Modeling the occupancy of the Eurasian Blue Tit in Finland

Bayesian Data Analysis course project

Severi Rissanen

November 2020

Contents

1 Introduction

One of the key issues in ecological and species conservation research is the estimation of species abundance and occupancy, and in particular their variability in time and across different locations. One growing trend is the use of citizen science data sets, which allow researchers to utilize species sighting data from hobbyists and amateurs [1]. There is some evidence that data from volunteers can produce reliable estimates [2, 3], but using such data can naturally be difficult, and care must be taken in the modeling phase for good results. The goal of this project was to use the public eBird data set [4] to conduct a case study on the occupancy of the Eurasian Blue Tit bird inside Finland using a fully Bayesian occupancy model.

The main quantity of interest in this project was the occupancy of a species, which can be defined as the probability of at least one bird occupying a location at any given time. It depends only on factors concerning the location, such as the vegetation. In occupancy models [5], this quantity is separated from the detectability, which depends on factors such as the effort put in to the observation process as well as the environment itself. The actual probability of observing the species in question is then the product of occupancy and detectability. Figure 1 illustrates the modeling idea. The goal of occupancy modeling is to move away from just predicting the variability in the data to estimating the actual distribution of the species by controlling for the detectability.

2 The data set and model

2.1 Description

The full eBird data set[6] contains over half a billion bird sightings around the world, and about eight hundred thousand sightings in Finland. It is collected by independent amateur bird watchers as checklists that are filled after each birding trip. The fields contain information about the species of birds observed, their amount, effort variables such as the time spent on observing and the distance traveled during the observation, as well as whether the checklist was "complete". Completeness means that the watchers reported all birds that they were able to during the trip, and it is crucial for occupancy modeling because it allows us to know whether the observed *didn't* observe the species in question.

eBird data has been used extensively in research [7], and to be honest I'm not sure about everything that has been done, but this study differs on existing ones on occupancy in that I haven't seen it done in Finland, and also with a full Bayesian model (although most likely that has been done in one way or the other). Also, the covariates for occupancy and detectability were chosen by me and the exact modeling approach was invented by myself, so it's unlikely that something identical has been done elsewhere.

Following the Cornell University tutorial on modeling eBird data [8], I also used satellite data from the MODIS satellite [9] to estimate the habitat type

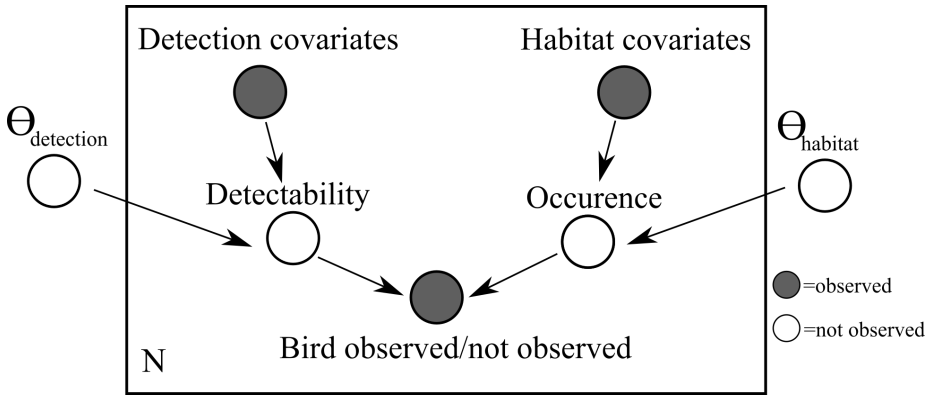

Figure 1: A graph illustrating the relationships between the data and the model. The variables inside the box correspond to measurements/unmeasured latent variables that were different for each observation, while $\theta_{detection}$ and $\theta_{habitat}$ correspond to the model parameters for estimating the detectability and occupancy.

across Finland. These estimates were used as the habitat covariates in the model. In the data, which uses the University of Maryland (UMD) classification, the habitats are divided in to 15 categories, such as "water bodies", "evergreen needleleaf forests", "mixed forests", "urban" etc.

2.2 Preprocessing

Preprocessing the data, especially the satellite data, was somewhat complicated, but I was able to mostly follow the instructions and best practises presented in the Cornell University tutorial [8]. The `auk` package for R was used to zero-fill (add non-observations of birds) the data efficiently, which was needed because one of the required files was 15GB in size. Following the tutorial, I restricted the analysis on observations with the duration of observations under 5 hours, the distance traveled under 5 km and the number of observers under 6 to reduce the variation in detectability and make modeling easier. I chose observations on the Eurasian Blue Tit (Finnish: sinitiainen) in the period 1.6.2020-31.7.2020. The period was chosen to be short enough that the occupancy hopefully shouldn't change much in time, but also to not reduce the amount of data too much. The Blue Tit is a good bird to do the analysis with since it has one of the highest amount of sightings in Finland according to a preliminary exploratory data analysis (see also Fig. 2). This should reduce the problem of imbalanced classes and make the analysis easier. After all the filters, the remaining data set had 2 197 observations on Blue Tit presence/absence.

The satellite data could have been combined with the bird observations so that we would extract the habitat for the longitude and latitude marked for the observation, but this would have been problematic because people sometimes walk around during observing and more importantly because birds don't live in single points. What matters is the general landscape around the observation.

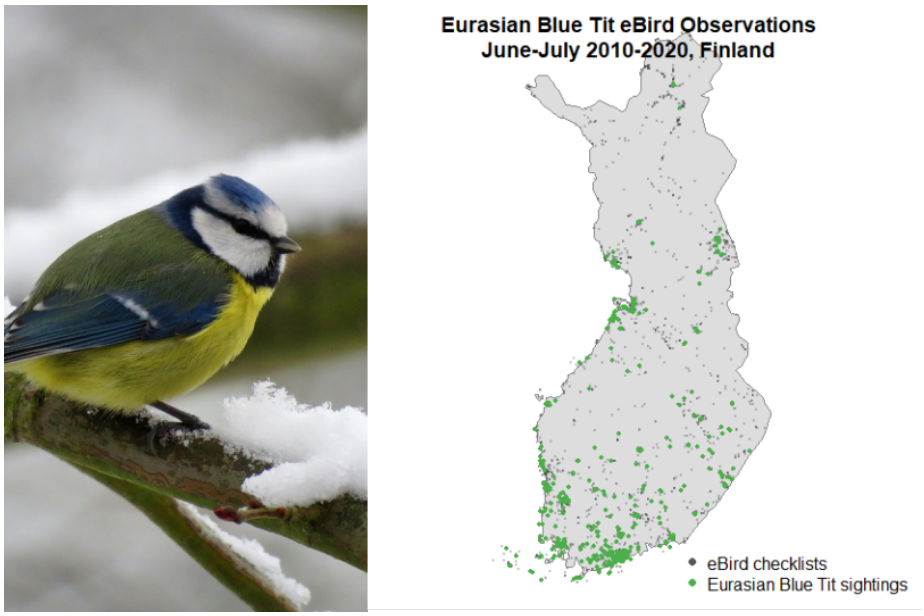

Figure 2: Left: The Eurasian Blue Tit. Right: Blue Tit sightings and non-sightings in June and July between 2010 and 2020.

Again following the tutorial, I chose a neighborhood centered on the observation so that its diameter was 5 MODIS satellite cells, and the habitat covariates were defined as the proportions of each landscape class in the neighborhood (also called PLAND).

2.3 The model

As a first attempt at a model, I chose the covariates for the occupancy to simply be the proportions of the different habitats around the observation spot (p_{ij} for observation i and habitat j). The covariates for the occupancy were chosen as the duration of the observation T_i (in minutes), distance travelled s_i (in kilometers) and the proportion of forest in the area, f_i. The proportion was defined as the sum of the four different forest class proportions, and was motivated by the fact that it's most likely more difficult to see far away in a dense forest. Some of the notable assumptions by a model like this are that the occupancy was constant for the chosen time period at all locations, the detection probability didn't depend on observers directly and that the time of the observation doesn't influence detectability either. The probabilistic model

was then defined as:

$$
\begin{aligned}
c_j &\sim N(0, 10) \\
c_T &\sim N(0, 0.1) \\
c_s &\sim N(0, 2) \\
c_f &\sim N(0, 10) \\
a_o &\sim N(0, 2) \\
a_d &\sim N(0, 2) \\
O_i &= \sigma(a_o + \sum_j c_j \mathrm{p}_{ij}) \\
D_i &= \sigma(a_d + c_T T_i + c_s s_i + c_f f_i) \\
y_i | O_i, D_i &\sim \mathrm{Bern}(O_i \cdot D_i)
\end{aligned}
$$

Here c_j are logistic regression coefficients on the PLAND habitat covariates and c_T, c_s and s_f are logistic regression coefficients on observability covariates. O_i and D_i are the occupancies and detectabilities for data points i, and σ is the logistic function. The priors on coefficients were chosen to be weakly informative based on the ranges of the variables. Noting that a total change of 1 unit in the sum inside the logistic functions usually makes a large difference in the output, a standard deviation of 10 for landscape cover covariates made sense since it's unlikely that a change of 0.1 in some proportion would have a larger effect than something like 0.1·20=2 (two standard deviations) in the input to the logistic function. Similarly, the durations in minutes ranged from 0 to 300 (with most closer to 0 than 300 in the entire Finnish data set according to preliminary exploratory analysis), and a change from 10 minutes to 20 minutes should be able to have a large effect. With two standard deviations, 10·0.2=2 is quite large indeed, and perhaps the prior could be even tighter. The prior for distance traveled was chosen similarly with a change in 1 km causing a change of 1·4=4 if the value of c_s is two standard deviations from the mean. The priors for the constant parameters a_0 and a_d were also chosen with the same principle.

Aside from the simple model, I also experimented with using the observation start time as a covariate for detectability as well. This makes sense since most likely it's more difficult to observe the bird in the night than during the day, but having the start time as a simple extra parameter in the logistic regression won't make sense either since we can expect the response to be nonlinear and periodic (0h and 24h from the start of the day should be equal). I decided to model the response based on the idea of a Fourier series decomposition of the probability of detection as a function of observation start time:

$$
p(detect|t) \approx \frac{a_0}{2} + \sum_{n=1}^{N} \left(a_n \cos(\frac{2\pi n t}{24}) + b_n \sin(\frac{2\pi n t}{24}) \right)
$$

with some coefficients a_n and b_n and for start time t in hours. With high enough N, the decomposition should be close to the true function. We can control the

wigliness of the estimate by adjusting it to a lower value. To incorporate the idea to the model, I included similar terms inside the logistic function in the detectability model:

$$
\begin{aligned}
a_n &\sim N(0, 2) \\
b_n &\sim N(0, 2) \\
D_i &= \sigma(a_d + c_T T_i + c_s s_i + c_f f_i + \sum_{n=1}^{N} (a_n \cos(\frac{2\pi n t_i}{24}) + b_n \sin(\frac{2\pi n t_i}{24}))
\end{aligned}
$$

where the weakly informative priors were again chosen to be reasonably wide considering that the outputs of sin and cos functions are between -1 and 1. I tried two models: One with $N = 1$ and one with $N = 2$.

3 Results

All of the Stan code and most relevant R code is in the Appendix, as I felt that they would unnecessarily clutter this section.

3.1 Setup and convergence diagnostics

I ran Stan on all three modes with 2000 MCMC iterations per chain (1000 warmup, 1000 sampling) and 4 chains. Table 1 lists the relevant convergence diagnostics for all three models. For all models, $\hat{R}$ values were at most 1.01, which indicates that there were no problems with the mixing of the four chains, and our samples were likely from the correct posterior. Effective sample size values for the first model were not great for all parameters (491 at lowest out of 4000 actual samples), but acceptable. Perhaps a bit surprisingly, the ESS values were much better for the models with additional start time parameters (2 and 3). A larger effective sample size means that we can be more sure of estimates made using the sampled values, and it looks like most of the time we have a pretty good number of effective samples to calculate expectations with, for example. Also, there were no divergent transitions in any of the models. This means that the HMC sampler didn't encounter "difficult" regions in the target distribution which it couldn't explore well enough, and we should be able to trust the inferences. The maximum tree-depth wasn't exceeded, meaning that the no-U-turn sampler didn't have to terminate prematurely at any point.

3.2 Posterior predictive checking and model selection

I conducted three different posterior predictive checks for all models. First, following the lecture slides, I plotted the actual observed/not observed values w.r.t. the predicted probabilities of observing them, and binned them in to eight groups w.r.t. their predicted probabilities. I then calculated the proportions of "observed" data points inside each bin and their binomial confidence intervals (with the normal approximation). If the model is able to explain the data well

	Model 1		Model 2		Model 3	
Parameter	$\hat{R}$	Bulk ESS	$\hat{R}$	Bulk ESS	$\hat{R}$	Bulk ESS
c_1	1.00	763	1.00	1192	1.00	1249
c_2	1.01	491	1.00	2722	1.00	2838
c_3	1.00	4229	1.00	5525	1.00	6211
c_4	1.00	4572	1.00	5556	1.00	5914
c_5	1.01	617	1.00	1988	1.00	1851
c_6	1.00	2833	1.00	2470	1.00	2894
c_7	1.00	784	1.00	1118	1.00	1169
c_8	1.00	758	1.00	1336	1.00	1338
c_9	1.00	834	1.00	1116	1.00	1161
c_{10}	1.01	535	1.00	3728	1.00	3707
c_{11}	1.00	4302	1.00	5879	1.00	4172
c_{12}	1.00	782	1.00	1859	1.00	1787
c_{13}	1.00	3201	1.00	4766	1.00	5971
a_o	1.00	748	1.00	1076	1.00	1118
c_T	1.00	637	1.00	3598	1.00	4323
c_s	1.00	3660	1.00	5210	1.00	4872
c_f	1.00	1922	1.00	3294	1.00	3280
a_d	1.01	641	1.00	3599	1.00	3164
a_1	-	-	1.00	5608	1.00	3535
b_1	-	-	1.00	4688	1.00	3524
a_2	-	-	-	-	1.00	3578
b_2	-	-	-	-	1.00	4852

Table 1: Convergence diagnostics for the three models considered. Model 1 corresponds to the model with no start time dependence on detectability, model 2 uses only the first two Fourier coefficients (N=1) and model 3 uses the first four (N=2).

enough, then this should result in the proportions being close to the predicted probabilities inside the corresponding bins. These are plotted in Fig.3. We can see that all models are OK for the most part, but the first model doesn't match as well with the data as the other two. The third (N=2) model seems somewhat better than the second model. I also tried more concrete predictive checks by plotting the marginal probability of observing the bird with respect to distance traveled and the starting time, estimated directly from the counts in the data and from the predicted probabilities from the model. This also involved binning of the distances traveled and starting times in to different groups and calculating proportions inside them. These are plotted in Figs.4 and 5. Looking at the error bars on the distance plot, it's clear that even though the probability predictions don't match that well with the directly estimated means, this could easily be due to random chance. The most striking differences are seen in the starting time plot, with the estimates matching better and better with the data as we increase the complexity of the model. This is not surprising, but we see that including the starting time directly is important.

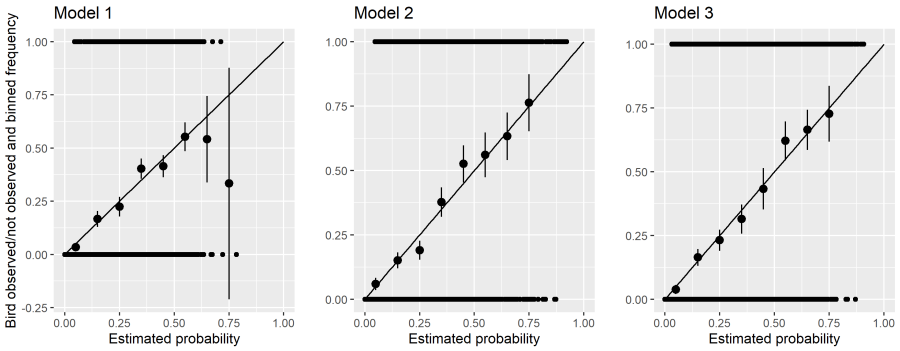

Figure 3: The logistic regression -type posterior predictive check for all three models. Explained in the main text.

I also tried comparing the models with the PSIS-LOO values, but some of the pareto-k-diagnostic values were quite bad for all models, and the estimates weren't reliable out of the box. Judging from Fig.5 and especially Fig.3, I would, however, say that the third model was the best out of these. I also compared the predictive accuracies of the model (predicting whether the bird is observed or not), and the values were 0.73, 0.76 and 0.78 for models 1, 2 and 3, respectively. The accuracies also imply that model 3 is the best out of these.

3.3 Prior sensitivity checking

As we decided to select the third model based on the previous section, I decided to investigate the sensitivity to prior choice only on that one. I fitted two additional models: one with the standard deviations of parameters reduced to one fifth of the original values, and one with standard deviations increased to three

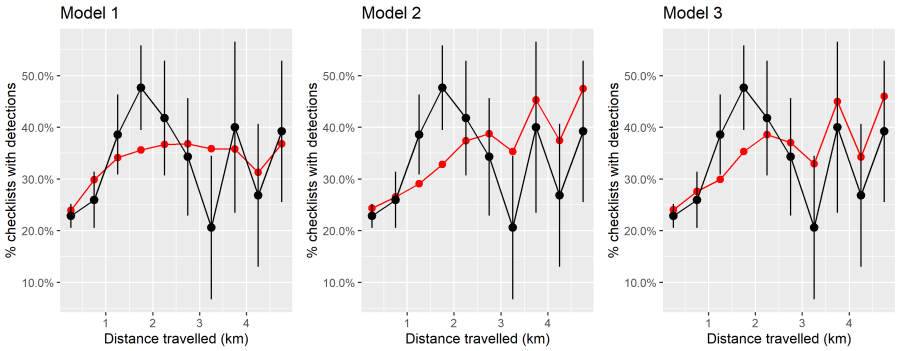

Figure 4: Posterior predictive check of marginal probability given distance traveled, predicted by the model and estimated directly from the data (black is data, red is model prediction)

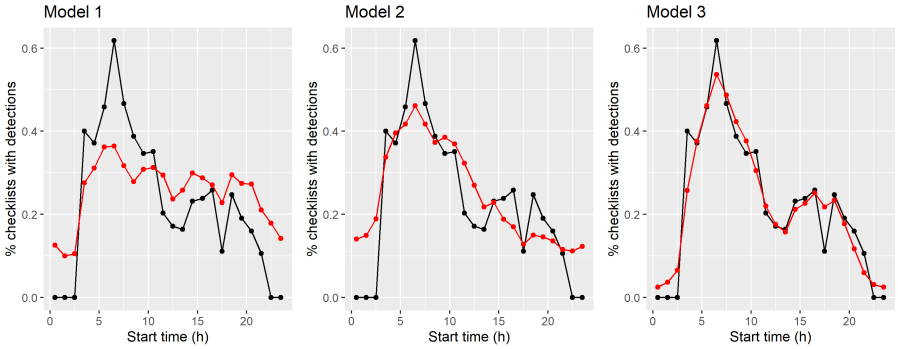

Figure 5: Posterior predictive check of marginal probability given observation starting time, predicted by the model and estimated directly from the data (black is data, red is model prediction)

times the original values. Figure 6 shows the logistic regression-type posterior predictive check for all three fits. We see that the model is somewhat sensitive to prior choice, and both drastically decreasing or increasing the standard deviations results in decreased model performance.

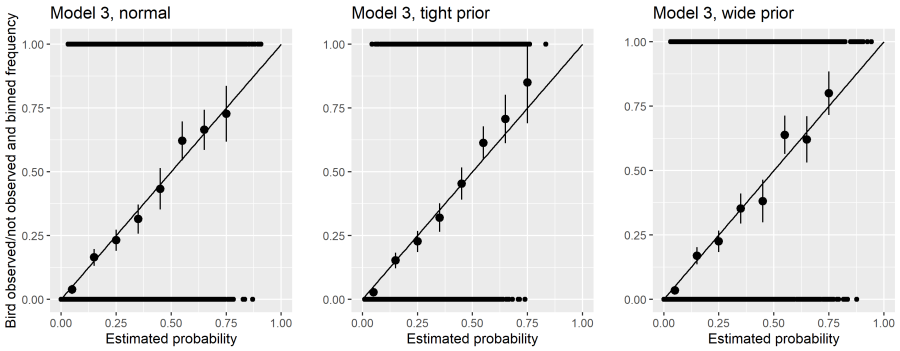

Figure 6: The logistic regression type posterior predictive checks for the different priors. The standard deviations are reduced to one fifth for the tight priors and increased three-fold for the wide priors.

3.4 Analysis of results

Figure 7 shows the estimates of the coefficients for the third model. We can immediately draw some conclusions: It seems that the blue tit prefers evergreen needleleaf forests and wetlands the most, but urban areas and mixed forests are probably positive factors as well. Open areas such as open shrublands or croplands are not preferred by the bird. As expected, the distance travelled and observation duration contribute positively to detectability, while forest cover contributes negatively.

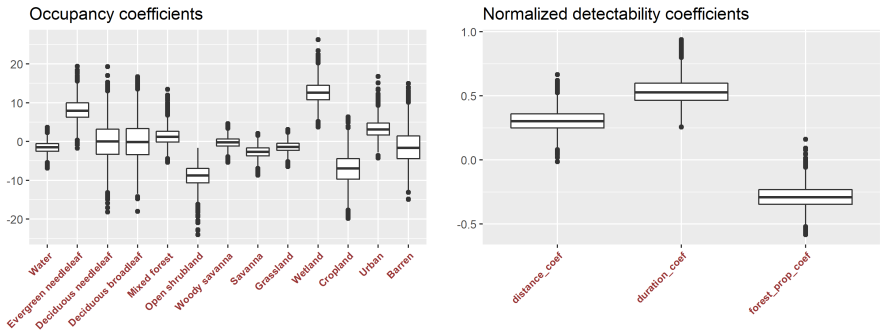

Figure 7: Selected logistic regression coefficient distributions from the third model. The detectability coefficients were normalized by multiplying with the standard deviation of the corresponding covariate in the data set. Note that the habitat names are based on the University of Maryland land cover classification system, and may not reflect the corresponding habitat in Finland accurately (e.g. Savanna). Search the UMD classification system for exact interpretations.

Since we assumed that the occupancy is dependent only on the habitat data, which we got from the MODIS satellite, we can use the generated coefficient

draws to estimate the occupancy elsewhere in Finland as well. This was accomplished by dividing Finland in to cells that were the same size as the cells that were used to estimate the habitat distribution around birding sites, and for each cell estimated the expected value and standard deviation of occupancy based on the 3rd model draws. The result is shown in Fig. 8. We see that especially in southern Finland there are areas which have a high occupancy, but for instance northern Lapland has a notably low occupancy, aside from the region around lake Inari.

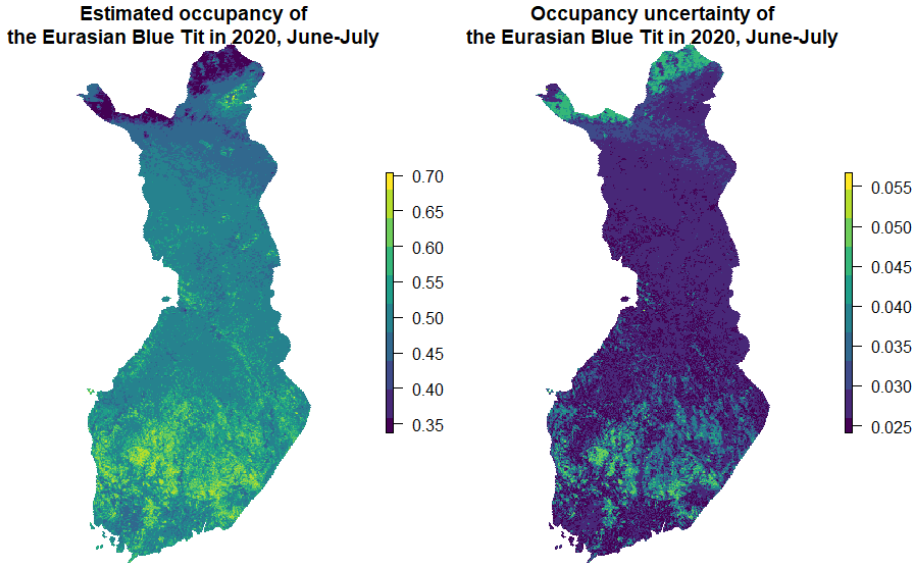

Figure 8: The expected occupancy of the Blue Tit in Finland and the standard deviation of occupancy in 2020 June-July, according to the third model.

4 Discussion

4.1 Model issues and improvements

There are many things that could possibly be improved with the model. To relax the assumptions made about what factors exactly influence the occupancy and detectability variables, we could increase the amount of covariates. For instance, having the coordinates as additional covariates for the occupancy model or somehow including variables that determine how good the individual observers are could make the results more accurate. On the other hand, especially the third model seemed to explain the data quite well, and the results were quite sensible. Creating a too complex model could have some disadvantages as well, since we want the model to stay interpretable to ensure that our conclusions make sense.

A second, slightly different type of extension to the model would be to add the time of year as a covariate for occupancy. The advantage for this would be that we wouldn't need to assume a constant occupancy for the chosen time period, and our results would be applicable to other times of year outside June and July.

A third possible improvement would be to replace the simple logistic regression terms in the probabilistic model with something more complex. One simple option would be to add interaction terms (like $c_{T,s}T_i s_i$) to the logistic regression. This could make sense especially with observation duration and start times, since those two variables together determine what the actual period of observation was.

4.2 Conclusions

A simple, but fully Bayesian, occupancy model using logistic regression and in particular Fourier features for the observation start time can evidently produce sensible and interpretable results concerning the occupancy and detectability of a species. A fully Bayesian approach is clearly useful in this context, since it allows us to determine the uncertainty in our conclusions regarding the model. This is very important if we want to make rigorous scientific research that could impact conservation action, for instance. The eBird data set seems to be well fitting for this type of modeling, and although the amount of observations in Finland is smaller than in the US, for instance, using a common species like the Eurasian Blue Tit seems to bypass this problem. The next thing to try out would naturally be a species that is less common or even endangered.

I personally learned a lot of stuff making this project. I had never heard about occupancy or abundance models and in that way I learned some basics of research in ecology. I had never used satellite data before and only a little bit of map data in general. I also learned quite a bit of R, especially things relating to tidyverse like ggplot. I got some hands-on experience in formulating a fully Bayesian model to solve a scientific problem and especially got more familiar with Stan and the Bayesian workflow.

References

[1] B. L. Sullivan, T. Phillips, A. A. Dayer, C. L. Wood, A. Farnsworth, M. J. Iliff, I. J. Davies, A. Wiggins, D. Fink, W. M. Hochachka, *et al.*, "Using open access observational data for conservation action: A case study for birds," *Biological Conservation*, vol. 208, pp. 5–14, 2017.

[2] A. W. Crall, G. J. Newman, T. J. Stohlgren, K. A. Holfelder, J. Graham, and D. M. Waller, "Assessing citizen science data quality: an invasive species case study," *Conservation Letters*, vol. 4, no. 6, pp. 433–442, 2011.

[3] A. J. van Strien, C. A. van Swaay, and T. Termaat, "Opportunistic citizen science data of animal species produce reliable estimates of distribu-

tion trends if analysed with occupancy models," *Journal of Applied Ecology*, vol. 50, no. 6, pp. 1450–1458, 2013.

[4] B. L. Sullivan, C. L. Wood, M. J. Iliff, R. E. Bonney, D. Fink, and S. Kelling, "ebird: A citizen-based bird observation network in the biological sciences," *Biological conservation*, vol. 142, no. 10, pp. 2282–2292, 2009.

[5] D. I. MacKenzie, J. D. Nichols, G. B. Lachman, S. Droege, J. Andrew Royle, and C. A. Langtimm, "Estimating site occupancy rates when detection probabilities are less than one," *Ecology*, vol. 83, no. 8, pp. 2248–2255, 2002.

[6] "`ebird basic dataset. version: Ebd_reloct-2020.`," *Cornell Lab of Ornithology, Ithaca, New York.*

[7] "`https://ebird.org/science/publications`,"

[8] "`https://cornelllabofornithology.github.io/ebird-best-practices/index.html`,"

[9] "`https://lpdaac.usgs.gov/products/mcd12q1v006/`,"

A Stan code for model 1

```
data {
  int<lower=0> N;//Amount of observations
  int observed[N];//Whether the bird was observed or not
  int<lower=0> N_habitats;
  int<lower=0> N_localities;
  vector[N_habitats] habitat_prop[N_localities];
  int localities[N];//The ids have to be preprocessed go 1..N_localities
  vector[N] duration_minutes;
  vector[N] distance_traveled;//Perhaps should also have the protocol type?
  vector[N] forest_prop;//Used for detection model
  int<lower=0> check_N;
  int check_indices_habprop[check_N];
  int check_indices_others[check_N];
}

// The parameters accepted by the model. Our model
// accepts two parameters 'mu' and 'sigma'.
parameters {
  //Obs. model parameters
  vector[N_habitats] habitat_coef;
  real habitat_bias;

  //Detection model parameters
  real duration_coef;
  real distance_coef;
  real forest_prop_coef;
  real detectability_bias;
}

// The model to be estimated. We model the output
// 'y' to be normally distributed with mean 'mu'
// and standard deviation 'sigma'.
model {
  vector[N_localities] occupied_prob;
  vector[N] detected_prob;

  //Occupancy model
  for(j in 1:N_habitats){//priors for coefs
    habitat_coef[j] ~ normal(0,5);
  }
  habitat_bias ~ normal(0,2);
  for(i in 1:N_localities){
    occupied_prob[i] = inv_logit(habitat_bias + sum(habitat_coef .* habitat_prop[i,]));
  }


  //Detection model
  duration_coef ~ normal(0,0.5);
```

```
  distance_coef ~ normal(0,2);
  forest_prop_coef ~ normal(0,5);
  detectability_bias ~ normal(0,2);
  for(k in 1:N){
    detected_prob[k] = inv_logit(detectability_bias + duration_coef*duration_minutes[k] +
                                 distance_coef*distance_traveled[k] +
                                 forest_prop_coef*forest_prop[k]);
  }

  //Combining the two
  for(k in 1:N){
    observed[k] ~ bernoulli(occupied_prob[localities[k]] * detected_prob[k]);
  }
}

generated quantities{
  int observed_check[check_N];
  vector[check_N] occupied_prob;
  vector[check_N] detected_prob;

  for(i in 1:check_N){
    occupied_prob[i] = inv_logit(habitat_bias + sum(habitat_coef .* habitat_prop[check_indices_habprop[i],]));
  }
  for(k in 1:check_N){
    detected_prob[k] = inv_logit(detectability_bias + duration_coef*duration_minutes[check_indices_others[k]] +
                                 distance_coef*distance_traveled[check_indices_others[k]] +
                                 forest_prop_coef*forest_prop[check_indices_others[k]]);
  }

  for(k in 1:check_N){
    observed_check[k] = bernoulli_rng(occupied_prob[k] * detected_prob[k]);
  }
}
```

B Stan code for model 2

```
data {
  int<lower=0> N;//Amount of observations
  int observed[N];//Whether the bird was observed or not
  int<lower=0> N_habitats;
  int<lower=0> N_localities;
  vector[N_habitats] habitat_prop[N_localities];
  int localities[N];//The ids have to be preprocessed go 1..N_localities
  vector[N] duration_minutes;
  vector[N] distance_traveled;//Perhaps should also have the protocol type?
  vector[N] forest_prop;//Used for detection model
  vector[N] start_time;
  int<lower=0> check_N;
  int check_indices_habprop[check_N];
```

```
  int check_indices_others[check_N];
}

// The parameters accepted by the model. Our model
// accepts two parameters 'mu' and 'sigma'.
parameters {
  //Obs. model parameters
  vector[N_habitats] habitat_coef;
  real habitat_bias;

  //Detection model parameters
  real duration_coef;
  real distance_coef;
  real forest_prop_coef;
  real detectability_bias;
  //Start time model parameters
  real a1;
  real b1;
}

// The model to be estimated. We model the output
// 'y' to be normally distributed with mean 'mu'
// and standard deviation 'sigma'.
model {
  vector[N_localities] occupied_prob;
  vector[N] detected_prob;

  //Occupancy model
  for(j in 1:N_habitats){//priors for coefs
    habitat_coef[j] ~ normal(0,5);
  }
  habitat_bias ~ normal(0,2);
  for(i in 1:N_localities){
    occupied_prob[i] = inv_logit(habitat_bias + sum(habitat_coef .* habitat_prop[i,]));
  }


  //Detection model
  duration_coef ~ normal(0,0.5);
  distance_coef ~ normal(0,2);
  forest_prop_coef ~ normal(0,5);
  detectability_bias ~ normal(0,2);
  a1 ~ normal(0,2);
  b1 ~ normal(0,2);
  for(k in 1:N){
    detected_prob[k] = inv_logit(detectability_bias + duration_coef*duration_minutes[k] +
                                 distance_coef*distance_traveled[k] +
                                 forest_prop_coef*forest_prop[k] +
                                 a1*cos(2*pi()*start_time[k]/24) + b1*sin(2*pi()*start_time[k]/24));
  }
```

```
  //Combining the two
  for(k in 1:N){
    observed[k] ~ bernoulli(occupied_prob[localities[k]] * detected_prob[k]);
  }
}

generated quantities{
  int observed_check[check_N];
  vector[check_N] occupied_prob;
  vector[check_N] detected_prob;

  for(i in 1:check_N){
    occupied_prob[i] = inv_logit(habitat_bias + sum(habitat_coef .* habitat_prop[check_indices_habprop[i],]));
  }
  for(k in 1:check_N){
    detected_prob[k] = inv_logit(detectability_bias + duration_coef*duration_minutes[check_indices_others[k]] +
                                 distance_coef*distance_traveled[check_indices_others[k]] +
                                 forest_prop_coef*forest_prop[check_indices_others[k]] +
                                 a1*cos(2*pi()*start_time[check_indices_others[k]]/24) +
                                 b1*sin(2*pi()*start_time[check_indices_others[k]]/24));
  }

  for(k in 1:check_N){
    observed_check[k] = bernoulli_rng(occupied_prob[k] * detected_prob[k]);
  }
}
```

C Stan code for model 3

```
data {
  int<lower=0> N;//Amount of observations
  int observed[N];//Whether the bird was observed or not
  int<lower=0> N_habitats;
  int<lower=0> N_localities;
  vector[N_habitats] habitat_prop[N_localities];
  int localities[N];//The ids have to be preprocessed go 1..N_localities
  vector[N] duration_minutes;
  vector[N] distance_traveled;//Perhaps should also have the protocol type?
  vector[N] forest_prop;//Used for detection model
  vector[N] start_time;
  int<lower=0> check_N;
  int check_indices_habprop[check_N];
  int check_indices_others[check_N];
}

// The parameters accepted by the model. Our model
// accepts two parameters 'mu' and 'sigma'.
parameters {
```

```
  //Obs. model parameters
  vector[N_habitats] habitat_coef;
  real habitat_bias;

  //Detection model parameters
  real duration_coef;
  real distance_coef;
  real forest_prop_coef;
  real detectability_bias;
  //Start time model parameters
  real a1;
  real b1;
  real a2;
  real b2;
}

// The model to be estimated. We model the output
// 'y' to be normally distributed with mean 'mu'
// and standard deviation 'sigma'.
model {
  vector[N_localities] occupied_prob;
  vector[N] detected_prob;

  //Occupancy model
  for(j in 1:N_habitats){//priors for coefs
    habitat_coef[j] ~ normal(0,5);
  }
  habitat_bias ~ normal(0,2);
  for(i in 1:N_localities){
    occupied_prob[i] = inv_logit(habitat_bias + sum(habitat_coef .* habitat_prop[i,]));
  }

  //Detection model
  duration_coef ~ normal(0,0.5);
  distance_coef ~ normal(0,2);
  forest_prop_coef ~ normal(0,5);
  detectability_bias ~ normal(0,2);
  a1 ~ normal(0,2);
  b1 ~ normal(0,2);
  a2 ~ normal(0,2);
  b2 ~ normal(0,2);
  for(k in 1:N){
    detected_prob[k] = inv_logit(detectability_bias + duration_coef*duration_minutes[k] +
                                 distance_coef*distance_traveled[k] +
                                 forest_prop_coef*forest_prop[k] +
                                 a1*cos(2*pi()*start_time[k]/24) + b1*sin(2*pi()*start_time[k]/24) +
                                 a2*cos(2*pi()*2*start_time[k]/24) + b2*sin(2*pi()*2*start_time[k]/24));
  }

  //Combining the two
```

```
  for(k in 1:N){
    observed[k] ~ bernoulli(occupied_prob[localities[k]] * detected_prob[k]);
  }
}

generated quantities{
  int observed_check[check_N];
  vector[check_N] occupied_prob;
  vector[check_N] detected_prob;

  for(i in 1:check_N){
    occupied_prob[i] = inv_logit(habitat_bias + sum(habitat_coef .* habitat_prop[check_indices_habprop[i],]));
  }
  for(k in 1:check_N){
    detected_prob[k] = inv_logit(detectability_bias + duration_coef*duration_minutes[check_indices_others[k]] +
                                 distance_coef*distance_traveled[check_indices_others[k]] +
                                 forest_prop_coef*forest_prop[check_indices_others[k]] +
                                 a1*cos(2*pi()*start_time[check_indices_others[k]]/24) + b1*sin(2*pi()*start_time[check_indi
                                 a2*cos(2*pi()*2*start_time[check_indices_others[k]]/24) + b2*sin(2*pi()*2*start_time[check_
  }

  for(k in 1:check_N){
    observed_check[k] = bernoulli_rng(occupied_prob[k] * detected_prob[k]);
  }
}
```

D Code for preprocessing Ebird data

```
library(auk)
library(lubridate)
library(sf)
library(gridExtra)
library(tidyverse)
select <- dplyr::select
dir.create("data", showWarnings = FALSE)

ebd <- auk_ebd("ebd_FI_relOct-2020.txt",
               file_sampling = "ebd_sampling_relOct-2020.txt")
ebd

ebd_filters <- ebd %>%
  auk_species("Eurasian Blue Tit") %>%
  auk_country("Finland") %>%
  # june and july, use * to get data from any year
  auk_date(date = c("*-06-01", "*-07-31")) %>%
  # restrict to the standard traveling and stationary count protocols
  auk_protocol(protocol = c("Stationary", "Traveling")) %>%
  auk_complete()
ebd_filters
```

```
data_dir <- "data"
f_ebd <- file.path(data_dir, "ebd_bluetit_junejuly.txt")
f_sampling <- file.path(data_dir, "ebd_checklists_junejuly.txt")

if (!file.exists(f_ebd)) {
  auk_filter(ebd_filters, file = f_ebd, file_sampling = f_sampling, overwrite = TRUE)
}

ebd_zf <- auk_zerofill(f_ebd, f_sampling, collapse = TRUE)

time_to_decimal <- function(x) {
  x <- hms(x, quiet = TRUE)
  hour(x) + minute(x) / 60 + second(x) / 3600
}

# clean up variables
ebd_zf <- ebd_zf %>%
  mutate(
    # convert X to NA
    observation_count = if_else(observation_count == "X",
                                NA_character_, observation_count),
    observation_count = as.integer(observation_count),
    # effort_distance_km to 0 for non-travelling counts
    effort_distance_km = if_else(protocol_type != "Traveling",
                                 0, effort_distance_km),
    # convert time to decimal hours since midnight
    time_observations_started = time_to_decimal(time_observations_started),
    # split date into year and day of year
    year = year(observation_date),
    day_of_year = yday(observation_date)
  )

ebd_zf$observation_count

# additional filtering
ebd_zf_filtered <- ebd_zf %>%
  filter(
    # effort filters
    duration_minutes <= 5 * 60,
    effort_distance_km <= 5,
    # last 10 years of data
    year >= 2010,
    # 10 or fewer observers
    number_observers <= 10)

ebird <- ebd_zf_filtered %>%
  select(checklist_id, observer_id, sampling_event_identifier,
         scientific_name,
         observation_count, species_observed,
```

```
         state_code, locality_id, latitude, longitude,
         protocol_type, all_species_reported,
         observation_date, year, day_of_year,
         time_observations_started,
         duration_minutes, effort_distance_km,
         number_observers)
write_csv(ebird, "data/ebd_bluetit_junejuly_zf.csv", na = "")
```

E Code for running the models and producing plots

```
library(lubridate)
library(sf)
library(dggridR)
library(raster)
library(ebirdst)
library(fields)
library(tidyverse)
library(rstan)

# resolve namespace conflicts
select <- dplyr::select
projection <- raster::projection

ebird <- read_csv("data/ebd_bluetit_junejuly_zf.csv") %>%
  mutate(year = year(observation_date),
         # occupancy modeling requires an integer response
         species_observed = as.integer(species_observed))
# modis land cover covariates
habitat <- read_csv("data/modis_pland_location-year.csv") %>%
  mutate(year = as.integer(year))

# combine ebird and habitat data
ebird_habitat <- inner_join(ebird, habitat, by = c("locality_id", "year"))

# filter prior to creating occupancy model data
ebird_filtered <- filter(ebird_habitat,
                         number_observers <= 5,
                         year == max(year))

time_freq <- ebird_filtered %>%
  mutate(tod_bins = cut(time_observations_started,
                        breaks = 0:24,
                        labels = 0:23,
                        include.lowest = TRUE),
         tod_bins = as.numeric(as.character(tod_bins))) %>%
  group_by(tod_bins) %>%
  summarise(n_checklists = n(),
            n_detected = sum(species_observed),
            det_freq = mean(species_observed))
```

```
plot(time_freq$tod_bins, time_freq$det_freq )

# load gis data for making maps
map_proj <- st_crs(4326)
ne_land <- read_sf("data/gis-data.gpkg", "ne_land") %>%
  st_transform(crs = map_proj) %>%
  st_geometry()

N_localities <- n_distinct(ebird_filtered$locality_id)
loc_ids <- tibble(locality_id = unique(ebird_filtered$locality_id), locality_simpleid = 1:N_localities)

ebird_filtered <- inner_join(ebird_filtered, loc_ids, by = c("locality_id"))
ebird_filtered$locality_simpleid

habitats <-
  c("pland_00_water",
    "pland_01_evergreen_needleleaf",
    "pland_03_deciduous_needleleaf",
    "pland_04_deciduous_broadleaf",
    "pland_05_mixed_forest",
    "pland_07_open_shrubland",
    "pland_08_woody_savanna",
    "pland_09_savanna",
    "pland_10_grassland",
    "pland_11_wetland",
    "pland_12_cropland",
    "pland_13_urban",
    "pland_15_barren")

habitat_prop <- distinct(inner_join(loc_ids, ebird_filtered[,append(c("locality_id"),habitats)], by=c("locality_id")))
habitat_prop

forest_prop <- ebird_filtered %>%
  select(c("pland_01_evergreen_needleleaf",
           "pland_03_deciduous_needleleaf",
           "pland_04_deciduous_broadleaf",
           "pland_05_mixed_forest")) %>%
  mutate(sum = rowSums(.[1:4])) %>%
  select("sum")

ppcheck_indices <- 1:dim(ebird_filtered)[1]#sample(1:dim(ebird_filtered)[1],300)
ppcheck_indices_habprop <- ebird_filtered$locality_simpleid#ebird_filtered[ppcheck_indices,]$locality_simpleid

stan_fit_1 <- stan("model.stan", data=list(N=dim(ebird_filtered)[1],
                                           observed=ebird_filtered$species_observed,
                                           N_habitats=length(habitats),
                                           N_localities=dim(loc_ids)[1],
                                           habitat_prop=habitat_prop[,habitats],
                                           localities=ebird_filtered$locality_simpleid,
                                           duration_minutes=ebird_filtered$duration_minutes,
```

```
                                        distance_traveled=ebird_filtered$effort_distance_km,
                                        forest_prop=forest_prop$sum,
                                        check_N=length(ppcheck_indices),
                                        check_indices_habprop=ppcheck_indices_habprop,
                                        check_indices_others=ppcheck_indices),
                  iter=2000, chains=4)
save("stan_fit_1",file="data/model1fit")
load("data/model1fit")

stan_fit_3 <- stan("model3.stan", data=list(N=dim(ebird_filtered)[1],
                                          observed=ebird_filtered$species_observed,
                                          N_habitats=length(habitats),
                                          N_localities=dim(loc_ids)[1],
                                          habitat_prop=habitat_prop[,habitats],
                                          localities=ebird_filtered$locality_simpleid,
                                          duration_minutes=ebird_filtered$duration_minutes,
                                          distance_traveled=ebird_filtered$effort_distance_km,
                                          forest_prop=forest_prop$sum,
                                          start_time=ebird_filtered$time_observations_started,
                                          check_N=length(ppcheck_indices),
                                          check_indices_habprop=ppcheck_indices_habprop,
                                          check_indices_others=ppcheck_indices),
                   iter=2000, chains=4)
save("stan_fit_3",file="data/model3fit")
load("data/model3fit")

stan_fit_3 <- stan("model3.stan", data=list(N=dim(ebird_filtered)[1],
                                           observed=ebird_filtered$species_observed,
                                           N_habitats=length(habitats),
                                           N_localities=dim(loc_ids)[1],
                                           habitat_prop=habitat_prop[,habitats],
                                           localities=ebird_filtered$locality_simpleid,
                                           duration_minutes=ebird_filtered$duration_minutes,
                                           distance_traveled=ebird_filtered$effort_distance_km,
                                           forest_prop=forest_prop$sum,
                                           start_time=ebird_filtered$time_observations_started,
                                           check_N=length(ppcheck_indices),
                                           check_indices_habprop=ppcheck_indices_habprop,
                                           check_indices_others=ppcheck_indices),
                   iter=2000, chains=4)
save("stan_fit_3",file="data/model3fit")
load("data/model3fit")

stan_fit_4 <- stan("model4.stan", data=list(N=dim(ebird_filtered)[1],
                                           observed=ebird_filtered$species_observed,
                                           N_habitats=length(habitats),
                                           N_localities=dim(loc_ids)[1],
                                           habitat_prop=habitat_prop[,habitats],
                                           localities=ebird_filtered$locality_simpleid,
                                           duration_minutes=ebird_filtered$duration_minutes,
```

```
                                        distance_traveled=ebird_filtered$effort_distance_km,
                                        forest_prop=forest_prop$sum,
                                        start_time=ebird_filtered$time_observations_started,
                                        check_N=length(ppcheck_indices),
                                        check_indices_habprop=ppcheck_indices_habprop,
                                        check_indices_others=ppcheck_indices),
                  iter=2000, chains=4)
save("stan_fit_4",file="data/model4fit")
load("data/model4fit")

draws_model1 <- rstan::extract(stan_fit_1, permuted = T)
draws_model3 <- rstan::extract(stan_fit_3, permuted = T)
draws_model4 <- rstan::extract(stan_fit_4, permuted = T)


#----------------------DO THE LOGREG POSTERIOR PREDICTIVE THING FROM THE LECTURE SLIDES---------------

plot_ppcheck <- function(draws_model,modeltitle){
  ebird_ppcheck <- tibble(ebird_filtered[ppcheck_indices,], estimated_prob = rowMeans(t(draws_model$observed_check)))
  breaks <- seq(0,0.8,0.1)
  labels <- (breaks[-1] + breaks[-(length(breaks))])/2
  binomial_ppcheck <- ebird_ppcheck %>%
    mutate(bin=cut(ebird_ppcheck$estimated_prob, breaks = breaks,
                   labels = labels,
                   include.lowest = TRUE),
           bin = as.numeric(as.character(bin))) %>%
    group_by(bin) %>%
    summarise(n_checklists = n(),
              n_detected = sum(species_observed),
              det_freq = mean(species_observed),
              sd = sqrt(det_freq*(1-det_freq)/n_checklists))

  plot <- ggplot(binomial_ppcheck) +
    geom_pointrange(aes(x=bin, y=det_freq, ymin=det_freq-2*sd,ymax=det_freq+2*sd)) +
    geom_point(data=ebird_ppcheck,aes(x=estimated_prob, y=species_observed)) +
    geom_line(data=tibble(prob=seq(0,1,0.01),prop=seq(0,1,0.01)),aes(x=prob,y=prop)) +
    xlab("Estimated probability") + ylab("") + labs(title=modeltitle)
  return(plot)
}
library(patchwork)
g <- plot_ppcheck(draws_model1, "Model 1")+ ylab("Bird observed/not observed and binned frequency") +
  plot_ppcheck(draws_model3, "Model 2") + plot_ppcheck(draws_model4, "Model 3")
g
ggsave(g, file="posterior_predictive_1.png" , width=10, height=4)


#--------------------DISTANCE TRAVELED----------------------
library(gridExtra)
plot_ppcheck_dist <- function(draws_model,modeltitle){
# summarize data by 500m bins
```

```
  breaks <- seq(0, 5, by = 0.5)
  labels <- breaks[-length(breaks)] + diff(breaks) / 2
  ebird_dist <- ebird_filtered[ppcheck_indices,] %>%
    mutate(dist_bins = cut(effort_distance_km,
                           breaks = breaks,
                           labels = labels,
                           include.lowest = TRUE),
           dist_bins = as.numeric(as.character(dist_bins))) %>%
    group_by(dist_bins) %>%
    summarise(n_checklists = n(),
              n_detected = sum(species_observed),
              det_freq = mean(species_observed),
              sd = sqrt(det_freq*(1-det_freq)/n_checklists))

  #summarize posterior predictive draws
  pp_dist <- tibble(effort_distance_km=ebird_filtered[ppcheck_indices,]$effort_distance_km,t(draws_model$observed_check)) %
    mutate(mean_probs = rowMeans(.),
           dist_bins = cut(effort_distance_km,
                           breaks = breaks,
                           labels = labels,
                           include.lowest = TRUE),
           dist_bins = as.numeric(as.character(dist_bins))) %>%
    group_by(dist_bins) %>%
    summarise(det_freq = mean(mean_probs))

  #frequency of detection for posterior predictive draws
  g_dist_freq <- ggplot(pp_dist) +
    aes(x = dist_bins, y = det_freq) +
    geom_line(col="red") +
    geom_point(col="red", size=2.4) +
    scale_x_continuous(breaks = 0:5) +
    scale_y_continuous(labels = scales::percent) +
    labs(x = "Distance travelled (km)",
         y = "% checklists with detections",
         title = modeltitle) +
    geom_line(data=ebird_dist,aes(x = dist_bins, y = det_freq)) +
    geom_point(data=ebird_dist,aes(x = dist_bins, y = det_freq)) +
    geom_pointrange(data=ebird_dist, aes(x=dist_bins, y=det_freq, ymin=det_freq-2*sd,ymax=det_freq+2*sd))
  return(g_dist_freq)
}

g <-plot_ppcheck_dist(draws_model1,"Model 1") + plot_ppcheck_dist(draws_model3,"Model 2") + plot_ppcheck_dist(draws_model4,
g
ggsave(g, file="posterior_predictive_dist.png" , width=10, height=4)

#----------------------START TIME PLOTS-------------------------
plot_ppcheck_startime <- function(draws_model,modeltitle){
  ebird_ppcheck <- tibble(ebird_filtered[ppcheck_indices,], estimated_prob = rowMeans(t(draws_model$observed_check)))
  breaks <- 0:24
  labels <- breaks[-length(breaks)] + diff(breaks) / 2
```

```
  starttime_ppcheck <- ebird_ppcheck %>%
    mutate(tod_bins = cut(time_observations_started,
                          breaks = breaks,
                          labels = labels,
                          include.lowest = TRUE),
           tod_bins = as.numeric(as.character(tod_bins))) %>%
    group_by(tod_bins) %>%
    summarise(n_checklists = n(),
              n_detected = sum(species_observed),
              det_freq = mean(species_observed),
              estimated_prob = mean(estimated_prob))
  g_return <- ggplot(starttime_ppcheck) +
    geom_line(aes(x=tod_bins,y=det_freq)) + geom_point(aes(x=tod_bins,y=det_freq)) +
    geom_line(aes(x=tod_bins,y=estimated_prob),color='red') + geom_point(aes(x=tod_bins,y=estimated_prob),color='red') +
    labs(x = "Start time (h)",
         y = "% checklists with detections",
         title = modeltitle)
  return(g_return)
}
g <-plot_ppcheck_startime(draws_model1,"Model 1") + plot_ppcheck_startime(draws_model3,"Model 2") + plot_ppcheck_startime(d
g
ggsave(g, file="posterior_predictive_time.png" , width=10, height=4)

#PSIS-LOO CHECKS
probs_model1 <- draws_model1$occupied_prob*draws_model1$detected_prob
probs_model3 <- draws_model3$occupied_prob*draws_model3$detected_prob
probs_model4 <- draws_model4$occupied_prob*draws_model4$detected_prob
loo(log(probs_model1))
loo(log(probs_model3))
loo(log(probs_model4))

#ACCURACIES
mean(ebird_filtered$species_observed == (colMeans(probs_model1) > 0.5))
mean(ebird_filtered$species_observed == (colMeans(probs_model3) > 0.5))
mean(ebird_filtered$species_observed == (colMeans(probs_model4) > 0.5))

#----------------------------PRIOR SENSITIVITY CHECKS---------------------------------

stan_fit_4_2 <- stan("model4.stan", data=list(N=dim(ebird_filtered)[1],
                                              observed=ebird_filtered$species_observed,
                                              N_habitats=length(habitats),
                                              N_localities=dim(loc_ids)[1],
                                              habitat_prop=habitat_prop[,habitats],
                                              localities=ebird_filtered$locality_simpleid,
                                              duration_minutes=ebird_filtered$duration_minutes,
                                              distance_traveled=ebird_filtered$effort_distance_km,
                                              forest_prop=forest_prop$sum,
                                              start_time=ebird_filtered$time_observations_started,
                                              check_N=length(ppcheck_indices),
                                              check_indices_habprop=ppcheck_indices_habprop,
```

```
                                          check_indices_others=ppcheck_indices),
                  iter=2000, chains=4)
save("stan_fit_4_2",file="data/model4fit_2")
load("data/model4fit_2")

stan_fit_4_3 <- stan("model4.stan", data=list(N=dim(ebird_filtered)[1],
                                              observed=ebird_filtered$species_observed,
                                              N_habitats=length(habitats),
                                              N_localities=dim(loc_ids)[1],
                                              habitat_prop=habitat_prop[,habitats],
                                              localities=ebird_filtered$locality_simpleid,
                                              duration_minutes=ebird_filtered$duration_minutes,
                                              distance_traveled=ebird_filtered$effort_distance_km,
                                              forest_prop=forest_prop$sum,
                                              start_time=ebird_filtered$time_observations_started,
                                              check_N=length(ppcheck_indices),
                                              check_indices_habprop=ppcheck_indices_habprop,
                                              check_indices_others=ppcheck_indices),
                     iter=2000, chains=4)
save("stan_fit_4_3",file="data/model4fit_3")
load("data/model4fit_3")

draws_model4_2 <- rstan::extract(stan_fit_4_2, permuted = T)
draws_model4_3 <- rstan::extract(stan_fit_4_3, permuted = T)

g <- plot_ppcheck(draws_model4, "Model 3, normal")+ ylab("Bird observed/not observed and binned frequency") +
  plot_ppcheck(draws_model4_2, "Model 3, tight prior") + plot_ppcheck(draws_model4_3, "Model 3, wide prior")
g
ggsave(g, file="posterior_predictive_sensitivity.png" , width=10, height=4)


#----------------------------RESULT ANALYSIS-----------------------------

habitat_names <- c("Water","Evergreen needleleaf","Deciduous needleleaf", "Deciduous broadleaf", "Mixed forest", "Open shru
                   "Savanna", "Grassland", "Wetland", "Cropland", "Urban", "Barren")

habitat_coef_draws <- data.frame(draws_model4$habitat_coef)
names(habitat_coef_draws) <- habitat_names
occ_plot <- ggplot(stack(habitat_coef_draws), aes(x=ind,y=values)) +
  geom_boxplot()+ggtitle("Occupancy coefficients") +
  theme(axis.text.x = element_text(face = "bold", color = "#993333", size = 8, angle = 45,hjust = 1)) +
  labs(x="",y="")

detection_coef_draws <- tibble(duration_coef = draws_model$duration_coef,
                               distance_coef=draws_model$distance_coef,
                               forest_prop_coef=draws_model$forest_prop_coef) %>%
                        mutate(duration_coef = duration_coef * sd(ebird_filtered$duration_minutes),
                               distance_coef = distance_coef * sd(ebird_filtered$effort_distance_km),
                               forest_prop_coef = forest_prop_coef * sd(forest_prop$sum))
det_plot <- ggplot(pivot_longer(detection_coef_draws,cols=c("duration_coef", "distance_coef", "forest_prop_coef")), aes(x=n
```

```
  geom_boxplot()+ggtitle("Normalized detectability coefficients") +
  theme(axis.text.x = element_text(face = "bold", color = "#993333", size = 8, angle = 45,hjust = 1)) +
  labs(x="",y="")

g <- occ_plot+det_plot
ggsave(g, file="coefs.png" , width=10, height=4)
```

F Code for producing the distribution map

```
library(sf)
library(raster)
library(MODIS)
library(exactextractr)
library(viridis)
library(tidyverse)
library(boot)
# resolve namespace conflicts
select <- dplyr::select
map <- purrr::map
projection <- raster::projection

#-------------LOAD MAP DATA---------------
ne_land <- read_sf("data/gis-data.gpkg", "ne_land") %>%
  #Project to the native MODIS projection
  st_transform(crs = paste("+proj=sinu +lon_0=0 +x_0=0 +y_0=0",
                           "+a=6371007.181 +b=6371007.181 +units=m +no_defs"))


#-------------LOAD MODIS DATA---------------
landcover <- list.files("data/modis", "^modis_mcd12q1_umd",
                        full.names = TRUE) %>%
  stack()
landcover <- names(landcover) %>%
  str_extract("(?<=modis_mcd12q1_umd_)[0-9]{4}") %>%
  paste0("y", .) %>%
  setNames(landcover, .)
landcover

max_lc_year <- names(landcover) %>%
  str_extract("[0-9]{4}") %>%
  as.integer() %>%
  max()

#-----------GET PREDICTION SURFACE----------
pland <-read_csv("data/modis_pland_location-year.csv") %>%
  mutate(year = as.integer(year))

agg_factor <- round(2 * neighborhood_radius / res(landcover))
```

```
r <- raster(landcover) %>%
  aggregate(agg_factor)
r <- ne_land %>%
  st_transform(crs = projection(r)) %>%
  rasterize(r, field = 1) %>%
  # remove any empty cells at edges
  trim()
r <- writeRaster(r, filename = "data/prediction-surface.tif", overwrite = TRUE)

# get cell centers and create neighborhoods
r_centers <- rasterToPoints(r, spatial = TRUE) %>%
  st_as_sf() %>%
  transmute(id = row_number())
r_cells <- st_buffer(r_centers, dist = neighborhood_radius)

# extract landcover values within neighborhoods, only needed most recent year
lc_extract_pred <- landcover[[paste0("y", max_lc_year)]] %>%
  exact_extract(r_cells, progress = FALSE) %>%
  map(~ count(., landcover = value)) %>%
  tibble(id = r_cells$id, data = .) %>%
  unnest(data)

# calculate the percent for each landcover class
pland_pred <- lc_extract_pred %>%
  group_by(id) %>%
  mutate(pland = n / sum(n)) %>%
  ungroup() %>%
  select(-n) %>%
  # remove NAs after tallying so pland is relative to total number of cells
  filter(!is.na(landcover))

# convert names to be more descriptive
pland_pred <- pland_pred %>%
  inner_join(lc_names, by = "landcover") %>%
  arrange(landcover) %>%
  select(-landcover)

# tranform to wide format, filling in implicit missing values with 0s
pland_pred <- pland_pred %>%
  pivot_wider(names_from = lc_name,
              values_from = pland,
              values_fill = list(pland = 0)) %>%
  mutate(year = max_lc_year) %>%
  select(id, year, everything())

# join in coordinates
pland_coords <- st_transform(r_centers, crs = 4326) %>%
  st_coordinates() %>%
  as.data.frame() %>%
  cbind(id = r_centers$id, .) %>%
```

```
  rename(longitude = X, latitude = Y) %>%
  inner_join(pland_pred, by = "id")

#---------------LOAD MODEL-------------------
load("data/model4fit")
draws_model4 <- rstan::extract(stan_fit_4, permuted = T)

pland_coords
habitats <-
  c("pland_00_water",
    "pland_01_evergreen_needleleaf",
    "pland_03_deciduous_needleleaf",
    "pland_04_deciduous_broadleaf",
    "pland_05_mixed_forest",
    "pland_07_open_shrubland",
    "pland_08_woody_savanna",
    "pland_09_savanna",
    "pland_10_grassland",
    "pland_11_wetland",
    "pland_12_cropland",
    "pland_13_urban",
    "pland_15_barren")

pland_coords[,5:17]

draws_model4$habitat_coef
occupancy_means <- 0*(1:dim(pland_coords)[1])
occupancy_sds <- 0*(1:dim(pland_coords)[1])

n_samples <- dim(draws_model4$habitat_coef)[1]
dim(pland_coords)

for(i in 1:dim(pland_coords)[1]){
  probs <- inv.logit(rowMeans(draws_model4$habitat_coef * pland_coords[i,5:17][rep.int(1, n_samples),]))
  occupancy_means[i] <- mean(probs)
  occupancy_sds[i] <- sd(probs)
  if(i %% 1000 == 0){
    print(i)
  }
}

occupancy_tibble <- tibble(id=1:dim(pland_coords)[1], occupancy_means=occupancy_means, occupancy_sds=occupancy_sds)

expected_occupancy_cover <- pland_coords %>% inner_join(occupancy_tibble, by="id") %>%
  # convert to spatial features
  st_as_sf(coords = c("longitude", "latitude"), crs = 4326) %>%
  st_transform(crs = projection(r)) %>%
  # rasterize points
  rasterize(r, field = occupancy_means) %>%
  # project to albers equal-area for mapping
```

```
  projectRaster(crs = st_crs(4326)$proj4string, method = "ngb") %>%
  # trim off empty edges of raster
  trim()

uncertainty_occupancy_cover <- pland_coords %>% inner_join(occupancy_tibble, by="id") %>%
  # convert to spatial features
  st_as_sf(coords = c("longitude", "latitude"), crs = 4326) %>%
  st_transform(crs = projection(r)) %>%
  # rasterize points
  rasterize(r, field = occupancy_sds) %>%
  # project to albers equal-area for mapping
  projectRaster(crs = st_crs(4326)$proj4string, method = "ngb") %>%
  # trim off empty edges of raster
  trim()

# make a map
par(mfrow=c(1,2),mar = c(0.25, 0.25, 2, 0.25))

t <- str_glue("Estimated occupancy of\n",
              "the Eurasian Blue Tit in 2020, June-July")
plot(expected_occupancy_cover, axes = FALSE, box = FALSE, col = viridis(10), main = t)


t <- str_glue("Occupancy uncertainty of\n",
              "the Eurasian Blue Tit in 2020, June-July")
p2 <- plot(uncertainty_occupancy_cover, axes = FALSE, box = FALSE, col = viridis(10), main = t)
```