Vignette: Application of Bayesian single-season, single-species occupancy model for assessing silver-haired bat (*Lasionycteris noctivagans*) occurrence in Oregon and Washington, 2019

Camille Rieber[1] Christian Stratton[2] Thomas J. Rodhouse[3]
Kathryn M. Irvine[1,*]

Abstract

Our vignette demonstrates an application of a single-season, single-species Bayesian occupancy model to silver-haired bat (*Lasionycteris noctivagans*) detection/non-detection data using the `SpOccupancy` R package. The detailed methods along with annotated code provide a step-by-step guide to applying one type of occupancy model for analyzing summertime occurrence data collected following a NABat regional stationary acoustic protocol. We outline procedures for data cleaning and manipulation, model fitting, assessment of model convergence and fit, interpretation of parameters, and producing predictive maps. Emphasis is placed on demonstrating the steps of model evaluation and assumption testing that are necessary not only for this specific model and analysis, but generally for any statistical analysis. The code made available herein is broadly applicable for evaluating the structural components of a model using residual diagnostics and for interrogating model assumptions, such as independence, Markov Chain Monte Carlo (MCMC) convergence, etc. that is encouraged for any NABat analysis to infer status and trends in species occurrence.

[1] U.S. Geological Survey, Northern Rocky Mountain Science Center, Bozeman, MT, USA
[2] Department of Mathematical Sciences, Montana State University, Bozeman, MT, USA
[3] U.S. National Park Service, Upper Columbia Basin Network, Bend, OR, USA

[*] Correspondence: Kathryn M. Irvine <kirvine@usgs.gov>

Contents

1 Introduction

The purpose of this document is to demonstrate how data collected by the Pacific Northwest Bat Hub, a member of the North American Bat Monitoring Program (NABat), may be used to make inferences about species occurrence. As motivation, we consider occurrence data collected on Silver-haired bats (*Lasionycteris noctivagans*, LANO) in Oregon and Washington in 2019. In particular, we are interested in the following: 1) understanding the relationships between various covariates and the probability of occurrence and detection, and 2) predicting species occurrence probabilities across the study area. Throughout this document, we assume the reader has a cursory understanding of the NABat sampling design and program; see Loeb et al. (2015) for a complete description of the NABat protocol. The Pacific NW Bat Hub selected cells for surveys following the NABat probabilistic master sample design (Rodriguez et al. 2019). The analytical unit is the 10km $\times$ 10km grid cell which corresponds to the resolution of the NABat grid (Talbert and Reichert 2018) and we will refer to it as the "NABat grid cell" or just "cell". The NABat grid cell is the unit at which we estimate occurrence (referred to as the "site" in the occupancy literature). Our detection events are the replicated visits to a cell (referred to as a "visit" in the occupancy literature). The revisit design could arise from multiple stationary detector locations (spatial replication within a grid cell) or multiple nightly surveys (temporal replication within a grid cell). We refer to each unique revisit as a "site-night." Because bats may be using a 10-km by 10-km cell for many reasons (e.g., foraging, roosting, etc), we use the term "occurrence" as opposed to "occupancy" throughout for our statistical inferences. While we are using the traditional framework referred to in the literature as "occupancy" modeling, we maintain the language of "occurrence" throughout the vignette to emphasize that we are estimating the probability that at least one individual from a species occurred in a cell.

For our analysis, we assume that all false-positive detections are removed prior to analysis (Banner et al. 2018), enabling the use of a single-season, single-species occupancy model (MacKenzie et al. 2002); following Banner et al. (2018), we refer to this model as the "remove" model. This document serves as a step-by-step guide to applying the remove model to an example dataset that was a portion of previously analyzed data that contained ecological interpretations (see Wright et al. (2021)). We provide annotated code to clean and prepare the data for analysis; fit the single-season, single-species Bayesian occupancy model to the cleaned data; assess the convergence of the fitted model and quality of the fitted model through residual diagnostics; and produce summaries of estimated model parameters and predicted occurrence. Throughout this document, considerable emphasis is placed on the assessment of model assumptions, and code is provided to allow researchers to make rigorous assessments of these assumptions for related analyses.

Application of this vignette and the Bayesian occupancy model assumes basic understanding of Bayesian modeling and operational knowledge of Markov chain Monte Carlo (MCMC) sampling. All model fitting and inference is completed in `R` and we assume base proficiency in `R` for use of this vignette (e.g., installing packages, familiarity with `tidyverse`). We use the `spOccupancy` package developed by Doser et al. (2022) to fit the occupancy model; for more information on this package see the `R` help file or package webpage. The `spOccupancy` package utilizes the desired Bayesian occupancy model, and includes user-friendly model-fitting syntax and output. This package provides far faster model fitting compared to alternative model fitting algorithms by utilizing Póly-Gamma data augmentation to induce Gibbs updates.

The remainder of this document is organized as follows: in Section 2, we describe the available data; in Section 3, we describe the model fit to the data, including model assumptions, code to fit the model, and assessment of the fitted model; in Section 4, we provided a summary of the fitted model, including interpretations of estimated parameters and maps of predicted occurrence.

2 Data

In this vignette, we will analyze one year of detection data for the Silver-haired bat (LANO) across Washington and Oregon in 2019. These data were previously analyzed by Wright et al. (2021); for a complete description of these data, see Wright et al. (2021) and Rodriguez et al. (2019). These data come in the standard occupancy data format of each row corresponding to a site-night revisit. Below, we briefly summarize key features of the data:

- Repeated detection non-detection data for sampled sites
  - Repeated measures for sites (e.g., NABat grid cell) can be spatial (e.g., multiple detectors within a site) or temporal (e.g., detectors record for multiple nights)
  - Data analyzed in this vignette contain only spatial replicates
- Values of detection covariates for all site-night visits
- GPS locations of detectors
- Values of occurrence covariates across the spatial extent of interest (e.g, Oregon and Washington)
- Spatial geometries for NABat grid across the spatial extent of interest

Because we are using the remove model (Banner et al. 2018), potential false-positives in the occurrence data must be removed prior to analysis. Once potential false-positives have been removed, the response variable is an indicator variable for whether at least one acoustic recording for the species of interest is manually confirmed for each revisit (each site-night). For example, if a confirmed echolocation sequence was recorded

for LANO at that site-night a 1-value is assigned, and 0-value is assigned otherwise.

2.1 Format of incoming data

Prior to analysis with the `spOccupancy` package, additional `R` packages are used for data cleaning, manipulation, and visualization, including:

- `tidyr` and `dplyr` for data cleaning and manipulation
- `ggplot2` and `viridis` for data visualization
- `sf` for manipulation of spatial objects

Throughout this vignette, we assume the reader has some familiarity with each of these additional packages; see the `R` help page and package vignettes for more detail.

```
library(spOccupancy)
library(dplyr)
library(tidyr)
library(sf)
library(ggplot2)
library(viridis)
```

Read in detection data:

```
data.0 <- read.csv("data/survey_data.csv")
names(data.0)
```

```
##  [1] "state"            "CONUS10km"        "path"             "Site"
##  [5] "WAVsRaw"          "WAVsProc"         "Start"            "End"
##  [9] "Year"             "SurveyData"       "ANPA"             "COTO"
## [13] "EPFU"             "EUMA"             "EUPE"             "LABL"
## [17] "LACI"             "LANO"             "MYCA"             "MYCI"
## [21] "MYEV"             "MYLU"             "MYTH"             "MYVO"
## [25] "MYYU"             "PAHE"             "TABR"             "lat"
## [29] "long"             "Clutter."         "ClutterType"      "ClutterMethod"
## [33] "LocalHabitat"     "BroadHabitat"     "DetectionTarget"  "TargetDescriptor"
## [37] "DateIssue"        "tmin"             "prcp"             "vp"
## [41] "dayl"
```

The data we will start with contains one row for each survey. For each survey, the data must have:

- an ID for the 10km x 10km NABat grid cell (in this case "CONUS10km")
- an identifier or identifiers for date and location of the survey within the grid cell
- an indicator for species detection (in this case "LANO")
- values of detection covariates for each survey (in this case "Clutter.", "DetectionTarget", "tmin")

Read in data for the NABat grid and associated occurrence covariates for the entire study area (e.g., Oregon

and Washington):

Covariate data were obtained from Wright et al. (2021).

```
plot_covs_all <- read.csv("data/OR-WA-FullGrid-Landcover-Covariates.csv")
names(plot_covs_all)
```

```
##  [1] "SampleUnitID.CONUS10k"
##  [2] "Xcoord_UTM"
##  [3] "Ycoord_UTM"
##  [4] "Elevation_Mean"
##  [5] "Elevation_StdDev"
##  [6] "SnagDensityMeanPerHa_25cmDBH"
##  [7] "NetPrimaryProductivity_avg.kg.C.sqmeter"
##  [8] "CliffsCanyons_PercentCover"
##  [9] "Forest_PercentCover"
## [10] "MeanAnnualPrecipitation_mm"
## [11] "State"
```

This data contains one row for each 10km x 10km NABat grid cell. For each cell it should have:

- an ID for the grid cell (in this case "SampleUnitID.CONUS10k")
- values of occurrence covariates for each cell (in this case "CliffsCanyons_PercentCover", "Forest_PercentCover", "MeanAnnualPrecipitation_mm")

Read in shapefile for the 10km x 10km NABat grid:

We used grid cells downloaded from Wright et al. (2021), which used the sciencebase.gov grid cells provided by USGS.

```
grid_shp <- read_sf("data/grid_shp.shp")
names(grid_shp)
```

```
##  [1] "Plot"     "EvalRsn"  "EvlStts"  "GRTS_ID"  "mdcaty"   "panel"
##  [7] "stratum"  "wgt"      "xcoord"   "ycoord"   "water_p"  "outsd_p"
## [13] "stt_n_1"  "stt_p_1"  "stt_n_2"  "stt_p_2"  "stt_n_3"  "stt_p_3"
## [19] "stt_n_4"  "stt_p_4"  "cnty_n_1" "cnty_p_1" "cnty_n_2" "cnty_p_2"
## [25] "cnty_n_3" "cnty_p_3" "cnty_n_4" "cnty_p_4" "cnty_n_5" "cnty_p_5"
## [31] "own_BLM"  "ow_CITY"  "ow_CNTY"  "own_DOD"  "own_FWS"  "own_JNT"
## [37] "own_NGO"  "ow_NOAA"  "o_NODAT"  "own_NPS"  "ow_OTHF"  "own_PVT"
## [43] "own_REG"  "o_STATE"  "ow_TRIB"  "own_UNK"  "o_USACE"  "ow_USBR"
## [49] "ow_USFS"  "watr_p2"  "otsd_p2"  "grd_dpl"  "geometry"
```

This shapefile should have the same ID for each 10km x 10km grid cell (in this case "Plot") as the previous data file. This file will allow for spatial mapping and needs to contain the associated geometry for each cell, and is an sf object. It additionally contains points for the centroid of each grid cell in "xcoord" and "ycoord."

Analytical unit labeling

For consistency, we will call each 10km x 10km grid cell determined by the NABat master sample structure the "Cell" across all data objects.

```
data <- data.0 %>% rename("Cell" = "CONUS10km")
grid_shp <- grid_shp %>% rename( "Cell"="Plot" )
plot_covs_all<- plot_covs_all %>% rename( "Cell" = "SampleUnitID.CONUS10k")
```

We will use "Replicate" to describe site-nights (in this case spatial replicates) within a cell. Replicate will then be a unique identifier for the site-night.

```
data <- data %>% rename("Replicate" = "Site")
```

2.1.1 Explore data

First, we explore the data to inform assessment of spatial and temporal scope of inference.

Investigate species, spatial extent, seasons, and replicates:

```
unique(data$state) #see spatial extent of surveys
```

```
## [1] "CA" "WA" "OR"
```

```
unique(plot_covs_all$State) #see spatial extent of covariates
```

```
## [1] "Oregon"     "Washington"
```

```
# remove CA data because we only have covariate data for OR and WA
data <- data %>% filter(!(state== "CA"))
```

Perform any additional data processing specific to the dataset:

```
#Only keep data rows where Surveydata = potential bats
#This removes survey rows that were labeled, "Malfunction," or "NoData,"
# as well as removing survey rows labeled "NoBats" which had WAVsProc = 0
data <- data %>% filter(SurveyData == "PotentialBats")
summary(data$Year)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2019    2019    2019    2019    2019    2019
```

We have one season (the year 2019), and our spatial extent is Oregon and Washington. The probabilistic sampling design coordinated by the Pacific Northwest Bat Hub enables inferences for this spatial extent of interest.

Investigate how many surveys (replicates) were completed at each site:

```
data$Cell <- as.factor(data$Cell)
summary(data$Cell)[1:10]
```

```
## 109767 100495  96809  97272  99612 100518 102350 112067 119485  95426
##      7      6      5      5      5      5      5      5      5      4
```

```
#Examine what kind of replication is happening at these > 4 cells:
data %>%
  group_by(Cell) %>%
  filter(n() > 4) %>%
  select("Cell", "Replicate", "lat", "long", "Start") %>%
  print(n = nrow(.))
```

```
## # A tibble: 48 x 5
## # Groups:   Cell [9]
##    Cell   Replicate    lat  long Start
##    <fct>  <chr>      <dbl> <dbl> <chr>
##  1 96809  96809_NE2   42.1 -120. 2019-06-06
##  2 96809  96809_NW1   42.1 -120. 2019-07-08
##  3 96809  96809_SW1   42.1 -120. 2019-06-06
##  4 96809  96809_SW1   42.1 -120. 2019-07-20
##  5 96809  96809_SW2   42.1 -120. 2019-07-16
##  6 97272  97272_NE1   42.2 -120. 2019-06-05
##  7 97272  97272_NE2   42.2 -120. 2019-06-05
##  8 97272  97272_SW1   42.2 -120. 2019-06-05
##  9 97272  97272_SW2   42.2 -120. 2019-06-06
## 10 97272  97272_SW2   42.2 -120. 2019-07-20
## 11 100495 100495_NW1  42.4 -122. 2019-08-08
## 12 100495 100495_NW2  42.4 -122. 2019-08-08
## 13 100495 100495_SW1  42.4 -122. 2019-08-27
## 14 100495 100495_SW3  42.4 -122. 2019-08-27
## 15 100495 100495_SW3  42.4 -122. 2019-09-10
## 16 100495 100495_SW4  42.4 -122. 2019-09-10
## 17 102350 102350_NE1  42.9 -122. 2019-08-06
## 18 102350 102350_NW1  42.8 -122. 2019-08-06
## 19 102350 102350_SE1  42.8 -122. 2019-08-26
## 20 102350 102350_SE1  42.8 -122. 2019-09-25
## 21 102350 102350_SW2  42.8 -122. 2019-08-06
## 22 119485 119485_NW1  46.1 -123. 2019-08-07
## 23 119485 119485_NW1  46.1 -123. 2019-08-12
## 24 119485 119485_SE2  46.1 -123. 2019-08-07
## 25 119485 119485_SW1  46.1 -123. 2019-08-07
## 26 119485 119485_SW3  46.0 -123. 2019-08-07
## 27 100518 100518_NE1  42.9 -120. 2019-06-17
## 28 100518 100518_NW1  42.9 -120. 2019-06-17
## 29 100518 100518_NW3  42.9 -120. 2019-06-17
## 30 100518 100518_NW3  42.9 -120. 2019-07-10
## 31 100518 100518_SW1  42.9 -120. 2019-06-17
## 32 99612  99612_NE1   43.2 -117. 2019-06-19
## 33 99612  99612_NW1   43.2 -117. 2019-07-11
## 34 99612  99612_NW2   43.2 -117. 2019-06-19
## 35 99612  99612_SE1   43.1 -117. 2019-06-19
## 36 99612  99612_SW1   43.2 -117. 2019-07-11
## 37 109767 109767_SE1  44.4 -122. 2019-06-17
## 38 109767 109767_SW1  44.4 -122. 2019-07-08
## 39 109767 109767_SW2  44.4 -122. 2019-07-17
## 40 109767 109767_SW3  44.4 -122. 2019-08-05
```

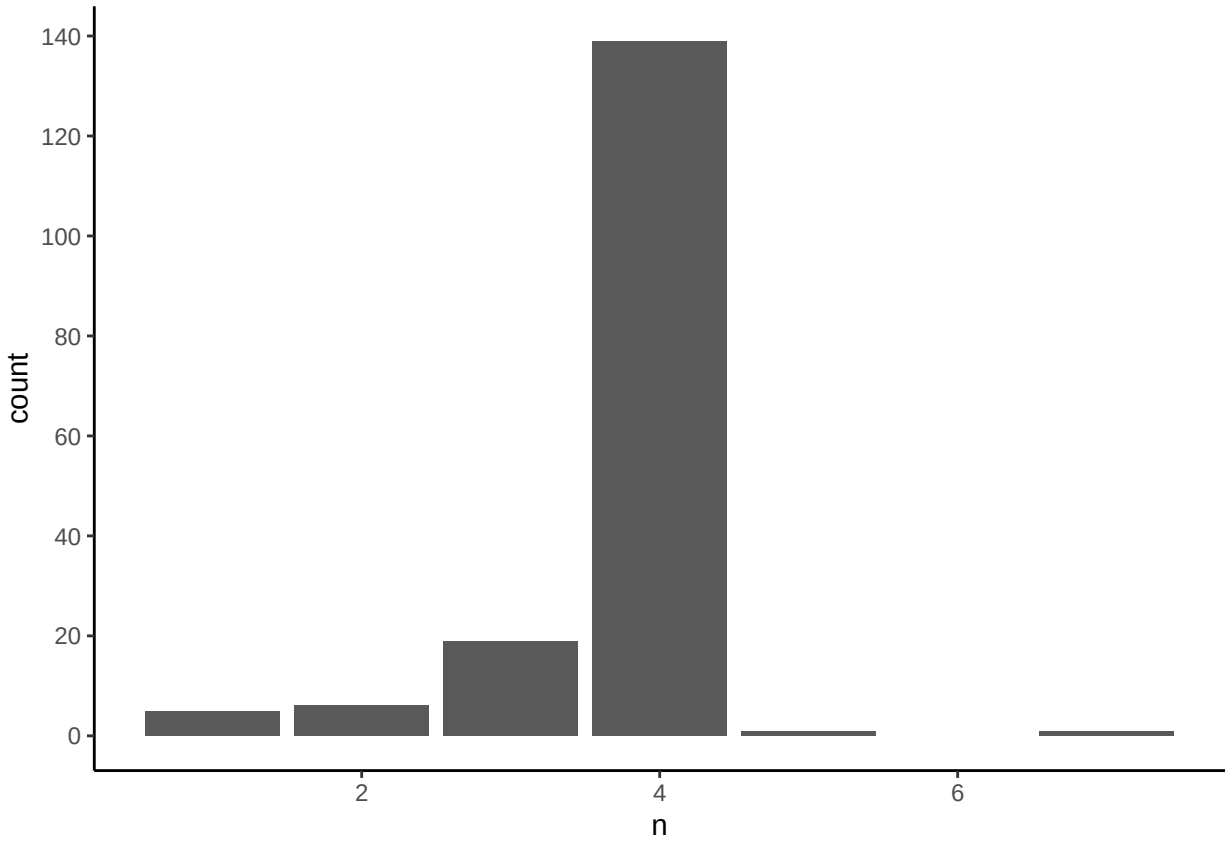
```
## 41 109767 109767_SW4  44.4 -122. 2019-07-30
## 42 109767 109767_SW5  44.4 -122. 2019-07-30
## 43 109767 109767_SW6  44.4 -122. 2019-08-05
## 44 112067 112067_NE1  44.5 -124. 2019-08-19
## 45 112067 112067_NE2  44.5 -124. 2019-08-19
## 46 112067 112067_NE2  44.5 -124. 2019-08-20
## 47 112067 112067_NW1  44.5 -124. 2019-08-19
## 48 112067 112067_SE4  44.5 -124. 2019-08-19
```

The standard revisit design for these data is to have four spatial replicates (different detector locations) within each grid cell (Rodriguez et al. (2019)). We see that 9 of these sites were surveyed >4 times. Looking at the data, this is either because >4 spatial replicates were completed in 2 cells (e.g., 109767_SW1:109767_SW6); or because temporal replicates were completed in 7 cells (e.g., 119485_NW1 surveyed on two separate dates). To match the revisit design and minimize data manipulation in this vignette, we remove the data for the 7 cells with temporal replicates.

```
# filter out cells with temporal replicates (visually identified from the above output)
data <- data %>% filter(!(Cell %in% c("112067","100518","102350","100495",
                                      "97272","119485","96809")))
```

View the spread of number of surveys conducted at each surveyed cell:

```
data %>% count(Cell) %>%
  ggplot( aes(n)) +
  geom_bar() +
  scale_y_continuous(n.breaks=8) +
  theme_classic()
```



Pare the data down to just the variables of interest for model-fitting formatting. This includes Cell ID, Replicate ID, species detection, and detection/survey level covariates.

```
data <- data %>% select("Cell", "Replicate",  "LANO", "DetectionTarget",
                        "Clutter.","tmin", "prcp", "vp","dayl", "lat", "long" )
```

Check for NA values in included covariates, because all covariates used to fit the model must be complete:

```
which(is.na(plot_covs_all), arr.ind=TRUE)
```

```
##      row col
```

```
which(is.na(data), arr.ind=TRUE)
```

```
##       row col
##  [1,]   1   4
##  [2,]   2   4
##  [3,]   9   4
##  [4,]  10   4
##  [5,]  11   4
##  [6,]  12   4
##  [7,]  13   4
##  [8,]  14   4
##  [9,]  15   4
## [10,]  16   4
```

```
## [11,]  17   4
## [12,]  18   4
## [13,]  57   4
## [14,]  71   4
## [15,] 147   4
## [16,] 177   4
## [17,] 178   4
## [18,] 570   4
## [19,] 609   4
```

We see that there are 19 NAs in the `DetectionTarget` detection covariate. We assume that this detection target data is missing at random and, because we wish to include the water indicator detection covariate, we remove the 19 rows of detection data that do not have detection target information.

```
data <- data %>% filter(!is.na(DetectionTarget))
```

Maintaining consistent ordering of the data is necessary to relate the original `data` object to the output of the `spOccupancy` model fitting. We order the data by Cell number. This order is then used to fit the model, ensuring that model output is in this order as well. Occurrence covariates must also be in this order.

```
data <- data %>% arrange(Cell)
plot_covs_all <- plot_covs_all %>% arrange(Cell)
```

2.1.2 Reformat occurence data

The survey data has one row for each separate site-night ("Replicate"); the following code reformats it to one row for each cell with the entire detection history. This formatting is standard for occupancy modeling (MacKenzie et al. 2002) and will also be used later on for the `spOccupancy` package.

```
#let the max number of surveys be the max number of surveys in the data
max_reps <- max(table(data$Cell))

#J indexes the unique Cells:
J <- length(unique(data$Cell))
Cell.codes <- unique(data$Cell)
# the order we're maintaining throughout for cells is the
# numeric ordering already set in data

#set up data frame for storage:
y <- as.data.frame(matrix(NA, nrow=J, ncol=max_reps))
colnames(y) <- as.character(c(1:max_reps))
replicate_names <- as.data.frame(matrix(NA, nrow=J, ncol=max_reps))
rownames(replicate_names) <- Cell.codes
colnames(replicate_names) <- as.character(c(1:max_reps))

# loop through each Cell:
for (j in 1:J){
  Cell <- data %>% filter(Cell == Cell.codes[j])
```

```
  n.reps <- nrow(Cell)
  Cell.y <- Cell %>% select("Cell", "Replicate", "LANO")
  Cell.y <- Cell.y %>% spread(Replicate, LANO)
  Cellname <- Cell.y[,1]
  Cell.y <- Cell.y %>% select(!(Cell))

  if(n.reps < max_reps){ #when less than the max surveys are done, need to add NAs
    nas <- rep(NA, max_reps-n.reps)
    replicate_names[j,] <- c(colnames(Cell.y), nas) #save ordered list of Cell code names
    y[j,] <- c(Cell.y[1,], nas)
  }
  else{ #when all surveys are done
    replicate_names[j,] <- colnames(Cell.y) #save ordered list of Cell code names
    y[j,] <- Cell.y[1,]
  }
  rownames(y)[j] <- as.character(Cellname)
}

head(y)
```

```
##       1  2  3  4  5  6  7
## 95426 0  0  0  0 NA NA NA
## 95901 0  0  0  0 NA NA NA
## 96343 1 NA NA NA NA NA NA
## 96350 0  1  0  1 NA NA NA
## 96362 1  0  0  0 NA NA NA
## 96364 0  0  0  0 NA NA NA
```

2.2 Exploratory data analysis

2.2.1 Included covariates

We assume the user applies this vignette with pre-identified covariates based on understanding of their ecological system. Necessary considerations of these covariates should be made prior to analysis, such as appropriate resolution for covariates and corresponding data sources. For this vignette we use covariate data from Wright et al. (2021), including percent forest cover, percent cliff and canyon cover, and mean annual precipitation at the occurrence level. At the detection level, we include a categorical measure of the degree of clutter near the detector, the minimum nightly temperature, and an indicator for whether water is present near the detector. Since we are using the covariates selected in Wright et al. (2021), we do not perform any form of model comparison or selection, though an option for that is available in the results section. While we examine model fit, we do not adjust model formulation in this vignette.

2.2.1.1 Occurrence covariates We are using the variables percent forest cover, percent cover of cliffs and canyons, and mean annual precipitation (mm) as predictors for occurrence. Each occurence covariate must be at the resolution of one 10km x 10km cell and covariate data have been averaged to match that resolution before being read in. We need measures of occurrence covariates at both the surveyed cells (to fit

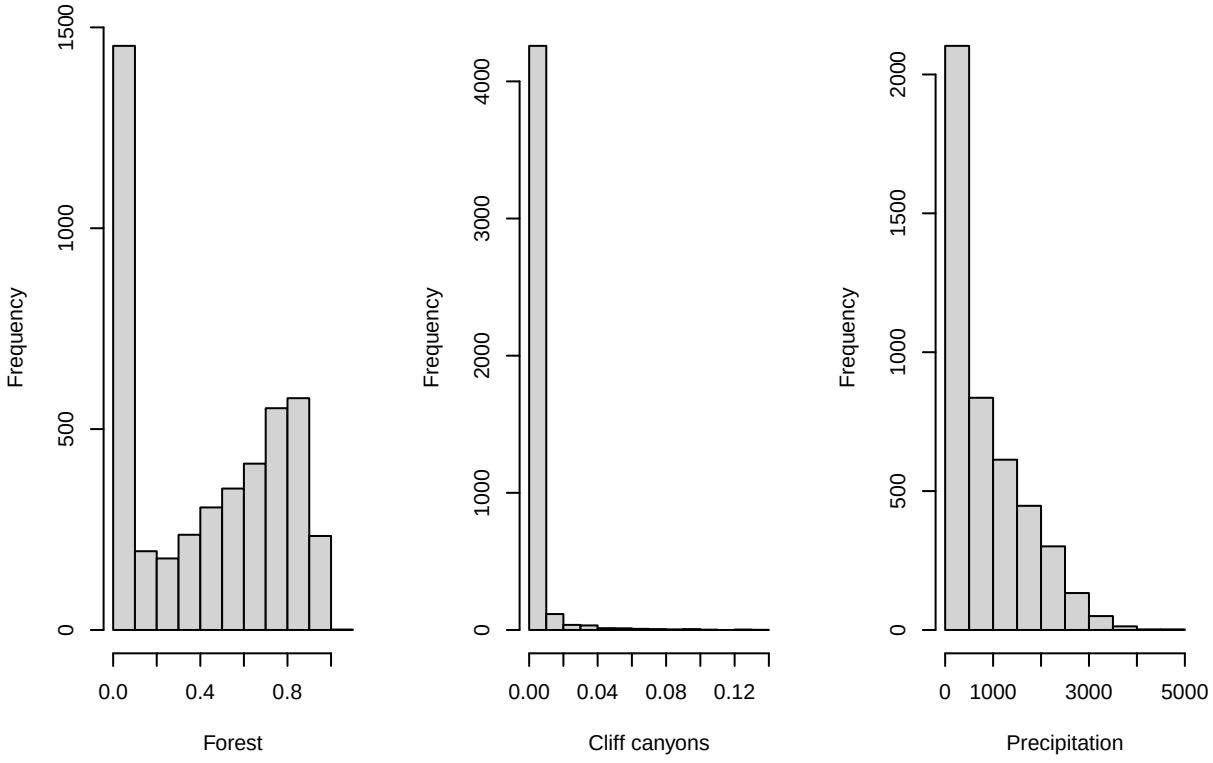
the model) and the entire study area (to create the predictive occurrence map).

Visualize covariate data across the study area and check for correlation between covariates used to fit the model:

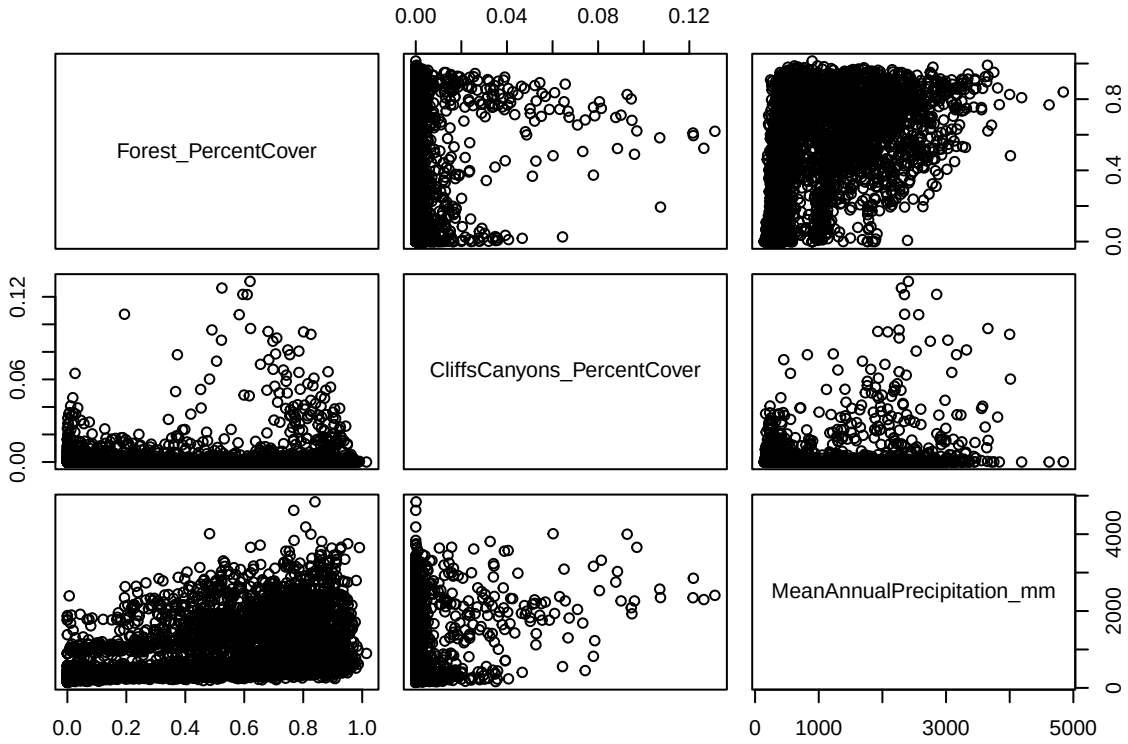
```
names(plot_covs_all) #contains our occurrence covariates across all the grid cells
```

```
##  [1] "Cell"
##  [2] "Xcoord_UTM"
##  [3] "Ycoord_UTM"
##  [4] "Elevation_Mean"
##  [5] "Elevation_StdDev"
##  [6] "SnagDensityMeanPerHa_25cmDBH"
##  [7] "NetPrimaryProductivity_avg.kg.C.sqmeter"
##  [8] "CliffsCanyons_PercentCover"
##  [9] "Forest_PercentCover"
## [10] "MeanAnnualPrecipitation_mm"
## [11] "State"
```

```
occ_covs_all <- plot_covs_all %>% #select the covariates of interest
  select("Cell", "Forest_PercentCover", "CliffsCanyons_PercentCover",
         "MeanAnnualPrecipitation_mm" )

par(mfrow= c(1,3))
hist(occ_covs_all$Forest_PercentCover, main="", xlab = "Forest" )
hist(occ_covs_all$CliffsCanyons_PercentCover,  main="", xlab = "Cliff canyons" )
hist(occ_covs_all$MeanAnnualPrecipitation_mm,  main="", xlab = "Precipitation" )
```



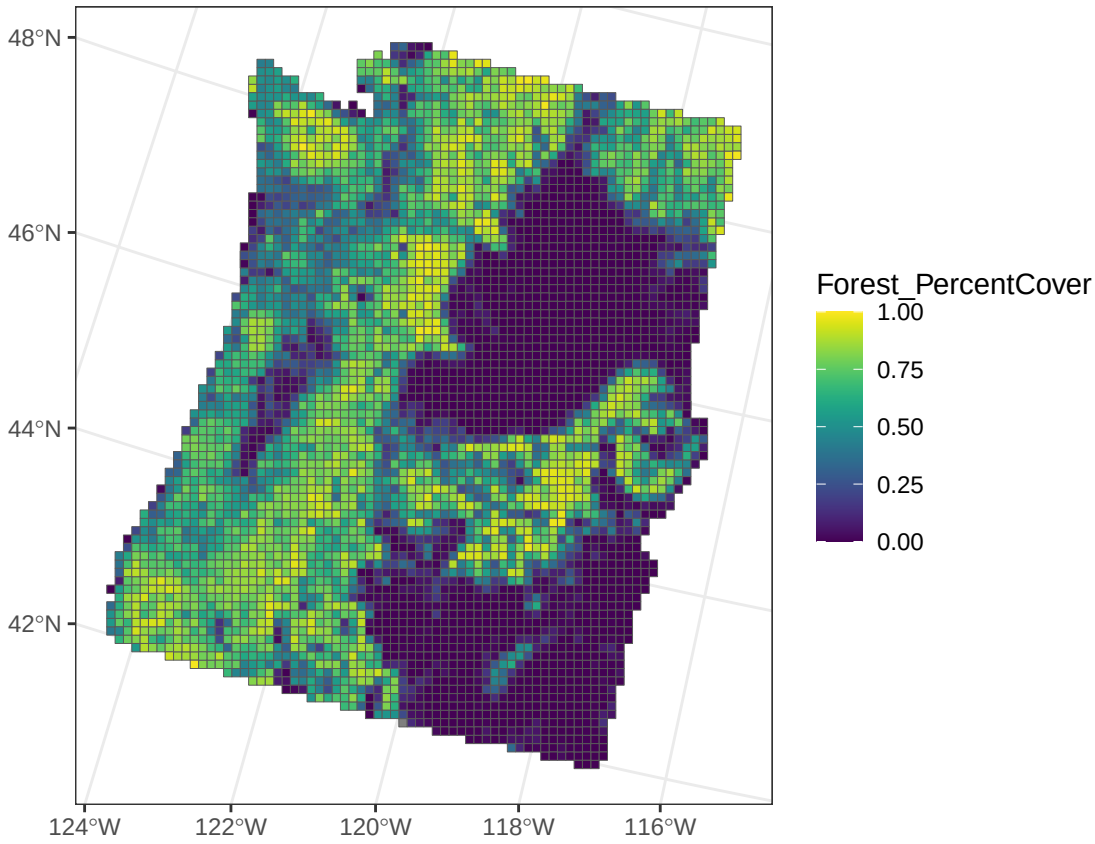
```r
pairs(occ_covs_all[,2:4])
```



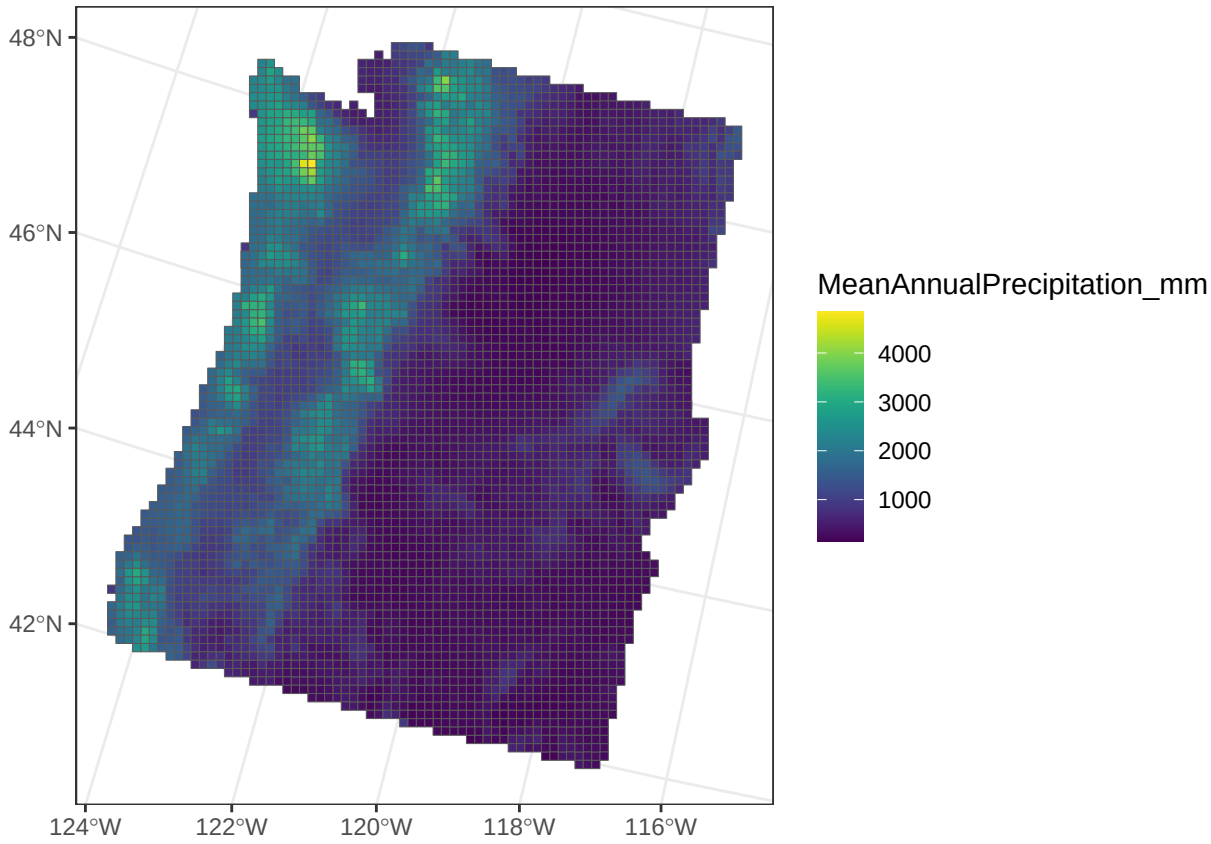
We don't see any obvious correlations (clear relationships in the scatterplots) between these covariates. If a clear relationship between covariates is observed, a formal test of correlation should be performed and covariates reconsidered.

Map occurrence covariates across study area:

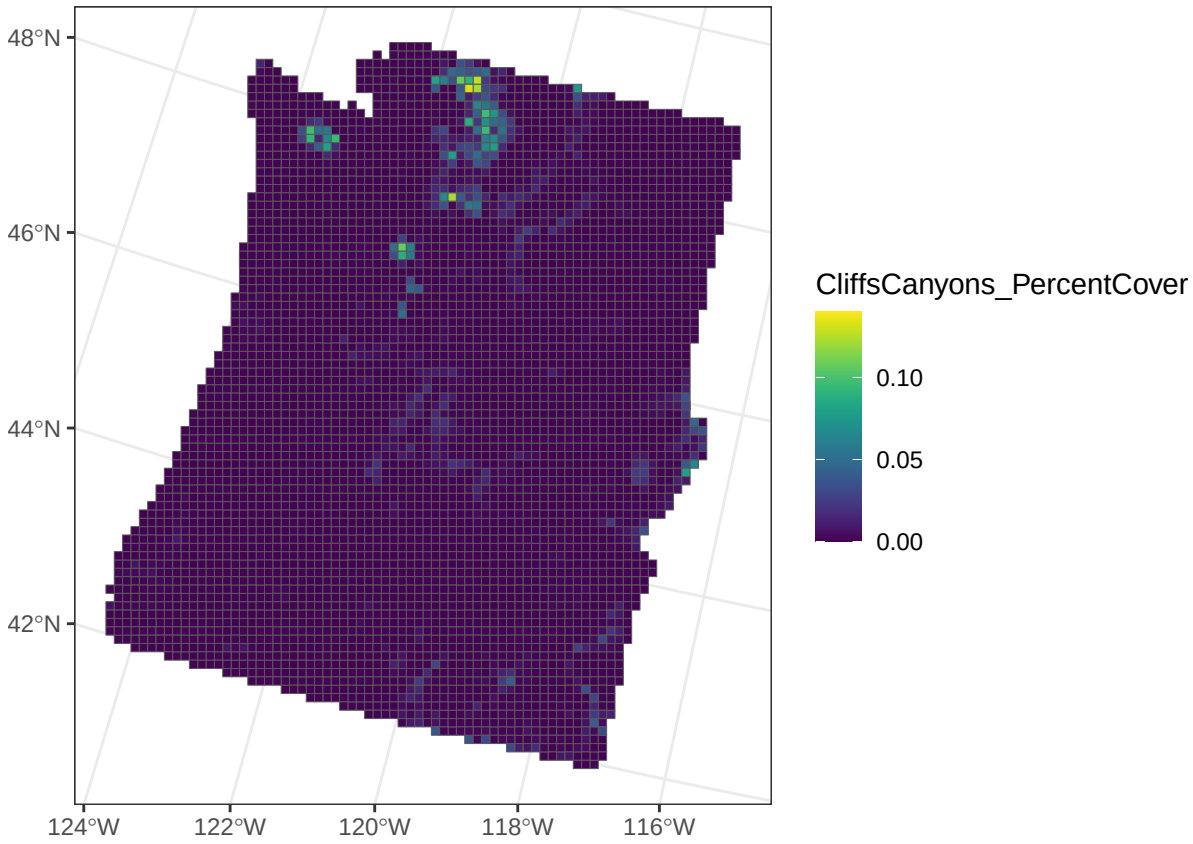
```r
#join the covariates to the spatial grid, using the plot name
plottable_covs <- left_join(select(grid_shp,Cell, geometry),
                        plot_covs_all,
                        by = 'Cell')

#there are some NA grid cells that didn't have covariate values and don't have predictions
#remove them:
plottable_covs <- na.omit(plottable_covs)

ggplot(data = plottable_covs) +
  geom_sf(aes(fill = Forest_PercentCover), lwd = 0.05 ) +
  theme_bw() +
  viridis::scale_fill_viridis(limits = c(0, 1))
```




```
ggplot(data = plottable_covs) +
  geom_sf(aes(fill = MeanAnnualPrecipitation_mm),  lwd = 0.05) +
  theme_bw() +
  viridis::scale_fill_viridis()
```




```
ggplot(data = plottable_covs) +
  geom_sf(aes(fill = CliffsCanyons_PercentCover),  lwd = 0.05) +
  theme_bw() +
  viridis::scale_fill_viridis(limits = c(0, 0.14))
```



2.2.1.2 Detection covariates We are using the variables average minimum temperature, vegetation clutter surrounding the acoustic detector, and whether the acoustic detector was at a water source as predictors for detection. Each detection covariate is recorded at each unique survey (e.g., the spatial replicates within the cell).

We create the water feature covariate using a binary classification from the data: 1 for detection target being a waterbody, 0 for not a waterbody.

```
data$DetectionTarget <- as.factor(data$DetectionTarget)
summary(data$DetectionTarget)
```

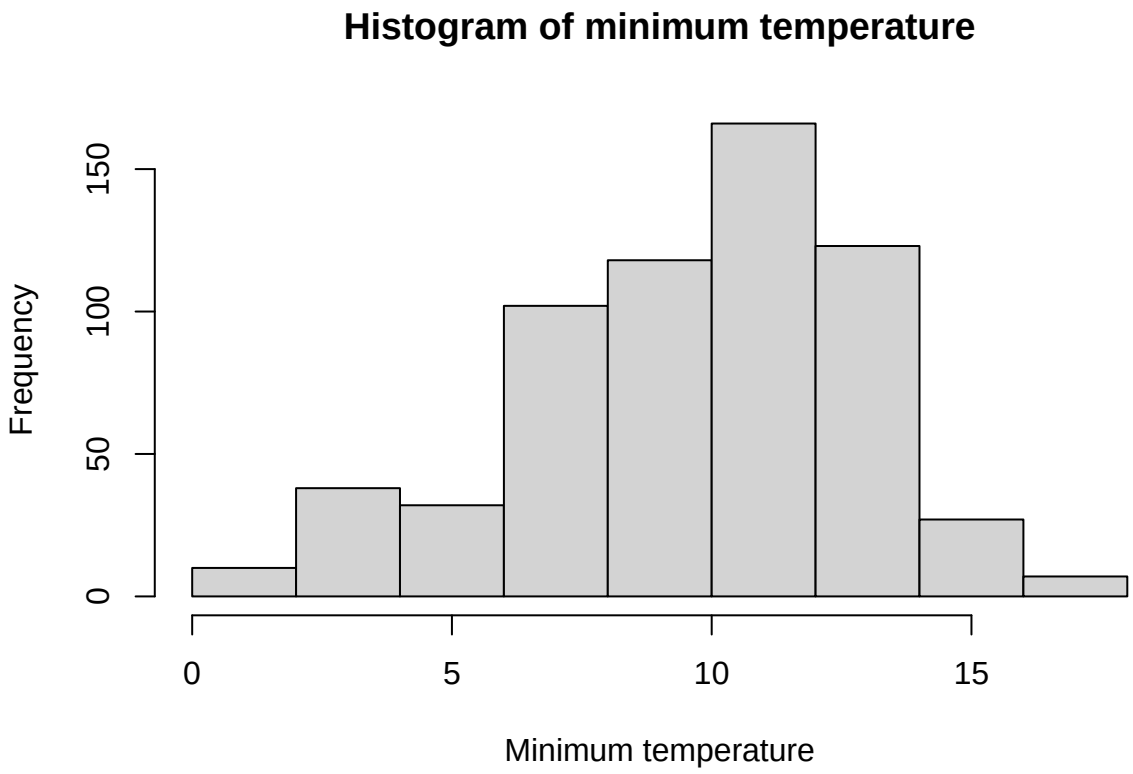
```
##                              Dry_Water_Feature
##                                             51
##                                   Forest_Edge
##                                             64
##                                Forest_Opening
##                                             83
##                                        Meadow
##                                             38
## Other_Detection_Target__if_none_of_the_above_
##                                             20
##                                  Rock_Feature
##                                             34
```

```
##                                     Waterbody
##                                           333
```

```
data$water_ind <- 0 #for not a waterbody
data$water_ind[which(data$DetectionTarget == "Waterbody")] <- 1 #for a waterbody
data$water_ind <- as.factor(data$water_ind)
```

Visualize the detection covariates:

```
hist(data$tmin, main = "Histogram of minimum temperature", xlab = "Minimum temperature")
```




```
table(data$Clutter.)
```

```
##
##   0   1   2   3   4
## 104 321 145  39  14
```

```
table(data$water_ind)
```

```
##
##   0   1
## 290 333
```

Since clutter is a categorical covariate (5 distinct clutter categories) we want to treat it as a categorical as opposed to numeric variable, and later on we will build our design matrix accordingly.

```
data$Clutter. <- as.factor(data$Clutter.)
```

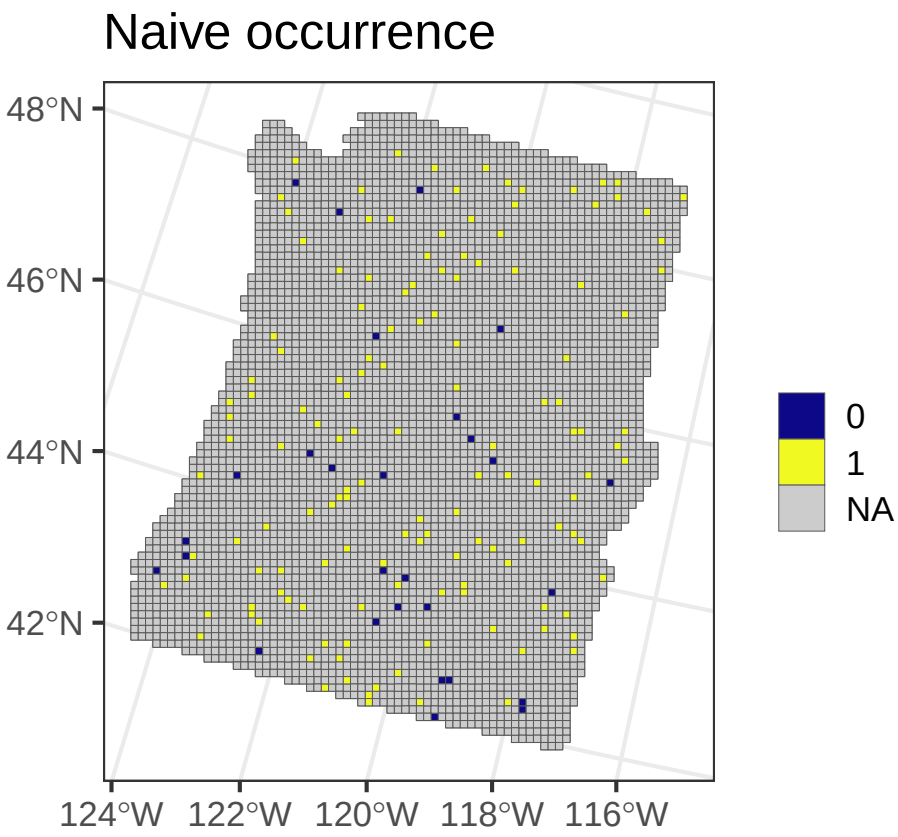
2.2.2 Naive occurrence map

First derive naive occurrence for each cell (0 if a species is never observed, 1 if a species is observed):

```
naive <- rowSums(y, na.rm = T)
naive <- as.data.frame(naive)
naive$Cell <- as.numeric(rownames(naive))
naive$Naive_occupancy <- as.factor(ifelse(naive$naive == 0, 0, 1))
```

Creating a map requires linking the geometry of the grid cells to the occurrence detections. This can be done using the cell names ("Cell").

```
naive_plot <- left_join(select(grid_shp, Cell, geometry),
                        naive, by = 'Cell')

ggplot(naive_plot) +
  geom_sf() +
  geom_sf(aes(fill = Naive_occupancy),  lwd = 0.01) +
  theme_bw(base_size = 16) +
  labs(title = "Naive occurrence", fill = "") +
  viridis::scale_fill_viridis(discrete = TRUE,
                              option = 'plasma',
                              na.value = 'grey80')
```



2.3 Reformat data for spOccupancy package

The `spOccupancy` package has specific requirements to read in data. Formatting data into the package's intake format is the most time consuming part of fitting the model. We already formatted the occurrence data to plot naive occurrence. In addition, we need a list containing separate identically structured matrices for each detection covariate (with rows corresponding to grid cells and columns corresponding to surveys), and a single dataframe for the occurrence covariates. The following code walks through formatting the data that was read in with one row for each site-night (standard occupancy format). The final `spOccupancy` format is shown in section 2.3.3.

2.3.1 Format detection covariates

First we create our detection design matrix, with number of rows corresponding to the number of site-nights that were actually surveyed. Continuous variables are scaled within the design matrix, which is discussed in model specification.

```
design.matrix.surveyed <- model.matrix(~ Clutter. + scale(tmin) + water_ind, data)
```

```
# create a temporary dataframe to help with construction of the covariate dataframes:
## explicitly use model.matrix to ensure the design matrix is constructed
## correctly
tmp <- data %>%
  as_tibble %>%
  dplyr::select(Cell, Replicate, Clutter.) %>%
  group_by(Cell) %>%
  mutate(
    site_num = cur_group_id()
  ) %>%
  mutate(
    visit = 1:n()
  ) %>%
  ungroup %>%
  dplyr::select(site_num, visit) %>%
  mutate(
    clutter1 = design.matrix.surveyed[,2],
    clutter2 = design.matrix.surveyed[,3],
    clutter3 = design.matrix.surveyed[,4],
    clutter4 = design.matrix.surveyed[,5],
    tmin_scale = design.matrix.surveyed[,6],
    water_ind = design.matrix.surveyed[,7]
  )
```

We then use the design matrix to create the extended tables of detection covariates, and store them all in one list.

```
det_cov_list <- list()

det_cov_list[["clutter1"]] <- tmp %>%
  dplyr::select(site_num, visit, clutter1) %>%
  pivot_wider(names_from = "visit", values_from = "clutter1") %>%
  dplyr::select(-1) %>%
  as.matrix

det_cov_list[["clutter2"]] <- tmp %>%
  dplyr::select(site_num, visit, clutter2) %>%
  pivot_wider(names_from = "visit", values_from = "clutter2") %>%
  dplyr::select(-1) %>%
  as.matrix

det_cov_list[["clutter3"]] <- tmp %>%
  dplyr::select(site_num, visit, clutter3) %>%
  pivot_wider(names_from = "visit", values_from = "clutter3") %>%
  dplyr::select(-1) %>%
  as.matrix

det_cov_list[["clutter4"]] <- tmp %>%
  dplyr::select(site_num, visit, clutter4) %>%
  pivot_wider(names_from = "visit", values_from = "clutter4") %>%
  dplyr::select(-1) %>%
  as.matrix

det_cov_list[["tmin_scale"]] <- tmp %>%
  dplyr::select(site_num, visit, tmin_scale) %>%
  pivot_wider(names_from = "visit", values_from = "tmin_scale") %>%
  dplyr::select(-1) %>%
  as.matrix

det_cov_list[["water_ind"]] <- tmp %>%
  dplyr::select(site_num, visit, water_ind) %>%
  pivot_wider(names_from = "visit", values_from = "water_ind") %>%
  dplyr::select(-1) %>%
  as.matrix
```

2.3.2 Format occurrence covariates

Because they do not differ across surveys, the occurrence covariates are formatted in the same dataframe. From the `occ_covs_all` dataframe we filter down to only the variables we want to include, and only the surveyed cells.

```
#select just the sampled cells to use for fitting the model:
occ_covs0 <- occ_covs_all %>% filter(Cell %in% Cell.codes) %>%
  select("Cell", "Forest_PercentCover", "CliffsCanyons_PercentCover",
         "MeanAnnualPrecipitation_mm" )
rownames(occ_covs0) <- occ_covs0[,1]; occ_covs0 <- occ_covs0[,-1]

occ_covs_unscaled <- occ_covs0 %>% select("Forest_PercentCover", "CliffsCanyons_PercentCover",
```

```
                                          "MeanAnnualPrecipitation_mm")
```

We scale the occurrence covariates based on their values across the entire study area. This step is necessary to allow prediction to unsurveyed sample locations, and is further explained under model specification.

```
Forest_PercentCover_scaled <-
  (occ_covs_unscaled$Forest_PercentCover -  mean(occ_covs_all$Forest_PercentCover)) / sd(occ_covs_all$F

CliffsCanyons_PercentCover_scaled <-
  (occ_covs_unscaled$CliffsCanyons_PercentCover - mean(occ_covs_all$CliffsCanyons_PercentCover)) / sd(o

MeanAnnualPrecipitation_mm_scaled <-
  (occ_covs_unscaled$MeanAnnualPrecipitation_mm - mean(occ_covs_all$MeanAnnualPrecipitation_mm)) / sd(o

occ_covs <-
  as.data.frame(
    cbind(Forest_PercentCover_scaled, CliffsCanyons_PercentCover_scaled, MeanAnnualPrecipitation_mm_sca
```

2.3.3 Final formatting

Finally, we put the dataframes into the final list format that the `spOccupancy` package will accept.

```
data.LANO <- list(y = as.matrix(y), occ.covs =occ_covs, det.covs = det_cov_list)
```

The final data read into the `spOccupancy` package is as follows:

Presence absence data for each of the cells (rows) at each of the site-nights (columns)

```
head(data.LANO$y)
```

```
##       1  2  3  4  5  6  7
## 95426 0  0  0  0 NA NA NA
## 95901 0  0  0  0 NA NA NA
## 96343 1 NA NA NA NA NA NA
## 96350 0  1  0  1 NA NA NA
## 96362 1  0  0  0 NA NA NA
## 96364 0  0  0  0 NA NA NA
```

```
dim(data.LANO$y)
```

```
## [1] 168   7
```

Values of occurrence covariates (columns) for each of the cells (rows)

```
head(data.LANO$occ.covs)
```

```
##   Forest_PercentCover_scaled CliffsCanyons_PercentCover_scaled
## 1                 -1.2399718                        -0.2373433
## 2                 -1.2423919                        -0.2373433
## 3                  0.2406816                        -0.2373433
## 4                 -1.2268047                         0.6532027
```

```
## 5                 -1.2422066                          -0.1847246
## 6                 -1.2424183                          -0.2373433
##   MeanAnnualPrecipitation_mm_scaled
## 1                        -0.9130015
## 2                        -0.8768420
## 3                        -0.6897861
## 4                        -0.8944476
## 5                        -0.9205880
## 6                        -0.9197770
```

```
dim(data.LANO$occ.covs)
```

```
## [1] 168   3
```

List of dataframes for each of the detection covariates

```
summary(data.LANO$det.covs)
```

```
##            Length Class  Mode
## clutter1   1176   -none- numeric
## clutter2   1176   -none- numeric
## clutter3   1176   -none- numeric
## clutter4   1176   -none- numeric
## tmin_scale 1176   -none- numeric
## water_ind  1176   -none- numeric
```

Each detection covariate dataframe has values at each of the cells (rows) on each of the site-nights (columns)

```
head(data.LANO$det.covs$clutter1)
```

```
##      1  2  3  4  5  6  7
## [1,] 0  0  0  0 NA NA NA
## [2,] 0  1  1  0 NA NA NA
## [3,] 0 NA NA NA NA NA NA
## [4,] 1  1  0  1 NA NA NA
## [5,] 1  1  1  0 NA NA NA
## [6,] 0  1  0  0 NA NA NA
```

```
dim(data.LANO$det.covs$clutter1)
```

```
## [1] 168   7
```

3 Methods

We fit a single-season single-species nonspatial occupancy model, assuming no false-positives, using the `spOccupancy` package from Doser et al. (2022).

3.1 Model assumptions

It is important to explicitly consider all assumptions we make when drawing inferences from our estimated model. The first two of these assumptions cannot be evaluated with the available data, while the latter three

assumptions are assessed using residual diagnostics within the model assessment section.

- No false-positives present in data
  - Positive detection of a species at a site-night (replicate) is not prone to false-positive errors. E.g., if a site-night has a 1 for the species, there has been at least one manual confirmation of the species at that site-night.
- Closure
  - Cells are closed to changes in occurrence within the season (in this case, the summer). The bat species is assumed to either be present or absent from each cell for the entire summer survey season. This means *all* bats of a species do not migrate from a cell or die within a cell within the season, and bats do not emigrate to an unoccupied cell within the season (though an individual bat may leave a cell or die within a cell).
- No unmodeled heterogeneity in p or ψ
  - Variation in detection probability and occurrence probability are fully explained by the detection and occurrence covariates included in the model.
  - Residuals are used to assess if there is unmodeled heterogeneity after fitting a given model.
- Independence of occurrence across cells
  - Occurrence of the bat species is both spatially and temporally independent across cells. This means there is no unmodeled correlation remaining for occurrence probabilities in nearby cells or in a single cell over seasons, if multiple seasons of data were available.
  - Because we have only a single year of data, we do not check for temporal independence. However, we do assume that there is no spatial correlation in the occurrence states.
  - We evaluate the assumption of spatial independence using the Moran's I statistic for the occurrence residuals.
- Independence of detection across site-nights
  - Detection of the bat species is both spatially and temporally independent across surveys.
  - If there is spatial replication, then we assume there is no unmodeled spatial correlation in detections at nearby acoustic detectors. If temporal replicates are used, then there is assumed to be no unmodeled temporal correlation in detections. In a spatial replicate design, temporal correlation in detection may manifest as spatial correlation in detection due to spatially close monitors being deployed on consecutive nights (Wright, Irvine, and Higgs 2019).
  - Temporal correlation in temporal replicates can be evaluated by examining detection residuals over time, however we do not check for residual temporal correlation in our model because the

data does not contain temporal replicates.

- We evaluate the assumption of spatial independence using the Moran's I statistic for the detection residuals.

3.2 Model specification

We specify a classic single-season, single-species occupancy model following MacKenzie et al. (2002). For cells i, $i = 1, \ldots, N$, with N cells in the study area, the true presence z_i is modeled as

$$z_i \sim Bernoulli(\psi_i).$$

The probability of occurrence in cell i (ψ_i) is modeled using the logit-link based on occurrence covariates, $\boldsymbol{X}$:

$$logit(\psi_i) = \boldsymbol{x}_i^T \boldsymbol{\beta}.$$

Data sampled with imperfect probability of detection ($p_{i,j}$) at cell i on survey j, $j = 1, \ldots, 4$, are modeled as

$$y_{i,j} \sim \ Bernoulli(p_{i,j} z_i).$$

Detection probability is modeled using the logit link based on detection covariates $\boldsymbol{V}$:

$$\text{logit}(p_{i,j}) = \boldsymbol{v}_{i,j}^T \boldsymbol{\alpha}.$$

Priors

Priors are specified as multivariate normal for occurrence and detection coefficients $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$. Priors are automatically set to be weakly informative with means 0 and variances 2.72, which translates to an approximately uniform prior for the scaled variables on the probability scale. These hyperparameters for the priors can be changed using `priors = list(beta.normal = list(mean = 0, var = 2.72), alpha.normal = list(mean = 0, var = 2.72))`, but we utilize the automatic weakly informative priors.

Scale variables for model fitting

We scale continuous occurrence and detection covariates to have mean 0 and standard deviation of 1 for model fitting, with occurrence covariates scaled across the entire study area. Scaling facilitates selection of priors and manages possible differences in scales of covariate measurements. Scaling was performed in the data formatting section.

3.3 Fit model to data

The `spOccupancy` package utilizes Póly-Gamma data augmentation to implement speedy Gibbs updates for sampling of all model parameters.

Setting MCMC sampling parameters

`spOccupancy` model fitting requires the number of samples per sample chain (`n.samples`), burn-in to discard from these samples (`n.burn`), interval to thin remaining samples by (`n.thin`), and the number of chains to sample (`n.chains`). These settings result in the following number of total samples: `((n.samples - n.burn)/n.thin)*n.chains`. For this analysis, we selected the following settings:

- We sampled 3 chains, which allows for confirmation of convergence.
- We set a large burn in of 2000 and ensured this was sufficiently large by examining traceplots in the convergence assessment section.
- Initial values for parameters can be set using `inits = list(alpha =  , beta =  , z = )`. Setting initial values for parameters based on one run of model fitting may accelerate convergence but is not necessary.
  - Default initial values are set as random values from the prior distributions for $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ respectively, and as the recorded naive occurrence value for z (1 if observed, 0 if unobserved).

`n.thin` and `n.samples` should be set at some initial value (e.g., those provided) then evaluated in the convergence assessment section and iteratively changed if necessary based on the convergence evaluation.

Fit model to data

```
n.chains <- 3
out <- PGOcc(
  occ.formula = ~
    Forest_PercentCover_scaled + CliffsCanyons_PercentCover_scaled +
    MeanAnnualPrecipitation_mm_scaled,
    #include linear effects of all occurrence covariates
  det.formula = ~ clutter1 + clutter2 + clutter3 + clutter4 + tmin_scale +  water_ind,
  #include linear effects of all detection covariates
  data = data.LANO,
  verbose = F,
  n.samples = 40000,
  n.thin = 4,
  n.burn = 2000,
  n.chains = n.chains)
# Model fit is cached so that the same model fit is used each time the document is knit.
```

3.4 Model assessment

3.4.1 Convergence assessment

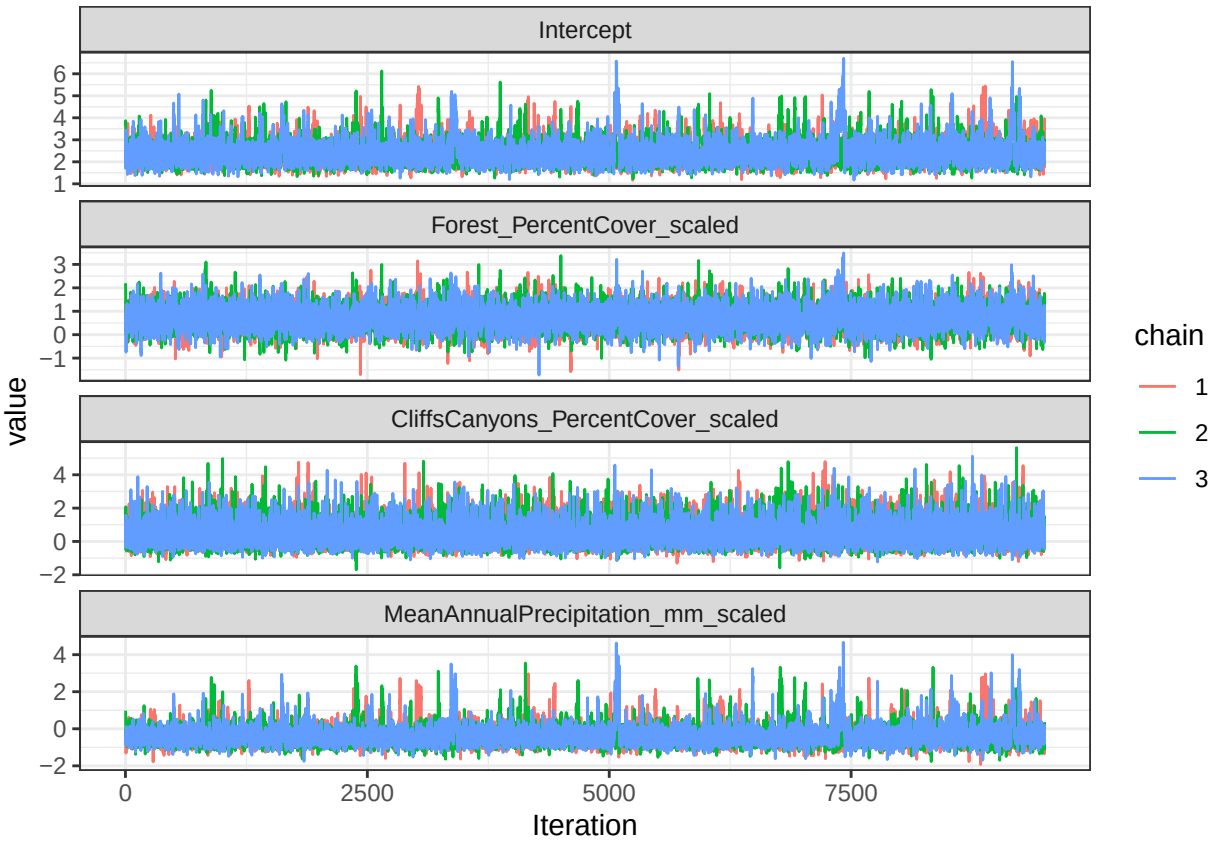
Before making inference from our fitted model, we must first confirm that chains have converged within our MCMC sampling algorithm (Hobbs and Hooten 2015). Converged chains are well mixed and explore the posterior efficiently to represent the true shape of the posterior distribution. Additionally, they contain large enough sample sizes to remain stationary with additional samples. We investigate chain convergence by both examining the chains visually and using diagnostic metrics.

Visually examine trace plots

We can first visually examine trace plots for convergence by plotting chains for each parameter. Trace plots indicative of convergence look like random scatter around a mean value and do not display any trends. For these reasons, trace plots indicative of convergence are often described as looking like "fuzzy caterpillars." Below, the trace plots for both the occurrence and detection coefficients are consistent with model convergence.

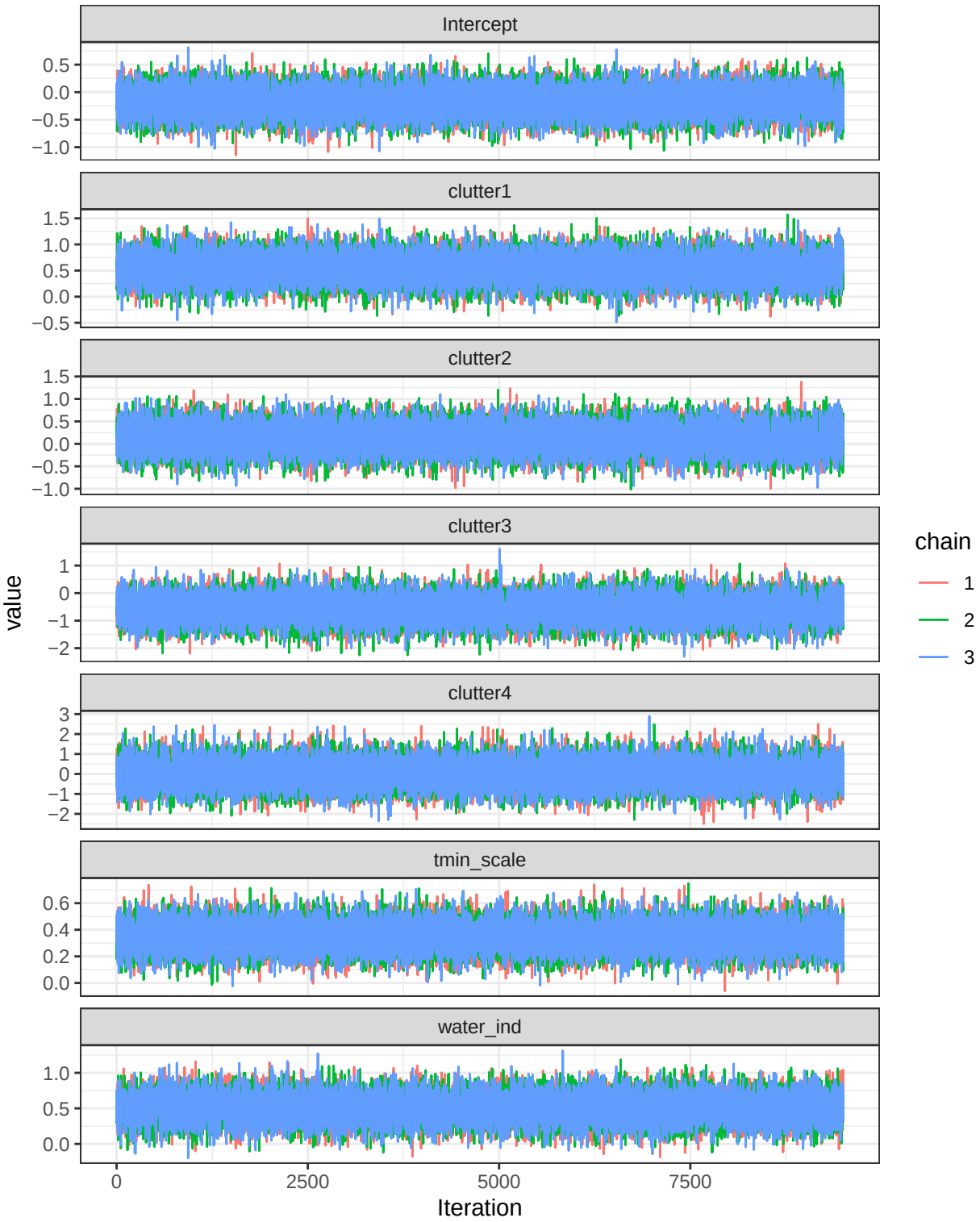
Occurrence parameters:

```
occ.samps <- as.data.frame(out$beta.samples)
#the spocc sampling did sample 3 chains, but then stacked them on top of eachother
#to plot them separately we have to divide them back into chains:
occ.samps$x <- rep(1:(nrow(occ.samps)/n.chains), n.chains)
occ.samps$chain <- as.factor(rep(1:3, each =nrow(occ.samps)/n.chains))

# convert to long for plotting
names(occ.samps)[1] <- "Intercept"
occ.samps.long <- gather(
    occ.samps, parameter, value,
    Intercept:MeanAnnualPrecipitation_mm_scaled, factor_key = T
  )

ggplot(occ.samps.long) +
  geom_line(
    aes(x = x, y = value,
      col = chain, group = chain)) +
  labs(x ="Iteration") +
  facet_wrap(~ parameter, nrow = length(unique(occ.samps.long$parameter)), scales = "free_y") +
  theme_bw()
```



Detection parameters:

```
det.samps <- as.data.frame(out$alpha.samples)
#the spocc sampling did sample 3 chains, but then stacked them on top of eachother
#to plot them separately we have to divide them back into chains:
det.samps$x <- rep(1:(nrow(det.samps)/n.chains), n.chains)
det.samps$chain <- as.factor(rep(1:3, each =nrow(det.samps)/n.chains))

# convert to long for plotting
names(det.samps)[1] <- "Intercept"
det.samps.long <- gather(det.samps, parameter, value, Intercept:water_ind, factor_key = T )

ggplot(det.samps.long) +
  geom_line(
    aes(x = x, y = value,
      col = chain, group = chain)) +
  labs(x ="Iteration") +
  facet_wrap(~ parameter, nrow = length(unique(det.samps.long$parameter)), scales = "free_y") +
  theme_bw()
```



Gelman-Rubin diagnostics

The Gelman-Rubin diagnostic, $\hat{r}$, formally assesses model convergence by comparing variance of parameter estimates among chains to variance of parameter estimates within chains. A large $\hat{r}$ means variance among

chains is greater than within chains, implying lack of convergence. When assessing convergence it is standard to accept $\hat{r}$ values below 1.1 (Hobbs and Hooten 2015). The $\hat{r}$ value can be found for each parameter in the summary output under `Rhat`.

```
summary(out)
```

```
##
## Call:
## PGOcc(occ.formula = ~Forest_PercentCover_scaled + CliffsCanyons_PercentCover_scaled +
##     MeanAnnualPrecipitation_mm_scaled, det.formula = ~clutter1 +
##     clutter2 + clutter3 + clutter4 + tmin_scale + water_ind,
##     data = data.LANO, n.samples = 40000, verbose = F, n.burn = 2000,
##     n.thin = 4, n.chains = n.chains)
##
## Samples per Chain: 40000
## Burn-in: 2000
## Thinning Rate: 4
## Number of Chains: 3
## Total Posterior Samples: 28500
## Run Time (min): 0.4002
##
## Occurrence (logit scale):
##                                    Mean     SD    2.5%     50%  97.5%   Rhat
## (Intercept)                      2.4210 0.5327  1.6675  2.3308 3.8254 1.0065
## Forest_PercentCover_scaled       0.7319 0.4827 -0.1811  0.7092 1.7621 1.0008
## CliffsCanyons_PercentCover_scaled  0.4357 0.7347 -0.5665  0.2738 2.3300 1.0008
## MeanAnnualPrecipitation_mm_scaled -0.3388 0.5165 -1.0794 -0.4117 1.0111 1.0051
##                                    ESS
## (Intercept)                       2331
## Forest_PercentCover_scaled        7070
## CliffsCanyons_PercentCover_scaled 10904
## MeanAnnualPrecipitation_mm_scaled  2473
##
## Detection (logit scale):
##                Mean     SD    2.5%     50%  97.5%   Rhat   ESS
## (Intercept) -0.1729 0.2377 -0.6430 -0.1727 0.2935 1.0003 26332
## clutter1     0.5396 0.2503  0.0502  0.5411 1.0313 1.0007 26854
## clutter2     0.1226 0.2983 -0.4561  0.1229 0.7040 1.0001 16787
## clutter3    -0.5726 0.4497 -1.4601 -0.5736 0.2958 1.0006 17956
## clutter4    -0.0061 0.6325 -1.2322 -0.0137 1.2777 1.0000 29062
## tmin_scale   0.3474 0.1011  0.1493  0.3468 0.5446 1.0002 22135
## water_ind    0.5006 0.1846  0.1422  0.5004 0.8626 1.0002 28011
```

We conclude satisfactory $\hat{r}$ values for all parameters in this model.

Effective sample size

Effective sample size (ESS) estimates the number of samples the MCMC sample would contain if correlated samples were removed, providing a measure of the information contained in the MCMC sample. If ESS is low compared to the MCMC sample size this may imply poor mixing and high autocorrelation, and thinning may

help to address this. Larger ESS indicates a better sample, but there is no distinct rule for a "large enough" ESS.

We compare the ESS (denoted as `ESS` in the summary output) to the MCMC sample size and confirm there is a large ESS for each parameter.

```
summary(out)
```

```
##
## Call:
## PGOcc(occ.formula = ~Forest_PercentCover_scaled + CliffsCanyons_PercentCover_scaled +
##     MeanAnnualPrecipitation_mm_scaled, det.formula = ~clutter1 +
##     clutter2 + clutter3 + clutter4 + tmin_scale + water_ind,
##     data = data.LANO, n.samples = 40000, verbose = F, n.burn = 2000,
##     n.thin = 4, n.chains = n.chains)
##
## Samples per Chain: 40000
## Burn-in: 2000
## Thinning Rate: 4
## Number of Chains: 3
## Total Posterior Samples: 28500
## Run Time (min): 0.4002
##
## Occurrence (logit scale):
##                                    Mean     SD    2.5%     50%  97.5%   Rhat
## (Intercept)                      2.4210 0.5327  1.6675  2.3308 3.8254 1.0065
## Forest_PercentCover_scaled       0.7319 0.4827 -0.1811  0.7092 1.7621 1.0008
## CliffsCanyons_PercentCover_scaled  0.4357 0.7347 -0.5665  0.2738 2.3300 1.0008
## MeanAnnualPrecipitation_mm_scaled -0.3388 0.5165 -1.0794 -0.4117 1.0111 1.0051
##                                    ESS
## (Intercept)                       2331
## Forest_PercentCover_scaled        7070
## CliffsCanyons_PercentCover_scaled 10904
## MeanAnnualPrecipitation_mm_scaled  2473
##
## Detection (logit scale):
##                Mean     SD    2.5%     50%  97.5%   Rhat   ESS
## (Intercept) -0.1729 0.2377 -0.6430 -0.1727 0.2935 1.0003 26332
## clutter1     0.5396 0.2503  0.0502  0.5411 1.0313 1.0007 26854
## clutter2     0.1226 0.2983 -0.4561  0.1229 0.7040 1.0001 16787
## clutter3    -0.5726 0.4497 -1.4601 -0.5736 0.2958 1.0006 17956
## clutter4    -0.0061 0.6325 -1.2322 -0.0137 1.2777 1.0000 29062
## tmin_scale   0.3474 0.1011  0.1493  0.3468 0.5446 1.0002 22135
## water_ind    0.5006 0.1846  0.1422  0.5004 0.8626 1.0002 28011
```

Set sampling parameters to reach satisfactory diagnostics

If any of the three diagnostics (trace plots, Gelman-Rubin Diagnostics, and ESS) are not satisfactory for any of the parameters, adjust the sampling parameters (`n.thin` and `n.samples`) and resample. For example, we first ran this model with `n.thin = 2` and `n.samples = 9000` and got unsatisfactory trace plots and ESSs

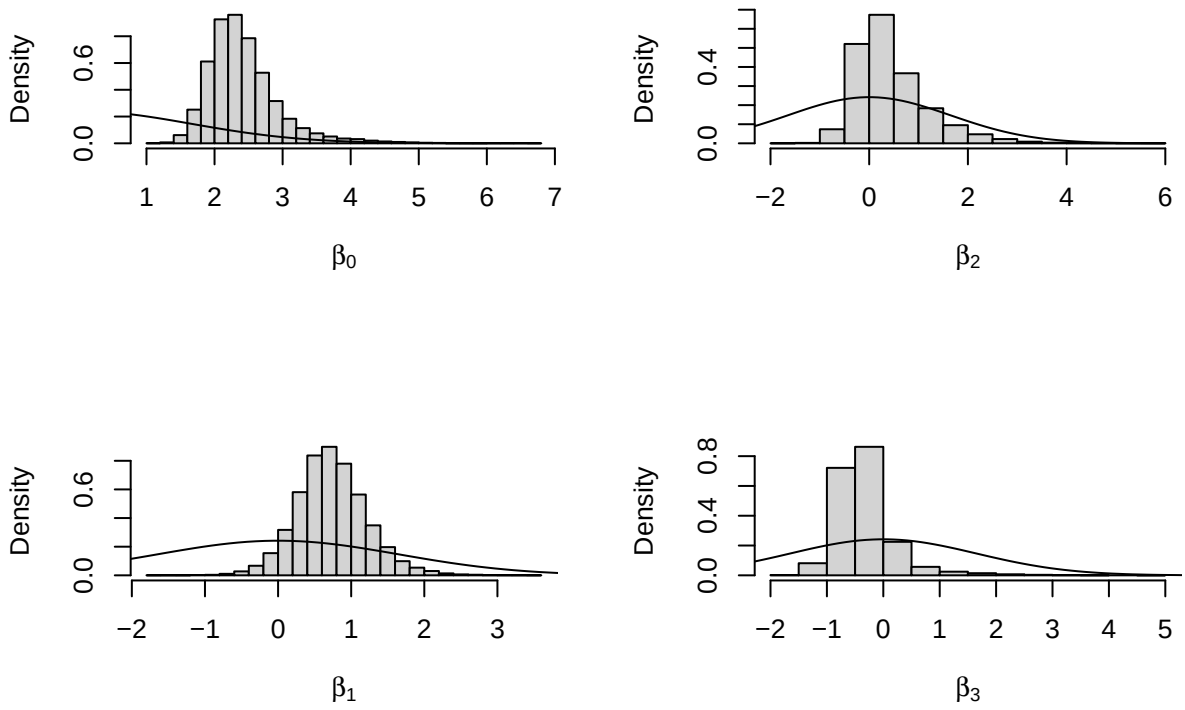
for the occurrence coefficients. Trace plots were showing spikes indicating high autocorrelation and poor mixing, so we increased the thinning interval and total number of posterior samples.

3.4.2 Sensitivity analysis

After evaluating convergence of our MCMC sampler, we check the effect of the priors on the posterior distributions of parameters. Plotting the posterior distributions of the estimated parameters ($\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$) over the prior density for those parameters (plotted by a solid line) evaluates how sensitive the model is to prior choice. In general, we prefer to see that the prior distributions are diffuse relative to the posterior distributions, suggesting our choice of prior distribution is weakly influential.

$\boldsymbol{\beta}$ (occurrence) plots:

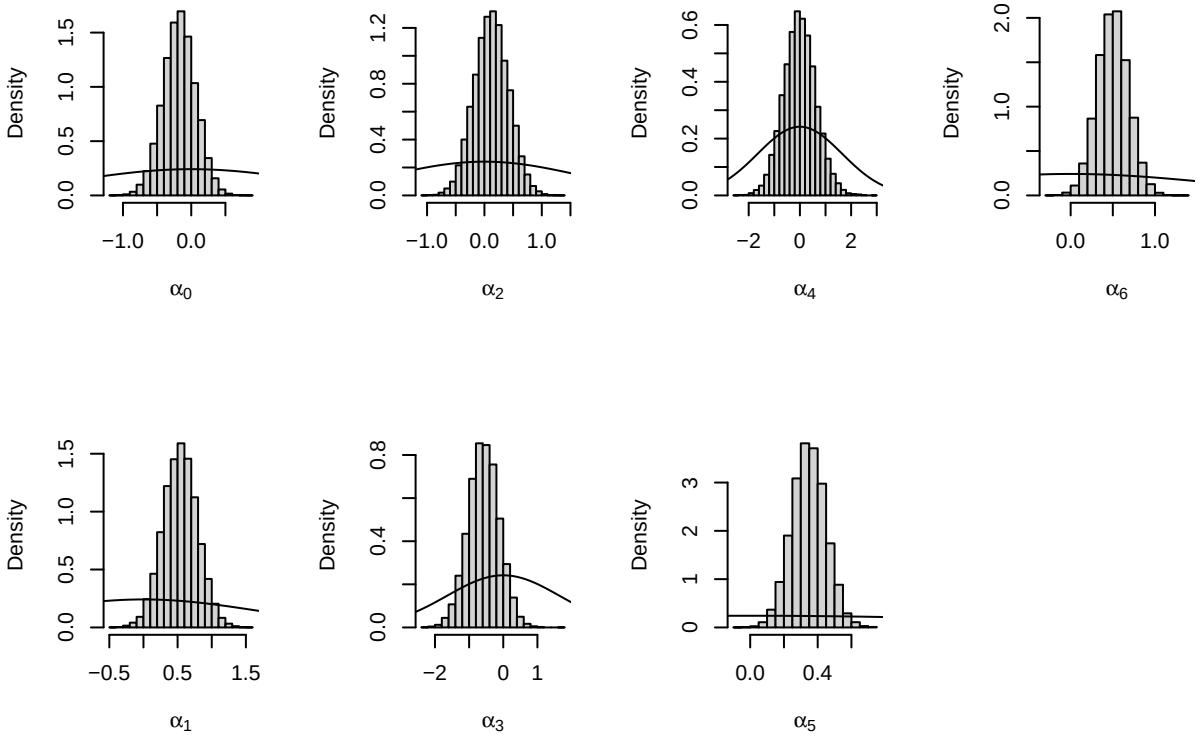
```
occ.samps <- as.data.frame(out$beta.samples)

layout(matrix(1:4,2,2))
x <- seq(-6, 6, length=100)

hist(
  occ.samps$`(Intercept)`,breaks = 20, freq = F,
  xlab = bquote(beta[0]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve
hist(
  occ.samps$Forest_PercentCover_scaled,breaks = 20,
  freq = F, xlab = bquote(beta[1]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve
hist(occ.samps$CliffsCanyons_PercentCover_scaled,breaks = 20, freq = F,
     xlab = bquote(beta[2]), main = "")
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve
hist(occ.samps$MeanAnnualPrecipitation_mm_scaled,breaks = 20, freq = F,
     xlab = bquote(beta[3]), main = "")
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve
```



$\boldsymbol{\alpha}$ *(detection) plots:*

```
det.samps <- as.data.frame(out$alpha.samples)

layout(matrix(1:8,2,4))
x <- seq(-6, 6, length=100)

hist(
  det.samps$`(Intercept)`,breaks = 20, freq = F,
  xlab = bquote(alpha[0]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve

hist(
  det.samps$clutter1, breaks = 20, freq = F,
  xlab = bquote(alpha[1]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve

hist(
  det.samps$clutter2, breaks = 20, freq = F,
  xlab = bquote(alpha[2]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve

hist(
  det.samps$clutter3, breaks = 20, freq = F,
  xlab = bquote(alpha[3]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve
```

```r
hist(
  det.samps$clutter4, breaks = 20, freq = F,
  xlab = bquote(alpha[4]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve

hist(
  det.samps$tmin_scale,breaks = 20, freq = F,
  xlab = bquote(alpha[5]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve

hist(
  det.samps$water_ind,breaks = 20, freq = F,
  xlab = bquote(alpha[6]), main = ""
)
lines(x, dnorm(x, 0, sqrt(2.72)), type = "l") #plot the prior curve
```



Sensitivity plots for both sets of parameters show that priors are not having substantial effects on the parameter estimates, which was our intention in setting weakly informative priors.

3.4.3 Residual diagnostics

After assessing MCMC samples for convergence and effects of priors, it is important to check for violations of model assumptions. First we use residuals to check for violations of spatial or temporal independence at both the occurrence and detection levels (Wright, Irvine, and Higgs 2019). Separately analyzing both occurrence

and detection residuals allows for separate investigation of the two components of the hierarchical model. After evaluating residuals for spatial or temporal dependence, we use residuals to evaluate overall goodness of fit of the model and check if there are unmodeled relationships remaining in the residuals, addressing the assumption of no unmodeled heterogeneity remaining in p or ψ. Plotting detection and occurrence residuals against fitted values and individual covariates allows for examination of where lack of fit may exist in the model (Warton et al. 2017).

We compute residuals as conditional on the latent z state (occupied or unoccupied) as defined by Wright et al. (2021). Though Warton et al. (2017) provides an alternative definition of occupancy model residuals, our model's hierarchical Bayesian formulation facilitates use of the estimated latent z state to compute residuals at each MCMC iteration.

Occurence residuals are computed at each MCMC iteration t from the draws of the latent occupied state and occurrence probability at that iteration:

$$o_i^{[t]} = z_i^{[t]} - \psi_i^{[t]}.$$

Detection residuals are computed at each MCMC iteration t where the draw of the latent occupied state is occupied. These conditional detection residuals are computed at each survey from the observed detection and the draw of the detection probability:

$$[d_{ij}^{[t]} | z_i^{[t]} = 1] = y_{ij} - p_{ij}^{[t]}.$$

Residuals are computed individually at each of the MCMC iterations, necessitating visual examination of multiple iterations of each residual plot in order to visualize the variation in residual diagnostics across the MCMC sample.

Binned residuals

To evaluate residuals from discrete results (e.g., occupied or not, detected or not) we assess binned residuals, as recommended by Gelman et al. (2000). Binned residuals offer increased interpretability of residual plots because they hold the property that the binned residuals are expected to be symmetric around 0 (given enough observations within a bin for the central limit theorem to hold) (Gelman et al. 2000). This means that when interpreting the binned residual plots we look for symmetry around zero and lack of a pattern. Stark deviations from this suggest a relationship in the data that was not captured by our model structure.

Bins are defined as having approximately equal numbers of observations within each bin, not by bin width

(Gelman et al. 2000). The binning executed in the following functions for detection and occurrence residuals can be summarized as:

1) Set the number of bins. This is a trade off between having enough bins that plots can be interpreted, and having enough observations in each bin that binned residuals are expected to display symmetry. This will depend on the number of cells and the number of site-nights.
2) Order the variable to be plotted against residuals (e.g., the covariate or fitted values). Divide the ordered variable and its associated residuals into the bins, with approximately equal numbers of observations in each bin.
3) Average variable values and residual values within each bin.

3.4.3.1 Residual analysis for spatial or temporal dependence Residuals allow us to check if there is unmodeled spatial or temporal autocorrelation remaining. Because our example contains spatial and not temporal replication, we use the Moran's I statistic to measure the spatial autocorrelation in our previously defined occurrence and detection residuals, following the methods of Wright, Irvine, and Higgs (2019). Checking for spatial autocorrelation evaluates our assumptions that detection events are independent across site-night surveys and that occurrence is independent across cells (Warton et al. 2017). The Moran's I statistic is used to create correlograms that summarize the spatial autocorrelation in residuals for continuous response variables (hence the use of binning for binary responses). We use distance classes of 15 km, following Wright, Irvine, and Higgs (2019).

We use the function written by Wright, Irvine, and Higgs (2019) to compute a Moran's I empirical correlegram from a distance matrix and a single draw of residuals.

```
correl_fun <- function(dist.mat, resid.vec, dist.incr){
  # function to calculate moran's from Wright et al 2021
  n <- length(resid.vec)
  resid.bar <- mean(resid.vec)
  resid.var <- var(resid.vec)
  resid.scale <- resid.vec - resid.bar
  resid.pairs <- outer(resid.scale, resid.scale)
  resid.pairs <- resid.pairs[lower.tri(resid.pairs)]

  dgrp.mat <- ceiling(dist.mat / dist.incr)
  dgrp.mat <- dgrp.mat[lower.tri(dgrp.mat)]

  dist.mat <- dist.mat[lower.tri(dist.mat)]

  moran <- tapply(resid.pairs, dgrp.mat, mean, na.rm = TRUE) * n / (n - 1) / resid.var
  dist.means <- tapply(dist.mat, dgrp.mat, mean, na.rm = TRUE)
  return(list(dist = dist.means, moran = moran))
}
```

Because each set of residuals depends on the MCMC iteration, a single MCMC iteration will produce an empirical correlogram using the above function. We plot correlograms from 100 randomly drawn MCMC iterations simultaneously to express the posterior variability in spatial autocorrelation (Wright, Irvine, and Higgs 2019).

Occurrence

Checking for spatial autocorrelation in the occurrence residuals allows us to evaluate the assumption that occurrence is independent across cells. If clear violation of spatial independence across cells is found, a spatial model should be considered.

We construct a function to create plottable output for Moran's I statistics from occurrence residuals. The function computes occurrence residuals, creates a distance matrix from the spatial NABat grid cells, estimates Moran's I statistics from those occurrence residuals and the distance matrix using the Wright, Irvine, and Higgs (2019) function, and performs this at a chosen number of iterations that can then be simultaneously plotted.

```
morans_occ <- function(
    out, #fitted model output to get residuals from
    data, #occurrence data we've been using
    grid_shp, #shape file for grid cells -
    #does not need to contain center, it will be calculated within function
    niter){ #number of randomly selected MCMC samples to perform this over

  ###  build occurrence distance matrix:
  occ_dist_m <- data %>%
    dplyr::select(Cell) %>%
    distinct() %>%
    mutate(Cell = as.character(Cell)) %>%
    left_join(
      .,
      grid_shp %>%
        dplyr::select(Cell, geometry) %>%
        mutate(Cell = as.character(Cell)),
      by = "Cell"
    ) %>%
    st_as_sf %>%
    st_centroid %>%
    mutate(
      x_centroid = st_coordinates(.)[,1],
      y_centroid = st_coordinates(.)[,2]
    ) %>%
    as_tibble %>%
    dplyr::select(x_centroid, y_centroid) %>%
    as.matrix %>%
    dist() %>%
    as.matrix
```

```
  ## convert to km
  occ_dist_km <- occ_dist_m / 1000

  # create storage
  out_ <- list()

  z.rep <- as.data.frame(out$z.samples) # samples of latent z state across all sites
  psi <- as.data.frame(out$psi.samples) # samples of latent psi across all sites

  # select iterations
  total_samples <- nrow(z.rep)
  iters <- sample(1:total_samples, size = niter, replace = F)

  # loop through iterations and produce occurrence residual vector at one mcmc iter:
  for(ndx in 1:niter){

    iter_ <- iters[ndx]

    # get residual at ONE mcmc sample:
    resid_raw_occ <- as.numeric(z.rep[iter_, ] - psi[iter_, ])

    # calculate moran's i with Wright's function
    moran <- correl_fun(occ_dist_km, resid_raw_occ, 15) #Use distance lag of 15 km

    out_[[ndx]] <- tibble(
      dist = moran$dist,
      moran = moran$moran
    )  %>%
      mutate(
        iter = iter_
      )
  }

  return(do.call("rbind", out_))

}
```

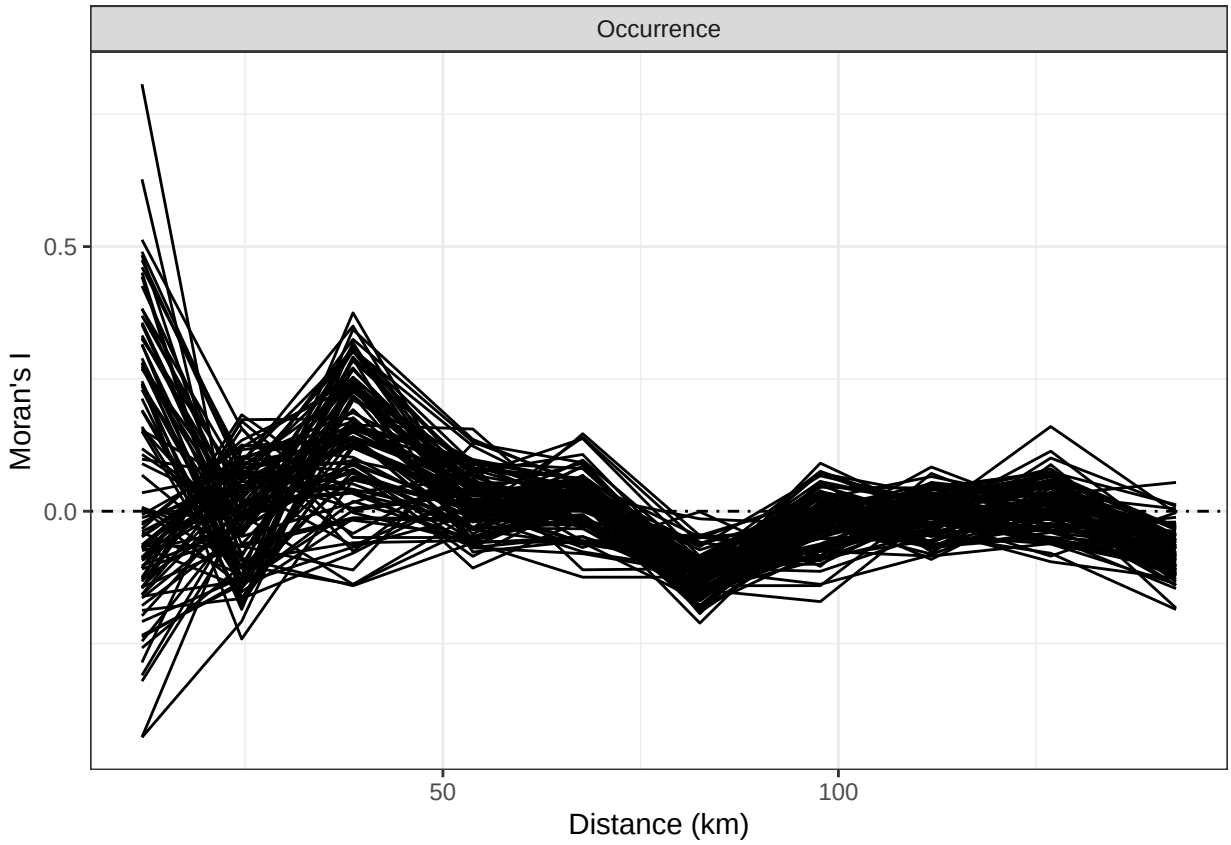
Run function and plot:

```
morans_occ <- morans_occ(out, data, grid_shp, 100)
```

```
## Warning: st_centroid assumes attributes are constant over geometries
```

```
morans_occ %>%
  mutate(type = "Occurrence") %>%
  filter(dist <= 150) %>%
  ggplot() +
  geom_line(aes(x = dist, y = moran, group = iter)) +
  theme_bw() +
  geom_hline(aes(yintercept = 0), linetype = "dotdash") +
  labs(
    y = "Moran's I",
    x = "Distance (km)"
```

```
) +
facet_wrap(~type)
```



We do not see a strong pattern in the occurrence correlogram and the value of Moran's I quickly approaches zero, suggesting that any remaining spatial correlation is negligible.

Detection

Checking for spatial autocorrelation in the detection residuals allows us to evaluate the assumption that site-night replicates are spatially independent with regards to detection. If clear violation of spatial independence across detectors is found, a model should be considered that explicitly accounts for this autocorrelation.

We construct a function to create plottable output for Moran's I statistics from detection residuals. The function computes detection residuals, creates a distance matrix from the detector locations, estimates Moran's I statistics from those detection residuals and the distance matrix using the Wright, Irvine, and Higgs (2019) function, and performs this at a chosen number of iterations that can then be simultaneously plotted.

```r
morans_det <- function(
    out, #fitted model output to get residuals from
    data_spatial, #occurrence data with spatial information for each detection
    design.matrix.surveyed,
    niter){ #number of randomly selected MCMC samples to perform this over

  # create storage
  out_ <- list()

  # select iterations
  iters <- sample(1:nrow(out$alpha.samples), size = niter, replace = F)

  # loop through iterations
  pb <- txtProgressBar(min = 0, max = niter, style = 3,  width = 50, char = "=")
  for(ndx in 1:niter){
    iter_ <- iters[ndx]

    # construct replicated z vector
    z.rep <- out$z.samples[iter_,] # samples of latent z state across all sites
    psi <- out$psi.samples[iter_,] # samples of latent psi across all sites

    # have to expand across the visits so each z is replicated for its corresponding visits:
    nvisits <- data_spatial %>% dplyr::select(Cell, Replicate) %>%
      group_by(Cell) %>%
      summarize(nvisit = n()) %>%
      dplyr::select(nvisit) %>%
      unlist %>% unname
    z.rep.exp <- rep(z.rep, nvisits)
    y_ <- data_spatial$LANO

    # filter to z == 1
    keep_ndx <- which(z.rep.exp == 1)

    # grab y's associated with filtered z's
    y_filtered <- y_[keep_ndx]

    # grab p's associated with filtered z's
    alpha <- as.vector(out$alpha.samples[iter_,])
    mult <- design.matrix.surveyed[keep_ndx,] %*% alpha

    # convert to probability
    p <- c(exp(mult) / (1 + exp(mult)))
    resid_raw_det <- y_filtered - p

    # create distance matrix
    det_dist_m <- data_spatial %>%
      slice(keep_ndx) %>%
      mutate(Cell = as.character(Cell)) %>%
      st_as_sf %>%
      mutate(
        x_tmp = st_coordinates(.)[,1],
        y_tmp = st_coordinates(.)[,2]
      ) %>%
```

```
      as_tibble %>%
      dplyr::select(x_tmp, y_tmp) %>%
      as.matrix %>%
      dist() %>%
      as.matrix
    det_dist_km <- det_dist_m/1000

    # calculate moran's i with Wright's function
    moran <- correl_fun(det_dist_km, resid_raw_det, 15)
    out_[[ndx]] <- tibble(
      dist = moran$dist,
      moran = moran$moran
    )  %>%
      mutate(
        iter = iter_
      )
    setTxtProgressBar(pb, ndx)
  }
  close(pb)

  return(do.call("rbind", out_))
}
```

Our `data` object only contains lat and long, so before running the spatial function for detection, we need to add spatial geometries and UTM coordinates to the detection data.

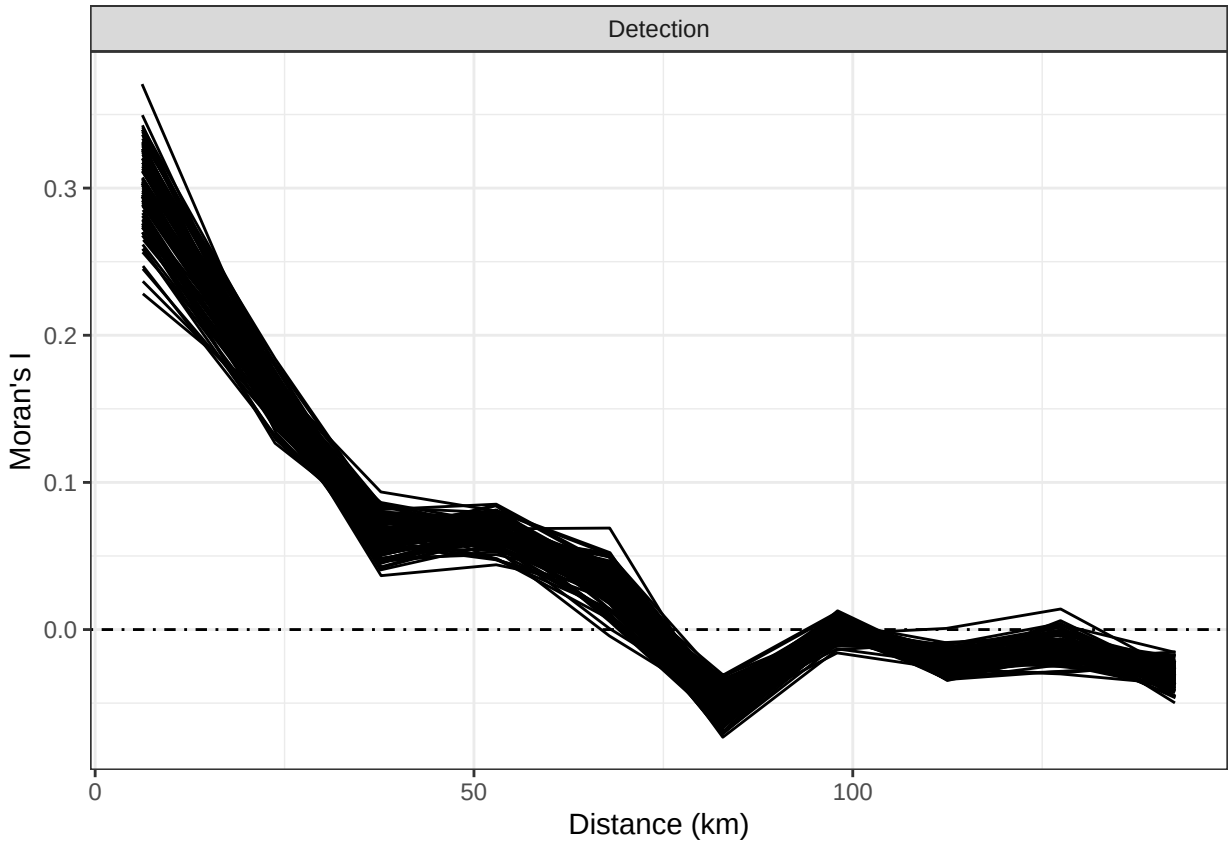
```
data_spatial = data %>%
  as_tibble %>%
  mutate(Cell = as.character(Cell)) %>%
  st_as_sf(coords = c("long", "lat"), crs = 4269) %>%
  mutate(
    x_coord = st_coordinates(.)[,1],
    y_coord = st_coordinates(.)[,2]
  ) %>%
  st_transform(., crs = "+proj=utm +zone=10") %>%
    mutate(
      x_utm = st_coordinates(.)[,1],
      y_utm = st_coordinates(.)[,2]
    ) %>%
  as_tibble
```

Run function and plot:

```
morans_det(out, data_spatial, design.matrix.surveyed , 100) %>%
  mutate(type = "Detection") %>%
  filter(dist <= 150) %>%
  ggplot() +
  geom_line(aes(x = dist, y = moran, group = iter)) +
  theme_bw() +
  geom_hline(aes(yintercept = 0), linetype = "dotdash") +
  labs(
    y = "Moran's I",
```

```
    x = "Distance (km)"
  ) +
  facet_wrap(~type)
```

```
##   |                                                                      |
```



We do see a pattern in the Moran's I correlogram for detection residuals, with correlation in detection residuals decreasing as distance between detectors increases. This implies unmodeled spatial autocorrelation remains between detection probabilities at nearby detectors, suggesting the need to explicitly model spatial autocorrelation of detection probabilities; see Wright, Irvine, and Higgs (2019) for an example.

Temporal replication

If our example had temporal replication in addition to spatial replication, we would similarly want to check the assumption of independence of detection events in temporal replicates. Additionally, if our design included multiple seasons of data we would want to check for temporal autocorrelation in occurrence of cells over seasons (years). Both types of temporal autocorrelation would be assessed by examining plots of binned residuals over lagged time.

3.4.3.2 Residual analysis for model fit

3.4.3.2.1 Residuals vs. fitted values Analysis of residuals plotted against fitted occurrence or detection probabilities allows for evaluation of the structural components of the model and determining if all relevant relationships have been modeled.

Occurrence

We plot the fitted occurrence values against binned residuals to check for overall lack of fit in the occurrence component of the model. In general, we do not see any consistent trends or patterns in the plots, suggesting no obvious unaccounted for structure in the occurrence portion of the model.

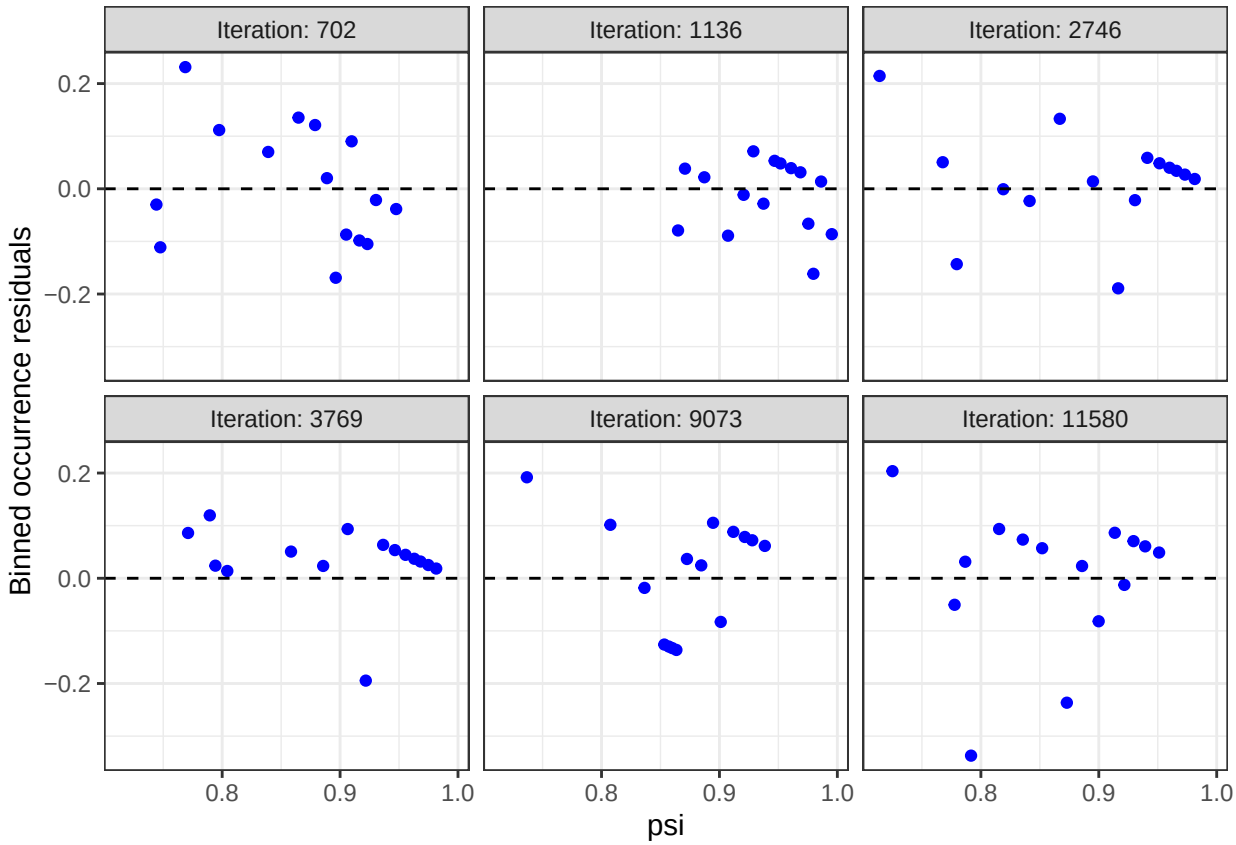
Function to compute binned occurrence residuals (based on a set number of bins) at a chosen number of iterations to plot:

```
occ_binned <- function( out, #fitted model output to get residuals from
                        data,
                        cov_vec, #covariates from the data (just surveyed site-nights)
                        numbins,
                        niter){ # the number of randomly selected MCMC samples

  # create storage
  out_ <- list()

  # select iterations
  iters <- sample(1:nrow(out$z.samples), size = niter, replace = F)

  # loop through iterations
  for(ndx in 1:niter){
    iter_ <- iters[ndx]

    # construct replicated z vector
    z.rep <- out$z.samples[iter_,] # samples of latent z state across all sites
    psi <- out$psi.samples[iter_,] # samples of latent psi across all sites
    #remember that model output and data are ordered to match.

    # get residual at one mcmc sample:
    resid_raw_occ <- as.numeric(z.rep - psi)

    ### now do binned residuals at that iteration:

    #in each bin we have:
    binsize <- floor(length(psi)/numbins)
    bin <- rep(1:numbins, each = binsize)
    leftover <- length(psi)%%numbins # with some leftover (why they're approx equal sized
    # groups) which will be added to the first group
    #(adding to first groups because anticipating having big zero groups)
    if (leftover > 0) {bin <- c(rep(1, leftover), bin)} #add leftover to first group

    # put it in a table so that each covariate value is staying with it's residual:
    binned <- as.data.frame(cbind(psi, resid_raw_occ, cov_vec))
```

```
    # 1) order the covariates (or in this case the psi)
    binned <- binned %>% arrange(psi)
    # 2) sort into bins with approx equal number of points in each
    binned$bin <- bin
    # 3) average within the bins
    df <- aggregate(x = binned, by = list(binned$bin), FUN = mean)
    df$sample <- iter_

    ## store:
    out_[[ndx]] <- tibble(
      bin = df$bin,
      psi = df$psi,
      cov_vec = df$cov_vec,
      resid_raw = df$resid_raw_occ,
      Iteration = df$sample)
  }

  return(do.call("rbind", out_))
}
```

Plot:

```
# set seed for reproducible iterations for .pdf document
set.seed(12042023)
ggplot( occ_binned(out, data, cov_vec=NA ,15, 6) ) +
  geom_point(aes(x = psi, y = resid_raw), color = "blue") +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw() +
  facet_wrap(~ Iteration, labeller = "label_both") +
  labs( y = "Binned occurrence residuals")
```



Detection

We plot the fitted detection values against binned residuals to check for overall lack of fit in the detection component of the model. Here again, we do not see any consistent trends or structure in the detection residuals.

Function to compute binned detection residuals (based on a set number of bins) at a chosen number of iterations to plot:

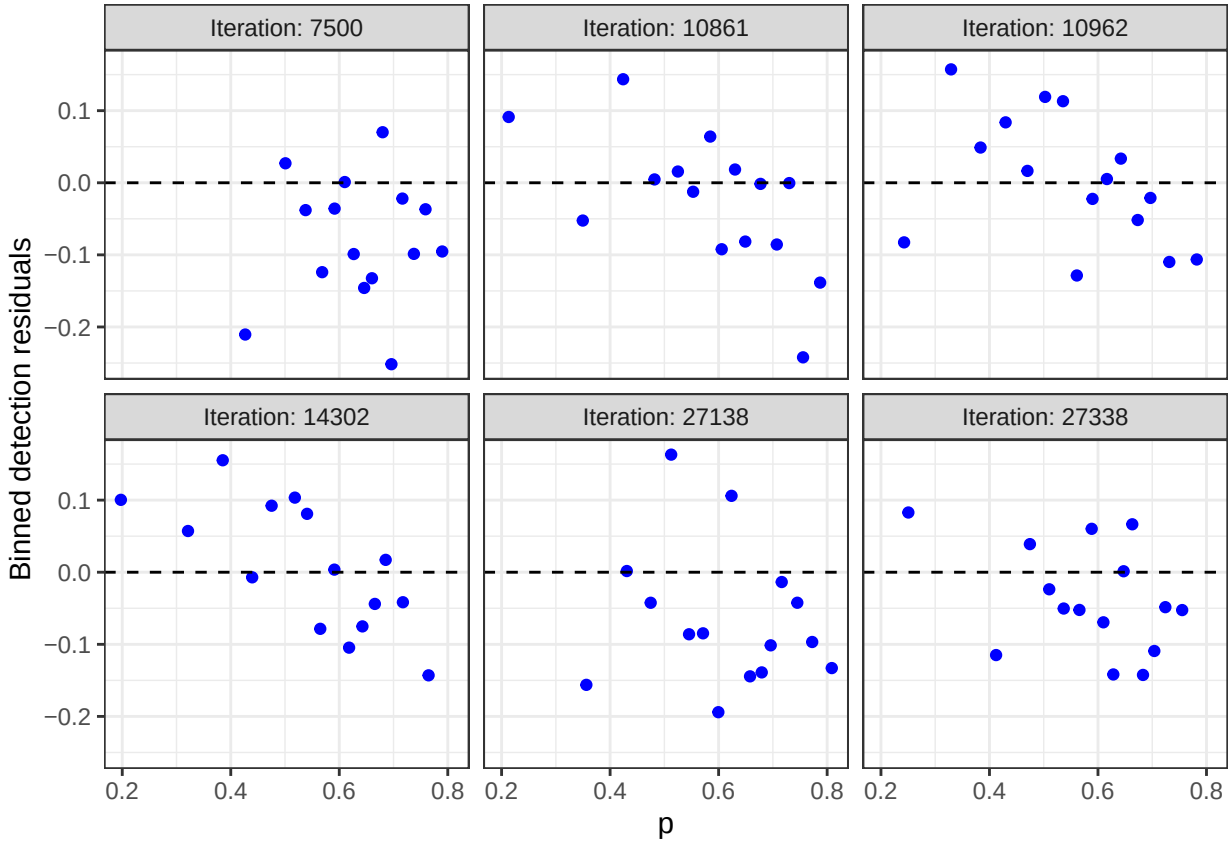
```
det_binned <- function( out, #fitted model output to get residuals from
                        data,
                        design.matrix.surveyed,
                        cov_vec, #covariates from the data (just surveyed site-nights)
                        numbins,
                        niter){ # the number of randomly selected MCMC samples

  # create storage
  out_ <- list()

  # select iterations
  iters <- sample(1:nrow(out$alpha.samples), size = niter, replace = F)

  # loop through iterations
  for(ndx in 1:niter){
```

```
  iter_ <- iters[ndx]

  # construct replicated z vector
  z.rep <- out$z.samples[iter_,] # samples of latent z state across all sites
  psi <- out$psi.samples[iter_,] # samples of latent psi across all sites
  #remember that model output and data are ordered to match.

  # have to expand across the visits so each z is replicated for its corresponding visits:
  #(this is the same as remaking the y vector from the data, but with no unsurveyed sites,
  #to match the no unsurveyed sites design matrix)
  nvisits <- data_spatial %>% dplyr::select(Cell, Replicate) %>%
    group_by(Cell) %>%
    summarize(nvisit = n()) %>%
    dplyr::select(nvisit) %>%
    unlist %>% unname
  z.rep.exp <- rep(z.rep, nvisits)
  y_ <- data_spatial$LANO

  # filter to z == 1
  keep_ndx <- which(z.rep.exp == 1)

  # grab y's and cov_vecs associated with filtered z's
  y_filtered <- y_[keep_ndx]
  cov_vec_filtered <- cov_vec[keep_ndx]

  # grab p's associated with filtered z's
  alpha <- as.vector(out$alpha.samples[iter_,])
  mult <- design.matrix.surveyed[keep_ndx,] %*% alpha

  # convert to probability
  p <- c(exp(mult) / (1 + exp(mult)))
  resid_raw_det <- y_filtered - p

  ### now do binned residuals at that iteration:

  #in each bin we have:
  binsize <- floor(length(p)/numbins)
  bin <- rep(1:numbins, each = binsize)
  leftover <- length(p)%%numbins # with some leftover (why they're approx equal sized
  # groups) which will be added to the first group
  #(adding to first groups because anticipating having big zero groups)
  if (leftover > 0) {bin <- c(rep(1, leftover), bin)} #add leftover to first group

  # put it in a table so that each covariate value is staying with it's residual:
  binned <- as.data.frame(cbind(p, resid_raw_det, cov_vec_filtered))

  # 1) order the covariates (or in this case the probability)
  binned <- binned %>% arrange(p)
  # 2) sort into bins with approx equal number of points in each
  binned$bin <- bin
  # 3) average within the bins
  df <- aggregate(x = binned, by = list(binned$bin), FUN = mean)
  df$sample <- iter_
```

```
    ## store:
    out_[[ndx]] <- tibble(
      bin = df$bin,
      p = df$p,
      cov_vec = df$cov_vec_filtered,
      resid_raw = df$resid_raw_det,
      Iteration = df$sample)
  }

  return(do.call("rbind", out_))
}
```

Plot:

```
ggplot(  det_binned(out, data, design.matrix.surveyed, cov_vec = NA,15, 6)  ) +
  geom_point(aes(x = p, y = resid_raw), color = "blue") +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw() +
  facet_wrap(~ Iteration, labeller = "label_both") +
  labs( y = "Binned detection residuals")
```



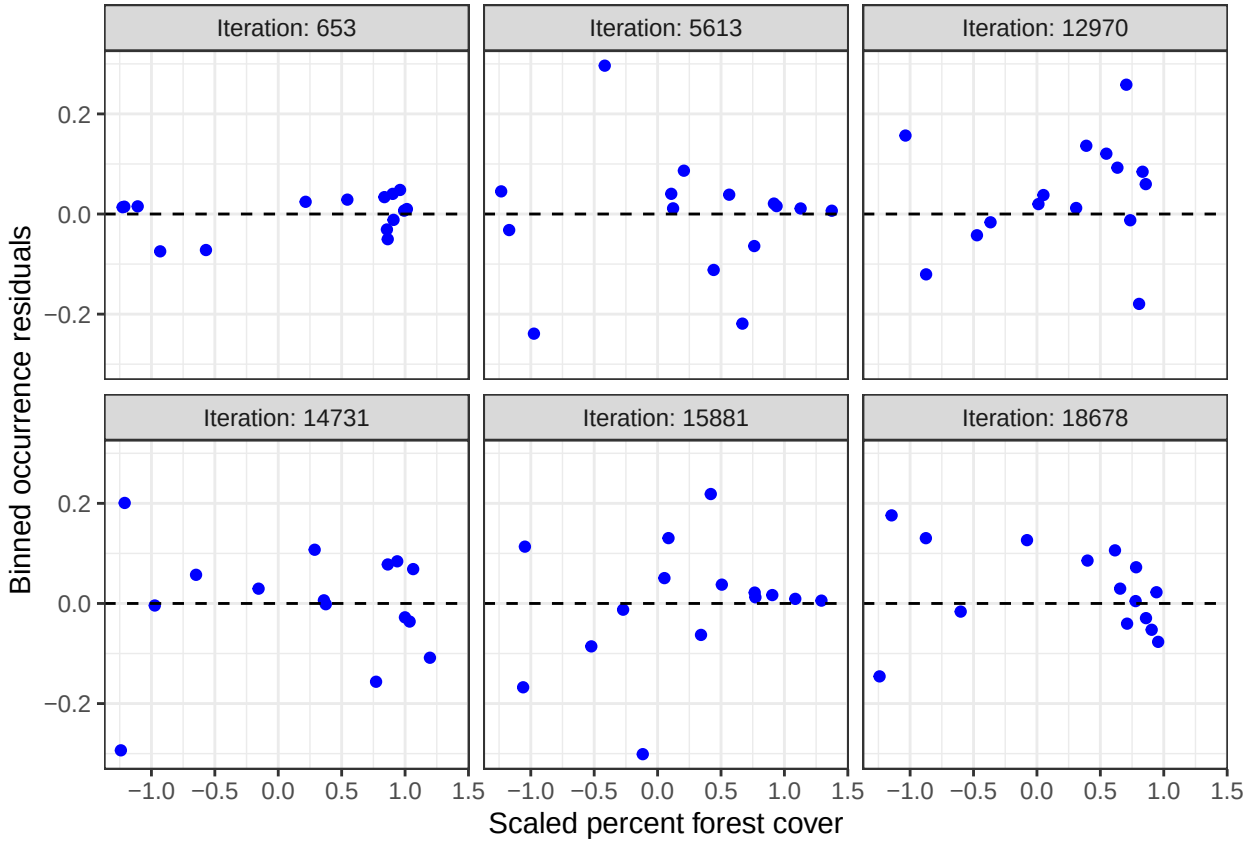
3.4.3.2.2 Residuals vs. covariates

Occurrence

We plot each of the occurrence covariates against their binned residuals to check if our model is capturing the relationship between that covariate and occurrence well (e.g., if another functional form of a covariate may be required in the model).

Forest cover:

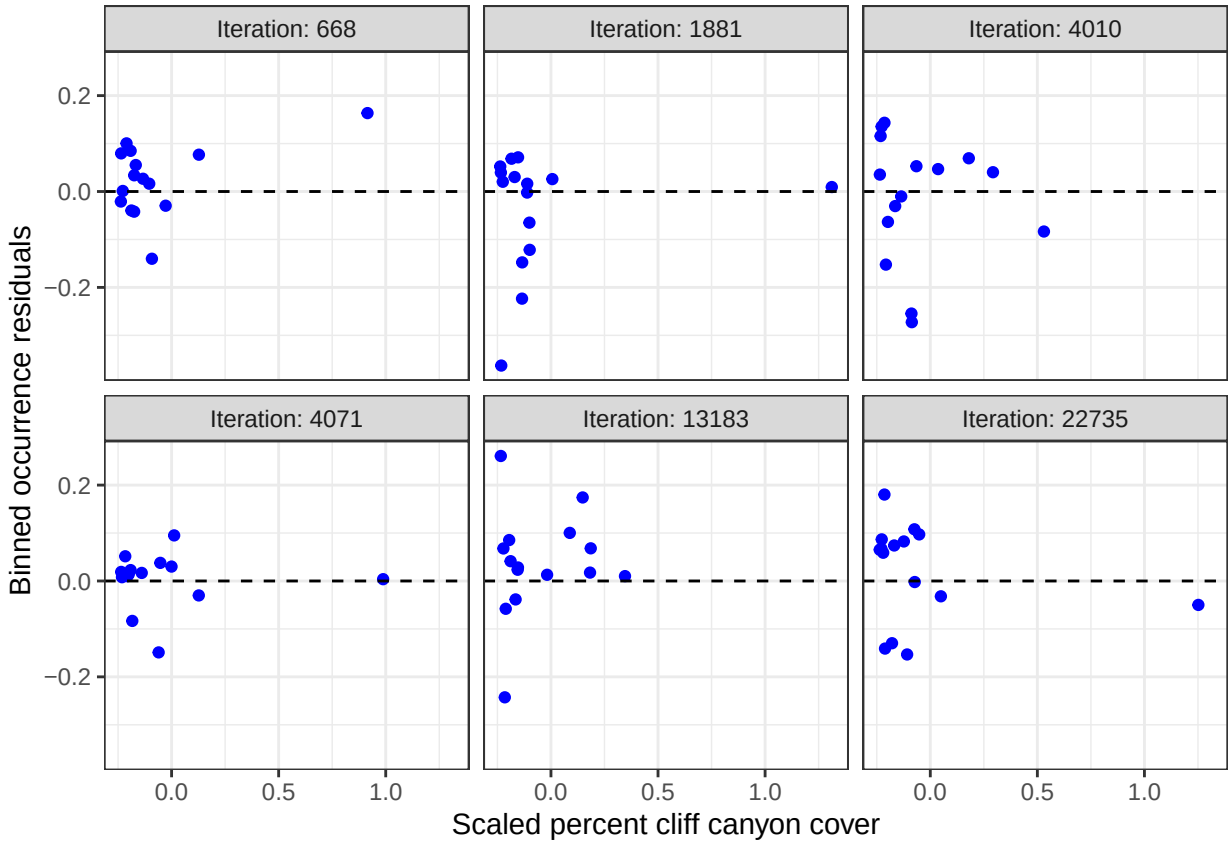
```
#covariance vector is for each cell:
cov_vec <- occ_covs$Forest_PercentCover_scaled

ggplot(  occ_binned(out, data, cov_vec,15, 6) ) +
  geom_point(aes(x = cov_vec, y = resid_raw), color = "blue") +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw() +
  facet_wrap(~ Iteration, labeller = "label_both") +
  labs( y = "Binned occurrence residuals", x = "Scaled percent forest cover")
```



Cliff canyon cover:

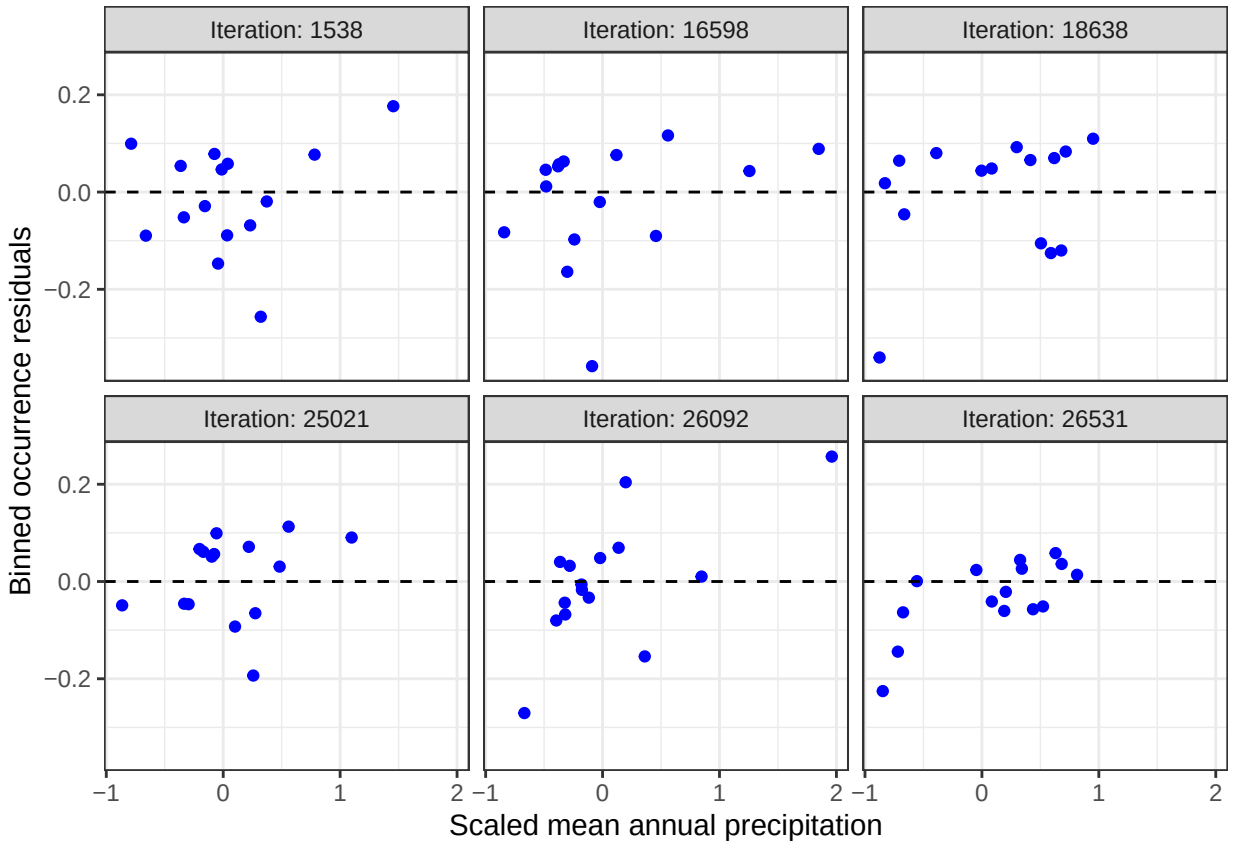
```
#covariance vector is for each cell:
cov_vec <- occ_covs$CliffsCanyons_PercentCover_scaled

ggplot(  occ_binned(out, data, cov_vec,15, 6) ) +
  geom_point(aes(x = cov_vec, y = resid_raw), color = "blue") +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw() +
```

```
  facet_wrap(~ Iteration, labeller = "label_both") +
  labs( y = "Binned occurrence residuals", x = "Scaled percent cliff canyon cover")
```



Mean annual precipitation:

```
#covariance vector is for each cell:
cov_vec <- occ_covs$MeanAnnualPrecipitation_mm_scaled

ggplot(  occ_binned(out, data, cov_vec,15, 6) ) +
  geom_point(aes(x = cov_vec, y = resid_raw), color = "blue") +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw() +
  facet_wrap(~ Iteration, labeller = "label_both") +
  labs( y = "Binned occurrence residuals", x = "Scaled mean annual precipitation")
```



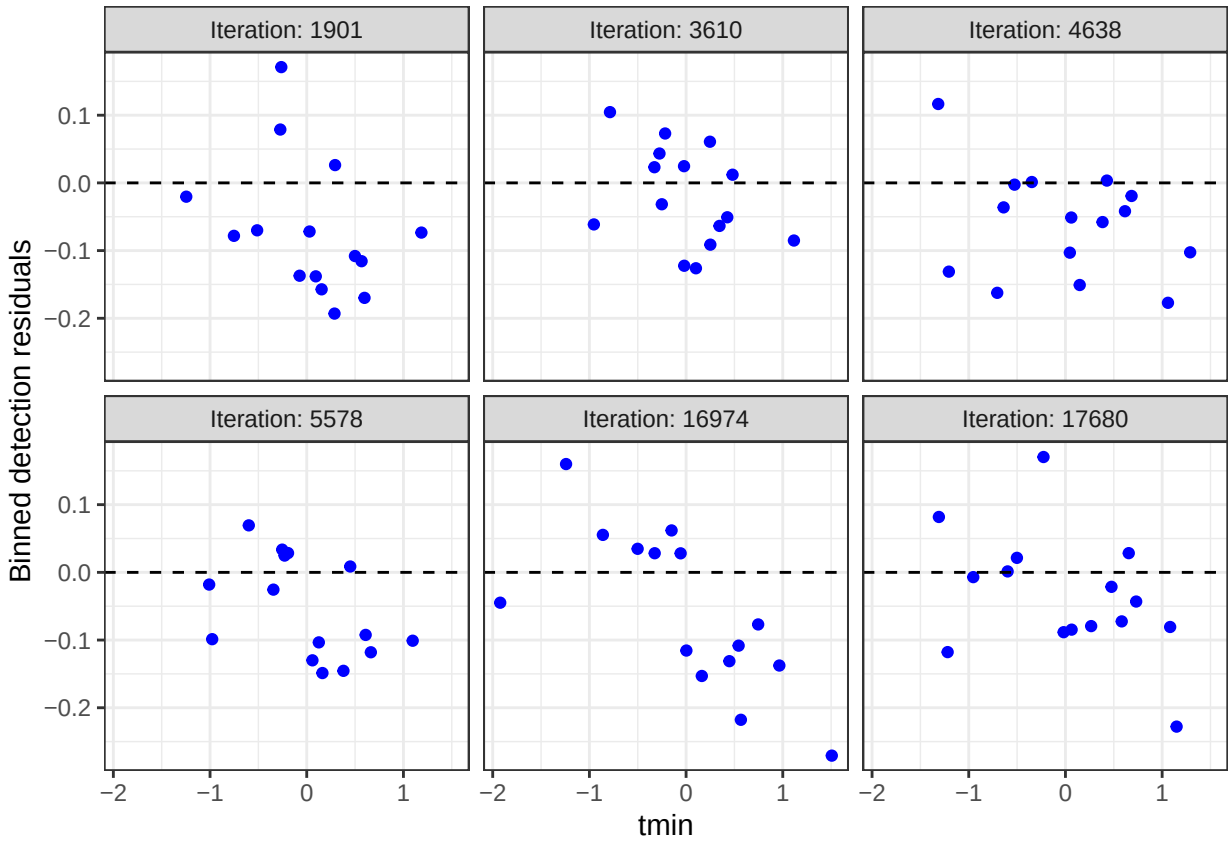
We do not see any consistent trends or patterns in the plots, meaning there is no implied lack of fit for the model with regards to that covariate.

Detection

We plot each of the continuous detection covariates against their binned residuals to check if our model is capturing the relationship between that covariate and detection well (e.g., if another functional form of a covariate may be required in the model).

Minimum temperature:

```
#covariate vector is for all surveyed site-nights
cov_vec <- as.numeric(t(scale(data$tmin)))

ggplot( det_binned(out, data, design.matrix.surveyed, cov_vec,15, 6) ) +
  geom_point(aes(x = cov_vec, y = resid_raw), color = "blue") +
  geom_hline(yintercept = 0, lty = 2) +
  theme_bw() +
  facet_wrap(~ Iteration, labeller = "label_both") +
  labs( y = "Binned detection residuals", x = "tmin")
```



We do not see any consistent trends or patterns in the plots, meaning there is no implied lack of fit for the model with regards to that covariate.

3.4.4 Posterior predictive checks

After using residuals to confirm that there are no substantial unmodeled relationships or violations of independence, we now assess the ability of the model to capture the underlying process. Posterior predictive checks are the final component of model assessment that allows us to evaluate general goodness of fit of the model (i.e., if we've chosen a good deterministic model to represent the data). Posterior predictive checks look for systematic differences between the model and the data and evaluate if the model is predicting values that make sense (Gelman et al. 2000).

Posterior predictive distributions are defined as the distribution of a new predicted data point conditional on the existing data; for the occupancy model, we can use the posterior predictive distribution of the site-night observations. In practice, this is done by generating new Bernoulli random variables, representing new data, for each MCMC iteration based on the estimated occurrence state and probability of detection.

$$y_{i,j}^{new,[t]}|y_{i,j} \sim \ Bernoulli(p_{i,j}^{[t]} z_i^{[t]})$$

We obtain MCMC samples of this posterior predictive distribution from our original MCMC samples of the parameters and latent state z using composition sampling (Hobbs and Hooten 2015). The posterior predictive distribution for binary detected or undetected data can be difficult to represent, so we use multiple iterations of bar plots, with bars representing the total number of detections or nondetections at the sampled sites, for that MCMC iteration. Posterior predictive distributions that deviate strongly from observed data would motivate use of a different model (Hobbs and Hooten 2015; Gelman et al. 1995).

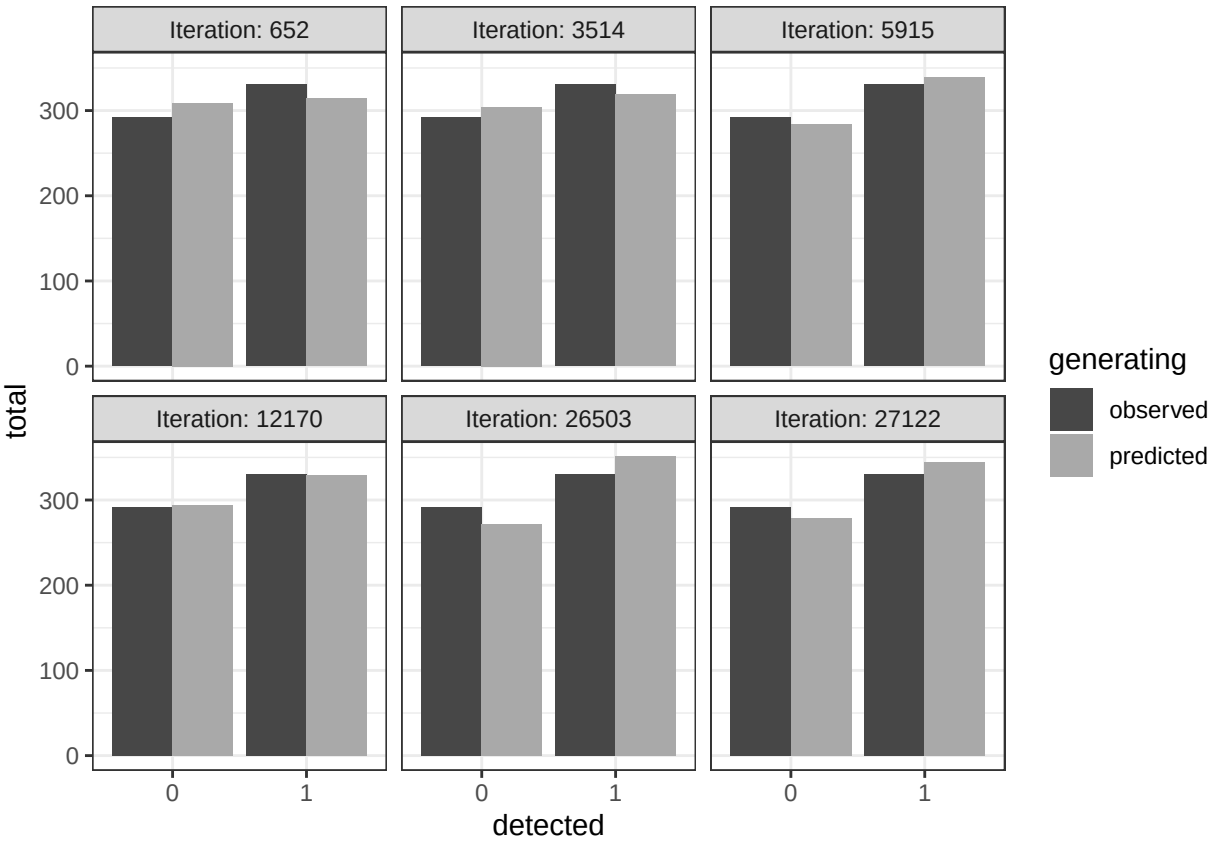
Custom function to output posterior predictive y values at one MCMC iteration:

```
y_new <- function( design.matrix.surveyed, out ){

  total_samples <- nrow(out$z.samples)
  iter_ = sample(1:total_samples, 1) #the mcmc sample we're examining

  # construct replicated z vector
  z.rep <- out$z.samples[iter_,] # samples of latent z state across all sites
  #remember that model output and data are ordered to match.

  # have to expand across the visits so each z is replicated for its corresponding visits:
  #(this is the same as remaking the y vector from the data, but with no unsurveyed sites,
  #to match the no unsurveyed sites design matrix)
  nvisits <- data_spatial %>% dplyr::select(Cell, Replicate) %>%
    group_by(Cell) %>%
    summarize(nvisit = n()) %>%
    dplyr::select(nvisit) %>%
    unlist %>% unname
  z.rep.exp <- rep(z.rep, nvisits)

  alpha <- as.vector(out$alpha.samples[iter_, ])
  mult <- design.matrix.surveyed %*% alpha

  # convert to probability
  p <- c(exp(mult) / (1 + exp(mult)))

  #formula for y_new:
  y_new <- rbinom(length(p), 1, prob= z.rep.exp*p)

  return(as.data.frame(cbind(y_new, iter_) ))
}
```

Output from this function is a vector of predicted detections for each surveyed site-night, at one MCMC iteration. To evaluate the posterior predictive distribution, it is necessary to compare predicted to observed values at multiple iterations. This can be accomplished by simultaneously examining plots for multiple randomly selected MCMC iterations.

Plot 6 realizations against the observed y values:

```r
plots <- list()
for(i in 1:6){ #loop through plots for each of the iterations
  generating <- c("observed", "observed", "predicted", "predicted" )
  detected <- c(0,1,0,1)
  total <- rep(NA, 4)
  plot <- as.data.frame(cbind(generating, detected, total))

  observed <- as.numeric(t(y)) # collapse detections into ij vector form
  #remove site-nights where surveys were not conducted (all NAs)
  observed <- na.omit(observed)
  plot[1,3]<- sum(observed==0)
  plot[2,3] <- sum(observed)

  predicted <- y_new(design.matrix.surveyed, out)
  plot[3,3]<- sum(predicted[,1]==0)
  plot[4,3] <- sum(predicted[,1])

  plot$total <- as.numeric(plot$total)
  plot$iter_ <- predicted$iter_ %>% unique

  plots[[i]] <- plot
}

do.call("rbind", plots) %>%
  rename(Iteration = iter_) %>%
  ggplot(aes(x=detected, y=total, fill=generating)) +
  geom_bar(stat="identity", position=position_dodge()) +
  scale_fill_manual(values = c("grey28", "dark gray"))  +
  facet_wrap(~ Iteration, labeller = "label_both") +
  theme_bw()
```



We do not see strong differences or a clear pattern between predicted and observed y values.

Bayesian p-value

A Bayesian p-value can be used to measure the significance of the deviation of the data from the model predictions, i.e., summarize the previous posterior predictive distribution plots in a single value (Gelman et al. 1995). To evaluate the level to which the distribution of predicted values aligns with the data, the Bayesian p-value relies on a test statistic or discrepency measure, which provides a scalar value that can be compared for observed data versus the posterior predictive distribution. The Bayesian p-value is then defined as the probability that the test statistic for the predicted data is more extreme than the test statistic for the observed data, given the observed data. Multiple test statistics can be used to evaluate model fit, and we use the Freeman-Tukey statistic recommended by Kéry and Royle (2016) and included within the `Spocc` package. Grouping or binning the binary data based on cells (`group=1`) or site-nights (`group=2`) provides two test statistics that may reveal different shortcomings of the model (i.e., problems of model fit in occurrence versus detection).

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

```
##
```

```
## Call:
## ppcOcc(object = out, fit.stat = "freeman-tukey", group = 1)
##
## Samples per Chain: 40000
## Burn-in: 2000
## Thinning Rate: 4
## Number of Chains: 3
## Total Posterior Samples: 28500
##
## Bayesian p-value:  0.1266
## Fit statistic:  freeman-tukey
```

```
ppc.out2 <- ppcOcc(out, fit.stat = 'freeman-tukey', group = 2)
summary(ppc.out2)
```

```
##
## Call:
## ppcOcc(object = out, fit.stat = "freeman-tukey", group = 2)
##
## Samples per Chain: 40000
## Burn-in: 2000
## Thinning Rate: 4
## Number of Chains: 3
## Total Posterior Samples: 28500
##
## Bayesian p-value:  0.9597
## Fit statistic:  freeman-tukey
```

A Bayesian p-value close to 0 or 1 implies the model does not accurately represent the distribution of the data (Hobbs and Hooten 2015). Note that the Bayesian p-value is a general goodness of fit statistic and won't provide more information about where in the model potential lack of fit may be occurring (Warton et al. 2017).

3.5 Prediction

We now use our fitted model to predict occurrence across all cells in Washington and Oregon. We use the predict function within the `spOcc` package, which intakes a design matrix of the covariate values to predict over. Note that the design matrix must match the order of covariates used to fit the model. The covariate values that we predict over must be standardized (scaled) in the same way that the variables were standardized to fit the model. This means they must be scaled against all covariate values.

```
fc.0 <- scale(occ_covs_all$Forest_PercentCover)
Cliffs.0 <- scale(occ_covs_all$CliffsCanyons_PercentCover)
Precip.0 <- scale(occ_covs_all$MeanAnnualPrecipitation_mm)

# Create prediction design matrix
X.0 <- cbind(1, fc.0, Cliffs.0, Precip.0)
```

```
out.pred <- predict(out, X.0)
psi.0.samples <- out.pred$psi.0.samples
#Prediction is not cached (due to storage constraints of git)
#and is computed each time the document is knit.
```

4 Results

4.1 Parameter plots

From the output of the fitted occupancy model, we can extract posterior distributions of model parameters and interpret them. These regression coefficients allow us to interpret effects of the covariates on detection and probability of occurrence. Ninety-five percent CIs for regression coefficients can be accessed quickly through the output summary.

```
summary(out)
```

```
##
## Call:
## PGOcc(occ.formula = ~Forest_PercentCover_scaled + CliffsCanyons_PercentCover_scaled +
##     MeanAnnualPrecipitation_mm_scaled, det.formula = ~clutter1 +
##     clutter2 + clutter3 + clutter4 + tmin_scale + water_ind,
##     data = data.LANO, n.samples = 40000, verbose = F, n.burn = 2000,
##     n.thin = 4, n.chains = n.chains)
##
## Samples per Chain: 40000
## Burn-in: 2000
## Thinning Rate: 4
## Number of Chains: 3
## Total Posterior Samples: 28500
## Run Time (min): 0.3962
##
## Occurrence (logit scale):
##                                    Mean     SD    2.5%     50%  97.5%   Rhat
## (Intercept)                      2.4210 0.5327  1.6675  2.3308 3.8254 1.0065
## Forest_PercentCover_scaled       0.7319 0.4827 -0.1811  0.7092 1.7621 1.0008
## CliffsCanyons_PercentCover_scaled  0.4357 0.7347 -0.5665  0.2738 2.3300 1.0008
## MeanAnnualPrecipitation_mm_scaled -0.3388 0.5165 -1.0794 -0.4117 1.0111 1.0051
##                                     ESS
## (Intercept)                        2331
## Forest_PercentCover_scaled         7070
## CliffsCanyons_PercentCover_scaled 10904
## MeanAnnualPrecipitation_mm_scaled  2473
##
## Detection (logit scale):
##                Mean     SD    2.5%     50%  97.5%   Rhat   ESS
## (Intercept) -0.1729 0.2377 -0.6430 -0.1727 0.2935 1.0003 26332
## clutter1     0.5396 0.2503  0.0502  0.5411 1.0313 1.0007 26854
## clutter2     0.1226 0.2983 -0.4561  0.1229 0.7040 1.0001 16787
## clutter3    -0.5726 0.4497 -1.4601 -0.5736 0.2958 1.0006 17956
## clutter4    -0.0061 0.6325 -1.2322 -0.0137 1.2777 1.0000 29062
```

```
## tmin_scale   0.3474 0.1011  0.1493  0.3468 0.5446 1.0002 22135
## water_ind    0.5006 0.1846  0.1422  0.5004 0.8626 1.0002 28011
```

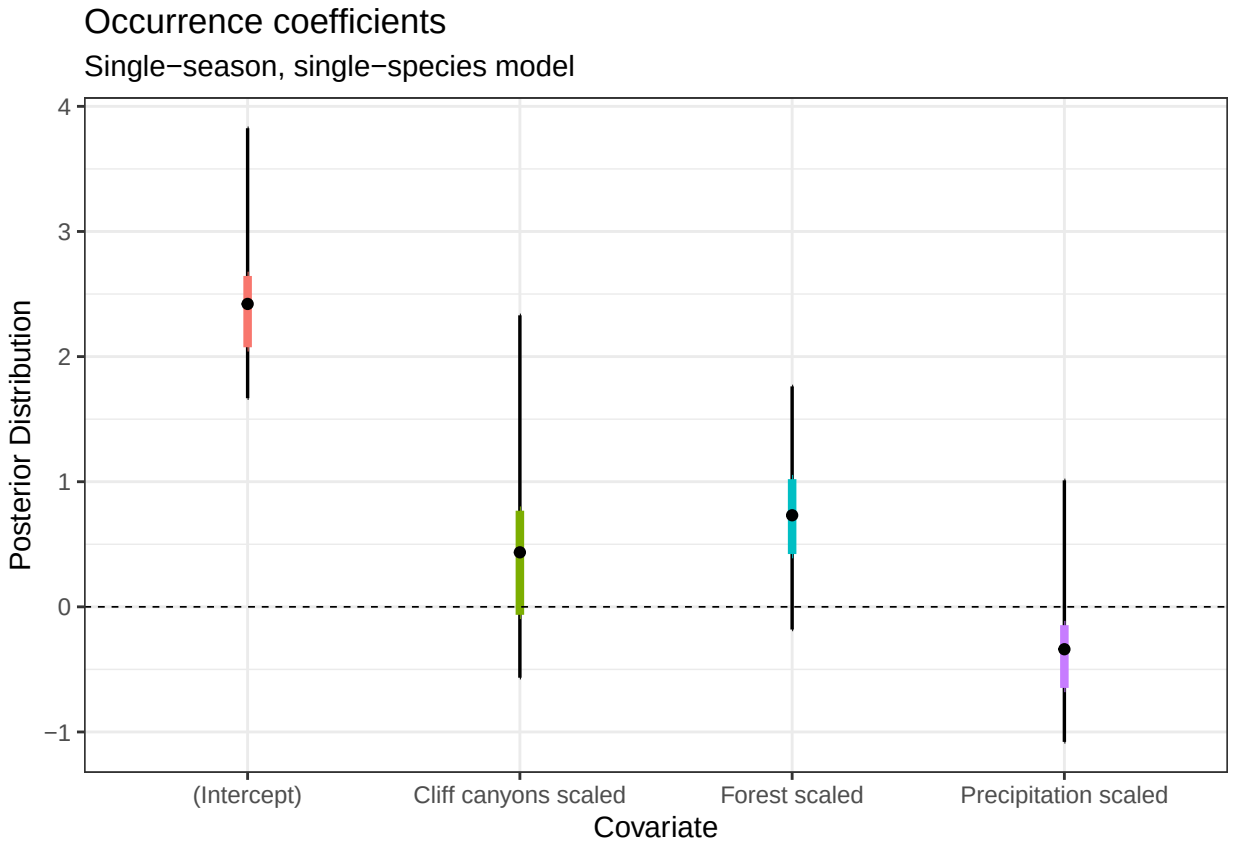
In the following subsections, we make custom parameter plots by accessing and summarizing the full posterior distributions of the parameters. We then provide interpretations of these estimates.

4.1.1 Occurence

Access the full samples of the occurrence parameters and save their quantiles within `occ_quantiles`.

```
occ.samps <- out$beta.samples #this holds the full outputted samples
#create a dataframe to store quantiles in:
occ_quantiles <- as.data.frame(matrix(NA, ncol(occ.samps),6))
colnames(occ_quantiles) <- c("covariate", "mean", "q2.5", "q25", "q75", "q97.5")
occ_quantiles$covariate <- colnames(occ.samps)
occ_quantiles$covariate <- c(
  "(Intercept)", "Forest scaled", "Cliff canyons scaled", "Precipitation scaled"
)
occ_quantiles$mean <- colMeans(occ.samps)
occ_quantiles[,3:6] <- t(apply(occ.samps ,2,quantile,probs=c(0.025, 0.25, 0.75, 0.975)))
```

```
ggplot(occ_quantiles, aes(x = covariate, y = mean)) +
  geom_errorbar(aes(ymin = q2.5, ymax = q97.5), width = 0, linewidth = 0.6) +
  geom_errorbar(aes(ymin = q25, ymax = q75, color = covariate), width = 0,
                linewidth = 1.5, show.legend = FALSE) +
  geom_point() +
  theme_bw() +
  geom_hline(yintercept = 0, lty = 2, linewidth = 0.3) +
  xlab('Covariate') +
  ylab('Posterior Distribution') +
  ggtitle('Occurrence coefficients',
          'Single-season, single-species model')
```



Interpretations:

- Coefficients are estimated for the $logit(\psi) = \beta X$ model and thus must be interpreted on the logit scale
- Positive beta coefficents imply that an increase in the value of that covariate results in an increase in the probability of occurrence, when all other covariate values are held constant.
- Negative beta coefficents imply that an increase in the value of that covariate results in a decrease in the probability of occurrence, when all other covariate values are held constant.
- Example interpretations for LANO occurrence in the states of Oregon and Washington:
  - Percent forest cover is positively associated with LANO occurrence.
  - The estimated effect size of scaled percent forest cover was 0.73 with a 95% credible interval of (-0.18, 1.76) (logit scale). We interpret this value as the following: A one standard deviation increase in scaled percent forest cover is associated with a 0.73 unit increase in the log-odds of LANO occurrence, after accounting for cliff canyon percentage and annual precipitation.
  - On the probability scale: When all other occurrence covariates are held constant, a one standard deviation increase in scaled percent forest cover is associated with an increase in the odds of LANO occurrence of 2.08 (an approximate doubling in the odds of occurrence) with a credible interval of

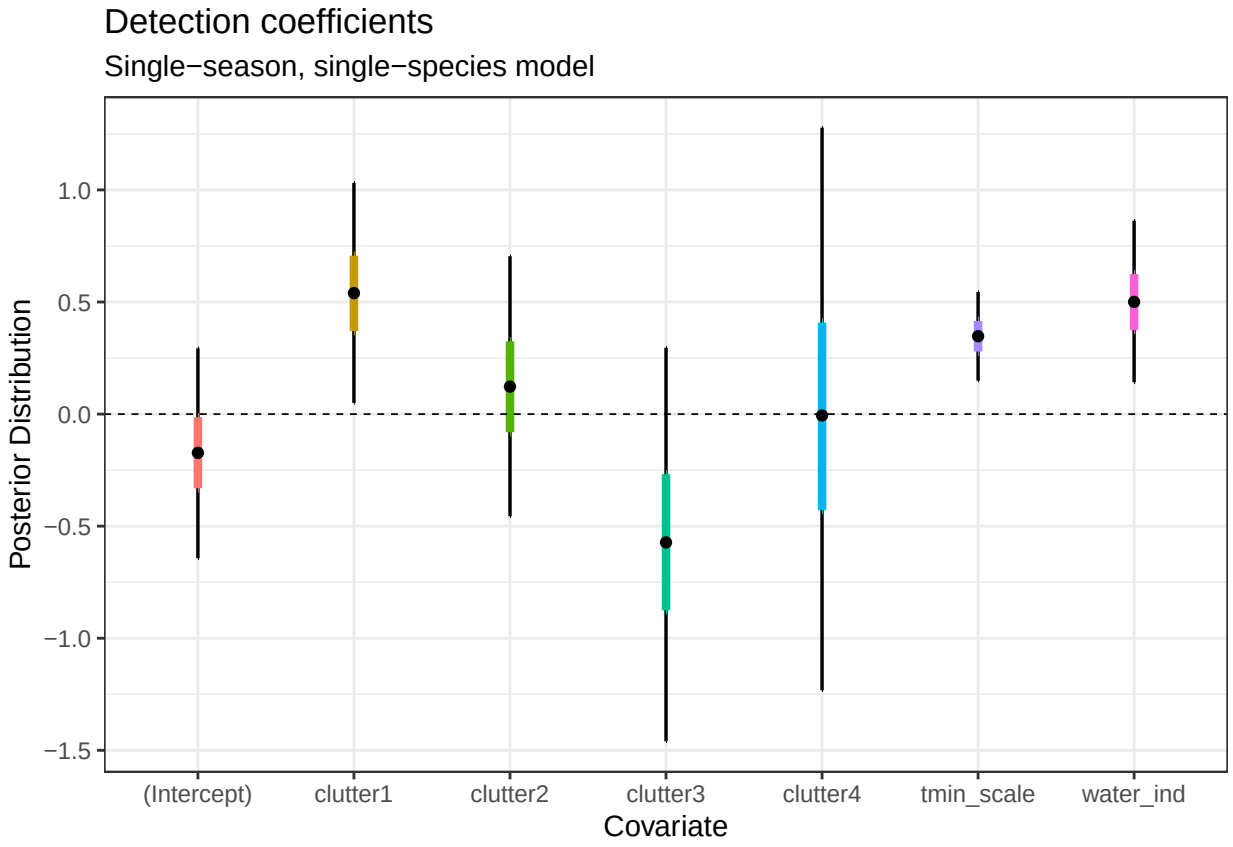
(0.83, 5.82).

4.1.2 Detection

Access the full samples of the occurrence paramaters and save their quantiles within `det_quantiles`.

```
det.samps <- out$alpha.samples
det_quantiles <- as.data.frame(matrix(NA, ncol(det.samps),6))
colnames(det_quantiles) <- c("covariate", "mean", "q2.5", "q25", "q75", "q97.5")
det_quantiles$covariate <- colnames(det.samps)
det_quantiles$mean <- colMeans(det.samps)
det_quantiles[,3:6] <- t(apply(det.samps ,2,quantile,probs=c(0.025, 0.25, 0.75, 0.975)))
```

```
ggplot(det_quantiles, aes(x = covariate, y = mean)) +
  geom_errorbar(aes(ymin = q2.5, ymax = q97.5), width = 0, linewidth = 0.6) +
  geom_errorbar(aes(ymin = q25, ymax = q75, color = covariate), width = 0,
                linewidth = 1.5, show.legend = FALSE) +
  geom_point() +
  theme_bw() +
  geom_hline(yintercept = 0, lty = 2, linewidth = 0.3) +
  xlab('Covariate') +
  ylab('Posterior Distribution') +
  ggtitle('Detection coefficients',
          'Single-season, single-species model')
```



Interpretations:

- Coefficient interpretation follows as for the occurrence coefficients.
- Indicator variable interpretation:
  - Positive beta coefficents imply that a 1 value for the covariate results in an increase in the probability of detection.
  - Negative beta coefficents imply that a 0 value for the covariate results in an increase in the probability of detection.
  - Example on the probability scale: Holding all other detection covariates constant, presence of water is associated with a exp(0.5) = 1.65 times increase in LANO detection odds (or a 65% increase in LANO detection odds) in the states of Oregon and Washington (CI 1.15, 2.37).

4.2 Predictive maps

We create our final predictive maps from the full posterior predictive distributions of ψ across the 4500 cells.

Occurrence probability means:

```
psi.0.mean <- apply(psi.0.samples, 2, mean)
#each column of psi.0.samples is an MCMC sample of the psi for one site
#these are the means ordered in that vector based on their cell
length(psi.0.mean) #this is mean values across all the cells, so we need a map to interpret it
```

```
## [1] 4500
```

Find 95% credible intervals and widths of 95% credible interval for occurrence probability estimates:

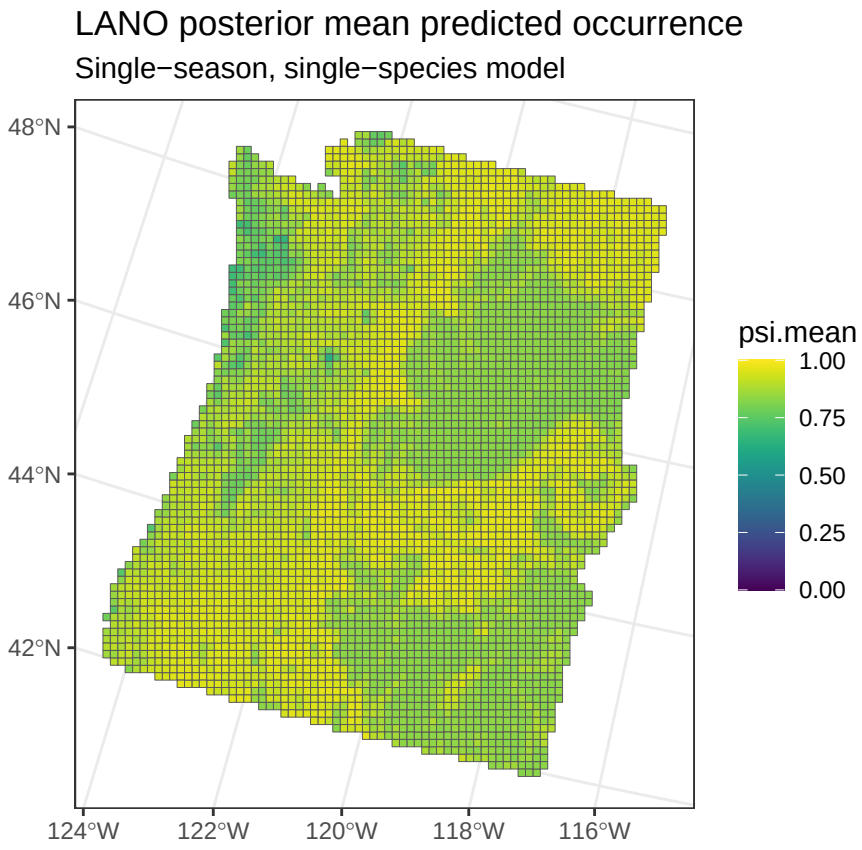
```
quantiles.ordered.samples <-  as.data.frame(
    t(apply(psi.0.samples ,2,quantile,probs=c(0.025,0.975)))
  )
quantiles.ordered.samples$width <- quantiles.ordered.samples[,2] -
  quantiles.ordered.samples[,1]
#these are ordered by cell
```

Predictive plots require the shapefile for the 10x10 grids that can be linked to the unique cell IDs.

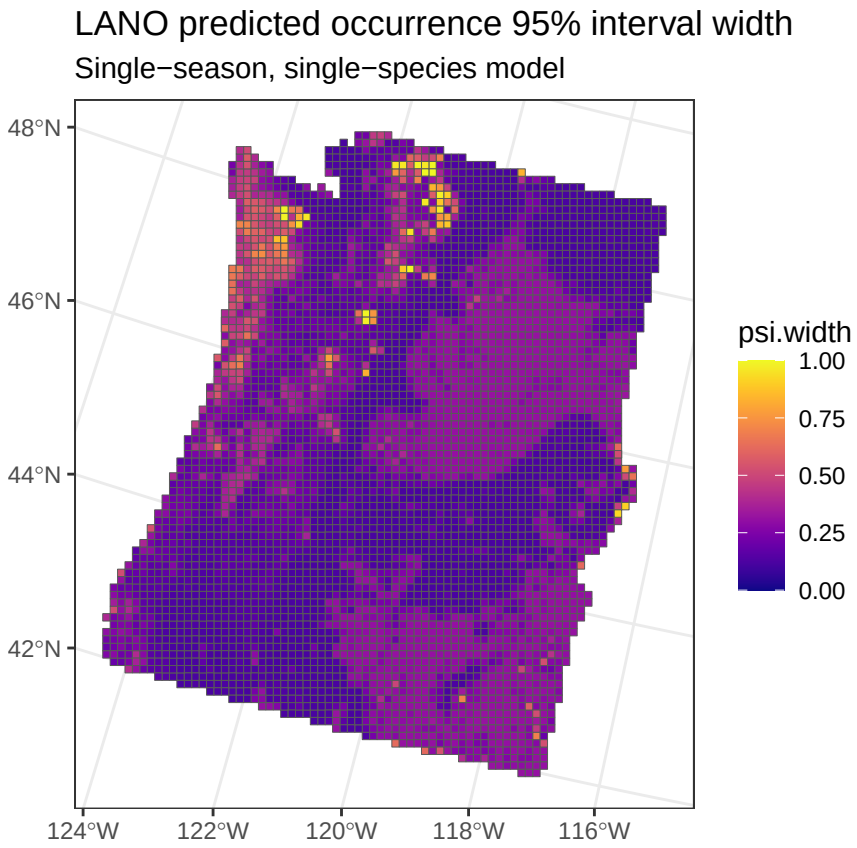
```
#Create a dataframe for plotting:
#Plot names in the covariates and in the prediction outputs line up
#because that is the order it was read into the prediction formula
predicted.df <- data.frame(Cell= occ_covs_all$Cell,
                      psi.mean = psi.0.mean,
                      psi.width = quantiles.ordered.samples$width)

#join the predicted results to the spatial grid, using the plot name
plottable <- left_join(select(grid_shp,Cell, geometry),
                       predicted.df,
                       by = 'Cell')

#there are some NA grid cells that didn't have covariate values and don't have predictions
```

```
#remove them:
plottable <- na.omit(plottable)
```

```
ggplot(data = plottable) +
  geom_sf(aes(fill = psi.mean), lwd = 0.05) +
  theme_bw() +
  viridis::scale_fill_viridis(limits = c(0, 1)) +
  ggtitle('LANO posterior mean predicted occurrence',
          'Single-season, single-species model')
```



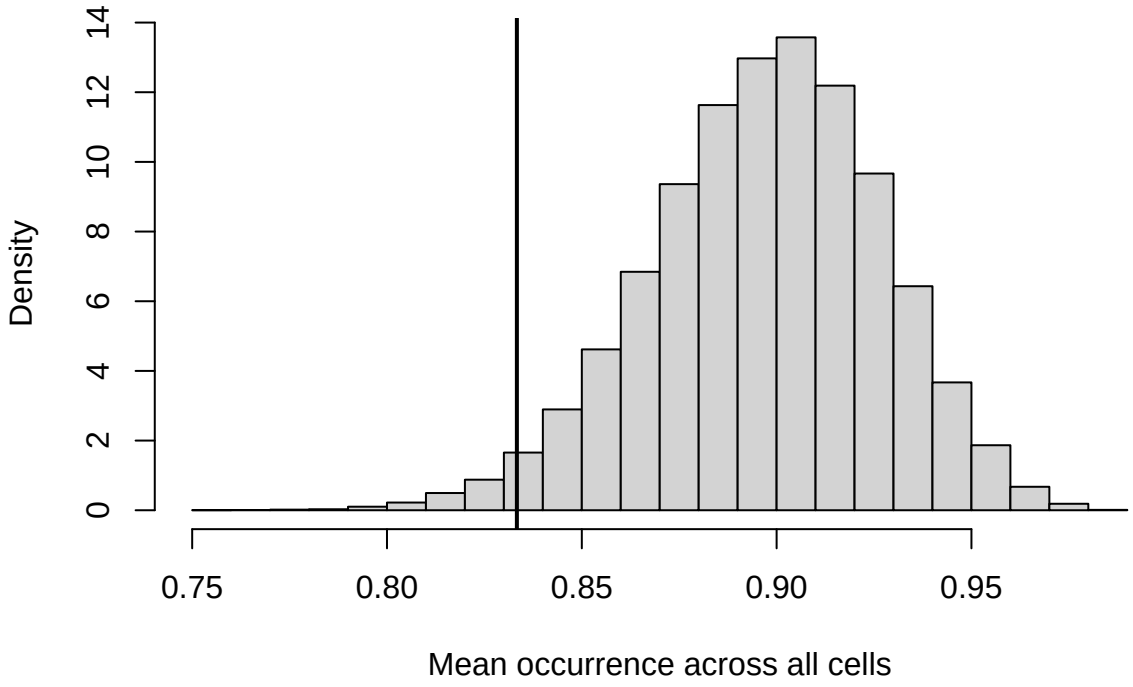

```
ggplot(plottable) +
  geom_sf(aes(fill = psi.width),  lwd = 0.05) +
  theme_bw() +
  viridis::scale_fill_viridis(option = 'plasma', limits = c(0, 1)) +
  ggtitle('LANO predicted occurrence 95% interval width',
          'Single-season, single-species model')
```



4.3 Occurrence across study area

We may be interested in average occurrence across the whole study area, which can be computed as a derived quantity of the cell-level occurrences. We include naive average occurrence across surveyed cells as a solid line.

```
psi <- as.data.frame(out$psi.samples) # samples of latent psi across all sites
#this is psi sampled across all 169 sites
avg_psi <- rowMeans(psi) #take averages across all columns (sites)
avg_psi_naive <- mean(as.numeric( as.character(naive$Naive_occupancy) ))

hist(avg_psi, breaks = 20, freq=F, main = "", xlab = "Mean occurrence across all cells")
abline(v = avg_psi_naive, lwd = 2)
```



4.4 Model comparison

If the covariates of the model have not been previously determined (as in our example), the user may wish to perform model selection to choose a model specification from a suite of possible models (see Hooten and Hobbs (2015)). Widely Applicable Information Criterion (WAIC) can be computed as a possible comparative measure of goodness of fit. WAIC can be computed for each fitted model and comparison can be made between models similar to any information criterion method (e.g., construct a set of candidate models and select models with lower information criterion values). See Gelman, Hwang, and Vehtari (2014) for more details on WAIC and model comparison.

```
waicOcc(out)
```

```
##       elpd         pD       WAIC
## -398.09459   12.70701  821.60320
```

In this document we have provided a single fitted model with the purpose of emphasizing and demonstrating model assessment, however, data analysis is an iterative process and in practice multiple models should be fit so that the data can inform the appropriate structure for the occupancy and detection components of the model. Residual diagnostics can guide that model selection process, and after a model is selected the full outlined diagnostics should be applied to the terminal model.

References

Banner, Katharine M., Kathryn M. Irvine, Thomas J. Rodhouse, Wilson J. Wright, Rogelio M. Rodriguez, and Andrea R. Litt. 2018. "Improving Geographically Extensive Acoustic Survey Designs for Modeling Species Occurrence with Imperfect Detection and Misidentification." *Ecology and Evolution* 8 (12): 6144–56. https://doi.org/10.1002/ece3.4162.

Doser, Jeffrey W., Andrew O. Finley, Marc Kéry, and Elise F. Zipkin. 2022. "spOccupancy: An R Package for Single-Species, Multi-Species, and Integrated Spatial Occupancy Models." *Methods in Ecology and Evolution* 13 (8): 1670–78. https://doi.org/10.1111/2041-210X.13897.

Gelman, Andrew, John B Carlin, Hal S Stern, and Donald B Rubin. 1995. *Bayesian Data Analysis*. Chapman; Hall/CRC.

Gelman, Andrew, Yuri Goegebeur, Francis Tuerlinckx, and Iven Van Mechelen. 2000. "Diagnostic Checks for Discrete Data Regression Models Using Posterior Predictive Simulations." *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 49 (2): 247–68. https://doi.org/10.1111/1467-9876.00190.

Gelman, Andrew, Jessica Hwang, and Aki Vehtari. 2014. "Understanding Predictive Information Criteria for Bayesian Models." *Statistics and Computing* 24 (6): 997–1016. https://doi.org/10.1007/s11222-013-9416-2.

Hobbs, N Thompson, and Mevin B Hooten. 2015. *Bayesian Models: A Statistical Primer for Ecologists*. 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. https://doi.org/10.1890/14-0661.1.

Kéry, Marc, and J Andrew Royle. 2016. *Applied Hierarchical Modeling in Ecology: Volume 1: Prelude and Static Models*. Academic Press.

Loeb, S. C., T. J. Rodhouse, L. E. Ellison, C. I. Lausen, J. D. Reichard, K. M. Irvine, T. E. Ingersoll, et al. 2015. "A plan for the North American Bat Monitoring Program (NABat). Gen. Tech. Report. SRS-208. Asheville, NC: U.S. Department of Agriculture Forest Service, Southern Research Station." https://doi.org/10.2737/SRS-GTR-208.

MacKenzie, Darryl I., James D. Nichols, Gideon B. Lachman, Sam Droege, J. Andrew Royle, and Catherine A. Langtimm. 2002. "Estimating Site Occupancy Rates When Detection Probabilities Are Less Than One." *Ecology* 83 (8): 2248–55. https://doi.org/10.1890/0012-9658(2002)083%5B2248:ESORWD%5D2.0.CO;2.

Rodriguez, Rogelio M, Thomas J Rodhouse, Jenny Barnett, Kathryn Irvine, Katharine M Banner, Jeff Lonneker, and Patricia C Ormsbee. 2019. "North American Bat Monitoring Program Regional Protocol for Surveying with Stationary Deployments of Echolocation Recording Devices: Narrative Version 1.0, Pacific Northwestern US." National Park Service. https://irma.nps.gov/DataStore/Reference/Profile/2265548.

Talbert, Colin, and B. E. Reichert. 2018. "North American Bat Monitoring Program (NABat) Master Sample and Grid-Based Sampling Frame." 10.5066/P9M00P17.

Warton, David I, Jakub Stoklosa, Gurutzeta Guillera-Arroita, Darryl I MacKenzie, and Alan H Welsh. 2017. "Graphical Diagnostics for Occupancy Models with Imperfect Detection." *Methods in Ecology and Evolution* 8 (4): 408–19. https://doi.org/10.1111/2041-210X.12761.

Wright, Wilson J, Kathryn M Irvine, and Megan D Higgs. 2019. "Identifying Occupancy Model Inadequacies: Can Residuals Separately Assess Detection and Presence?" *Ecology* 100 (6): e02703. https://doi.org/10.1002/ecy.2703.

Wright, Wilson J, Kathryn M Irvine, Thomas J Rodhouse, and Andrea R Litt. 2021. "Spatial Gaussian Processes Improve Multi-Species Occupancy Models When Range Boundaries Are Uncertain and Nonoverlapping." *Ecology and Evolution* 11 (13): 8516–27. https://doi.org/10.1002/ece3.7629.