                    cov.model = cov.model,
                    NNGP = TRUE,
                    n.neighbors = 5,
                    n.batch = n.batch,
                    n.burn = 5000,
                    batch.length = batch.length,
                verbose = FALSE,
                    k.fold = 4)
# Non-spatial model
out.int.k.fold$k.fold.deviance
```

```
[1] 1508.9607  157.0867
```

```r
# Spatial model
out.sp.int.k.fold$k.fold.deviance
```

```
[1] 1525.5126  148.8254
```

Again, we don't interpret these results here as the models have not fully converged. Further, we perform cross-validation using only the HBEF data source as a hold out data source to compare with single data source models to assess the benefit of integration.

```r
out.sp.int.k.fold.hbef <- spIntPGOcc(occ.formula = occ.formula.int,
                          det.formula = det.formula.int,
                          data = data.int,
                          starting = starting.list,
                          priors = prior.list,
                          tuning = tuning,
                          cov.model = cov.model,
                          NNGP = TRUE,
                          n.neighbors = 5,
                          n.batch = n.batch,
                          n.burn = 5000,
                          batch.length = batch.length,
                          verbose = FALSE,
                          k.fold = 4,
                                      k.fold.data = 1)
out.sp.int.k.fold.hbef$k.fold.deviance
```

```
[1] 1524.751
```

```r
# Non-spatial single data source model
out.k.fold$k.fold.deviance
```

```
[1] 1518.765
```

7.6 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.

```r
# Make sure to standardize using mean and sd from fitted model
elev.pred <- (hbefElev$val - mean(data.int$occ.covs[, 1])) / sd(data.int$occ.covs[, 1])
X.0 <- cbind(1, elev.pred, elev.pred^2)
coords.0 <- as.matrix(hbefElev[, c(2, 3)])
out.sp.int.pred <- predict(out.sp.int, X.0, coords.0)
```

References

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

Barnett, David T, Paul A Duffy, David S Schimel, Rachel E Krauss, Kathryn M Irvine, Frank W Davis, John E Gross, et al. 2019. "The Terrestrial Organism and Biogeochemistry Spatial Sampling Design for the National Ecological Observatory Network." *Ecosphere* 10 (2): e02540.

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.

Lunn, David, Christopher Jackson, Nicky Best, Andrew Thomas, and David Spiegelhalter. 2013. "The Bugs Book: A Practical Introduction to Bayesian Analysis."

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

Miller, David AW, Krishna Pacifici, Jamie S Sanderlin, and Brian J Reich. 2019. "The Recent Past and Promising Future for Data Integration Methods to Estimate Species' Distributions." *Methods in Ecology and Evolution* 10 (1): 22–37.

National Ecological Observatory Network (NEON). 2021. "Breeding Landbird Point Counts (Dp1.10003.001)." National Ecological Observatory Network (NEON). https://data.neonscience.org/data-products/DP1.10003.001.

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.

Simmonds, Emily G, Susan G Jarvis, Peter A Henrys, Nick JB Isaac, and Robert B O'Hara. 2020. "Is More Data Always Better? A Simulation Study of Benefits and Limitations of Integrated Distribution Models." *Ecography* 43 (10): 1413–22.

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).