A tutorial on the fitting of spatial occupancy models to Swiss Bird Atlas data in SE Switzerland with `spOccupancy` and `ubms`

Rock bunting (*Emberiza cia*), https://www.featherbase.info/uk/exhibit/9110/

MK, February-June 2024

More info on the R package `spOccupancy` by Jeff Doser:

- https://www.jeffdoser.com/files/spoccupancy-web/
- https://besjournals.onlinelibrary.wiley.com/doi/full/10.1111/2041-210X.13897
- https://cran.r-project.org/web/packages/spOccupancy/index.html
- https://www.jeffdoser.com/files/spoccupancy-web/articles/modelfitting

More info on the R package `ubms` by Ken Kellner:

- https://cran.r-project.org/web/packages/ubms/index.html
- https://besjournals.onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13777
- https://cran.r-project.org/web/packages/ubms/vignettes/spatial-models.html

Table of contents

1 Introduction 2
2 Preparation and summary of the Rock bunting and habitat data 5
3 Fitting nonspatial models in `unmarked`, `ubms` and `spOccupancy` 15
3.3 Spatial predictions from the nonspatial model: first species distribution maps 23
4 Fitting spatial models to the SE Swiss Rock Bunting data with `spOccupancy` 31
4.1 Spatial exponential model with 5-Nearest-neighbour Gaussian Process (NNGP 5) 31
4.2 Spatial exponential model with 15-Nearest-neighbour Gaussian Process (NNGP 15) 46
5 Fitting restricted spatial regression (RSR) models with `ubms` 51

1 Introduction

In this document, we show how to fit spatial occupancy models using functionality in the R packages `spOccupancy` (Doser et al., 2022) and `ubms` (Kellner et al. 2021). We restrict ourselves to the simplest case of a static occupancy model and will also compare non-spatial and spatial models. For the former, will use `unmarked` in addition to `spOccupancy` and `ubms` (both of the latter can also fit nonspatial occupancy models). While `unmarked` uses maximum likelihood and is a very general package for models of distribution (presence/absence) and abundance, `spOccupancy` and `ubms` use Bayesian inference with Markov chain Monte Carlo algorithms. However, in contrast to software such as JAGS, NIMBLE and Stan, setting up and running these algorithms is for the most part automated and will run with minimal or even no user input (although it can still be very helpful to understand some of what goes on under the hood of `spOccupancy` and `ubms`).

`spOccupancy` lets the user fit a wide range of occupancy models for single and multiple species and for single "seasons" (i.e., static) as well as for multiple "seasons" (i.e., with changes in occupancy over time). This includes models with "interactions" between species, or more accurately, with statistical residual correlations among their spatial or spatiotemporal occurrence. `umbs` could be called the Bayesian sister package of `unmarked` and provides Bayesian solutions to many (though not all) of the models implemented in `unmarked`. This can be very useful when you need to fit additional sets of random effects, e.g., for years or when sites are nested within regions. [Although, with the new TMB engine in `unmarked`, some of that can now also be done with `unmarked`].

Two of the key selling points of `spOccupancy` are the ability to deal with spatial formulations of all of its models, and in addition, its ability to fit these models even to very large data sets, such as when you have data from 50–100,000 sites. This opens up the option to fit spatial models and then obtain from them predictions for mapping even at subcontinental scales, as you can see in many recent papers by Jeff Doser and his colleagues. While `spOccupancy` is specialized for spatial models, adding spatial formulations in the model is only some of what `ubms` is meant for, and `ubms` is not optimized for large data sets.

Many ecologists may believe that a spatial model may be as simple as a GLM with spatially indexed covariates, from which predictions can be made onto some larger domain and a map can be produced. But this is not so. Statisticians call a spatial model one in which space appears in an explicit manner. Almost all of the times, this is via the matrix of the pairwise distances among all sites or by some other information that describes the spatial configuration of all of the study sites, i.e., their location relative to one another. A spatial model explicitly describes the so-called 'residual spatial autocorrelation', e.g., in occupancy. This is the remaining (therefore residual) correlation between sites, after everything that the covariates in the model can explain has been accounted for. This residual autocorrelation is formulated such that sites that are closer to one another have a greater correlation of the residuals than sites which are further apart. This distance-related relationship among sites is imposed in the model by one of a monotonically declining functions such as the (negative) exponential or Matérn correlation functions.

Spatial models have several advantages over non-spatial models if there indeed is such residual spatial autocorrelation:

- They get tests for fixed effects right (i.e., for the covariates that you have in the model), since they properly accommodate for any lack of independence among sites
- They can be much better for producing species distribution maps than nonspatial models, since in addition to the information coming from the covariates in the model, they also exploit the information that comes from the "neighbourhood relations".

However, spatial models are more complex than nonspatial models and thus, for instance, are harder to fit. That is, it usually takes longer to fit them, and sometimes you may not get convergence for them. Also, spatial models are mostly fit with Bayesian computational methods, while nonspatial models can very often be fit using straight maximum likelihood. It will not always be necessary to fit spatial models for distribution and abundance and if it is not, then the simpler non-Bayesian formulations or fitting algorithms may be better (since much faster). Some things to keep in mind when taking the decision between a spatial and a nonspatial formulation of a distribution or abundance model (in addition to the above) are these:

- When you fit a nonspatial model to data where there is residual spatial autocorrelation, then often you will not get biased estimates of the coefficients in your model, but "simply" get too optimistic uncertainty assessments. That is, you will tend to get too short standard errors and too narrow confidence intervals. Consequently, your statisticial tests will be too liberal, i.e., you will claim significance too often.
- (There may be cases when ignoring spatial autocorrelation will lead to bias, but I am frankly not sure when this happens.)
- Perhaps the most common mechanism creating residual spatial autocorrelation is by "forgotten covariates". That is, covariates that affect the quantity of your interest, such as occupancy or expected abundance, which are not in your model and which themselves are spatially structured. That is, their values are more similar at two sites that are near than at two sites that are more distant to each other. Therefore, if you know all the relevant covariates, then in theory, you would no longer need a spatial model. In general, the better your covariates, the less there may be a need for accommodating residual spatial autocorrelation in your model.
- The larger your study area (or the modelled 'domain', as statisticians say) the more there will usually be a need to accommodate in your model residual spatial autocorrelation.
- Fitting spatial models is virtually always more computationally costly than fitting nonspatial models and often very much so. Unfortunately, this may be a very practical reason for avoiding to fit a spatial formulation of a model in some cases.
- Spatial models are a truly vast field in statistics and there are very many different formulations of the "neighbourhood relations". It is my impression that at least for the purposes of prediction (i.e., when you basically want to make a species distribution map), often different spatial models lead to fairly similar maps. Therefore, as a first approximation, I'd not be too worried about which specific spatial model too choose. However, we will see some minimal comparison in this respect in this paper, since `ubms` and `spOccupancy` use different formulations of spatial autocorrelation. Indeed, in the latter package we can choose among a variety of them, e.g., an exponential or a Matérn correlation function.

In this tutorial, we will work with a subset of the data that were collected and used for the last Swiss breeding bird atlas, with its field work conducted during 2013–2016 (Knaus *et al.*, 2018). Specifically, we will use the data from one species, the Rock bunting (*Emberiza cia*); see title page. This is mostly a Mediterranean or at least warm-climate-loving species of open and semi-open landscapes. We will restrict attention to the South-Eastern quarter of Switzerland, in some parts of which the species is widespread, while in others, much less so. Spatial units (i.e., "sites") in our data are the 1 km^2 squares of the Swiss topographical grid system. Temporal units (i.e., "occasions" in the usual occupancy modeling parlance) are weeks. The original data (not seen here) had daily resolution, but we then aggregated these to weeks.

We will fit some regular, nonspatial occupancy models and then also spatial occupancy models with an exponential or other covariance matrix (in `spOccupancy`) and with a restricted spatial regression (RSR; Johnson et al. 2013) formulation in `ubms`. In the exponential correlation model, we assume that for each site there is a 'residual' spatial random effect *w* in the occupancy model on the logit scale, and the model for the *w*'s for all *J* sites is a multivariate normal distribution with a mean vector of zero and a variance-covariance matrix that is a function of the pairwise distance between all sites. The two parameters used to describe this residual autocorrelation in the exponential covariance model are: the spatial variance (sigma.sq) and a decay parameter (phi). The decay parameter is inversely related to the range of the spatial autocorrelation: if phi is small, the range is large, i.e., there is some residual autocorrelation even between sites that are far apart, and vice versa. One computationally highly beneficial approximation in `spOccupancy` is the so-called Nearest Neighbour Gaussian Process (NNGP), in which only the pairwise distances within some neighbourhood around each site (e.g., the closest 5, 10 or 15 neighbours) is considered. See the `spOccupancy` resources for much more information about the types of models that are fit and the computational techniques adopted for them.

The RSR formulation is a computationally beneficial approximation to an intrinsic CAR (conditionally autoregressive) model (Johnson et al., 2013), which in addition avoids some problems due to spatial collinearity that may arise in a CAR model. That is, spatial confounding, which denotes a correlation between the fixed effects in a model and the structure to accommodate the spatial covariance. CAR models are widely used to model spatial autocorrelation. Their computational tractability stems in part from their consideration of only a defined neighbourhood around each site, for which the spatial autocorrelation is defined directly in the model. See many resources for more about CAR and RSR models.

In this tutorial, we will first obtain, manage and summarize the Swiss Rock Bunting data in South-East Switzerland (often abbreviated as SE-CH or similar), along with the covariates used for the modeling (Section 2). Then, we use functions in `unmarked`, `spOccupancy` and `ubms` to fit a nonspatial occupancy model and to obtain predictions in geographic space such that we can produce first species distribution maps (Section 3). Then, we move to spatial models in Section 4, and use `spOccupancy` to explore two formulations of Nearest-Neighbour Gaussian Process (NNGP) spatial models, one with 5 and the other with 15 nearest neighbours. Finally, in Section 5 we fit some spatial occupancy models with an RSR formulation of space in `ubms`.

Throughout, the main goal of our modeling will be prediction (e.g., the production of accurate species distribution maps or derivatives, such as an estimate of the number of occupied sites in the modelled region), rather than inference about mechanism (Tredennick et al. 2021, Chapter 18 in Kéry & Kellner, 2024). Thus, we will pay some, but mostly passing only, attention to the estimates of the coefficients in the regression models.

A note on the format:
In this tutorial we have normal text, R and other computer code, and numerical as well as graphical output. Code is set in `Courier New` font, normal text in Calibri font, and numerical output is highlighted in grey boxes.

2 Preparation and summary of the Rock bunting and habitat data

We work with data in an R workspace named *"Data_RockBunting_SE_Switzerland.Rdata"*. We open the workspace and check out what's in it. We see two objects:

```
# Look at contents of the R workspace
ls()
```

```
[1] "AtlasData" "CovarData"
```

The first object in the R workspace contains the detection/nondetection data of Swiss Rock Buntings (*Emberiza cia*) in the SE quarter of the country, along with the environmental and survey covariates that were used to produce the species distribution maps in the last Swiss breeding bird atlas (Knaus *et al.* 2018). This data set contains data from those 6,338 1 km^2 quadrats that were surveyed at least once in this domain during the Atlas survey period 2013–2016. The temporal resolution of the data is 1 week, and we see that the data extend over a total period of 21 weeks (or almost 5 months). This is the breeding season of the Rock Bunting in Switzerland.

```
str(AtlasData)
```

```
List of 4
 $ y        : num [1:6338, 1:21] NA NA NA NA NA NA NA NA NA NA ...
 $ occ.covs: num [1:6338, 1:34] -0.427 -0.427 -0.433 -0.433 -0.425 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6338] "674_146" "674_147" "674_148" "674_156" ...
  .. ..$ : chr [1:34] "z.buildings" "z.buildings2" "z.elevation"
"z.elevation2" ...
 $ det.covs:List of 3
  ..$ date : num [1:6338, 1:21] 103 103 103 103 103 103 103 103 103 103
...
  ..$ date1: num [1:6338, 1:21] -1.07 -1.07 -1.07 -1.07 -1.07 ...
  ..$ date2: num [1:6338, 1:21] 1.14 1.14 1.14 1.14 1.14 ...
 $ coords  : num [1:6338, 1:2] 674 674 674 674 674 674 674 674 674 674
...
```

The detection/nondetection data are in object 'y'. They contain an NA when a site is not visited at least once during a week, a 0 when it was visited at least once and no Rock bunting was detected, and a 1 when it was visited at least once and a Rock Bunting was detected on at least one visit. When arrayed in a balanced manner, most of the data in 'y' will actually be missing values, or NAs; see later.

There are 17 environmental covariates, which come with both a linear and quadratic term, and which are standardized into standard normal deviates (that's why each starts with a z in the name). Note that in the other R object in the workspace, see below, we will also have these covariates in their original, i.e., not transformed, format. This is nicer for plotting and for understanding what they mean, while the standardized format is how we work with them when fitting them in a model.

```
head(AtlasData$occ.covs, 3)
```

```
        z.buildings z.buildings2 z.elevation z.elevation2
674_146  -0.4268965    0.3640705    1.346041  -0.07443274
674_147  -0.4272756    0.3647529    1.194785  -0.37562009
674_148  -0.4334047    0.3757985    1.496057   0.27202054
        z.farmland z.farmland2   z.forest z.forest2   z.glacier
674_146 -0.5120273   0.3091669 -0.9483997 0.7247866 -0.2097548
```

```
674_147 -0.5120273   0.3091669 -0.9483997 0.7247866 -0.2097548
674_148 -0.5120273   0.3091669 -0.9483997 0.7247866 -0.2097548
        z.glacier2 z.grassland z.grassland2    z.lakes
674_146   0.120621   0.6710748   -0.9707944 -0.2112924
674_147   0.120621   1.6771004    0.7108712 -0.2112924
674_148   0.120621   0.7917979   -0.8742177 -0.2112924
         z.lakes2 z.nitrogen z.nitrogen2 z.northness
674_146 0.1213143  -1.062655   0.3419142   0.5446114
674_147 0.1213143  -1.025065   0.2615459   0.3762253
674_148 0.1213143  -1.112776   0.4531331  -1.8368490
        z.northness2   z.rivers  z.rivers2    z.roads
674_146   -0.5597263 -0.1584931 -0.5429587 -0.7249713
674_147   -0.7149283  0.7563764 -0.8704290 -0.3924721
674_148    1.8893398  1.1126665 -0.6893568 -0.6600589
           z.roads2   z.rocks  z.rocks2 z.shoreline
674_146  0.60084795 1.6501944 -1.993127 -0.02552785
674_147 -0.01698022 0.6214527 -2.064699 -0.29644334
674_148  0.47014079 1.5644659 -2.083225  1.19064711
        z.shoreline2   z.slope   z.slope2 z.structures
674_146   -0.2025498 0.4011582 -0.9425887   -0.6171898
674_147    0.1942575 0.9190762 -0.2566352   -0.5091366
674_148   -1.5531720 0.2942862 -1.0122284   -0.6171898
        z.structures2 z.wetlands z.wetlands2 z.kfrivers
674_146     0.6014560 -0.1332426  0.09988408 -0.6187885
674_147     0.3835374 -0.1600886  0.14434276  1.6291665
674_148     0.6014560 -0.1600886  0.14434276  0.3890734
        z.kfrivers2
674_146   0.3295889
674_147  -0.8468447
674_148  -1.0823000
```

The detection covariates are the raw and the standardized, linear and quadratic dates of the survey week. Note that the detection covariates are complete, i.e., the have a non-missing value even when in the corresponding week there was no recorded visit at a site.

```
str(AtlasData$det.covs)
```

```
List of 3
 $ date : num [1:6338, 1:21] 103 103 103 103 103 103 103 103 103 103 ...
 $ date1: num [1:6338, 1:21] -1.07 -1.07 -1.07 -1.07 -1.07 ...
 $ date2: num [1:6338, 1:21] 1.14 1.14 1.14 1.14 1.14 ...
```

The final object in the 'AtlasData' object contains the coordinates of the surveyed quadrats, in kilometric units of the Swiss topographic system.

The second object in the R workspace contains the landscape data for the entire modelled domain, and we need this to produce the species distribution maps, as we will see below. We have the covariates in their raw form and then standardized in linear and quadratic form. At the start we have again the coordinates of the 1 km^2 quadrats in kilometric units of the Swiss topographic system. We are not going to explain what these covariates are. Most names should be self-explanatory and if not, then the covariate doesn't matter. Also, we're not so much after the biology of the Rock bunting in this tutorial for now, but want to illustrate the modeling and mapping techniques in the first place.

```
str(CovarData)
head(CovarData)     # not shown
```

```
'data.frame':   12757 obs. of  53 variables:
 $ x            : num  674 675 676 677 678 679 680 681 682 683 ...
 $ y            : num  210 210 210 210 210 210 210 210 210 210 ...
 $ buildings    : int  30072 23316 7580 4474 23732 7973 2084 1336 3076
15234 ...
 $ elevation    : int  476 568 830 1194 1350 1397 1302 1109 855 675 ...
 $ farmland     : num  0.02 0 0 0 0 0 0 0 0 0 ...
 $ forest       : num  0.09 0.05 0.59 0.46 0.28 0.3 0.35 0.71 0.66 0.21
...
 $ glacier      : num  0 0 0 0 0 0 0 0 0 0 ...
 $ grassland    : num  0.63 0.75 0.28 0.39 0.46 0.51 0.6 0.21 0.25 0.62
...
 $ lakes        : num  0.15 0 0 0 0 0 0 0 0 0 ...
 $ nitrogen     : num  23.2 23 24.1 20.1 18.3 18.3 19.7 23.4 27.1 25.9
...
 $ northness    : num  0.05 -0.18 -0.71 -0.85 -0.84 -0.01 -0.36 0.23 0.64
0.58 ...
 $ rivers       : int  1466 581 797 3024 5100 2390 3710 6087 8433 4889
...
 $ roads        : int  3731 3707 2709 687 2081 1253 39 983 1945 4858 ...
 $ rocks        : num  0 0 0 0 0 0.01 0 0.03 0.01 0 ...
 $ shoreline    : int  780 107 0 0 0 0 0 0 0 80 ...
 $ slope        : num  8 14.8 29.4 28.3 22 26.1 24.5 30.5 28.5 15.9 ...
 $ structures   : num  0.01 0.1 0.08 0.09 0.17 0.1 0.03 0.03 0.04 0.05
...
 $ wetlands     : int  0 0 0 0 0 0 0 0 0 2988 ...
 $ kfrivers     : num  0 0 0 0 58.7 ...
 $ z.buildings  : num  0.517 0.303 -0.194 -0.292 0.316 ...
 $ z.buildings2 : num  -1.0297 -0.7677 -0.0367 0.1276 -0.7848 ...
 $ z.elevation  : num  -1.0331 -0.9191 -0.5943 -0.143 0.0504 ...
 $ z.elevation2 : num  0.786 0.472 -0.272 -0.936 -1.088 ...
 $ z.farmland   : num  -0.408 -0.512 -0.512 -0.512 -0.512 ...
 $ z.farmland2  : num  0.0257 0.3092 0.3092 0.3092 0.3092 ...
 $ z.forest     : num  -0.638 -0.776 1.088 0.639 0.018 ...
 $ z.forest2    : num  -0.113 0.232 -0.753 -1.241 -1.158 ...
 $ z.glacier    : num  -0.21 -0.21 -0.21 -0.21 -0.21 ...
 $ z.glacier2   : num  0.121 0.121 0.121 0.121 0.121 ...
 $ z.grassland  : num  1.3149 1.7978 -0.0935 0.3491 0.6308 ...
 $ z.grassland2 : num  -0.124 1.047 -0.916 -1.088 -0.997 ...
 $ z.lakes      : num  0.682 -0.211 -0.211 -0.211 -0.211 ...
 $ z.lakes2     : num  -3.974 0.121 0.121 0.121 0.121 ...
 $ z.nitrogen   : num  0.98 0.955 1.093 0.591 0.366 ...
 $ z.nitrogen2  : num  -0.2421 -0.2817 -0.0499 -0.7246 -0.8768 ...
 $ z.northness  : num  0.0635 -0.4898 -1.7647 -2.1015 -2.0774 ...
 $ z.northness2 : num  -0.871 -0.724 1.668 2.778 2.692 ...
 $ z.rivers     : num  -0.106 -0.683 -0.542 0.911 2.266 ...
 $ z.rivers2    : num  -0.5929 0.1594 -0.0661 -0.8131 1.0825 ...
 $ z.roads      : num  0.3326 0.3258 0.0429 -0.5302 -0.1351 ...
 $ z.roads2     : num  -0.919 -0.913 -0.632 0.223 -0.407 ...
 $ z.rocks      : num  -0.493 -0.493 -0.493 -0.493 -0.493 ...
 $ z.rocks2     : num  0.344 0.344 0.344 0.344 0.344 ...
 $ z.shoreline  : num  2.0004 0.0186 -0.2964 -0.2964 -0.2964 ...
 $ z.shoreline2 : num  -2.062 -0.264 0.194 0.194 0.194 ...
 $ z.slope      : num  -0.914 -0.355 0.845 0.755 0.237 ...
 $ z.slope2     : num  -0.0879 -0.9065 -0.39 -0.537 -1.0395 ...
 $ z.structures : num  -0.509 0.463 0.247 0.355 1.22 ...
```

```
 $ z.structures2: num  0.384 -1.063 -0.822 -0.948 -1.549 ...
 $ z.wetlands    : num  -0.16 -0.16 -0.16 -0.16 -0.16 ...
 $ z.wetlands2   : num  0.144 0.144 0.144 0.144 0.144 ...
 $ z.kfrivers    : num  -0.619 -0.619 -0.619 -0.619 -0.528 ...
 $ z.kfrivers2   : num  0.33 0.33 0.33 0.33 0.144 ...
```

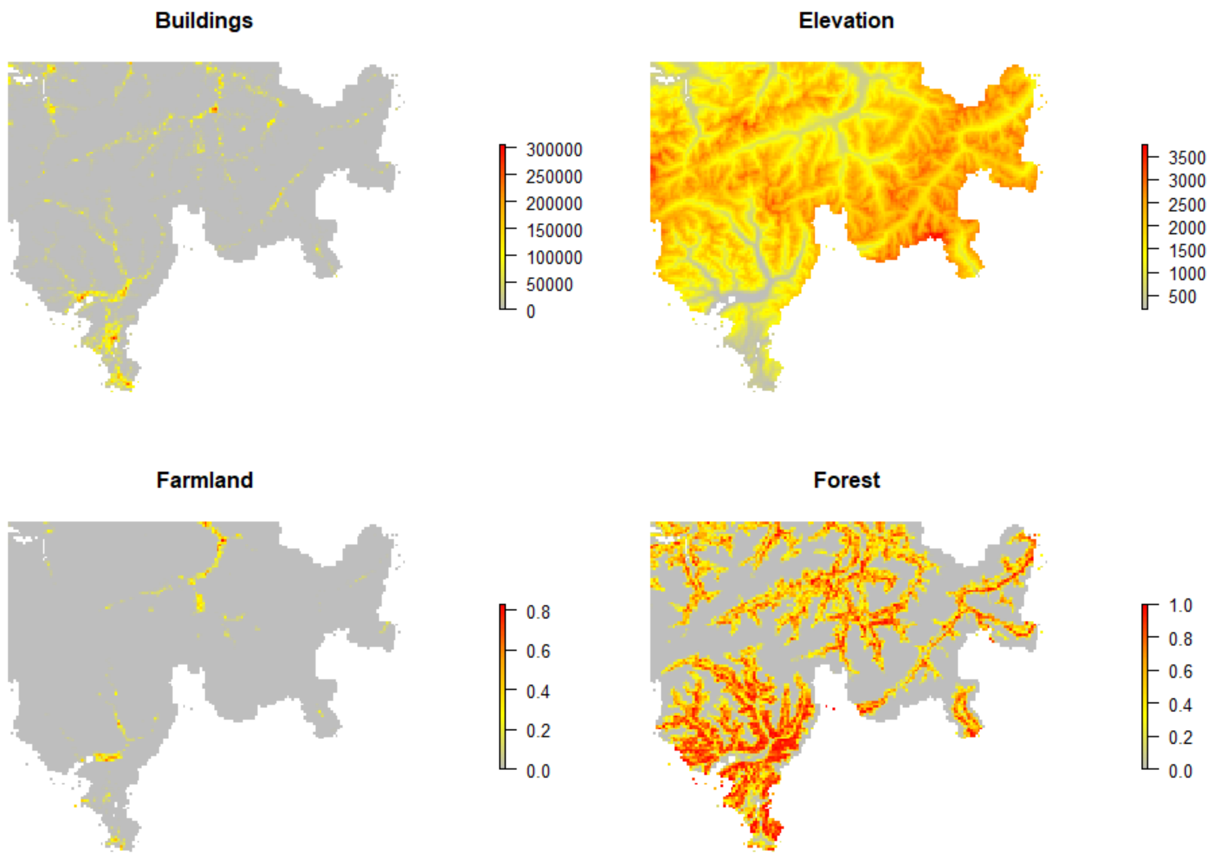
These two R objects contain all our data for playing around with `spOccupancy` and other functions for fitting a number of spatial and non-spatial occupancy models. As a final thing, note that for prediction, we have about 13,000 kilometric pixels and that the north-south and west-east extensions of the modelled domain are at most 136 and 163 kilometres.

```
# Compute extents of the prediction domain
diff(range(CovarData$x))
diff(range(CovarData$y))
```

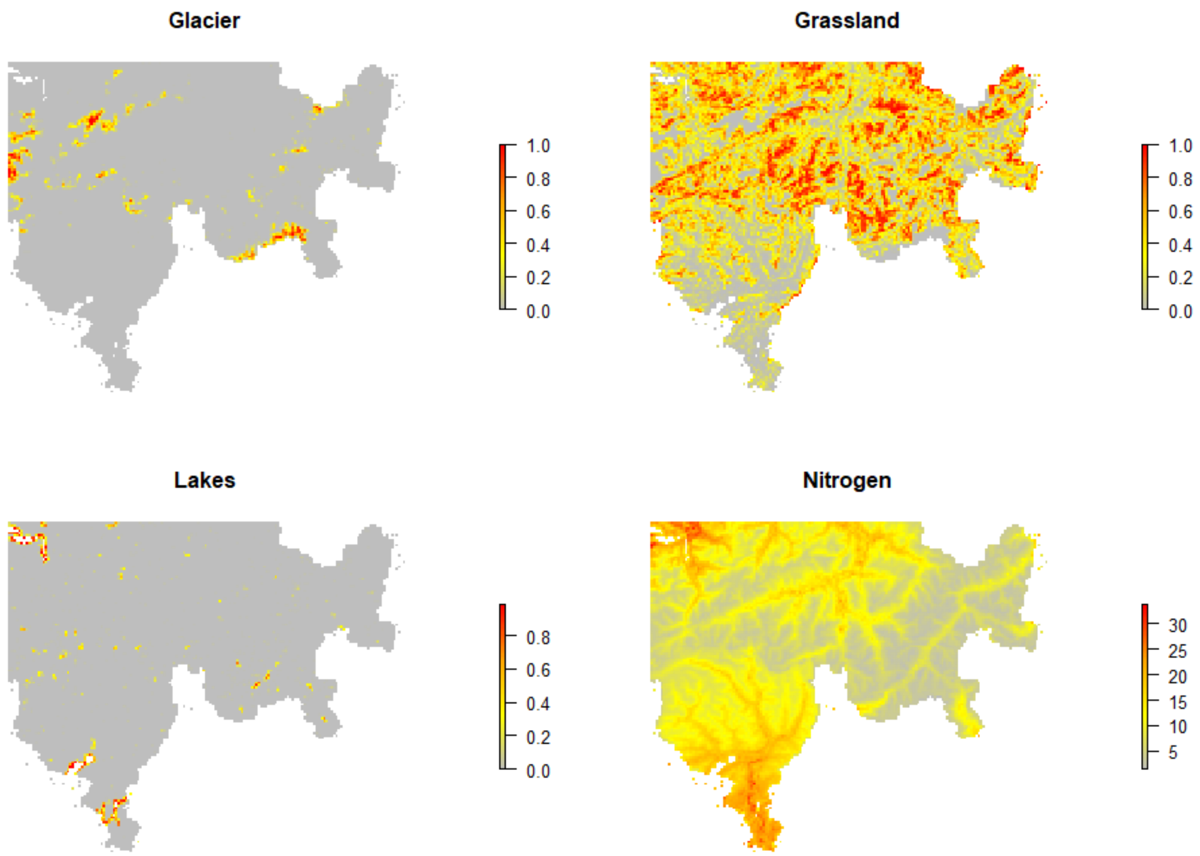
```
> diff(range(CovarData$x))
[1] 163
> diff(range(CovarData$y))
[1] 136
```

We now start by making maps of these covariates, so that we get a little bit of an understanding for the nature in the modelled domain.

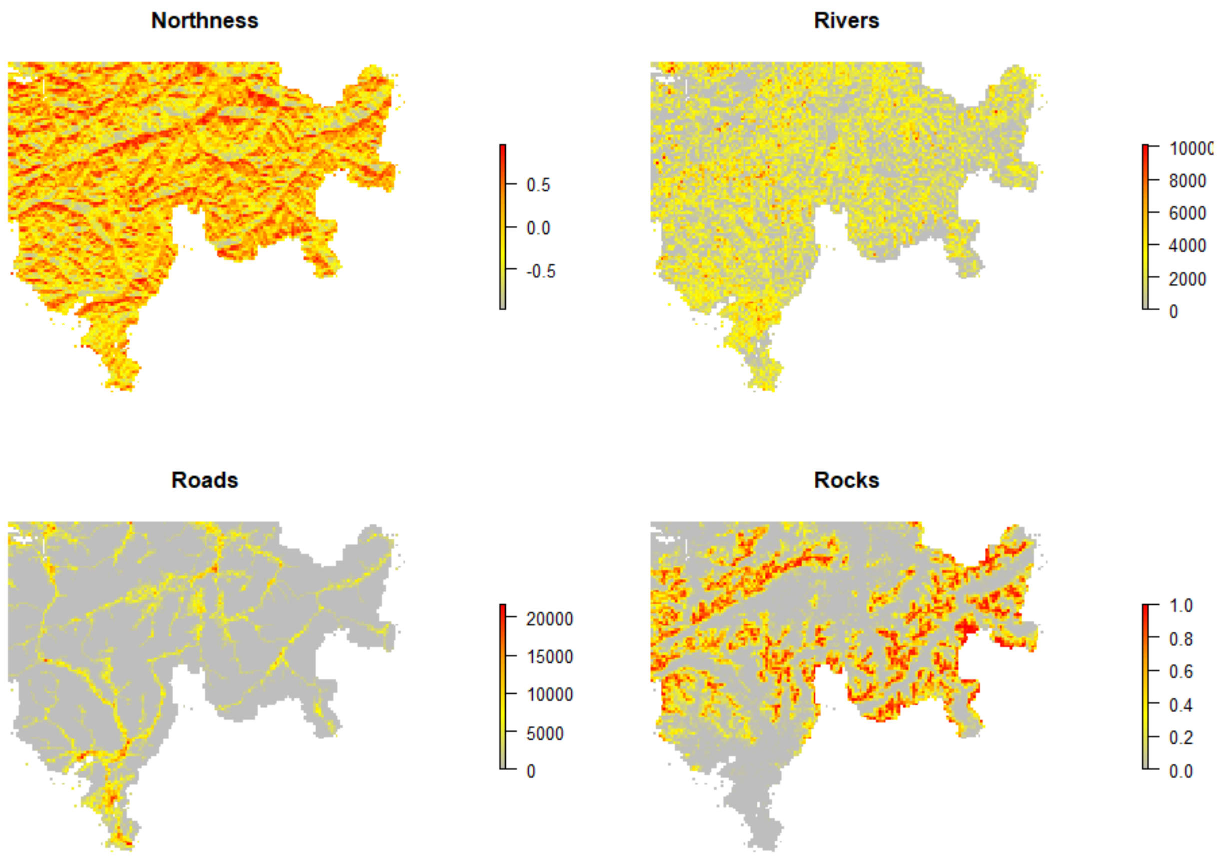
```
# Load needed packages
require(raster)      # Later have to use something else, e.g., sf
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))

# Buildings, elevation, farmland and forest
range(CovarData[, "buildings"])
range(CovarData[, "elevation"])
range(CovarData[, "farmland"])
range(CovarData[, "forest"])

par(mfrow = c(2, 2), mar = c(2,2,4,6), cex.main = 1.2)
r1 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "buildings"]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Buildings")
r2 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "elevation"]))
plot(r2, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Elevation")
r3 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "farmland"]))
plot(r3, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Farmland")
r4 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "forest"]))
plot(r4, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Forest")
```



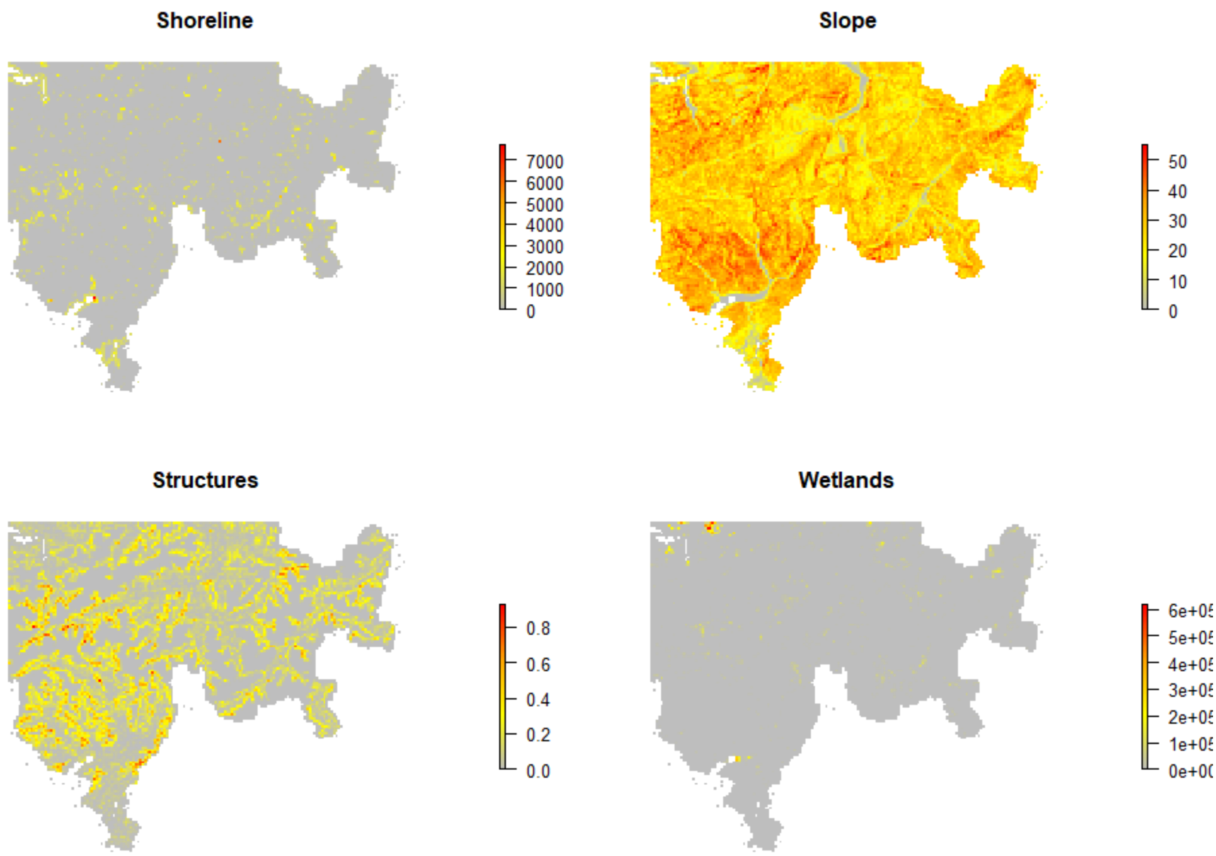

```
# Glacier, grassland, lakes and nitrogen
range(CovarData[, "glacier"])
range(CovarData[, "grassland"])
range(CovarData[, "lakes"])
range(CovarData[, "nitrogen"])

par(mfrow = c(2, 2), mar = c(2,2,4,6), cex.main = 1.2)
r1 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "glacier"]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Glacier")
r2 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "grassland"]))
plot(r2, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Grassland")
r3 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "lakes"]))
plot(r3, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Lakes")
r4 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "nitrogen"]))
plot(r4, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Nitrogen")
```



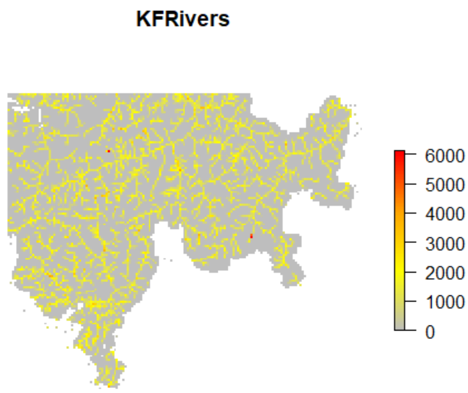

```
# Northness, Rivers, Roads and Rocks
range(CovarData[, "northness"])
range(CovarData[, "rivers"])
range(CovarData[, "roads"])
range(CovarData[, "rocks"])

par(mfrow = c(2, 2), mar = c(2,2,4,6), cex.main = 1.2)
r1 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "northness"]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Northness")
r2 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "rivers"]))
plot(r2, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Rivers")
r3 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "roads"]))
plot(r3, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Roads")
r4 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "rocks"]))
plot(r4, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Rocks")
```




```
# Shoreline, Slope, Structures and wetlands
range(CovarData[, "shoreline"])
range(CovarData[, "slope"])
range(CovarData[, "structures"])
range(CovarData[, "wetlands"])

r1 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "shoreline"]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Shoreline")
r2 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "slope"]))
plot(r2, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Slope")
r3 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "structures"]))
plot(r3, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Structures")
r4 <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "wetlands"]))
plot(r4, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Wetlands")
```




```
# kfrivers (no idea what that is...)
range(CovarData[, "kfrivers"])

rX <- rasterFromXYZ(data.frame(x = CovarData[,1], y = CovarData[,2],
  z = CovarData[, "kfrivers"]))
plot(rX, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"KFRivers")
```



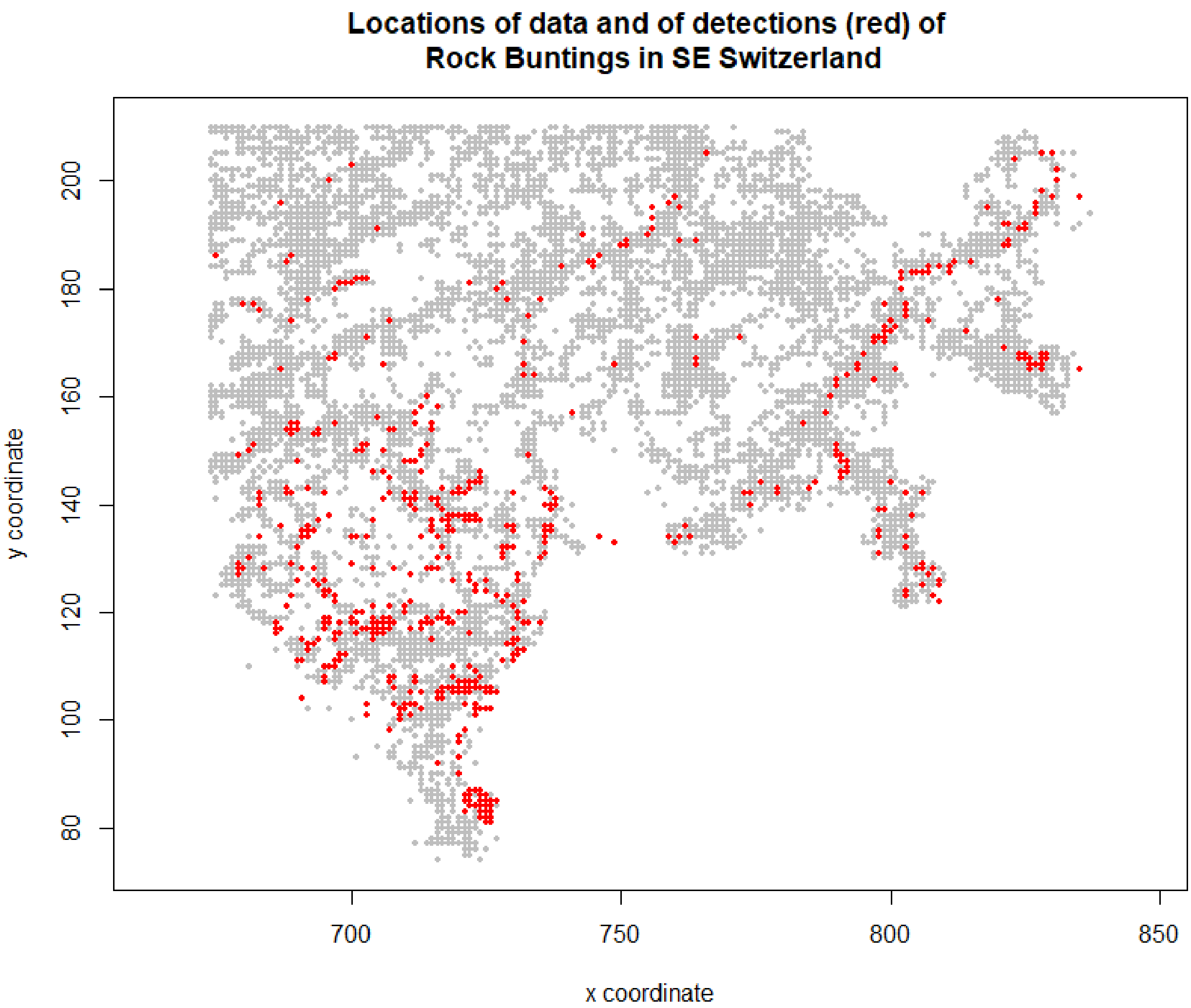
Next, we explore the survey data, which are available from 6,338 1km2 quadrats.

```
# Compute the observed presence/absence state (zobs)
zobs <- apply(AtlasData$y, 1, max, na.rm = TRUE)
table(zobs)
```

```
zobs
   0    1
5795  543
```

Thus, Rock Buntings were detected at least once in 543 1 km^2 quadrats, while they were never detected in 5795 surveyed quadrats. Thus, the observed mean occupancy probability (ignoring imperfect detection) is 543 / 6338 = 0.086. We make a map of these data.

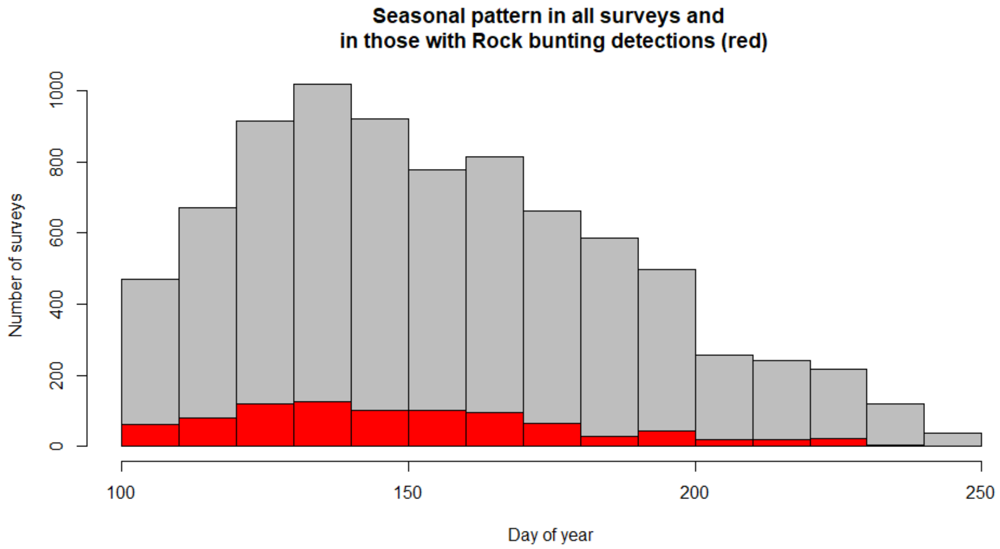
```
# Observed SDM in SE Switzerland
plot(AtlasData$coords[,1], AtlasData$coords[,2], pch = 16, asp = 1, main
= paste('Locations of data and of detections (red) of \n Rock Buntings in
SE Switzerland'), cex = 0.5, col = 'grey', xlab = 'x coordinate', ylab =
'y coordinate')
points(AtlasData$coords[,1][zobs == 1], AtlasData$coords[,2] [zobs == 1],
pch = 16, cex=0.6, col = 'red')
```



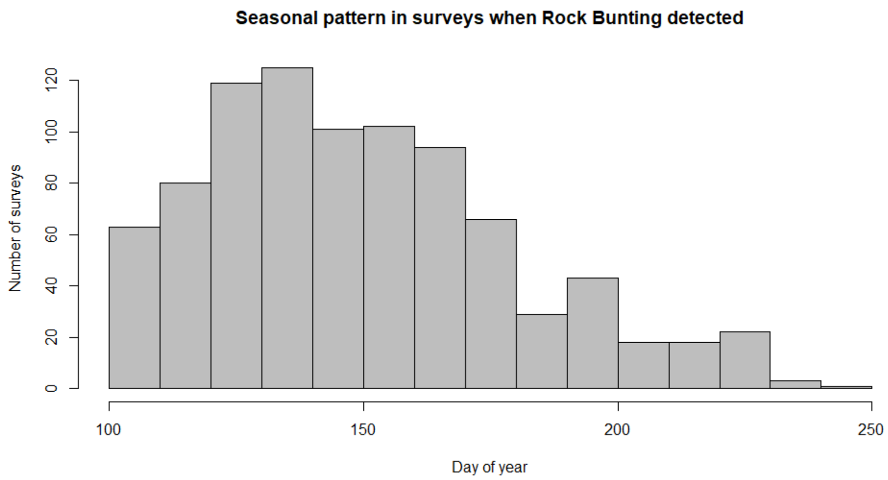
(Remember x and y units are kilometres.) Also note that our survey data are quite dense, in the sense that nearly 50% of the domain over which we want to produce a species distribution map has at least some surveys during 2013–2016. This proportion may be much lower in many other applications of species distribution models (SDMs).

####We end in drawing a plot for the seasonal profile of the data. DOES NOT WORK WITH BALANCED DETECTION DATES

```
# Seasonal distribution of survey days for all data
# and for the data with Rock bunting detections
#hist(AtlasData$det.covs$date, xlab = "Day of year", ylab = 'Number of
surveys', col = 'grey', main = 'Seasonal pattern in all surveys and \n in
those with Rock bunting detections (red)')
#hist(AtlasData$det.covs$date[AtlasData$y == 1], col = 'red', add = TRUE)
```




```
# Same only for surveys with Rock Bunting detections
#hist(AtlasData$det.covs$date[AtlasData$y == 1], xlab = "Day of year",
#ylab = 'Number of surveys', col = 'grey', main = 'Seasonal pattern in
#surveys when Rock Bunting detected')
```



In the rest of the tutorial, we are now going to use `spOccupancy`, `ubms` and `unmarked` to fit some nonspatial and some spatial occupancy species distribution models to these data and to produce species distribution maps for the Rock bunting in SE Switzerland.

3 Fitting nonspatial models in unmarked, ubms and spOccupancy

We start by fitting a nonspatial model with all covariates or at least some. In this section we will encounter the different formats of data entry etc. for the three packages.

```
library(unmarked)
library(ubms)
library(spOccupancy)
```

You might want to read up on some of the functions used here and later as well, by writing for instance `?occu` (in `unmarked`).

We wonder what's the proportion of missing values in the weekly detection history matrix now.

```
# Calculate proportion missing responses
mean(is.na(AtlasData$y))
```

```
[1] 0.8473681
```

So, even though we compressed the data along the time axis by aggregating the original daily raw data to the weekly occasions, we are still left with a detection history matrix with 85% of the cells having missing values.

We now model these data and fit the same model with `unmarked`, `ubms` and `spOccupancy`. We won't use all 17 x 2 covariate terms, but restrict ourselves to the linear and quadratic terms of those nine covariates which seemed to have some effect in an earlier, exploratory analysis (and yes, this is a little data dredging…).

We start with `unmarked`.

```
# Prepare environmental variates at site-level
covs <- as.data.frame(AtlasData$occ.covs)   # Grab all covariates

# Prepare detection variates
detcovs <- list(date1 = AtlasData$det.covs$date1, date2 =
AtlasData$det.covs$date2)

# Package all the data in an unmarked data frame for occu()
umf <- unmarkedFrameOccu(
  y = AtlasData$y,     # Pres/Abs measurements
  siteCovs = covs,     # Environmental covariates at site-level
  obsCovs = detcovs)   # Observation-specific covariates
summary(umf)
```

```
unmarkedFrame Object

6338 sites
Maximum number of observations per site: 21
Mean number of observations per site: 3.21
Sites with at least one detection: 543

Tabulation of y observations:
     0      1    <NA>
 19225   1090 112783

Site-level covariates:
```

```
  z.buildings         z.buildings2        z.elevation         z.elevation2
 Min.   :-0.4334   Min.   :-2.00417   Min.   :-1.3840   Min.   :-1.1696
 1st Qu.:-0.4332   1st Qu.: 0.10368   1st Qu.:-0.1616   1st Qu.:-1.0719
 Median :-0.4096   Median : 0.33389   Median : 0.5092   Median :-0.7675
 Mean   :-0.1660   Mean   : 0.09463   Mean   : 0.4017   Mean   :-0.4848
 3rd Qu.:-0.2742   3rd Qu.: 0.37580   3rd Qu.: 1.0175   3rd Qu.:-0.1106
 Max.   : 9.2546   Max.   :14.99065   Max.   : 2.6020   Max.   : 4.2930

    [TRUNCATED]

   z.kfrivers        z.kfrivers2
 Min.   :-0.6188   Min.   :-1.24937
 1st Qu.:-0.6188   1st Qu.:-0.91376
 Median :-0.5799   Median : 0.32959
 Mean   : 0.1954   Mean   :-0.08687
 3rd Qu.: 1.0137   3rd Qu.: 0.32959
 Max.   : 8.7331   Max.   :42.44775

Observation-level covariates:
     date1                date2
 Min.   :-1.06667   Min.   : 0.001111
 1st Qu.: 0.03333   1st Qu.: 0.444444
 Median : 1.20000   Median : 1.440000
 Mean   : 1.20238   Mean   : 3.423717
 3rd Qu.: 2.36667   3rd Qu.: 5.601111
 Max.   : 3.51667   Max.   :12.366944
```

```
# Fitting a large model with effects of 9 covariates (ART = 40 sec)
system.time(
  summary(
  fm1 <- occu(
   ~ date1 + date2
   ~ z.buildings + z.buildings2 + z.elevation + z.elevation2 +
     z.northness + z.northness2 + z.rivers + z.rivers2 +
     z.rocks + z.rocks2 + z.slope + z.slope2 +
     z.structures + z.structures2 + z.wetlands + z.wetlands2 +
     z.kfrivers + z.kfrivers2,
     se = TRUE,   # Could set to FALSE for exploration
     data=umf, control=list(trace=T, REPORT=5, maxit = 500))) )
```

```
Call:
occu(formula = ~date1 + date2 ~ z.buildings + z.buildings2 +
    z.elevation + z.elevation2 + z.northness + z.northness2 +
    z.rivers + z.rivers2 + z.rocks + z.rocks2 + z.slope + z.slope2 +
    z.structures + z.structures2 + z.wetlands + z.wetlands2 +
    z.kfrivers + z.kfrivers2, data = umf, se = TRUE,
    control = list(trace = T, REPORT = 5, maxit = 500))

Occupancy (logit-scale):
              Estimate     SE      z  P(>|z|)
(Intercept)     -3.546 0.3844 -9.224 2.87e-20
z.buildings     -0.388 0.1536 -2.526 1.16e-02
z.buildings2     0.296 0.1348  2.194 2.83e-02
z.elevation     -1.127 0.1527 -7.383 1.54e-13
```

```
z.elevation2    -0.196 0.1462 -1.339 1.81e-01
z.northness     -0.722 0.0741 -9.751 1.82e-22
z.northness2    -0.375 0.0636 -5.890 3.87e-09
z.rivers         0.129 0.0749  1.729 8.39e-02
z.rivers2       -0.124 0.0661 -1.879 6.03e-02
z.rocks         -0.990 0.4282 -2.313 2.07e-02
z.rocks2         0.240 0.2088  1.150 2.50e-01
z.slope          1.223 0.1576  7.761 8.44e-15
z.slope2        -0.152 0.1008 -1.505 1.32e-01
z.structures     0.272 0.0506  5.366 8.05e-08
z.structures2   -0.097 0.0401 -2.418 1.56e-02
z.wetlands      -4.835 4.8693 -0.993 3.21e-01
z.wetlands2     -3.000 3.2564 -0.921 3.57e-01
z.kfrivers      -0.144 0.0625 -2.305 2.12e-02
z.kfrivers2      0.172 0.0409  4.205 2.61e-05

Detection (logit-scale):
            Estimate     SE       z  P(>|z|)
(Intercept)  -0.6124 0.0532 -11.513 1.14e-30
date1        -0.2096 0.0661  -3.172 1.52e-03
date2        -0.0278 0.0302  -0.919 3.58e-01

AIC: 6186.299
Number of sites: 6338
optim convergence code: 0
optim iterations: 225
Bootstrap iterations: 0

   user  system elapsed
  71.19   11.39   39.91
```

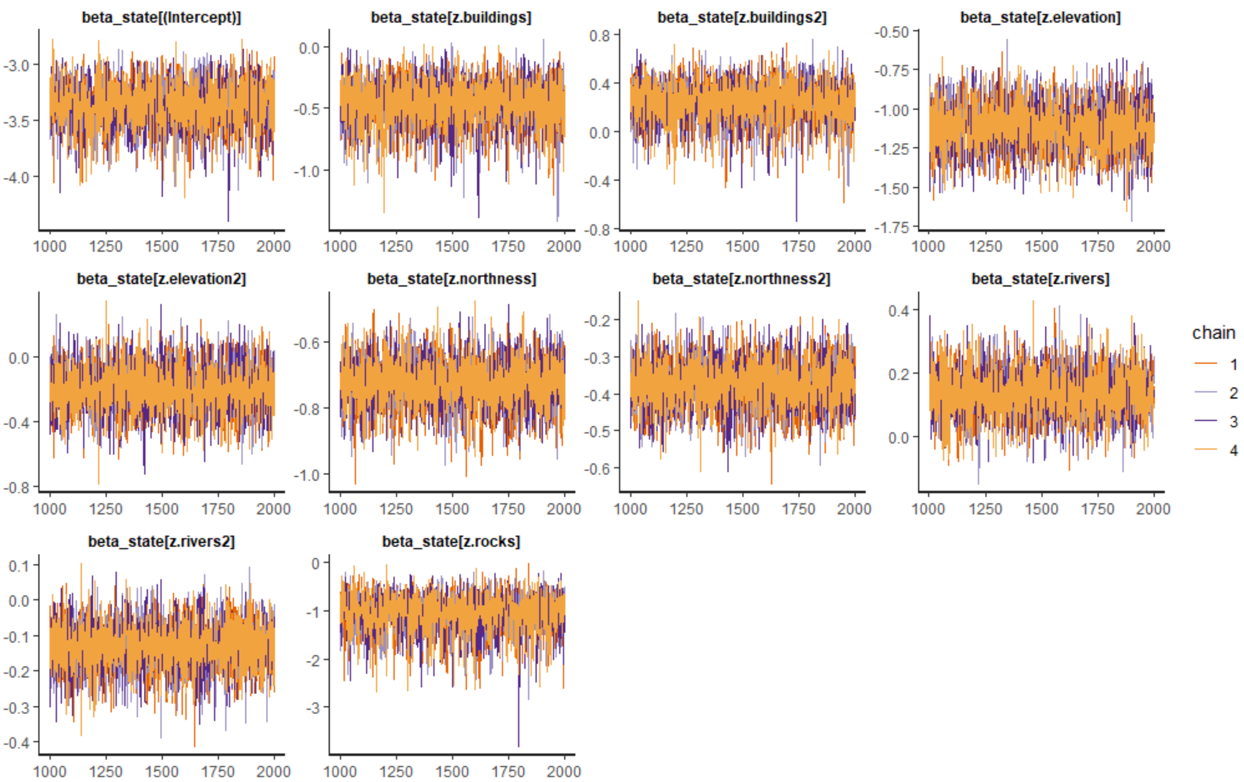
Things work.

We continue with `ubms`, which is a wrapper for `Stan`. It takes a hellufalot of time to produce the results…

```
# Fitting the model in Stan using ubms (ART = 51 min)
system.time(
  fm2 <- stan_occu(
   ~ date1 + date2
   ~ z.buildings + z.buildings2 + z.elevation + z.elevation2 +
     z.northness + z.northness2 + z.rivers + z.rivers2 +
     z.rocks + z.rocks2 + z.slope + z.slope2 +
     z.structures + z.structures2 + z.wetlands + z.wetlands2 +
     z.kfrivers + z.kfrivers2,
     data=umf) )
ubms::traceplot(fm2)        # Check convergence of chains
print(fm2)                  # Print posterior summaries
```

```
Call:
stan_occu(formula = ~date1 + date2 ~ z.buildings + z.buildings2 +
    z.elevation + z.elevation2 + z.northness + z.northness2 +
    z.rivers + z.rivers2 + z.rocks + z.rocks2 + z.slope + z.slope2 +
    z.structures + z.structures2 + z.wetlands + z.wetlands2 +
    z.kfrivers + z.kfrivers2, data = umf)

Occupancy (logit-scale):
```

```
                Estimate     SD   2.5%    97.5% n_eff  Rhat
(Intercept)      -3.3788 0.2106 -3.830 -2.99791  1638 1.000
z.buildings      -0.4636 0.1916 -0.898 -0.14528  2869 0.999
z.buildings2      0.2358 0.1730 -0.156  0.53168  2756 0.999
z.elevation      -1.1166 0.1539 -1.417 -0.82244  3207 1.001
z.elevation2     -0.1965 0.1445 -0.484  0.09149  3780 1.001
z.northness      -0.7245 0.0747 -0.882 -0.58297  3804 1.000
z.northness2     -0.3755 0.0653 -0.504 -0.25067  4304 1.000
z.rivers          0.1332 0.0757 -0.017  0.28160  3923 0.999
z.rivers2        -0.1312 0.0679 -0.268 -0.00369  5175 1.000
z.rocks          -1.1008 0.4178 -2.049 -0.44212  1707 1.001
z.rocks2          0.1980 0.2033 -0.254  0.53848  1889 1.000
z.slope           1.2160 0.1566  0.916  1.53022  3176 1.000
z.slope2         -0.1469 0.0987 -0.337  0.05239  3346 1.000
z.structures      0.2734 0.0507  0.172  0.37084  4475 1.001
z.structures2    -0.0977 0.0403 -0.177 -0.01818  4315 1.001
z.wetlands       -1.3869 0.8862 -3.634 -0.25275  1303 1.001
z.wetlands2      -0.6443 0.6183 -2.201  0.18083  1296 1.001
z.kfrivers       -0.1478 0.0622 -0.274 -0.03026  3650 1.000
z.kfrivers2       0.1812 0.0425  0.103  0.27023  4365 1.000

Detection (logit-scale):
            Estimate     SD    2.5%   97.5% n_eff Rhat
(Intercept)  -0.6168 0.0532 -0.7213 -0.5102  5677    1
date1        -0.2098 0.0645 -0.3352 -0.0823  3049    1
date2        -0.0285 0.0294 -0.0859  0.0281  3026    1

LOOIC: 6188.164
Runtime: 51.052 min
```



Finally, we fit the same model with spOccupancy.

```
str(AtlasData)
```

```
List of 4
 $ y       : num [1:6338, 1:21] NA NA NA NA NA NA NA NA NA NA ...
 $ occ.covs: num [1:6338, 1:34] -0.427 -0.427 -0.433 -0.433 -0.425 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:6338] "674_146" "674_147" "674_148" "674_156" ...
  .. ..$ : chr [1:34] "z.buildings" "z.buildings2" ...
 $ det.covs:List of 3
  ..$ date : num [1:6338, 1:21] 103 103 103 103 103 103 103 103 ...
  ..$ date1: num [1:6338, 1:21] -1.07 -1.07 -1.07 -1.07 -1.07 ...
  ..$ date2: num [1:6338, 1:21] 1.14 1.14 1.14 1.14 1.14 ...
 $ coords  : num [1:6338, 1:2] 674 674 674 674 674 674 674 674 ...
```

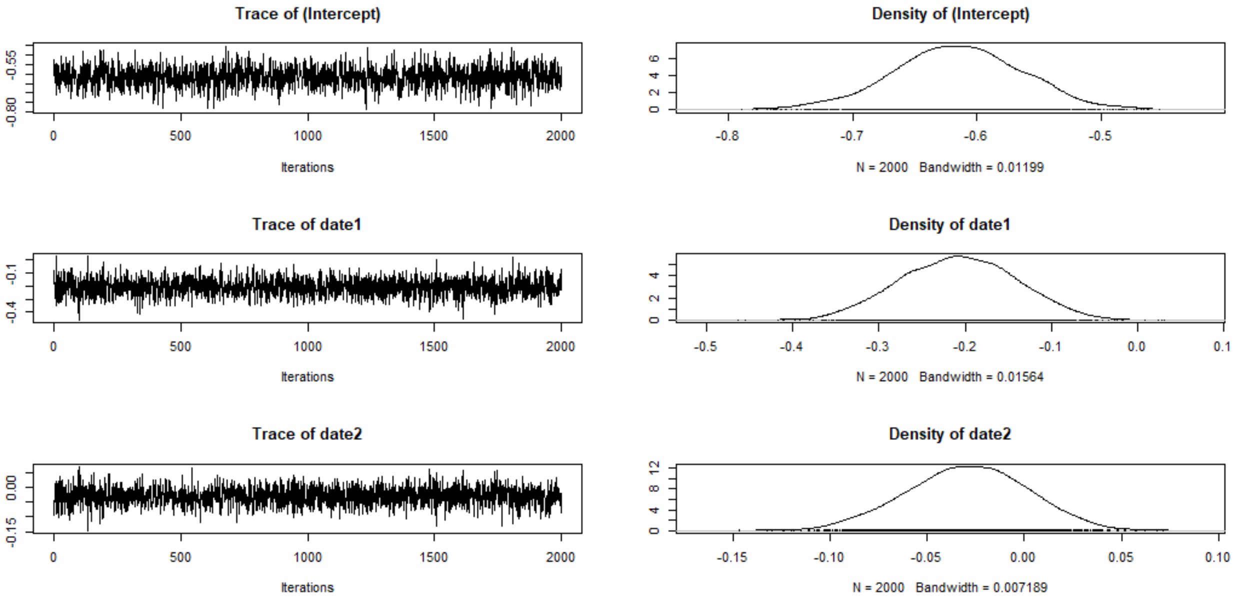
```
# Specify model formulas
# Occupancy
occ.formula <-
  ~ z.buildings + z.buildings2 + z.elevation + z.elevation2 +
    z.northness + z.northness2 + z.rivers + z.rivers2 +
    z.rocks + z.rocks2 + z.slope + z.slope2 +
    z.structures + z.structures2 + z.wetlands + z.wetlands2 +
    z.kfrivers + z.kfrivers2

# Next would be for all 17 * 2 terms …
full.occ.formula <- ~ z.buildings + z.buildings2 +
  z.elevation + z.elevation2 + z.farmland + z.farmland2 +
  z.forest + z.forest2 + z.glacier + z.glacier2 +
  z.grassland + z.grassland2 + z.lakes + z.lakes2 +
  z.nitrogen + z.nitrogen2 + z.northness + z.northness2 +
  z.rivers + z.rivers2 + z.roads + z.roads2 + z.rocks + z.rocks2 +
  z.shoreline + z.shoreline2 + z.slope + z.slope2 +
  z.structures + z.structures2 + z.wetlands + z.wetlands2 +
  z.kfrivers + z.kfrivers2

# Detection
det.formula <- ~ date1 + date2

# Fit the nonspatial occupancy model
fm3 <- PGOcc(occ.formula = occ.formula,
             det.formula = det.formula,
             data = AtlasData,
             n.samples = 11000,
             n.omp.threads = 6,
             n.burn = 1000,
             n.thin = 20,
             n.chains = 4,
             n.report = 1000, verbose = TRUE)
summary(fm3)
```

```
Call:
PGOcc(occ.formula = occ.formula, det.formula = det.formula, data =
AtlasData,
    n.samples = 11000, n.omp.threads = 6, verbose = TRUE, n.report =
1000,
```

```
    n.burn = 1000, n.thin = 20, n.chains = 4)

Samples per Chain: 11000
Burn-in: 1000
Thinning Rate: 20
Number of Chains: 4
Total Posterior Samples: 2000
Run Time (min): 10.2198

Occurrence (logit scale):
                 Mean     SD    2.5%     50%   97.5%   Rhat  ESS
(Intercept)   -3.3558 0.2027 -3.7826 -3.3439 -3.0010 1.0288  422
z.buildings   -0.4717 0.1897 -0.8920 -0.4510 -0.1685 1.0018 1751
z.buildings2   0.2312 0.1643 -0.1353  0.2481  0.5143 1.0017 1714
z.elevation   -1.1117 0.1506 -1.4047 -1.1090 -0.8020 1.0036 1980
z.elevation2  -0.1878 0.1471 -0.4773 -0.1898  0.0928 0.9997 1804
z.northness   -0.7242 0.0730 -0.8686 -0.7236 -0.5876 1.0015 1786
z.northness2  -0.3733 0.0635 -0.4939 -0.3744 -0.2496 1.0058 2000
z.rivers       0.1338 0.0748 -0.0133  0.1321  0.2839 1.0030 2000
z.rivers2     -0.1313 0.0653 -0.2557 -0.1315 -0.0084 1.0022 2000
z.rocks       -1.0722 0.3970 -2.0085 -1.0230 -0.4473 1.0388  326
z.rocks2       0.2138 0.1899 -0.2080  0.2296  0.5331 1.0178  390
z.slope        1.2092 0.1572  0.8973  1.2094  1.5220 1.0081 2000
z.slope2      -0.1422 0.1012 -0.3499 -0.1422  0.0554 1.0082 1827
z.structures   0.2750 0.0501  0.1752  0.2760  0.3705 1.0017 2000
z.structures2 -0.0984 0.0399 -0.1782 -0.0980 -0.0210 1.0005 1767
z.wetlands    -1.2927 0.8067 -3.2168 -1.1407 -0.2094 1.1139  217
z.wetlands2   -0.5790 0.5554 -1.9165 -0.4858  0.1714 1.1059  239
z.kfrivers    -0.1488 0.0630 -0.2753 -0.1479 -0.0286 1.0036 2000
z.kfrivers2    0.1792 0.0424  0.1010  0.1782  0.2644 1.0040 2000

Detection (logit scale):
               Mean     SD    2.5%     50%   97.5%   Rhat  ESS
(Intercept) -0.6159 0.0527 -0.7242 -0.6160 -0.5147 1.0005 1661
date1       -0.2077 0.0675 -0.3382 -0.2077 -0.0792 1.0063 2000
date2       -0.0295 0.0310 -0.0908 -0.0291  0.0282 1.0004 2000
```

Almost everything has converged fine according to Rhat. We check out the traceplots as well.

```
# Traceplots
plot(fm3$beta.samples)        # Occupancy params (not shown)
plot(fm3$alpha.samples)       # Detection params
```



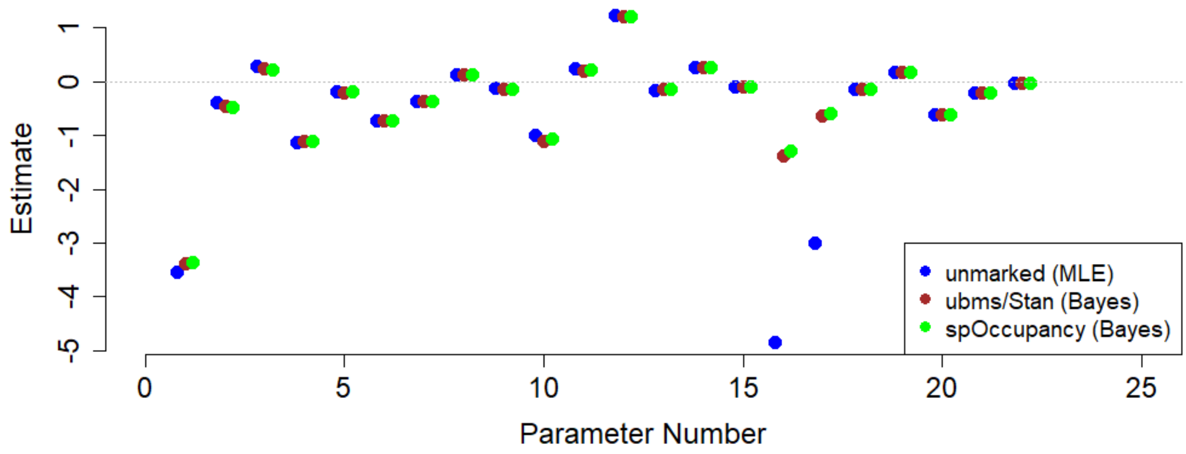
That looks good, too.

We compare the estimates from the three packages. For this, we produce the posterior means as our Bayesian point estimators for the solutions from `spOccupancy` (i.e., for `fm3`). (We could also use the median or the mode.)

```
# Get posterior means from spOccupancy output
coef_fm3 <- c(apply(fm3$beta.samples, 2, mean), apply(fm3$alpha.samples, 2, mean))
```

We compare them in a plot.

```
off <- 0.2
par(mar = c(6,6,6,3), cex.axis = 1.5, cex.lab = 1.5, cex.main = 1.5)
plot((1:22)-off, coef(fm1), pch = 16, cex = 1.5, col = 'blue', xlab = 'Parameter Number', ylab = 'Estimate', frame = FALSE, xlim = c(0, 25))
points((1:22), coef(fm2), pch = 16, cex = 1.5, col = 'brown')
points((1:22)+off, coef_fm3, pch = 16, cex = 1.5, col = 'green')
abline(h = 0, lwd = 1, lty = 3, col = 'grey')
legend('bottomright', pch = 16, cex = 1.2, col = c('blue', 'brown', 'green'), legend = c("unmarked (MLE)", "ubms/Stan (Bayes)", "spOccupancy (Bayes)"))   # bty = 'n'
```



Overall, we see very similar point estimates from all three packages, though two estimates from `unmarked` fall outside of this pattern. Here one can see the "regularizing" effect of the priors in a Bayesian analysis (plus the fact that one integrates over a posterior distribution, rather than taking the extreme of a likelihood function).

3.3 Spatial predictions from the nonspatial model: first species distribution maps

We next form predictions from the nonspatial models and produce the predicted species distribution map of the Rock Bunting in SE Switzerland. To do so means in a sense to go the opposite way of what we have been going when fitting the model, with relationship to the response variable y and the explanatory variables x. To fit the model, we took the observed response (i.e., the detection/nondetection data) and related it to the explanatory variables by fitting the models. The estimated parameters are then like "weights" that say how much (and in what direction) each explanatory variable affects (in the statistical sense of an association) the expected response. The whole takes place inside of a hierarchical model that, along the way, also corrects the observed detection/nondetection data for imperfect detection. Thus, the model relates an estimate of presence/absence to the known value of the explanatory variables.

When we fit the model:

y -> x # We find a rule for how x affects y: x -> y

When we make predictions from the model:

x -> y # We applly that rule for another set of covariates x

To make predictions, we take the 'rule', i.e., the fitted regression equation, which we obtained from the 6,338 quadrats where we did have some information about presence/absence of the Rock Bunting, and we apply it now to the entire set of 12,757 quadrats in the domain for which we want to produce a species distribution map (SDM).

Making predictions from a regression model can be notoriously confusing for ecologists, since we may have challenges due to having scaled the covariates and in addition also owing to having backtransform from the link scale (at which the linear effect of the covariates are estimated) to the probability scale, which is what we prefer the results to be presented. Importantly, the parameter estimates ONLY make sense in the context of the specific scale at which the covariates are expressed. Thus, for instance, if we have normalized an elevation covariate used to fit the model by subtracting a mean of, say, 1200 metres and dividing the result by a standard deviation of 800 metres, then we must also scale the elevation covariate in the data set used for prediction in exactly the same way. A common mistake is to standardize the prediction covariates instead by subtracting *their mean value* and dividing the result by *their standard deviation*. The result of this is an error which is similar to comparing the fuel efficiency of two cars in terms of either kilometres or of miles.

In our analysis, this first challenge is avoided, since the covariates in both the model fitting data set (i.e., "AtlasData") and the prediction data set ("CovarData") already come standardized in the identical manner. However, we still have to deal with the backtransformation from the link scale to the "natural" scale, which is the probability scale. This is done automatically when using the `predict()` function …… (or is it ?)

We start at making predictions from the fitted model objects in `unmarked`, `ubms` and the `spOccupancy`. For this, we always have to prepare a prediction (or "newdata") data set. Interestingly, for `unmarked` we can just supply all covariates, i.e., more than the ones actually used in the prediction for model `fm1`.

```
# Make a data frame containing the covariates for the prediction data set
cbind(names(CovarData))      # not shown
newdata <- data.frame(CovarData[20:53])

pred.um <- unmarked::predict(fm1, newdata = newdata, type = 'state')
options(scipen = 10)
```

```
head(pred.um)
```

```
   Predicted          SE      lower      upper
1 0.05104257 0.01319608 0.03057056 0.08403556
2 0.19145318 0.03115914 0.13763037 0.25997890
3 0.53163882 0.03909729 0.45486699 0.60694150
4 0.47405532 0.04738037 0.38311909 0.56674576
5 0.24000071 0.04734931 0.15956575 0.34436805
6 0.21807254 0.02065483 0.18029561 0.26124215
```

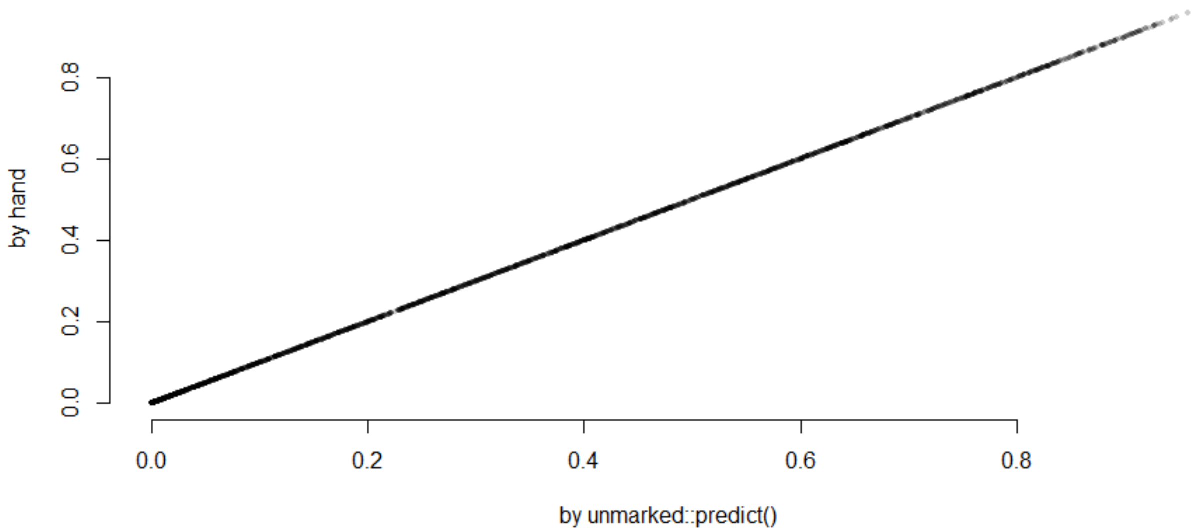
This data frame contains the point predictions (on the probability scale), the prediction standard error and the bounds of a 95% prediction interval, which can all be useful for producing a species distribution map and a map of the associated uncertainty.

As a side comment, we note that the `predict()` function simply takes the values of the covariates in the newdata data frame and plugs them into the linear model equation for (here) the occupancy part of the model, for each site, multiplying the covariate value with the value of the estimated coefficient, adding up and inverse-logit transforming the result, to get at the point prediction. For illustration, we quickly do the same by hand. We do make a big simplification by using matrix multiplication though … and we have to add a first column of ones for the intercept…

Remember the model for occupancy.

```
occ.formula <-
  ~ z.buildings + z.buildings2 + z.elevation + z.elevation2 +
    z.northness + z.northness2 + z.rivers + z.rivers2 +
    z.rocks + z.rocks2 + z.slope + z.slope2 +
    z.structures + z.structures2 + z.wetlands + z.wetlands2 +
    z.kfrivers + z.kfrivers2
```

```
# Same by hand for the point prediction
selected.covs <- c(1:4, 17:20, 23:24, 27:34)
link.scale.pred <- as.matrix(cbind(1, newdata[,selected.covs])) %*%
unmarked::coef(fm1)[1:19]
pred.by.hand <- plogis(link.scale.pred)    # Apply inverse link

# Test that we get the same as does unmarked with predict()
plot(pred.um[,1], pred.by.hand, xlab = 'by unmarked::predict()', ylab =
'by hand', pch = 16, cex = 0.6, col = rgb(0,0,0,0.2), frame = FALSE)
```



We continue now making predictions with the fitted model object from `ubms` and then from `spOccupancy`. As always, the syntax for `ubms` is identical to or very similar to that in `unmarked`; the only difference here is that the argument 'type' when predicting for an `unmarked` fitted model object is called 'submodel' when predicting with `ubms`.

```
pred.ubms <- ubms::predict(fm2, newdata = newdata, submodel = 'state')
str(pred.ubms)
head(pred.ubms)
```

```
> str(pred.ubms)
'data.frame':   12757 obs. of  4 variables:
 $ Predicted: num  0.0537 0.1952 0.5327 0.4789 0.2475 ...
 $ SD       : num  0.0138 0.0319 0.0394 0.0476 0.0491 ...
 $ 2.5%     : num  0.0312 0.138 0.4557 0.3898 0.1596 ...
 $ 97.5%    : num  0.0834 0.2632 0.6103 0.5728 0.353 ...

> head(pred.ubms)
   Predicted         SD       2.5%      97.5%
1 0.05370081 0.01378717 0.03118465 0.08339214
2 0.19517303 0.03192787 0.13802239 0.26322805
3 0.53265121 0.03939738 0.45573148 0.61031999
4 0.47885071 0.04756152 0.38976314 0.57276848
5 0.24745456 0.04914520 0.15957733 0.35302615
6 0.21940485 0.02127169 0.17966164 0.26295935
```

Finally, we make predictions for the `spOccupancy` fitted model object. Note that we need to add on one column with all ones for the intercept. We can use the same matrix as what we used above when we made predictions 'by hand'.

```
# Form predictions from the non-spatial model (takes 2 sec)
# (using the spOccupancy fitted model object)
spo.cov <- as.matrix(cbind(1, newdata[,selected.covs]))
system.time(
  pred.spo <- predict(fm3, spo.cov)
)
```

```
str(pred.spo)
```

```
List of 3
 $ psi.0.samples: 'mcmc' num [1:2000, 1:12757] 0.0367 0.063 0.0542 0.0465
0.0308 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:12757] "674_210" "675_210" "676_210" "677_210" ...
  ..- attr(*, "mcpar")= num [1:3] 1 2000 1
 $ z.0.samples  : 'mcmc' int [1:2000, 1:12757] 0 0 0 0 0 0 0 0 0 0 ...
  ..- attr(*, "mcpar")= num [1:3] 1 2000 1
 $ call         : language predict.PGOcc(object = fm3, X.0 = spo.cov)
 - attr(*, "class")= chr "predict.PGOcc"
```

Actually, what we get from the `predict` function is (here) 2000 draws from the posterior predictive distribution of both the expected (psi) and the realized presence/absence (z) for every quadrat in the prediction domain. To plot, we need to summarize them ourselves, for instance, by computing the posterior means, standard deviations and 95% credible intervals. We do the former two.

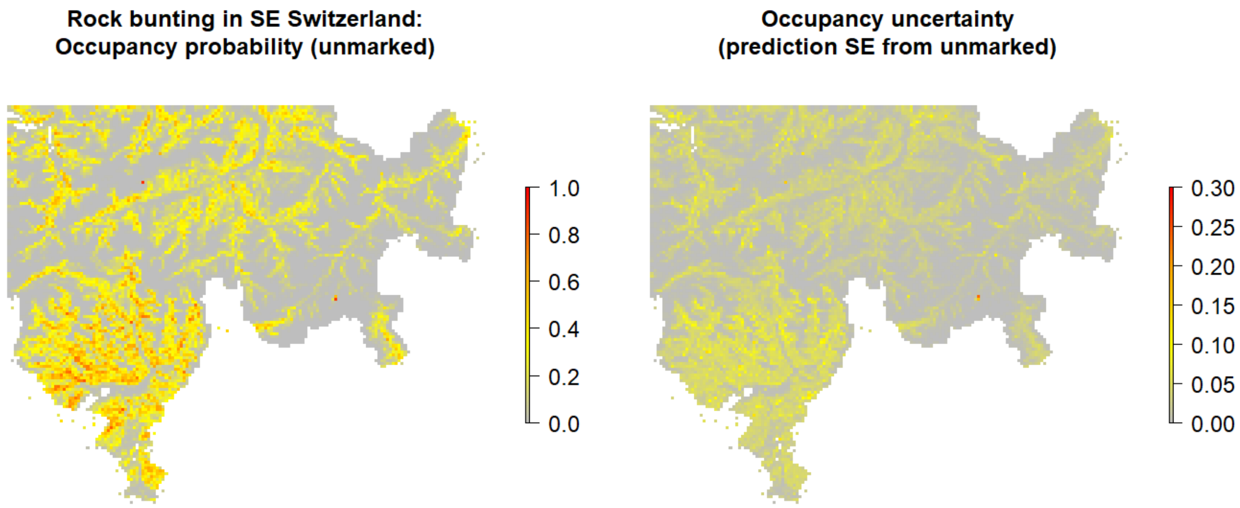
```
# Get posterior means and SDs for the model fit from spOccupancy
pm.psi <- apply(pred.spo$psi.0.samples, 2, mean)
psd.psi <- apply(pred.spo$psi.0.samples, 2, sd)

summary(pm.psi)
summary(psd.psi)
```

We can now plot these predictions in geographic space to obtain our species distribution maps.

```
# Load needed packages
require(raster)
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))
```

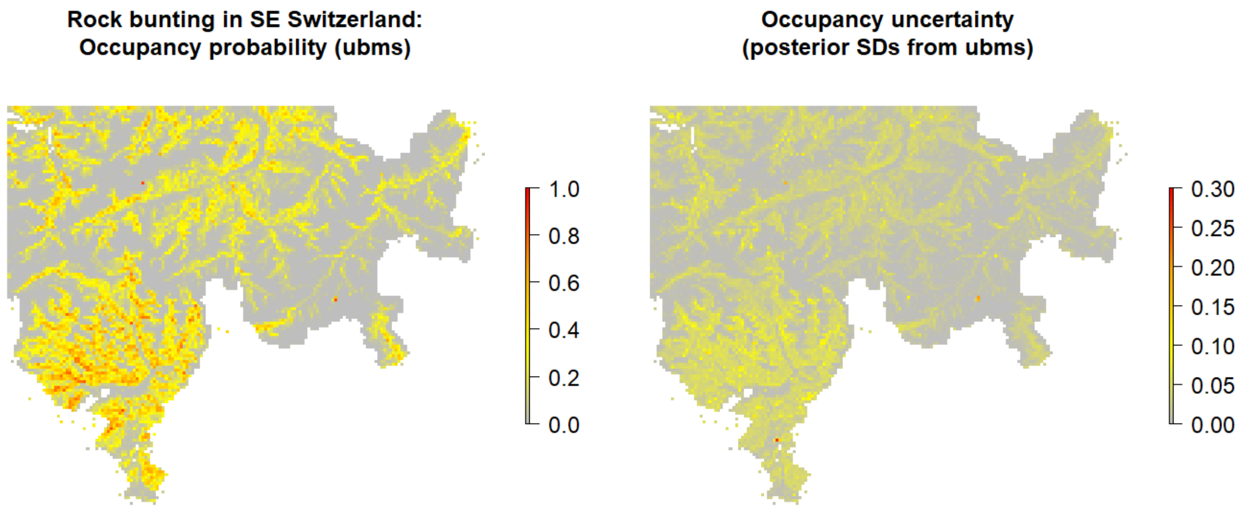
Bunting species distribution map in SE Switzerland from `unmarked`:

```
par(mfrow = c(1, 2), mar = c(1,3,4,6), cex.main = 1.5)
# Point predictions (based on MLEs of the parameters)
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.um[,1]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Rock
bunting in SE Switzerland:\nOccupancy probability (unmarked)", zlim =
c(0, 1))

# Prediction standard errors
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.um[,2]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Occupancy uncertainty\n(prediction SE from unmarked)", zlim = c(0, 0.3))
```



And the Rock Bunting species distribution map from `ubms`:

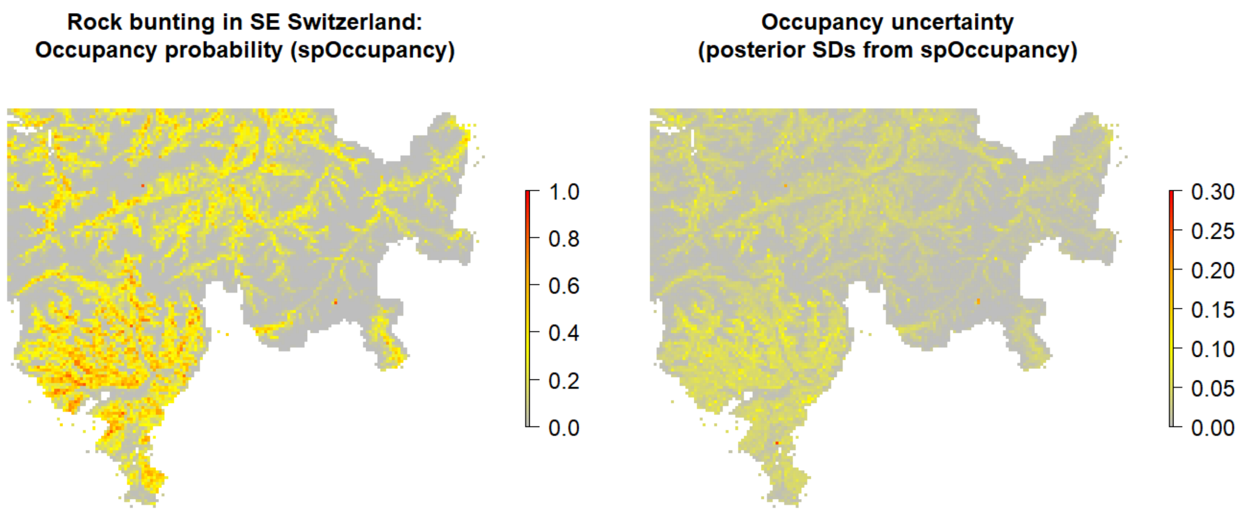
```
par(mfrow = c(1, 2), mar = c(1,3,4,6), cex.main = 1.5)
# Point predictions (posterior means)
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.ubms[,1]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Rock
bunting in SE Switzerland:\nOccupancy probability (ubms)", zlim = c(0,
1))

# Prediction uncertainty (posterior standard deviations)
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.ubms[,2]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Occupancy uncertainty\n(posterior SDs from ubms)", zlim = c(0, 0.3))
```



And, finally, the Rock Bunting species distribution map from `spOccupancy`:

```
par(mfrow = c(1, 2), mar = c(1,3,4,6), cex.main = 1.5)
# Point predictions (posterior means)
```

```
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pm.psi))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Rock
bunting in SE Switzerland:\nOccupancy probability (spOccupancy)", zlim =
c(0, 1))

# Prediction uncertainty (posterior standard deviations)
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = psd.psi))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Occupancy uncertainty\n(posterior SDs from spOccupancy)", zlim = c(0,
0.3))
```

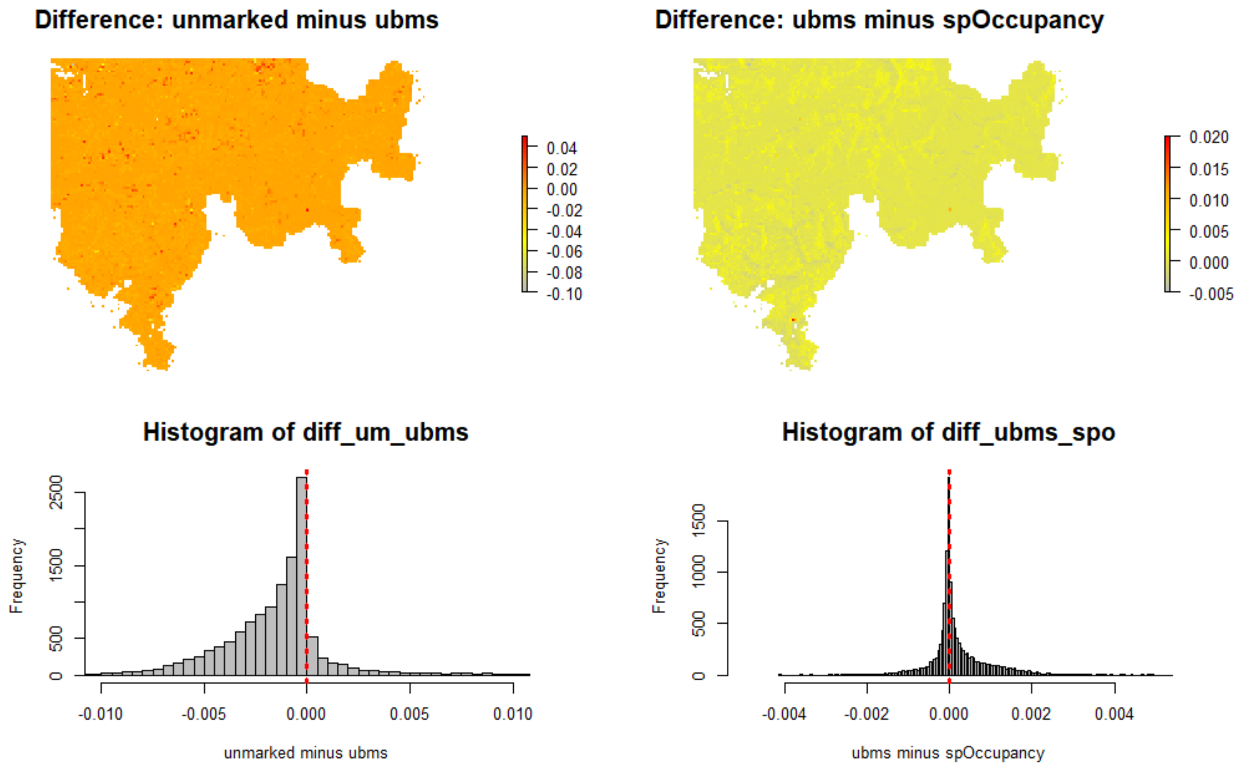


They should all look almost the same. To test this, we subtract the prediction from one of the other and plot these differences quickly.

Compute differences and plot them

```
diff_um_ubms <- pred.um[,1] - pred.ubms[,1]
diff_ubms_spo <- pred.ubms[,1] - pm.psi
range(diff_um_ubms)
range(diff_ubms_spo)

par(mfrow = c(2, 2), mar = c(1,3,4,8), cex.main = 1.5)
# Difference in point predictions between unmarked and ubms
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = diff_um_ubms))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Difference: unmarked minus ubms", zlim = c(-0.1, 0.05))

# Difference in point predictions between ubms and spOccupancy
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = diff_ubms_spo))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Difference: ubms minus spOccupancy", zlim = c(-0.005, 0.02))

par(mar = c(6,6,4,4))
```

```
hist(diff_um_ubms, xlab = "unmarked minus ubms", col = 'grey', breaks =
500, xlim = c(-0.01, 0.01))
abline(v = 0, col = 'red', lwd = 3, lty = 3)

hist(diff_ubms_spo, xlab = "ubms minus spOccupancy", col = 'grey', breaks
= 500, xlim = c(-0.005, 0.005))
abline(v = 0, col = 'red', lwd = 3, lty = 3)
```

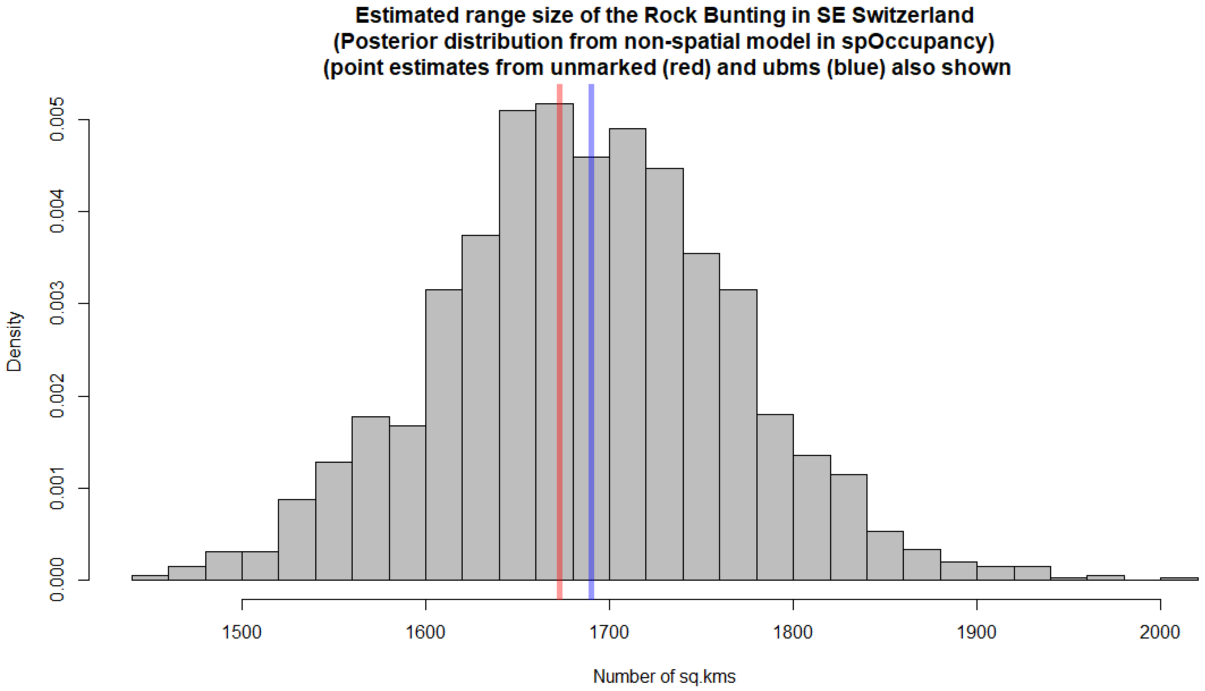


Basically, hardly any big differences, but slightly more between MLE (unmarked) and Bayes (ubms; left) and much less between the two Bayesian engines (note the narrower range in the map on the right !).

One common summary of a species distribution is the range size. For an occupancy model, we can obtain this by simply adding up the predictions over the desired area in which we want to summarize the distribution. For the Bayesian analyses, we can simply compute the sum(psi) as a derived quantity and then summarize its posterior distribution.

```
# Compute range size from unmarked (would have to bootstrap for SE)
rangeSize.um <- sum(pred.um[,1])

# Compute range size from ubms FOR NOW
rangeSize.ubms <- sum(pred.ubms[,1])

# Compute posterior distribution of range size from spOccupancy
rangeSize.spo <- apply(pred.spo[[1]], 1, sum)
hist(rangeSize.spo, breaks = 30, col = 'grey', main = 'Estimated range
size of the Rock Bunting in SE Switzerland\n(Posterior distribution from
non-spatial model in spOccupancy)\n (point estimates from unmarked (red)
and ubms (blue) also shown', xlab = 'Number of sq.kms', freq = FALSE)
abline(v = rangeSize.um, lwd = 5, lty = 1, col = rgb(1,0,0,0.4))
```

```
abline(v = rangeSize.ubms, lwd = 5, lty = 1, col = rgb(0,0,1,0.4))
```



So, this is surely a quite super-cool analysis that lets us learn a lot about the Rock Bunting in SE Switzerland during the years 2013–2016, both in terms of its occurrence (which is where we have put the emphasis in this section) and also in terms of likely factors that govern it (i.e., the covariates determining occurrence, on which we have however not put much emphasis here).

We have just seen how by plotting predictions into a 'real map' of the modelled domain, with known values for all of the covariates that we have used in the model, we can obtain a nice species distribution map.

However, this is NOT a spatial model, since space does NOT appear anywhere in the whole section. For instance, we could permute the x and y coordinates in both data sets and we would still get exactly the same parameter estimates and exactly the same predictions. In the next section, we will now progress to spatial versions of these models, where in addition to the effects of covariates, we will also use what we like to call "neighbourhood relations" to refine predictions. That is, in addition to the effects of the covariates, these models will implicitly estimate site-specific residuals, which measure by how much locally, the realized expected occurrence is more or less than what we would expect based on the estimated effects of the covariates alone. In the model, these site-level random effects are given a zero-mean multivariate normal distribution, where the variance-covariance matrix is expressed as a function (usually with 1-3 parameters only) of the pairwise distances between all sites. This has the effect of "smearing out" spatially the site random effects, such that we get "islands" of over- and islands of underpredictions (relative to what the covariates alone predict). The map of these spatially correlated residuals is called a random (spatial) field and will be interesting to look at. It may help to form hypotheses about which additional covariates might be needed to explain the species distribution. OK, here we go !

4 Fitting spatial models to the SE Swiss Rock Bunting data with spOccupancy

We will now fit some spatial models to the weekly-aggregated Rock Bunting data first with `spOccupancy` (in this section) and then, in Section 5, with `ubms`. In both cases, we will use the fitted spatial model to make predictions to come up with more species distribution maps.

4.1 Spatial exponential model with 5-Nearest-neighbour Gaussian Process (NNGP 5)

We now fit the spatial version of the model, using the nearest-neighbour Gaussian Process with 5 nearest neighbours. This only evaluates the spatial correlation matrix for the nearest 5 neighbours instead of for all neighbours, which results in a substantial gain in computation time, as we will see when we will also fit the model with 15 neighbours, in one of the later sections below (or if we did fit the full Gaussian process model).

```
# Compute distances between sites
distMat <- dist(AtlasData$coords)

# Select exponential covariance model
cov.model <- "exponential"

# Choose inits
# as in the model-fitting package vignette
spo.inits <- list(alpha = 0,
                  beta = 0,
                  z = apply(AtlasData$y, 1, max, na.rm = TRUE),
                  sigma.sq = 2,
                  phi = 3 / mean(distMat),
                  w = rep(0, nrow(AtlasData$y)))

# Set some tuning param
spo.tuning <- list(phi = 1)

# Define some priors (as in vignette)
min.dist <- min(distMat)
max.dist <- max(distMat)
spo.priors <- list(beta.normal = list(mean = 0, var = 2.72),
                   alpha.normal = list(mean = 0, var = 2.72),
                   sigma.sq.ig = c(2, 1),
                   phi.unif = c(3/max.dist, 3/min.dist))
```

We launch the model a first time, with just short chains.

```
# Resultant number of draws for each chain
4 * ((50 * 100) - 2000) / 12
```

```
[1] 1000
```

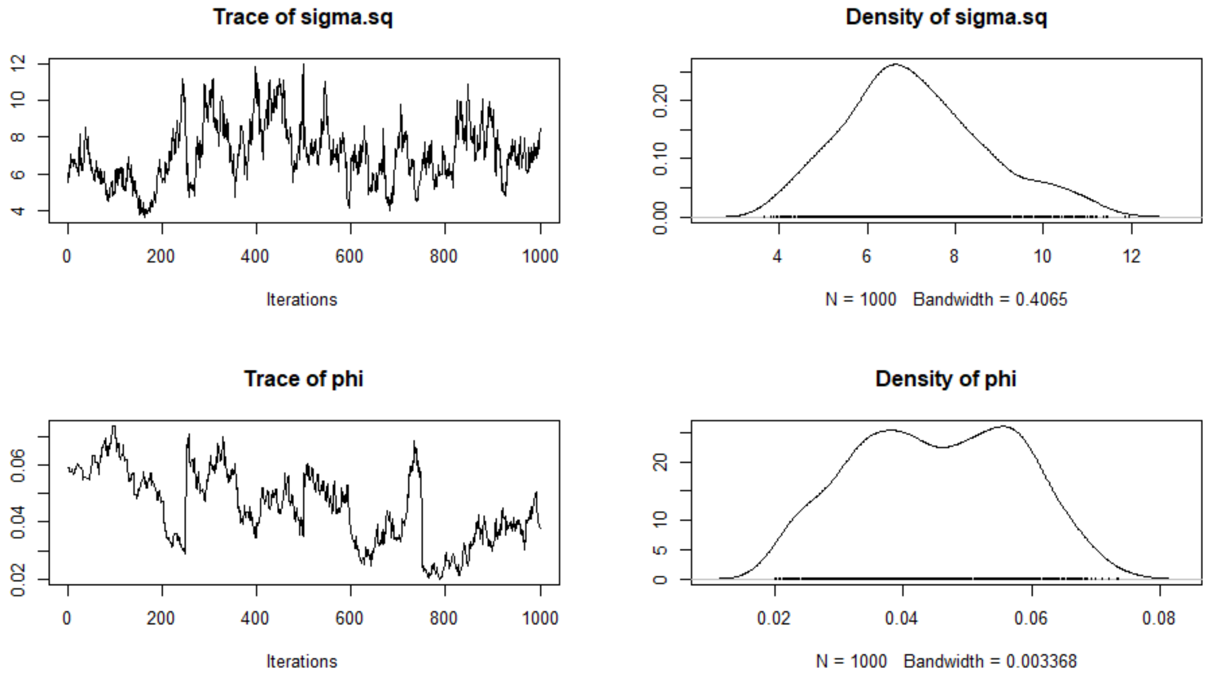
```
# Run a quickie to get ballpark estimates (ART 3.4 mins)
fm4a <-
    spPGOcc(occ.formula = occ.formula,
       det.formula = det.formula,
       data = AtlasData, inits = spo.inits,
       n.batch = 100, batch.length = 50,
       priors = spo.priors, cov.model = cov.model,
       NNGP = TRUE, n.neighbors = 5, tuning = spo.tuning,
```

```
        n.omp.threads = 6,
        n.burn = 2000, n.thin = 12, n.chains = 4,
        n.report = 100, verbose = TRUE)
summary(fm4a)
```

```
Call:
spPGOcc(occ.formula = occ.formula, det.formula = det.formula, data =
AtlasData,
    inits = spo.inits, priors = spo.priors, tuning = spo.tuning,
cov.model = cov.model,
    NNGP = TRUE, n.neighbors = 5, n.batch = 100, batch.length = 50,
    n.omp.threads = 6, verbose = TRUE, n.report = 100, n.burn = 2000,
    n.thin = 12, n.chains = 4)

Samples per Chain: 5000
Burn-in: 2000
Thinning Rate: 12
Number of Chains: 4
Total Posterior Samples: 1000
Run Time (min): 7.5268

Occurrence (logit scale):
                   Mean     SD    2.5%     50%   97.5%   Rhat ESS
(Intercept)     -4.4254 0.4418 -5.2748 -4.4023 -3.6240 1.3293  22
z.buildings     -0.4507 0.2211 -0.9365 -0.4373 -0.0602 1.0061 535
z.buildings2     0.2823 0.1945 -0.1431  0.2977  0.6190 1.0039 542
z.elevation     -1.8882 0.2724 -2.4106 -1.8899 -1.3556 1.1303 146
z.elevation2    -1.3190 0.2265 -1.7590 -1.3093 -0.8712 1.1660 135
z.northness     -0.8170 0.0994 -1.0184 -0.8173 -0.6356 1.0727 227
z.northness2    -0.2873 0.0882 -0.4530 -0.2887 -0.1220 1.0214 531
z.rivers         0.0573 0.1149 -0.1617  0.0582  0.2876 1.0291 362
z.rivers2       -0.0921 0.0868 -0.2649 -0.0957  0.0740 1.0077 517
z.rocks         -0.8340 0.5435 -2.0099 -0.7580  0.1124 1.1386  78
z.rocks2        -0.0377 0.2585 -0.5771 -0.0296  0.4375 1.0966 106
z.slope          0.8947 0.2089  0.4715  0.8867  1.3135 1.0374 330
z.slope2        -0.2809 0.1433 -0.5673 -0.2773 -0.0126 1.0152 338
z.structures     0.2533 0.0738  0.1133  0.2530  0.3962 1.0127 472
z.structures2 -0.1061 0.0567 -0.2156 -0.1082  0.0069 1.0276 699
z.wetlands      -1.0969 0.7398 -3.0365 -0.9709 -0.0552 1.1223  87
z.wetlands2     -0.6322 0.5277 -1.9521 -0.5652  0.1229 1.1408 113
z.kfrivers      -0.1144 0.0960 -0.3022 -0.1172  0.0755 1.1429 324
z.kfrivers2      0.1442 0.0595  0.0421  0.1386  0.2772 1.0204 593

Detection (logit scale):
                 Mean     SD    2.5%     50%   97.5%   Rhat  ESS
(Intercept) -0.5852 0.0510 -0.6841 -0.5860 -0.4877 1.0113 1000
date1       -0.1899 0.0635 -0.3164 -0.1894 -0.0645 1.0009 1000
date2       -0.0324 0.0293 -0.0902 -0.0329  0.0244 1.0036 1000

Spatial Covariance:
           Mean     SD   2.5%    50%   97.5%   Rhat ESS
sigma.sq 7.0934 1.6299 4.2521 6.8961 10.7274 1.7883  33
phi      0.0452 0.0126 0.0225 0.0451  0.0675 2.4033  10
```

According to the convergence test criterion Rhat, most parameters have converged, but not the two parameters in the Spatial Covariance model, nor the occupancy intercept. The former is fairly typical, since such variance parameters are often the hardest to estimate in spatial models. Just for fun we look at the traceplots (which, note, simply chains the draws from the 4 chains, which in my opinion does not make so much sense!).

```
plot(fm4a$alpha.samples)    # Traceplots of detection params, not shown
plot(fm4a$beta.samples)     # Traceplots of occupancy params, not shown
plot(fm4a$theta.samples)    # Traceplots of covariance params
```



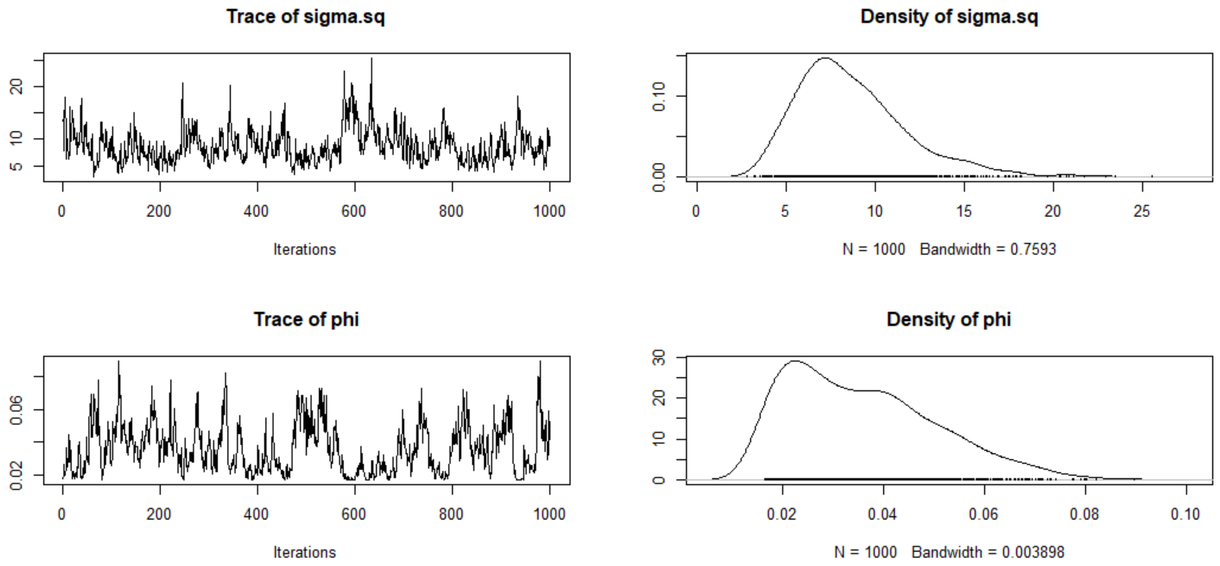
We repeat this with longer chains and with more burnin. Also, we use as inits for the spatial covariance parameters the estimates from the short run.

```
# Resultant number of draws for each chain
4 * ((500 * 200) - 50000) / 200
[1] 1000


# Choose inits, parly on solution from short run
# as in the model-fitting package vignette
spo.inits <- list(alpha = 0,
                  beta = 0,
                  z = apply(AtlasData$y, 1, max, na.rm = TRUE),
                  sigma.sq = 7.0934,
                  phi = 0.0452,
                  w = rep(0, nrow(AtlasData$y)))


# Launch model in spOccupancy
fm4b <-
    spPGOcc(occ.formula = occ.formula,
       det.formula = det.formula,
       data = AtlasData, inits = spo.inits,
       n.batch = 500, batch.length = 200,
       priors = spo.priors, cov.model = cov.model,
       NNGP = TRUE, n.neighbors = 5, tuning = spo.tuning,
```

```
        n.omp.threads = 6,
        n.burn = 50000, n.thin = 200, n.chains = 4,
        n.report = 100, verbose = TRUE)
summary(fm4b)
```

```
Call:
spPGOcc(occ.formula = occ.formula, det.formula = det.formula, data = AtlasData,
    inits = spo.inits, priors = spo.priors, tuning = spo.tuning, cov.model =
cov.model,
    NNGP = TRUE, n.neighbors = 5, n.batch = 500, batch.length = 200,
    n.omp.threads = 6, verbose = TRUE, n.report = 100, n.burn = 50000,
    n.thin = 200, n.chains = 4)

Samples per Chain: 100000
Burn-in: 50000
Thinning Rate: 200
Number of Chains: 4
Total Posterior Samples: 1000
Run Time (min): 141.215

Occurrence (logit scale):
                 Mean     SD    2.5%     50%   97.5%   Rhat  ESS
(Intercept)   -3.7388 0.8226 -5.0194 -3.8696 -1.8721 1.7253   17
z.buildings   -0.4438 0.2112 -0.8938 -0.4339 -0.0399 1.0108 1000
z.buildings2   0.2821 0.1978 -0.1621  0.3011  0.6175 1.0007 1000
z.elevation   -1.9004 0.2899 -2.4731 -1.8846 -1.3535 1.0176  522
z.elevation2  -1.3339 0.2173 -1.7661 -1.3334 -0.9094 1.0031  790
z.northness   -0.8169 0.1003 -1.0245 -0.8188 -0.6340 1.0005  675
z.northness2  -0.2837 0.0918 -0.4705 -0.2813 -0.1110 1.0056  868
z.rivers       0.0686 0.1082 -0.1634  0.0693  0.2754 1.0054  909
z.rivers2     -0.0964 0.0841 -0.2555 -0.0969  0.0651 1.0011 1000
z.rocks       -0.7627 0.5750 -2.0881 -0.6912  0.1587 1.0122  685
z.rocks2      -0.0182 0.2878 -0.6373  0.0138  0.4738 1.0233  662
z.slope        0.9103 0.2142  0.4902  0.9065  1.3330 1.0070 1000
z.slope2      -0.2793 0.1389 -0.5513 -0.2807  0.0043 1.0050 1000
z.structures   0.2594 0.0760  0.1092  0.2600  0.4020 1.0163  834
z.structures2 -0.1068 0.0558 -0.2214 -0.1058 -0.0017 1.0018 1000
z.wetlands    -1.1928 0.8440 -3.1586 -1.0571 -0.0082 1.0062  742
z.wetlands2   -0.7235 0.6036 -2.1253 -0.6440  0.1547 1.0058  779
z.kfrivers    -0.1132 0.0959 -0.2983 -0.1103  0.0671 1.0167  849
z.kfrivers2    0.1431 0.0636  0.0372  0.1365  0.2793 1.0038 1000

Detection (logit scale):
               Mean     SD    2.5%     50%   97.5%   Rhat  ESS
(Intercept) -0.5853 0.0526 -0.6885 -0.5848 -0.4804 1.0024 1000
date1       -0.1931 0.0652 -0.3216 -0.1942 -0.0667 1.0039 1026
date2       -0.0302 0.0300 -0.0904 -0.0294  0.0270 1.0044  852

Spatial Covariance:
           Mean     SD   2.5%    50%   97.5%   Rhat ESS
sigma.sq 8.7545 3.1907 4.3085 8.1101 16.4078 1.1330  85
phi      0.0359 0.0146 0.0170 0.0336  0.0687 1.0565  53
```

```
plot(fm4b$alpha.samples)   # Traceplots of detection params, not shown
plot(fm4b$beta.samples)    # Traceplots of occupancy params, not shown
plot(fm4b$theta.samples)   # Traceplots of covariance params
```



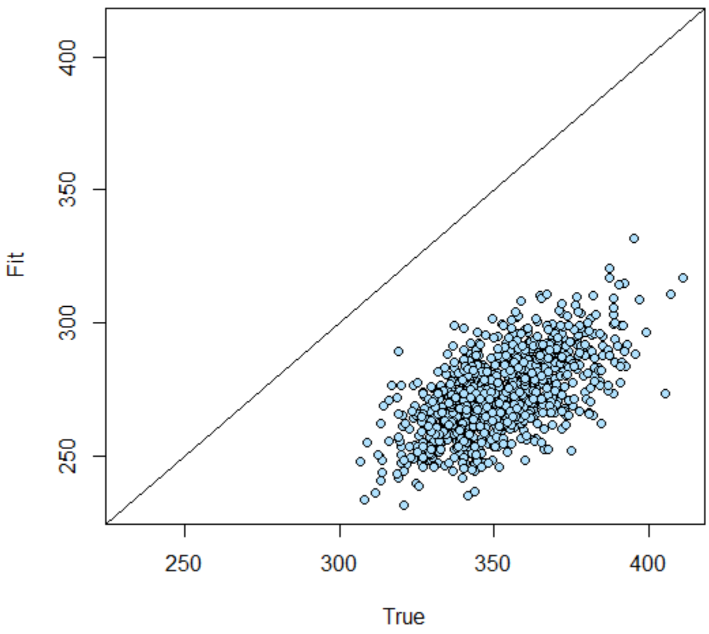
This looks much better now.

As a goodness of fit test, we run a posterior predictive check now. We use a Freeman-Tukey discrepancy measure, which is $D(\mathbf{y},\theta)=\sum_i(\sqrt{y_i}-\sqrt{E(y_i)})^2$, where where "y_i *is the observed value* i *and* $E(y_i)$ *its expected value. In contrast to a Chi-square discrepancy, the Freeman–Tukey statistic obviates the need to pool cells with small expected values and moreover is insensitive to unstable results due to small expected cell frequencies* " (AHM1, page 76).

```
system.time(              # 56 sec
  ppc.out <- ppcOcc(fm4b, fit.stat = 'freeman-tukey', group = 1)
  )
summary(ppc.out)
```

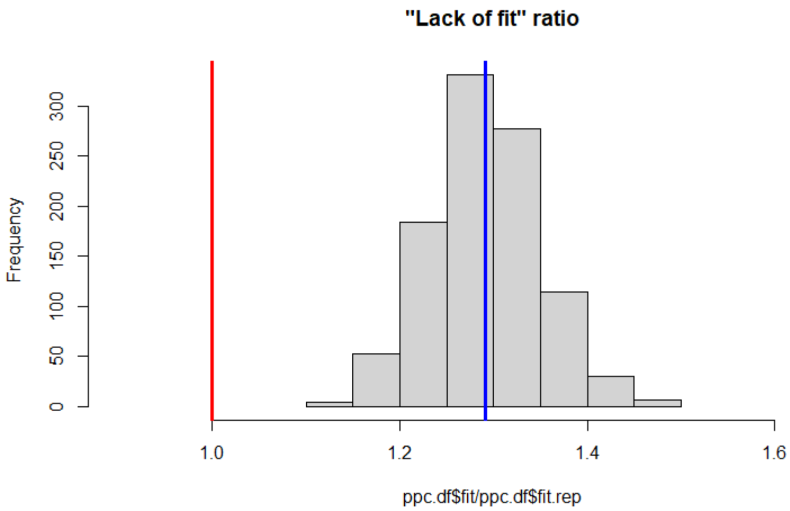
```
Call:
ppcOcc(object = fm4b, fit.stat = "freeman-tukey", group = 1)

Samples per Chain: 100000
Burn-in: 50000
Thinning Rate: 200
Number of Chains: 4
Total Posterior Samples: 1000

Bayesian p-value:  0
Fit statistic:  freeman-tukey
```

```
ppc.df <- data.frame(fit = ppc.out$fit.y,
                     fit.rep = ppc.out$fit.y.rep,
                     color = 'lightskyblue1')
ppc.df$color[ppc.df$fit.rep > ppc.df$fit] <- 'lightsalmon'
xylim <- range(c(ppc.df$fit, ppc.df$fit.rep))
plot(ppc.df$fit, ppc.df$fit.rep, bg = ppc.df$color, pch = 21,
     ylab = 'Fit', xlab = 'True', xlim = xylim, ylim = xylim)
abline(0, 1)
#lines(ppc.df$fit, ppc.df$fit, col = 'black')
```



OK, formally, our model does not fit at all. We have quite a bit of overdispersion (OD)…. ☹ We can form the ratio between the two fit statistics to get a metric for the magnitude of that OD. We find (next code block) that it is about 1.3, which is perhaps moderately strong.

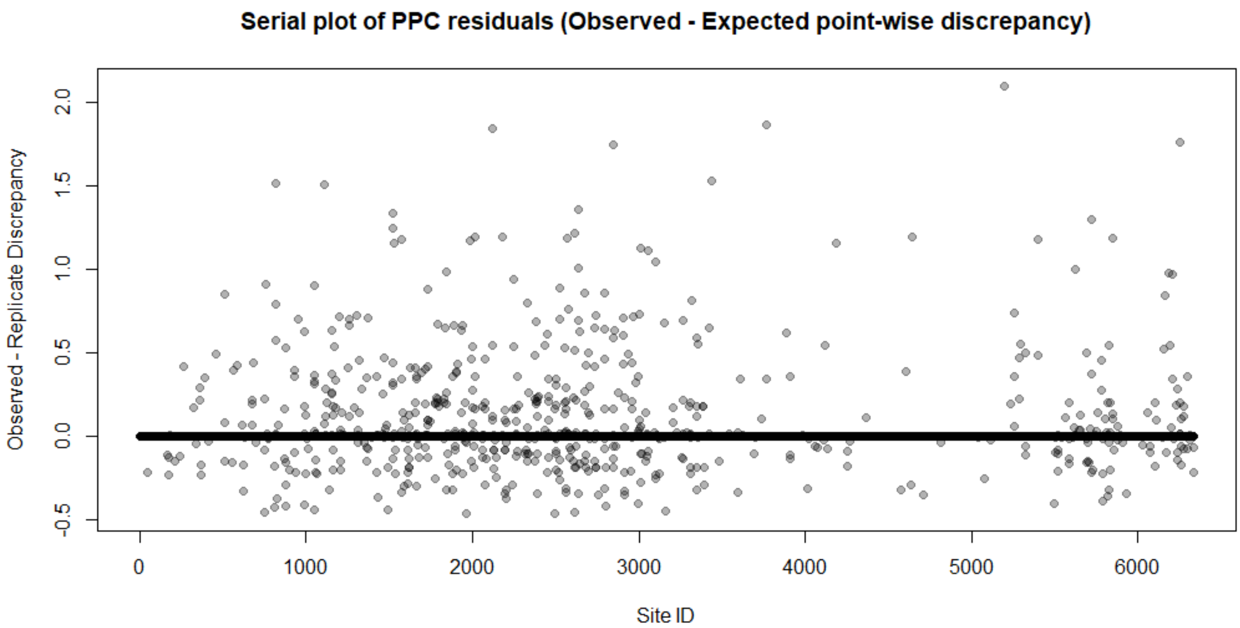
```
# Compute informal lack of fit ratio
hist(ppc.df$fit/ppc.df$fit.rep, xlim = c(0.9, 1.7), main = '"Lack of fit"
ratio')
abline(v = 1, col = 'red', lwd = 3)
abline(v = mean(ppc.df$fit/ppc.df$fit.rep), col = 'blue', lwd = 3)
```



This just gives a one-number summary of goodness of fit, but more insightful perhaps is it to inspect some sort of a residual of a model: this quantifies lack of fit for each individual data point by giving a distance between the observed data and those expected under the model in units of the metric chosen for the PPC.

We can compute a metric that has some spirit of a residual from the output of our ppc for the level of the sites in the analysis. We take the difference between the Freeman-Tukey statistic for the actual data and that for the replicate data.

```
# Compute a residual-like quantity from the PPC results
resi <- ppc.out$fit.y.group.quants[3, ] -
ppc.out$fit.y.rep.group.quants[3, ]
plot(resi, pch = 19, cex = 1, col = rgb(0,0,0,0.3),
xlab = 'Site ID', ylab = 'Observed - Replicate Discrepancy', main =
'Serial plot of PPC residuals (Observed - Expected point-wise
discrepancy)')
```

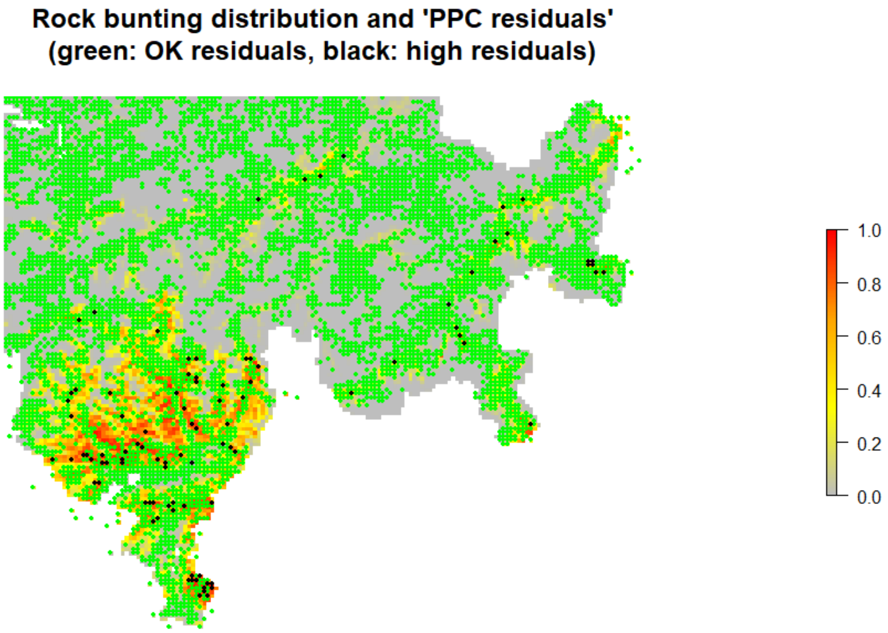


We see many sites where the discrepancy for the observed data is much greater than for the replicated data. We now want to relate these quantities against other things to see whether we can see some patterns in the lack of fit. We make a map of these "residual-like quantities" and then also plot them against the values of all site covariates, perhaps see some function form which is not quite right.

We get a little bit ahead of ourselves and plot the residuals within the posterior mean map of occupancy, which we produce about 3 pages down from here. Thus, to execute the next thing here, you will first have to execute the code over the next about 3 pages.

```
# Make a map of the posterior means of predicted occupancy
# with 'residuals' plotted over it
par(mfrow = c(1, 1), mar = c(4,4,6,8), cex.main = 1.5)
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.psi4b))

plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Rock
bunting distribution and 'PPC residuals'\n(green: OK residuals, black:
high residuals)", zlim = c(0, 1))

points(x = AtlasData$coords[,1][abs(resi) < 0.5],
   y = AtlasData$coords[,2][abs(resi) < 0.5],
   cex = 0.5, pch = 16, col = 'green')
```

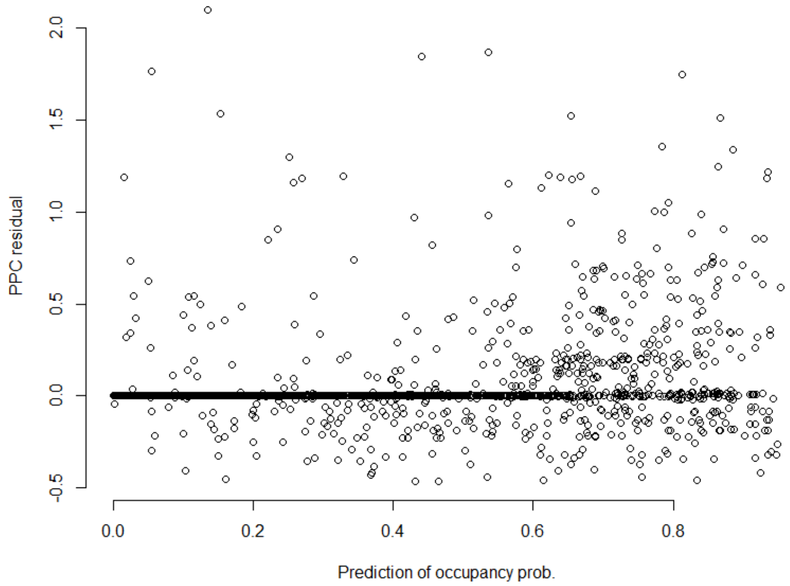
```
points(x = AtlasData$coords[,1][resi > 0.5],
   y = AtlasData$coords[,2][resi > 0.5],
   cex = 0.5, pch = 16, col = 'black')
points(x = AtlasData$coords[,1][resi < -0.5],
   y = AtlasData$coords[,2][ resi < -0.5],
   cex = 0.5, pch = 16, col = 'purple')
```



We see that the model lack of fit is concentrated in the high-density areas, especially in the Ticino (that's the triangle in the SW part of the domain). Have to think what we could do about this.

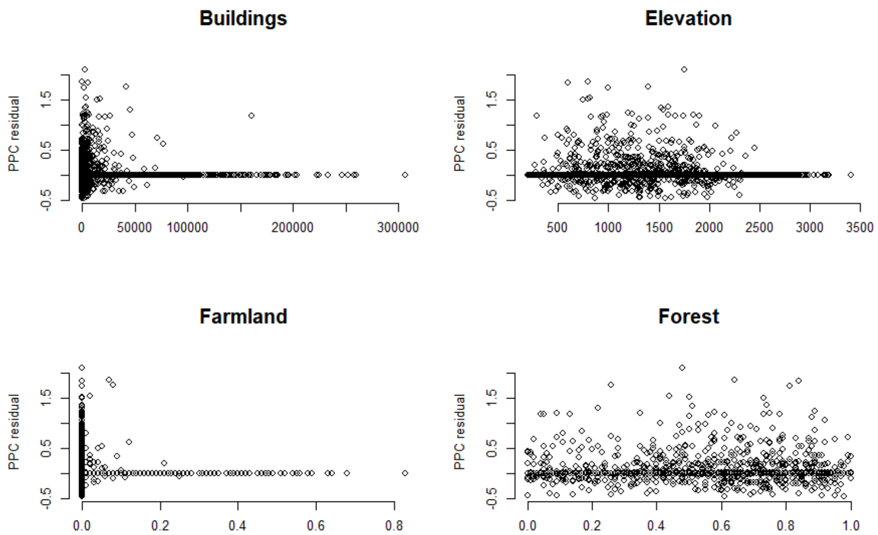
Let's plot them directly against the predictions of psi.

```
# Plot 'PPC residuals' against the predictions directly
quad1 <- paste(AtlasData$coords[,1], AtlasData$coords[,2], sep = '.')
quad2 <- paste(CovarData$x, CovarData$y, sep = '.')
idx <- pmatch(quad1, quad2)
plot(pm.psi4b[idx], resi, xlab = 'Prediction of occupancy prob.', ylab =
'PPC residual', frame = FALSE)
```



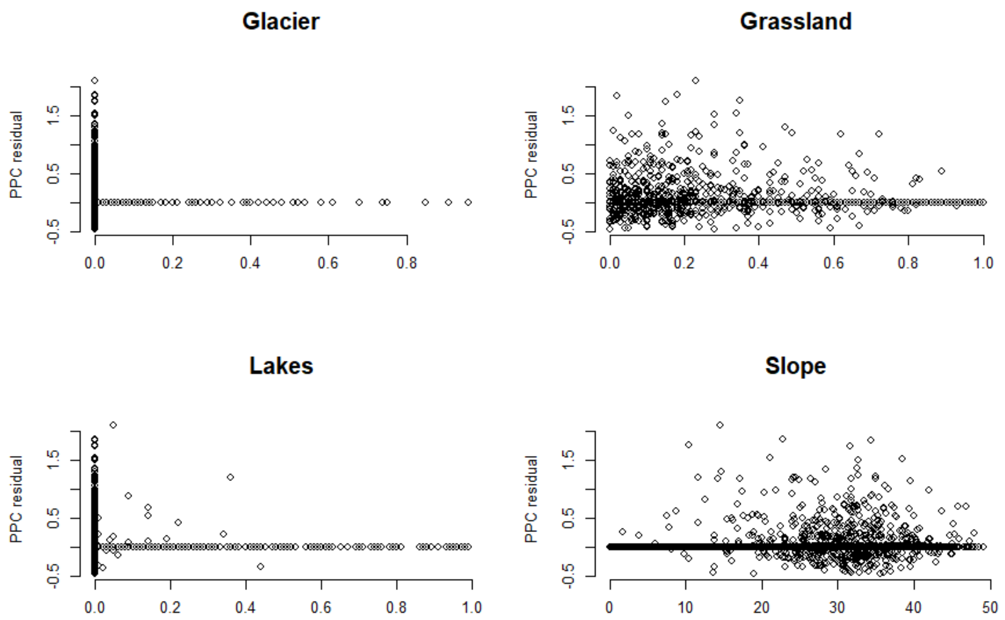
Next, we plot the PPC residuals against all covariates.

```
# Plot 'PPC residuals' against the covariates
par(mfrow = c(2, 2), mar = c(4,4,6,2), cex.main = 1.5)
plot(CovarData$buildings[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Buildings', frame = FALSE)
plot(CovarData$elevation[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Elevation', frame = FALSE)
plot(CovarData$farmland[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Farmland', frame = FALSE)
plot(CovarData$forest[idx], resi, xlab = '', ylab = 'PPC residual', main
= 'Forest', frame = FALSE)
```

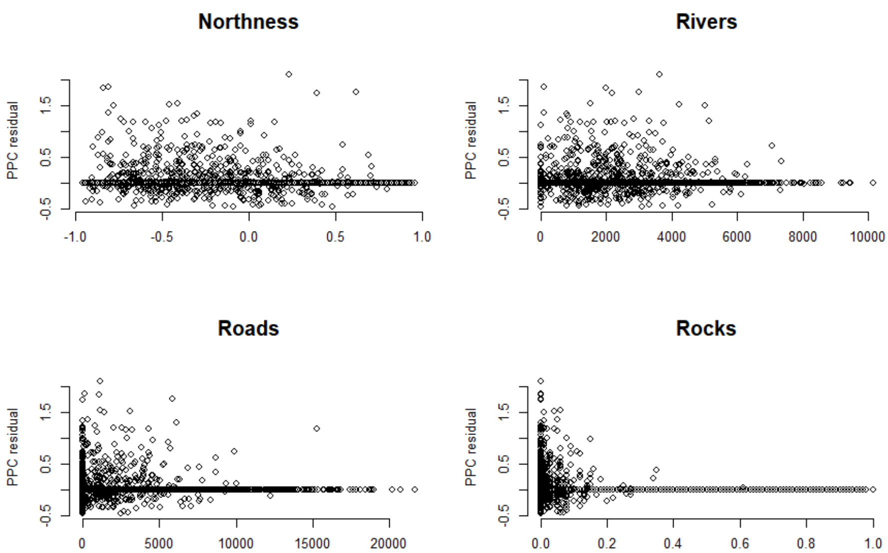



```
plot(CovarData$glacier[idx], resi, xlab = '', ylab = 'PPC residual', main
= 'Glacier', frame = FALSE)
```

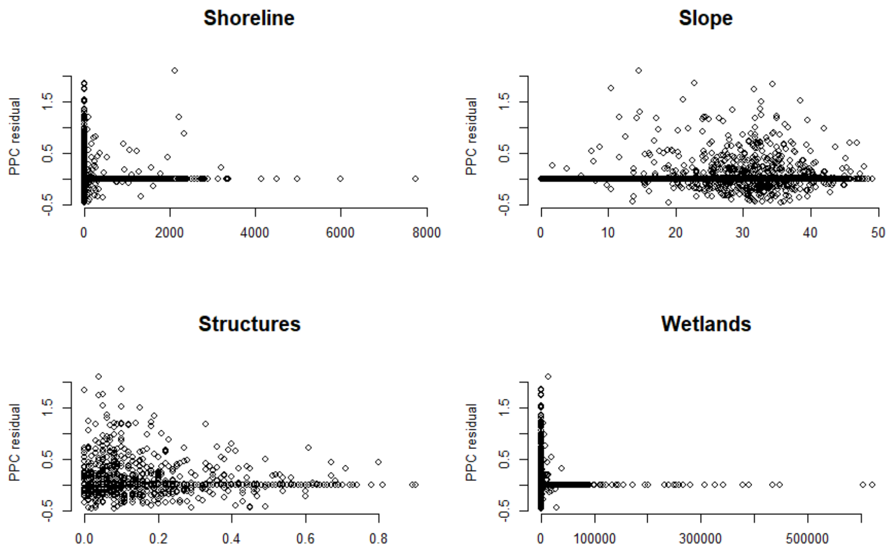
```
plot(CovarData$grassland[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Grassland', frame = FALSE)
plot(CovarData$lakes[idx], resi, xlab = '', ylab = 'PPC residual', main =
'Lakes', frame = FALSE)
plot(CovarData$slope[idx], resi, xlab = '', ylab = 'PPC residual', main =
'Slope', frame = FALSE)
```




```
plot(CovarData$northness[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Northness', frame = FALSE)
plot(CovarData$rivers[idx], resi, xlab = '', ylab = 'PPC residual', main
= 'Rivers', frame = FALSE)
plot(CovarData$roads[idx], resi, xlab = '', ylab = 'PPC residual', main =
'Roads', frame = FALSE)
plot(CovarData$rocks[idx], resi, xlab = '', ylab = 'PPC residual', main =
'Rocks', frame = FALSE)
```



```
plot(CovarData$shoreline[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Shoreline', frame = FALSE)
plot(CovarData$slope[idx], resi, xlab = '', ylab = 'PPC residual', main =
'Slope', frame = FALSE)
plot(CovarData$structures[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Structures', frame = FALSE)
plot(CovarData$wetlands[idx], resi, xlab = '', ylab = 'PPC residual',
main = 'Wetlands', frame = FALSE)
```



Have to think more about this ….

(One thing we note is that the data from each site stem from different years among the 2013–2016 period. If there were systematic differences between the years, then our failure to include the survey year may perhaps be responsible for the lack of fit ? We could in principle get this information, but it is not in our data sets here and so we can't follow up this hunch.)

Before we produce SDMs from this model we want to compare the two models using the WAIC, i.e., the nonspatial model and the spatial model.

```
(waic1 <- waicOcc(fm3))            # nonspatial model
(waic2b <- waicOcc(fm4b))           # spatial model
```

```
> (waic1 <- waicOcc(fm3))            # nonspatial model
       elpd          pD        WAIC
-3070.89071    22.71815  6187.21773

> (waic2b <- waicOcc(fm4b))           # spatial model
      elpd          pD       WAIC
-2650.0382   185.1032  5670.2830
```

So, the spatial model is preferred by a large margin over the corresponding spatial model.

We go on forming and the plotting predictions now. This proceeds very similarly as what we did above for the non-spatial model, but now we must also supply the coordinates for the prediction function.

```
spo.cov <- as.matrix(cbind(1, newdata[,selected.covs]))
str(spo.cov)
```

```
num [1:12757, 1:19] 1 1 1 1 1 1 1 1 1 1 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:12757] "674_210" "675_210" "676_210" "677_210" ...
  ..$ : chr [1:19] "1" "z.buildings" "z.buildings2" "z.elevation" ...
```

```
spo.coords <- CovarData[1:2]
head(spo.coords)
```

```
          x   y
674_210 674 210
675_210 675 210
676_210 676 210
677_210 677 210
678_210 678 210
679_210 679 210
```

```
# Form predictions from the spatial model (NNGP with 5 neighbours)
system.time(
  pred.fm4b <- predict(fm4b, spo.cov, spo.coords, verbose = TRUE)
)
```

```
----------------------------------------
        Prediction description
----------------------------------------
NNGP Occupancy model with Polya-Gamma latent
variable fit with 6338 observations.

Number of covariates 19 (including intercept if specified).

Using the exponential spatial correlation model.

Using 5 nearest neighbors.

Number of MCMC samples 1000.

Predicting at 6419 non-sampled locations.


Source compiled with OpenMP support and model fit using 1 threads.
-------------------------------------------------
                Predicting
-------------------------------------------------
Location: 100 of 6419, 1.56%
Location: 200 of 6419, 3.12%
Location: 300 of 6419, 4.67%
...
Location: 6300 of 6419, 98.15%
Location: 6400 of 6419, 99.70%
Location: 6419 of 6419, 100.00%
Generating latent occupancy state
   user  system elapsed
  19.20    0.42   19.66
```

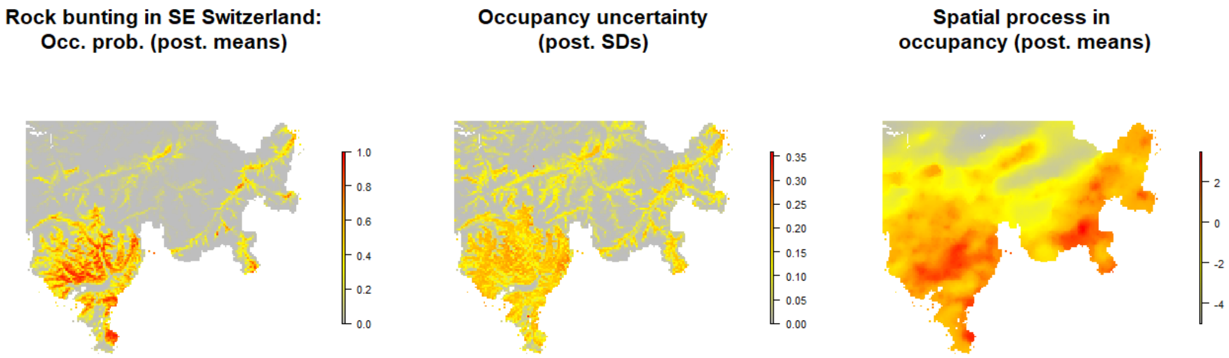
Don't quite understand that "out-of-sample" nature of the predictions: `spOccupancy` claims to only do this for about 2 k sites, but what happens to the other 1k sites ? After all: when we plot the predictions, we don't get to see any holes !

Whatever it is .... we produce those maps now.

```
# Load needed packages
require(raster)

# Map posterior mean and posterior SD of occupancy probability
# We also compute posterior means of w
pm.psi4b <- apply(pred.fm4b$psi.0.samples, 2, mean)  # Post mean
psd.psi4b <- apply(pred.fm4b$psi.0.samples, 2, sd)   # Post sd
pm.w4b <- apply(pred.fm4b$w.0.samples, 2, mean)      # Post mean process

summary(pm.psi4b)
summary(psd.psi4b)
summary(pm.w4b)
```

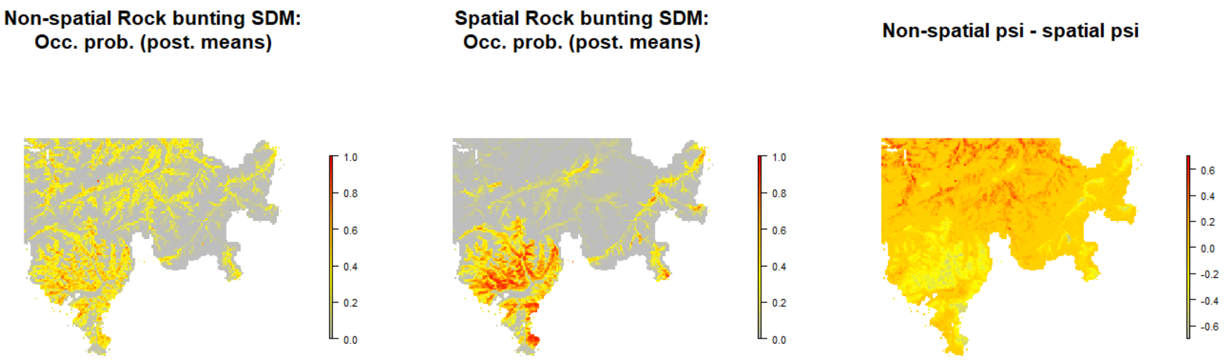
```
> summary(pm.psi4b)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.000001 0.002104 0.018732 0.125414 0.129971 0.957386
> summary(psd.psi4b)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0000038 0.0053283 0.0308190 0.0649085 0.1135202 0.3296820
> summary(pm.w4b)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-5.0112 -2.7754 -0.9570 -1.1684  0.4152  3.4594
```

```
mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))

par(mfrow = c(1, 3), mar = c(4,4,6,8), cex.main = 2)
# Posterior means
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.psi4b))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Rock
bunting in SE Switzerland:\nOcc. prob. (post. means)", zlim = c(0, 1))

# Posterior sds
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = psd.psi4b))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Occupancy uncertainty\n(post. SDs)", zlim = c(0, 0.36))

# Posterior means of spatial process w
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.w4b))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial process in\n occupancy (post. means)", zlim = c(-5, 3.5))
```



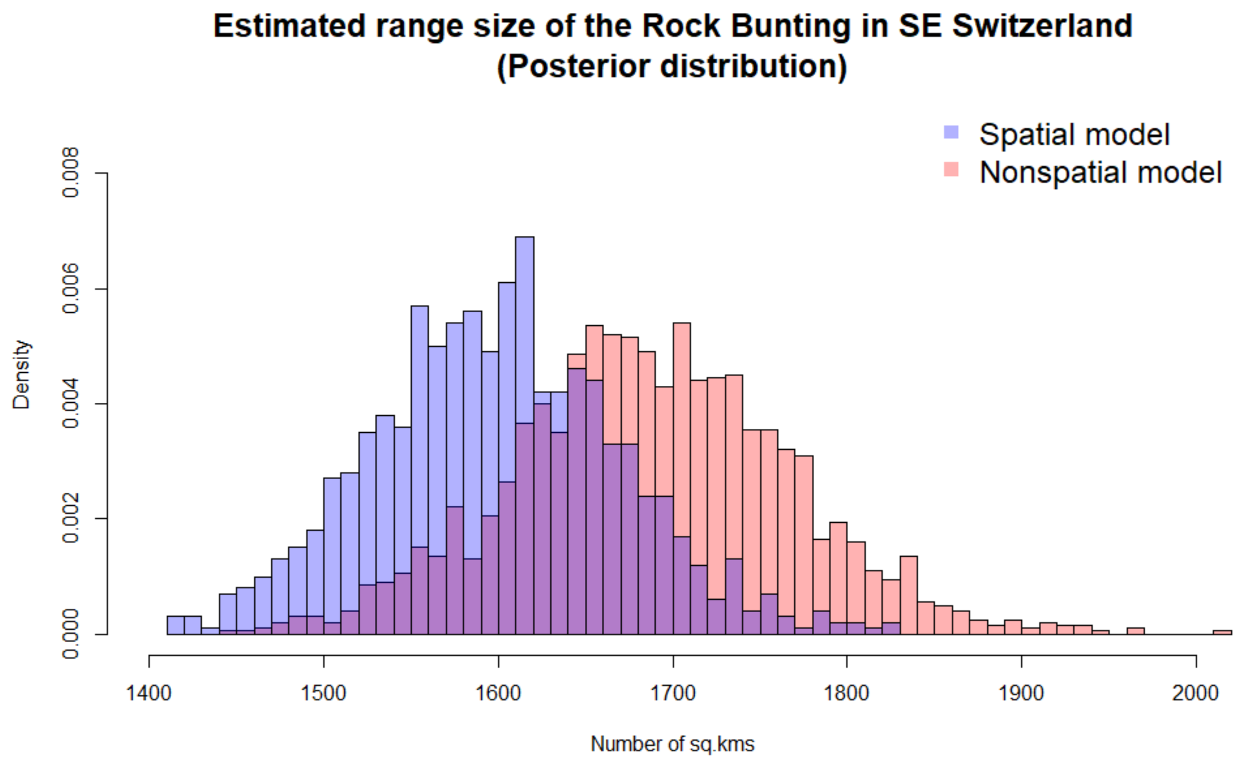
We want to compare the two maps from the spatial and nonspatial models

```
# Compute difference map: non-spatial psi minus spatial psi
diff.psi <- pm.psi - pm.psi4b

par(mfrow = c(1, 3), mar = c(2, 4, 6, 10), cex.main = 2)
# Posterior means nonspatial model
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
   z = pm.psi))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Non-
spatial Rock bunting SDM:\nOcc. prob. (post. means)", zlim = c(0, 1))

# Posterior means spatial model
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
   z = pm.psi4b))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial Rock bunting SDM:\nOcc. prob. (post. means)", zlim = c(0, 1))

# Occupancy difference map
range(diff.psi)
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
   z = diff.psi))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Non-
spatial psi - spatial psi", zlim = c(-0.7, 0.7))
```



Let's compute the expected range size of the rock bunting in SE Switzerland. For this we can simply compute the sum(psi) as a derived quantity and then summarize its posterior distribution.

```
# Compute range size and plot (not shown)
rangeSize2 <- apply(pred.fm4b[[1]], 1, sum)
hist(rangeSize2, breaks = 40, col = 'grey', main = 'Estimated range size
of the Rock Bunting in SE Switzerland\n(Posterior distribution from
spatial model in spOccupancy)', xlab = 'Number of sq.kms', freq = FALSE)


# Plot posterior distributions from non-spatial and spatial next to each
other
rangeSize1 <- rangeSize.spo
hist(rangeSize1, breaks = 60, col = rgb(1,0,0,0.3), main = 'Estimated
range size of the Rock Bunting in SE Switzerland\n(Posterior
distribution)', xlab = 'Number of sq.kms', freq = FALSE, xlim = c(1400,
2000), ylim = c(0, 0.009), cex.main = 1.6)
hist(rangeSize2, breaks = 40, col = rgb(0,0,1,0.3), freq = FALSE, add =
TRUE)
legend('topright', pch = 15, col = c(rgb(0,0,1,0.3), rgb(1,0,0,0.3)),
legend = c('Spatial model', 'Nonspatial model'), bty = 'n', cex = 1.5)
```



Thus, in this case, if we can believe WAIC and the spatial model is really better, it seems that with the nonspatial model we overestimate the range size of the Rock bunting.

4.2 Spatial exponential model with 15-Nearest-neighbour Gaussian Process (NNGP 15)

We wonder what's the main differences between an NNGP with fewer or with more nearest neighbours included in the calculations. Hence, we now fit the NNGP (15) model to the Rock bunting data as well. Essentially, we can recycle almost everything from Section 4.1. We launch the model directly with even longer chains, using as inits some of the solutions from the NNGP(5) run.

We launch the model again, using much larger MCMC settings than before.

```
# Resultant number of draws for each chain
4 * ((500 * 400)-50000) / 600
```

```
[1] 1000
```

```
# Launch model in spOccupancy (key change marked in grey)
fm4c <-
    spPGOcc(occ.formula = occ.formula,
       det.formula = det.formula,
       data = AtlasData, inits = spo.inits,
       n.batch = 500, batch.length = 400,
       priors = spo.priors, cov.model = cov.model,
       NNGP = TRUE, n.neighbors = 15, tuning = spo.tuning,
       n.omp.threads = 6,
       n.burn = 50000, n.thin = 600, n.chains = 4,
       n.report = 100, verbose = TRUE)
```

That takes probably between 5 and 10 hours to finish (my laptop falls asleep ….), i.e., quite a bit longer than with NNGP 5 before. We check the results.

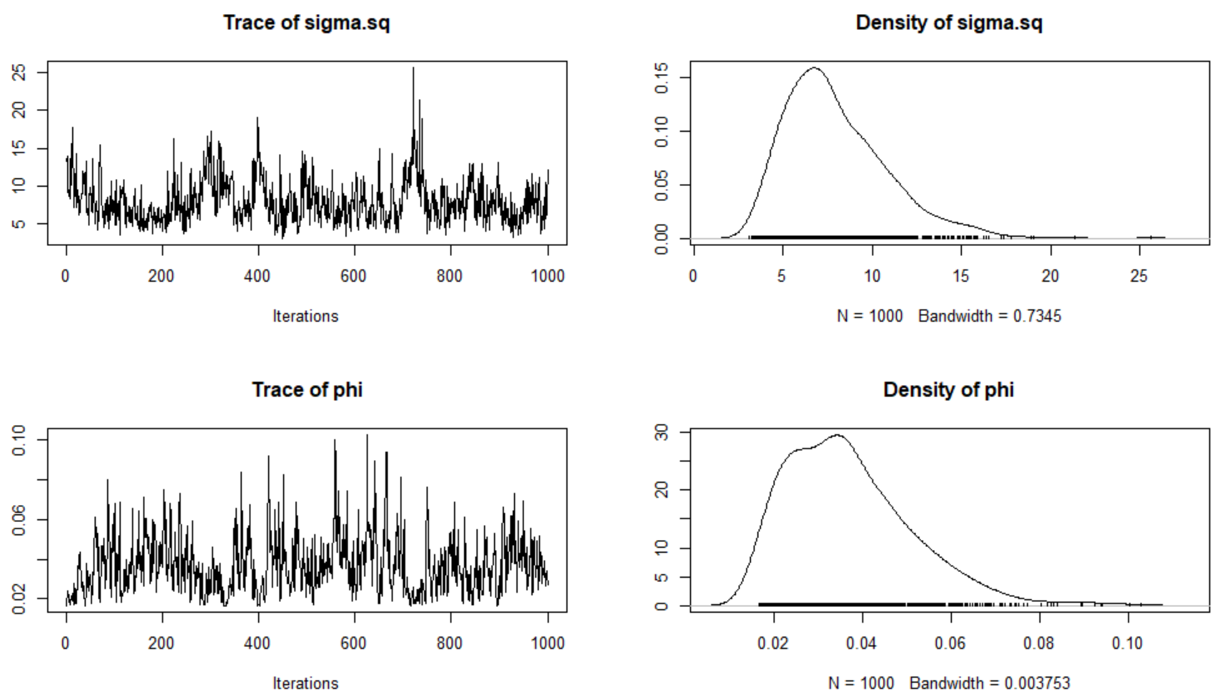
```
summary(fm4c)
```

```
Call:
spPGOcc(occ.formula = occ.formula, det.formula = det.formula, data = AtlasData,
    inits = spo.inits, priors = spo.priors, tuning = spo.tuning, cov.model =
cov.model,
    NNGP = TRUE, n.neighbors = 15, n.batch = 500, batch.length = 400,
    n.omp.threads = 6, verbose = TRUE, n.report = 100, n.burn = 50000,
    n.thin = 600, n.chains = 4)

Samples per Chain: 200000
Burn-in: 50000
Thinning Rate: 600
Number of Chains: 4
Total Posterior Samples: 1000
Run Time (min): 1016.0938

Occurrence (logit scale):
                 Mean     SD    2.5%     50%   97.5%   Rhat  ESS
(Intercept)   -3.4614 0.9345 -5.0004 -3.5708 -1.2481 1.1454   34
z.buildings   -0.4482 0.2162 -0.9299 -0.4290 -0.0716 1.0019 1163
z.buildings2   0.2802 0.1949 -0.1434  0.2956  0.6298 1.0039 1000
z.elevation   -1.9043 0.2922 -2.4765 -1.9069 -1.3519 1.0085  758
z.elevation2  -1.3378 0.2202 -1.7802 -1.3382 -0.9223 1.0076  908
z.northness   -0.8333 0.0969 -1.0237 -0.8293 -0.6594 1.0090  776
z.northness2  -0.2876 0.0880 -0.4570 -0.2864 -0.1148 1.0036 1200
z.rivers       0.0647 0.1125 -0.1490  0.0617  0.3090 1.0008 1000
z.rivers2     -0.0915 0.0840 -0.2590 -0.0904  0.0721 1.0059 1000
z.rocks       -0.8004 0.5801 -2.1651 -0.7256  0.1582 1.0021 1000
```

```
z.rocks2       -0.0362 0.2826 -0.6678 -0.0060  0.4196 1.0109 1000
z.slope         0.8880 0.2037  0.4913  0.8818  1.2903 1.0059 1000
z.slope2       -0.2689 0.1365 -0.5517 -0.2613 -0.0193 1.0068 1000
z.structures    0.2584 0.0758  0.1197  0.2580  0.4114 1.0083 1000
z.structures2 -0.1074 0.0557 -0.2149 -0.1054 -0.0010 1.0166  874
z.wetlands     -1.1772 0.8259 -3.1198 -1.0658 -0.0304 1.0003 1000
z.wetlands2    -0.7102 0.5827 -2.0947 -0.6142  0.1260 1.0004 1000
z.kfrivers     -0.1186 0.0974 -0.3056 -0.1198  0.0636 1.0056  866
z.kfrivers2     0.1410 0.0598  0.0326  0.1372  0.2660 1.0004 1000

Detection (logit scale):
               Mean     SD    2.5%     50%   97.5%   Rhat  ESS
(Intercept) -0.5888 0.0528 -0.6965 -0.5881 -0.4862 1.0198 1000
date1       -0.1909 0.0692 -0.3277 -0.1877 -0.0609 1.0094 1000
date2       -0.0316 0.0305 -0.0895 -0.0319  0.0304 1.0019 1000

Spatial Covariance:
           Mean     SD   2.5%    50%   97.5%   Rhat ESS
sigma.sq 7.9809 2.8606 3.9606 7.3983 14.9019 1.1235  83
phi      0.0369 0.0142 0.0174 0.0348  0.0689 1.0460 162
```

Based on the Rhat < 1.1 criterion, most chains have converged fine.

```
plot(fm4c$beta.samples)    # For the occupancy params
plot(fm4c$alpha.samples)   # For the detection params
plot(fm4c$theta.samples)   # For the spatial params
```



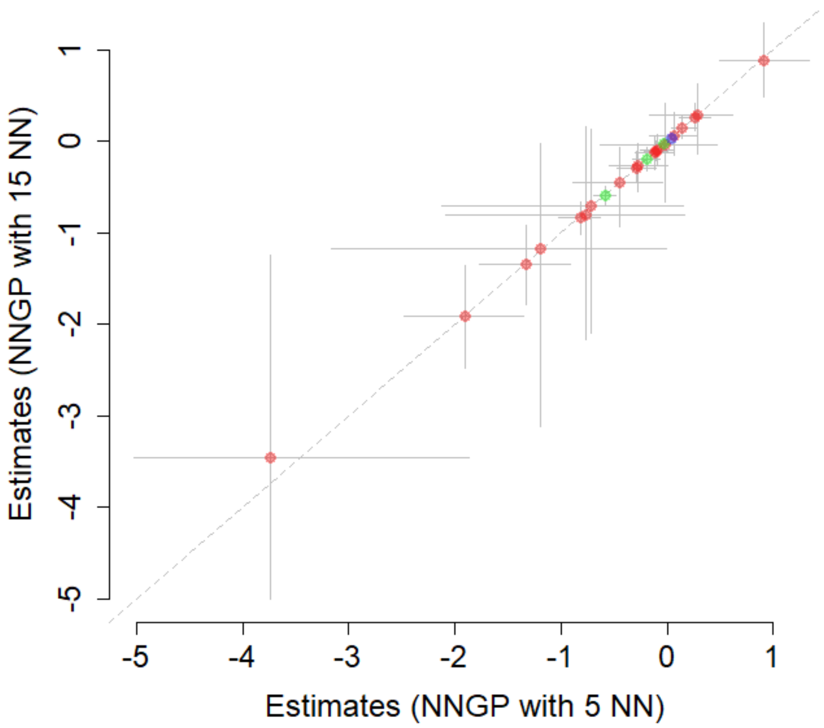
Visually though, they are not extremely bad.

We compare the estimates under the two versions of the model; that with 5 and that with 15 nearest neighbours.

```
library(abind)
```

```
# Compute point estimates
m4bpm <- c(apply(fm4b$beta.samples, 2, mean), apply(fm4b$alpha.samples,
2, mean), apply(fm4b$theta.samples, 2, mean))
m4cpm <- c(apply(fm4c$beta.samples, 2, mean), apply(fm4c$alpha.samples,
2, mean), apply(fm4c$theta.samples, 2, mean))

# Compute 95% CRIs
qt <- function(x) quantile(x, c(0.025, 0.975))
m4bCRI <- abind(apply(fm4b$beta.samples, 2, qt),
apply(fm4b$alpha.samples, 2, qt), apply(fm4b$theta.samples, 2, qt))
m4cCRI <- abind(apply(fm4c$beta.samples, 2, qt),
apply(fm4c$alpha.samples, 2, qt), apply(fm4c$theta.samples, 2, qt))

# Make plot
xylim <- c(-5, 1.2)
par(mar = c(5,5,3,2), cex.lab = 1.5, cex.axis = 1.5)
plot(m4bpm, m4cpm, xlab = 'Estimates (NNGP with 5 NN)', ylab = 'Estimates
(NNGP with 15 NN)', frame = FALSE, col = 'grey', pch = 1, cex = 1, xlim =
xylim, ylim = xylim)
abline(0, 1, col = 'grey', lty = 2)
segments(m4bCRI[1,], m4cpm, m4bCRI[2,], m4cpm, lwd = 1, col = 'grey')
segments(m4bpm, m4cCRI[1,], m4bpm, m4cCRI[2,], lwd = 1, col = 'grey')
points(m4bpm[1:19], m4cpm[1:19], col = rgb(1,0,0, 0.4), pch = 16, cex =
1.2)
points(m4bpm[20:22], m4cpm[20:22], col = rgb(0,1,0, 0.4), pch = 16, cex =
1.2)
points(m4bpm[23:24], m4cpm[23:24], col = rgb(0,0,1, 0.4), pch = 16, cex =
1.2)
```

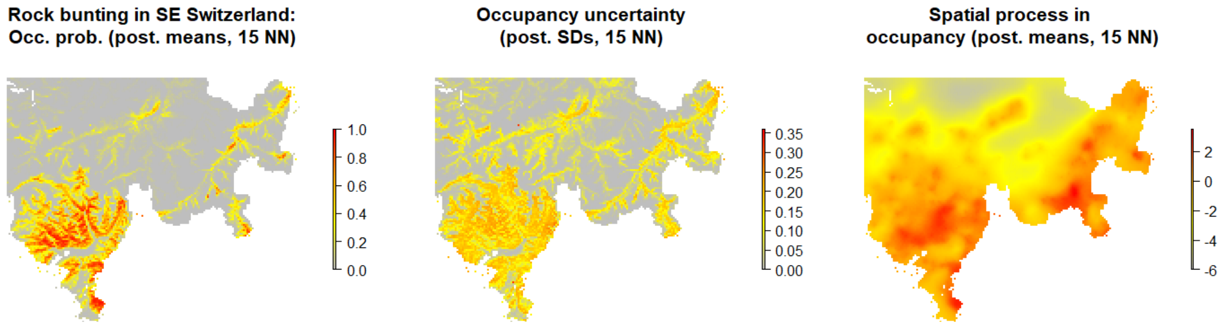
In this figure, red is for the occupancy, green for detection and blue for spatial params.



Estimates and their associated uncertainty look quite similar, with the exception of the occupancy intercept, which is about 0.5 units larger with NNGP 15.

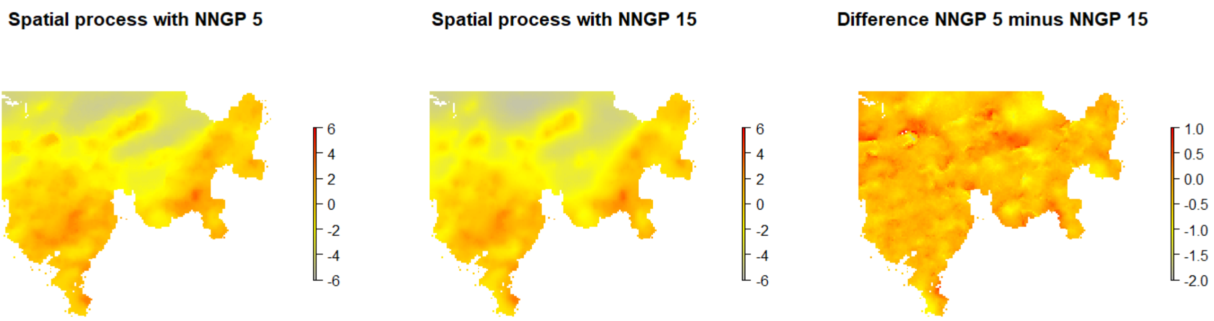
The estimates are about the same and so the predictions (i.e., the species distribution maps) should also come out about the same, we do form predictions now quickly. We can again recycle some from before.

```
# Form predictions from the spatial model (NNGP with 15 neighbours)
system.time(
  pred.fm4c <- predict(fm4c, spo.cov, spo.coords, verbose = TRUE)
)

# Load needed packages
require(raster)

# Map posterior mean and posterior SD of occupancy probability
# We also compute posterior means of w
pm.psi4c <- apply(pred.fm4c$psi.0.samples, 2, mean)  # Post mean
psd.psi4c <- apply(pred.fm4c$psi.0.samples, 2, sd)   # Post sd
pm.w4c <- apply(pred.fm4c$w.0.samples, 2, mean)      # Post mean process

summary(pm.psi4c)
summary(psd.psi4c)
summary(pm.w4c)


mapPalette1 <- colorRampPalette(c("grey", "yellow", "orange", "red"))

par(mfrow = c(1, 3), mar = c(4,4,6,8), cex.main = 2)
# Posterior means
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.psi4c))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main = "Rock
bunting in SE Switzerland:\nOcc. prob. (post. means, 15 NN)", zlim = c(0,
1))

# Posterior sds
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = psd.psi4c))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Occupancy uncertainty\n(post. SDs, 15 NN)", zlim = c(0, 0.36))

# Posterior means of spatial process w
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.w4c))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial process in\n occupancy (post. means, 15 NN)", zlim = c(-6, 3.5))
```



We want to compare the two maps of the spatial process from the NNGP 5 and the NNGP 15 runs.

```
# Compute difference map: NNGP 15 minus NNGP 5
diff.w <- pm.w4c - pm.w4b
summary(diff.w)

par(mfrow = c(1, 3), mar = c(2, 4, 6, 10), cex.main = 2)
# Posterior means of spatial process w with 5 nearest neighbours
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.w4b))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial process with NNGP 5", zlim = c(-6, 6))

# Posterior means of spatial process w with 15 nearest neighbours
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.w4c))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial process with NNGP 15", zlim = c(-6, 6))

# Spatial process difference map
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = diff.w))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Difference NNGP 5 minus NNGP 15", zlim = c(-2, 1))
```



Not sure what to do with this…

5 Fitting restricted spatial regression (RSR) models with ubms

R package ubms has functionality to fit spatial random fields (i.e., to fit truly "spatial models") using the formulation of a so-called 'restricted spatial regression', or RSR (Hodges & Reich, 2010; Johnson et al., 2013); see also the package vignette(cran.r-project.org/web/packages/ubms/vignettes/spatial-models.html). Here, we fit an occupancy model with an RSR formulation, drawing on that vignette. We also note that there is another package dedicated specifically to occupancy modeling with RSR spatial model, called stocc (Johnson et al., *Ecology*, 2013).

For model fitting with ubms, most importantly, we have to merge our previous two data sets, i.e., the one with all the covariates for the prediction domain and the other one with the survey data. And then we have to decide on a neighbourhood size. We begin with the former and merge the two data sets with respect to detection data in AtlasData.

```
library(ubms)    # Also automatically loads unmarked

# Remind ourselves of format
str(AtlasData)
str(CovarData)

# Make unique quadrat identifier in both data sets
ID_y <- paste(AtlasData$coords[,1], AtlasData$coords[,2])
ID_cv <- paste(CovarData[,1], CovarData[,2])
```

Now we loop over every site in the smaller list and check, which site in the larger list it corresponds to, and fill in its detection data 'AtlasData$y' at this place in a new matrix called 'y_ubms'. Also, we have to do the same with the detection covariates

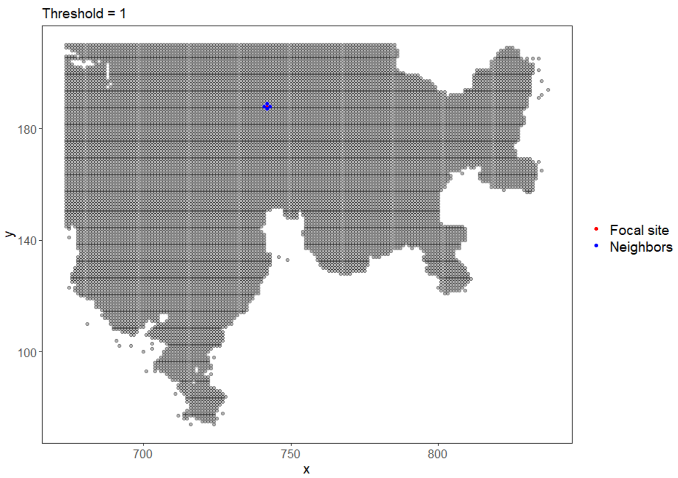
```
# Create new data frames for reponse (detection/nondetection) and for
detection covs
y_ubms <- matrix(NA, nrow = 12757, ncol = 21)
detcov_ubms <- matrix(NA, nrow = 12757, ncol = 21)
# Fill them
for(i in 1: length(ID_y)){
  sel.quad <- which(ID_cv == ID_y[i])
  y_ubms[sel.quad,] <- AtlasData$y[i,]
  detcov_ubms[sel.quad,] <- AtlasData$det.covs[[1]][i,]
}

# Sum tests ... look good (not shown)
sum(y_ubms, na.rm = TRUE) ; sum(AtlasData$y, na.rm = TRUE)
sum(detcov_ubms, na.rm = TRUE) ; sum(AtlasData$det.covs[[1]], na.rm =
TRUE)

# Repackage all the data in a new unmarked data frame for occu()
spatial_umf <- unmarkedFrameOccu(
  y = y_ubms,          # Reformatted Presence/Absence measurements
  siteCovs = CovarData[-(3:19)],  # Environmental covariates at site-
level
  obsCovs = list(date1 = detcov_ubms,
    date2 = detcov_ubms^2)) # Observation-specific covariates
summary(spatial_umf)
```

```
unmarkedFrame Object
```

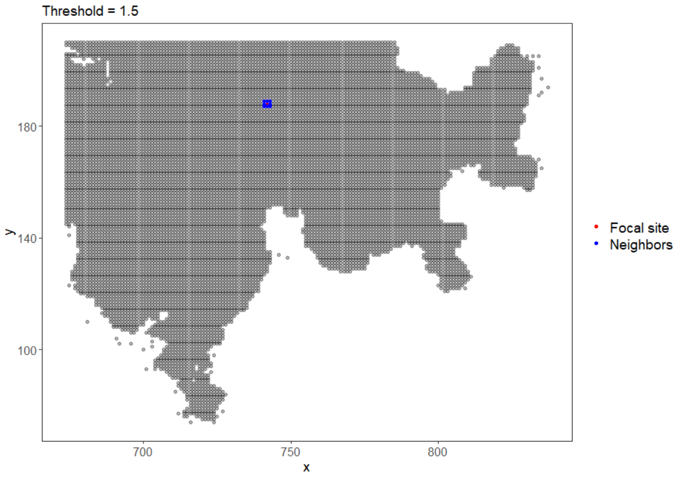
```
12757 sites
Maximum number of observations per site: 21
Mean number of observations per site: 1.59
Sites with at least one detection: 543

Tabulation of y observations:
     0      1   <NA>
 19225   1090 247582

Site-level covariates:
       x                y              z.buildings
 Min.   :674.0   Min.   : 74.0   Min.   :-0.4334
 1st Qu.:707.0   1st Qu.:142.0   1st Qu.:-0.4334
 Median :733.0   Median :167.0   Median :-0.4285
 Mean   :741.1   Mean   :163.2   Mean   :-0.2721
 3rd Qu.:775.0   3rd Qu.:187.0   3rd Qu.:-0.3730
 Max.   :837.0   Max.   :210.0   Max.   : 9.2546

...

Observation-level covariates:
     date1            date2
 Min.   :103.0    Min.   :10609
 1st Qu.:136.0    1st Qu.:18496
 Median :171.0    Median :29241
 Mean   :171.1    Mean   :31046
 3rd Qu.:206.0    3rd Qu.:42436
 Max.   :240.5    Max.   :57840
 NA's   :134799   NA's   :134799
```

Then, we decide on the size of the neighbourhood (see paper by Johnson et al., 2013, and 'Spatial modeling' vignette in ubms for details). We can plot the neighbourhood for a sample focal grid cell to get an impression of what it means.

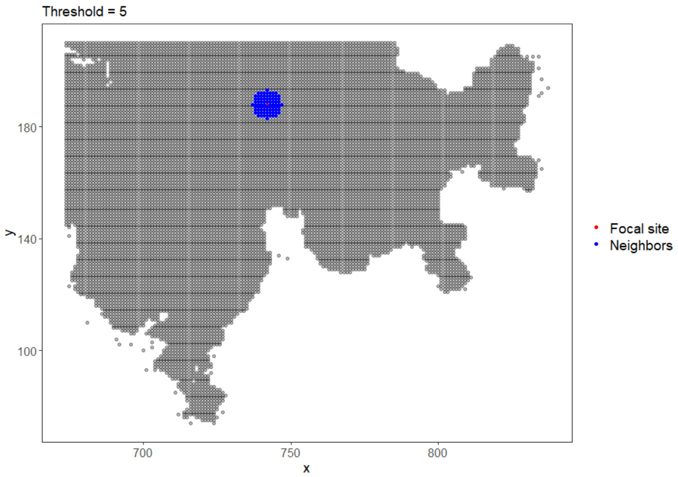
```
# A rook neighbourhood
site_cov <- CovarData[-(3:19)]
with(site_cov, RSR(x, y, threshold=1, plot_site = 3050))
```



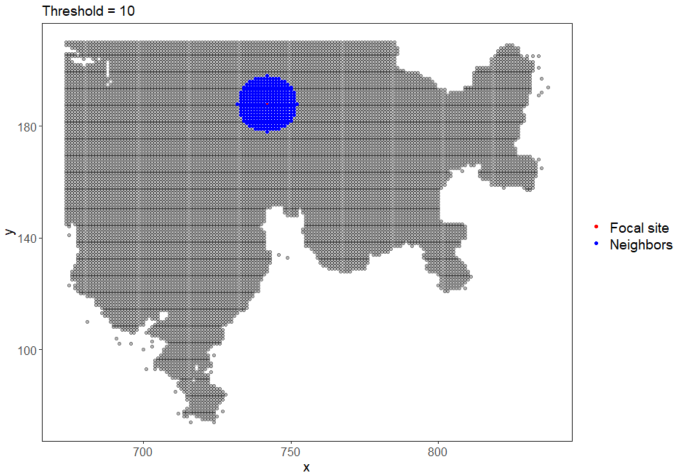

```
# A queen's neighbourhood
with(site_cov, RSR(x, y, threshold=1.5, plot_site = 3050))
```



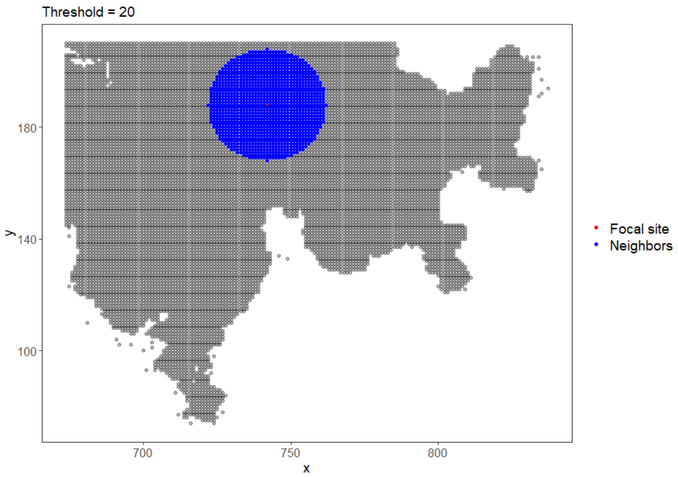

```
# A larger neighbourhood
with(site_cov, RSR(x, y, threshold=5, plot_site = 3050))
```




```
# A much larger neighbourhood
with(site_cov, RSR(x, y, threshold=10, plot_site = 3050))
```




```
# And EVEN larger neighbourhood
with(site_cov, RSR(x, y, threshold=20, plot_site = 3050))
```



This is f**** cool !!! Now we fit the last four of these spatial models.

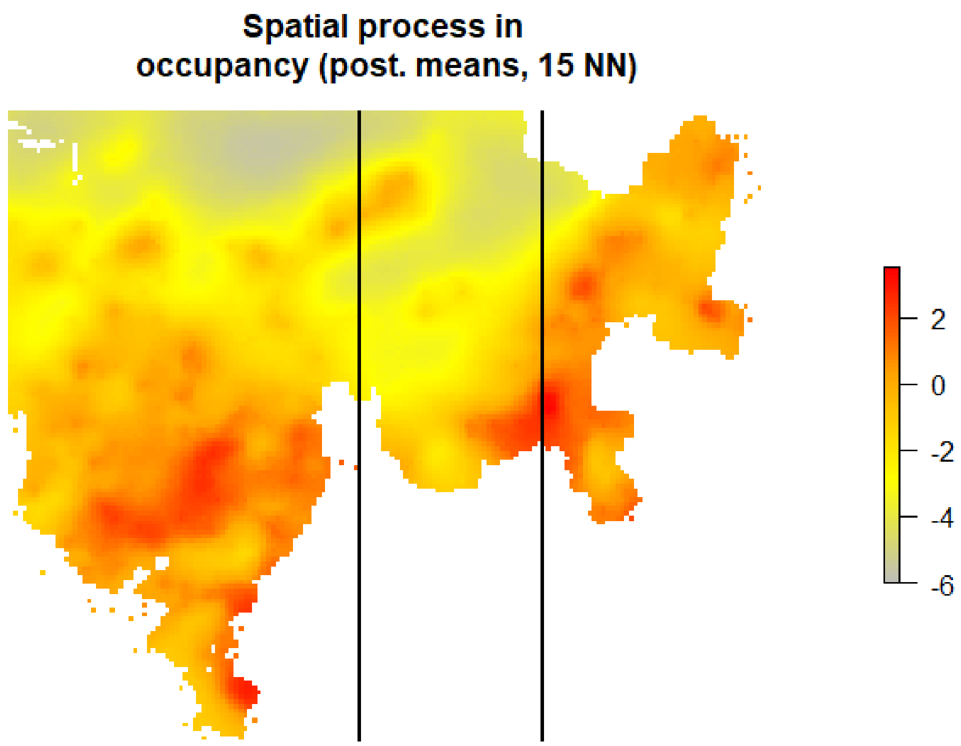
```
# Double formula: first part is for detection, second for occupancy
form1 <- ~ date1 + date2 ~ z.buildings + z.buildings2 +
  z.elevation + z.elevation2 + z.northness + z.northness2 +
  z.rivers + z.rivers2 + z.rocks + z.rocks2 + z.slope + z.slope2 +
  z.structures + z.structures2 + z.wetlands + z.wetlands2 +
  z.kfrivers + z.kfrivers2 + RSR(x, y, threshold = 1.5)

form2 <- ~ date1 + date2 ~ z.buildings + z.buildings2 +
  z.elevation + z.elevation2 + z.northness + z.northness2 +
  z.rivers + z.rivers2 + z.rocks + z.rocks2 + z.slope + z.slope2 +
  z.structures + z.structures2 + z.wetlands + z.wetlands2 +
  z.kfrivers + z.kfrivers2 + RSR(x, y, threshold = 5)

form3 <- ~ date1 + date2 ~ z.buildings + z.buildings2 +
  z.elevation + z.elevation2 + z.northness + z.northness2 +
  z.rivers + z.rivers2 + z.rocks + z.rocks2 + z.slope + z.slope2 +
  z.structures + z.structures2 + z.wetlands + z.wetlands2 +
  z.kfrivers + z.kfrivers2 + RSR(x, y, threshold = 10)

form4 <- ~ date1 + date2 ~ z.buildings + z.buildings2 +
  z.elevation + z.elevation2 + z.northness + z.northness2 +
  z.rivers + z.rivers2 + z.rocks + z.rocks2 + z.slope + z.slope2 +
  z.structures + z.structures2 + z.wetlands + z.wetlands2 +
  z.kfrivers + z.kfrivers2 + RSR(x, y, threshold = 20)
```

Unfortunately, on trying this, we find that `ubms` never gets started, since the step of doing the computations to setting up the RSR is never completed. So either 12,000 sites are too big of a sample size or else my old laptop is too weak. Whatever it is, we try to reduce the computational problem by only modeling in `ubms` part of previous domain.

Here's the map again of the spatial effect from `spOccupancy`.

```
# Posterior means of spatial process w, with possible geo-limits
library(raster)
r1 <- rasterFromXYZ(data.frame(x = CovarData[1], y = CovarData[2],
  z = pm.w4c))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial process in\n occupancy (post. means, 15 NN)", zlim = c(-6, 3.5))
abline(v = 750, col = 'black', lwd = 2) # Cut it here
```

```
abline(v = 790, col = 'black', lwd = 2) # Cut it here
```




```
# Current data for ubms
str(y_ubms)
str(CovarData)
str(detcov_ubms)

# Check out a new limit for defining smaller domain East of limit
plot(CovarData[,1:2], pch = '.')       # Current domain
abline(v = 750, col = 'red', lwd = 3) # Cut it here
abline(v = 790, col = 'red', lwd = 3) # Cut it here


# Restrict domain to Eastern half, with x coord >= 750
condition <- CovarData[, 1] > 749.5 & CovarData[, 1] < 790
sum(condition)                     # Results in 3022 quadrats

# Make new unmarked/ubms data frame
spatial_umf2 <- unmarkedFrameOccu(
  y = y_ubms[condition,],    # Presence/Absence measurements
  siteCovs = CovarData[-(3:19)][condition,],# Site-level covariates
  obsCovs = list(date1 = detcov_ubms[condition,],
    date2 = detcov_ubms[condition,]^2)) # Obs.level covariates
summary(spatial_umf2)
```

```
unmarkedFrame Object

3022 sites
Maximum number of observations per site: 21
```

```
Mean number of observations per site: 1.82
Sites with at least one detection: 33

Tabulation of y observations:
    0     1  <NA>
 5450    57 57955
...
```

```
# Try ubms again
system.time(
  fm51 <- stan_occu(form = form1, data = spatial_umf2,
  chains = 3, cores = 4) )
traceplot(fm51)        # Check convergence of chains
print(fm51)            # Print posterior summaries
```

From here on: have to run it for much longer

```
Call:
stan_occu(formula = form1, data = spatial_umf2, chains = 3, cores = 4)

Occupancy (logit-scale):
              Estimate    SD   2.5%   97.5% n_eff Rhat
(Intercept)     -1.554 2.023 -4.948  2.0082  2.40 2.43
z.buildings     -0.721 0.894 -1.980  1.7235 10.99 1.42
z.buildings2     0.345 0.785 -1.114  2.0827  4.25 1.62
z.elevation     -1.421 0.840 -3.028 -0.0575  8.78 1.42
z.elevation2     1.173 1.000 -0.936  3.1654  4.33 1.72
z.northness     -1.264 0.544 -2.296 -0.3807 11.65 1.35
z.northness2    -0.360 0.421 -1.213  0.5147 37.80 1.10
z.rivers        -0.214 0.505 -1.016  0.9640 13.71 1.32
z.rivers2        0.387 0.546 -0.853  1.2754 10.16 1.31
z.rocks          0.224 1.296 -2.212  2.6050  2.03 2.67
z.rocks2         0.136 0.786 -1.414  1.9480  5.69 1.38
z.slope          0.803 0.605 -0.423  1.9706  9.72 1.36
z.slope2         0.124 0.785 -1.331  1.5209  9.41 1.39
z.structures    -0.248 0.476 -1.319  0.5794  8.85 1.22
z.structures2   -1.635 0.741 -3.413 -0.5869 11.83 1.24
z.wetlands       1.398 0.845 -0.242  3.2768  8.34 1.54
z.wetlands2      0.554 0.807 -0.667  2.4102  8.45 1.28
z.kfrivers       0.238 0.350 -0.503  0.8635 20.88 1.17
z.kfrivers2      0.319 0.394 -0.138  1.3954  8.87 1.44
RSR [tau]        0.598 0.142  0.391  0.8817  4.08 1.80

Detection (logit-scale):
             Estimate       SD      2.5%    97.5% n_eff Rhat
(Intercept) -8.60e-01 1.70e+00 -4.548550 1.951162  6.08 1.82
date1       -1.50e-02 2.43e-02 -0.051718 0.042129  7.25 1.82
date2        1.06e-05 7.84e-05 -0.000167 0.000133  7.96 1.70

LOOIC: 587.008
Runtime: 3.452 hr
```

```
system.time(
  fm52 <- stan_occu(form = form2, data = spatial_umf,
  chains = 3, cores = 3) )
traceplot(fm52)        # Check convergence of chains
print(fm52)            # Print posterior summaries
```

```
Call:
```

```
stan_occu(formula = form2, data = spatial_umf, chains = 3, cores = 3)

Occupancy (logit-scale):
                Estimate       SD      2.5%    97.5%  n_eff  Rhat
(Intercept)    -5.290516 0.834757 -6.976841 -3.72671  560.9 1.012
z.buildings    -0.758707 0.464333 -1.677933  0.11995 1959.8 1.002
z.buildings2    0.024700 0.315633 -0.666229  0.54463 2598.3 0.999
z.elevation    -4.247360 0.778906 -5.925962 -2.82492  676.8 1.009
z.elevation2   -2.013594 0.495367 -3.037543 -1.05336  667.3 1.011
z.farmland     -0.235855 1.064590 -2.619319  1.60104 2503.3 1.000
z.farmland2     0.358896 0.798210 -1.268297  1.89459 2717.8 1.000
z.forest        0.844161 0.661278 -0.374320  2.19794 1139.8 1.004
z.forest2      -0.099094 0.154017 -0.411480  0.19718 2638.2 1.000
z.glacier       0.512192 1.734317 -2.794665  4.13826 3647.8 1.000
z.glacier2      0.061548 1.585591 -2.831322  3.43284 3527.0 1.000
z.grassland     1.269819 0.593430  0.174097  2.50891 1145.3 1.005
z.grassland2   -0.425495 0.207633 -0.833743 -0.01319 2747.4 1.000
z.lakes        -1.061160 1.161000 -3.956151  0.62686 1804.1 1.001
z.lakes2        0.281799 0.464490 -0.602052  1.19788 2008.2 1.000
z.nitrogen     -0.404664 0.488390 -1.388689  0.56869 2116.5 1.000
z.nitrogen2    -0.795501 0.513429 -1.824377  0.14905 2421.0 1.001
z.northness    -0.709011 0.141092 -0.996070 -0.44686  909.9 1.004
z.northness2   -0.331695 0.130093 -0.592709 -0.08170 3116.4 1.001
z.rivers       -0.005516 0.169969 -0.334634  0.32935 2578.8 1.001
z.rivers2      -0.268672 0.149004 -0.563735  0.00797 1390.8 1.004
z.roads         0.042460 0.325542 -0.603925  0.68045 2323.6 1.000
z.roads2        0.647240 0.237032  0.204770  1.12301 1873.9 1.003
z.rocks        -0.721682 1.181025 -3.292377  1.34921 1784.8 1.000
z.rocks2        0.163465 0.548424 -0.990303  1.15827 2000.1 1.000
z.shoreline    -0.135288 0.368543 -0.923070  0.52136 1539.1 1.002
z.shoreline2   -0.538139 0.361630 -1.360551  0.02548 1574.9 1.002
z.slope         1.120058 0.431085  0.277929  1.96546 2087.5 1.000
z.slope2       -0.288196 0.231221 -0.729726  0.17798 2248.3 1.000
z.structures    0.481916 0.250244 -0.003411  0.99585 1195.3 1.002
z.structures2 -0.060326 0.080806 -0.210866  0.10286 3452.6 1.000
z.wetlands     -0.585424 1.106953 -3.035314  1.22073 1281.9 1.001
z.wetlands2    -0.468261 0.838584 -2.195550  1.06709 1449.5 1.000
z.kfrivers     -0.253920 0.145648 -0.538998  0.03293 2176.9 1.001
z.kfrivers2     0.232661 0.102257  0.031424  0.43607 2711.3 1.000
RSR [tau]       0.000973 0.000437  0.000431  0.00209   69.5 1.050

Detection (logit-scale):
             Estimate     SD   2.5%  97.5% n_eff  Rhat
(Intercept)  -0.4452 0.0755 -0.594 -0.299  3068 1.000
date1        -0.2849 0.0729 -0.428 -0.140  3544 1.000
date2         0.0175 0.0787 -0.134  0.175  2654 0.999

LOOIC: 3175.862
Runtime: 33.610 min
```

```
system.time(
  fm53 <- stan_occu(form = form3, data = spatial_umf,
  chains = 3, cores = 3) )
traceplot(fm53)        # Check convergence of chains
print(fm53)            # Print posterior summaries
```

```
Call:
stan_occu(formula = form3, data = spatial_umf, chains = 3, cores = 3)

Occupancy (logit-scale):
                Estimate       SD      2.5%     97.5%  n_eff  Rhat
(Intercept)    -5.036296 7.68e-01 -6.672895 -3.624167 1700.8 1.000
```

```
z.buildings   -0.783345 4.44e-01 -1.736381  0.038386 2097.5 1.001
z.buildings2   0.001480 3.14e-01 -0.643558  0.530817 2305.8 1.002
z.elevation   -4.150793 7.61e-01 -5.643877 -2.716885  913.0 1.004
z.elevation2  -1.918139 4.71e-01 -2.874267 -1.037268 1153.8 1.004
z.farmland    -0.269642 1.07e+00 -2.669131  1.571881 1584.7 1.000
z.farmland2    0.319825 7.67e-01 -1.241112  1.782659 2710.1 1.001
z.forest       0.785141 6.45e-01 -0.438485  2.097630 1223.8 1.000
z.forest2     -0.081523 1.55e-01 -0.393677  0.221632 2706.6 1.001
z.glacier      0.488912 1.64e+00 -2.744625  4.007648 4195.6 1.000
z.glacier2     0.076367 1.52e+00 -2.607596  3.301367 3231.9 1.001
z.grassland    1.239746 5.63e-01  0.166789  2.382368 1205.1 1.001
z.grassland2  -0.424105 2.13e-01 -0.860063 -0.021542 2613.2 1.000
z.lakes       -1.086277 1.09e+00 -3.750795  0.524827 1913.5 1.001
z.lakes2       0.248763 4.32e-01 -0.654234  1.104851 2346.4 1.000
z.nitrogen    -0.298449 4.46e-01 -1.167301  0.555394 2593.1 1.000
z.nitrogen2   -0.625776 4.65e-01 -1.540808  0.261826 3087.5 1.000
z.northness   -0.705505 1.32e-01 -0.973062 -0.452952 1954.8 1.002
z.northness2  -0.304204 1.26e-01 -0.552288 -0.054521 2793.3 1.001
z.rivers      -0.033919 1.65e-01 -0.364202  0.291170 2700.6 0.999
z.rivers2     -0.258856 1.45e-01 -0.547107  0.017341 2819.8 1.000
z.roads        0.052698 3.26e-01 -0.580325  0.691180 2499.2 0.999
z.roads2       0.650465 2.27e-01  0.207186  1.101148 2520.8 1.001
z.rocks       -0.727005 1.20e+00 -3.298566  1.330992 2132.6 1.000
z.rocks2       0.215630 5.49e-01 -0.925595  1.188901 2267.5 1.000
z.shoreline   -0.116219 3.45e-01 -0.860152  0.474891 1770.6 1.002
z.shoreline2  -0.491407 3.48e-01 -1.281552  0.035828 1605.5 1.002
z.slope        1.062906 4.20e-01  0.241172  1.888272 2071.5 1.001
z.slope2      -0.237953 2.26e-01 -0.668175  0.222171 2480.3 1.000
z.structures   0.495545 2.39e-01  0.027202  0.972871 1267.9 1.001
z.structures2 -0.064964 8.10e-02 -0.224500  0.090176 3613.5 1.000
z.wetlands    -0.655568 1.07e+00 -3.052917  1.134910 2143.5 0.999
z.wetlands2   -0.503702 8.23e-01 -2.194062  0.958705 2218.4 1.000
z.kfrivers    -0.239700 1.41e-01 -0.512910  0.035434 1854.7 0.999
z.kfrivers2    0.242223 1.01e-01  0.045529  0.438652 2905.1 1.000
RSR [tau]      0.000257 9.41e-05  0.000132  0.000495   70.9 1.039

Detection (logit-scale):
            Estimate     SD   2.5%  97.5% n_eff Rhat
(Intercept)  -0.4471 0.0774 -0.594 -0.287  3511    1
date1        -0.2845 0.0745 -0.428 -0.136  4466    1
date2         0.0191 0.0782 -0.142  0.169  2794    1

LOOIC: 3193.158
Runtime: 42.285 min
```

```
system.time(
  fm54 <- stan_occu(form = form4, data = spatial_umf,
  chains = 3, cores = 3) )
traceplot(fm54)        # Check convergence of chains
print(fm54)            # Print posterior summaries
```

```
Call:
stan_occu(formula = form4, data = spatial_umf, chains = 3, cores = 3)

Occupancy (logit-scale):
               Estimate       SD      2.5%     97.5%  n_eff  Rhat
(Intercept)   -4.76e+00 7.61e-01 -6.35e+00 -3.363880 1308.3 1.001
z.buildings   -7.75e-01 4.29e-01 -1.64e+00  0.026810 2103.7 1.000
z.buildings2   7.70e-02 2.97e-01 -6.01e-01  0.608575 2695.6 1.001
z.elevation   -4.03e+00 7.35e-01 -5.49e+00 -2.616992  778.1 1.004
z.elevation2  -1.87e+00 4.51e-01 -2.73e+00 -1.000744 1267.7 1.003
z.farmland    -2.99e-01 1.07e+00 -2.70e+00  1.560407 2180.3 1.000
```

```
z.farmland2    3.53e-01 7.53e-01 -1.18e+00  1.796198 2952.0 1.000
z.forest       5.69e-01 6.43e-01 -6.72e-01  1.873471 1159.4 1.001
z.forest2     -6.80e-02 1.47e-01 -3.59e-01  0.220942 2592.0 0.999
z.glacier      4.48e-01 1.66e+00 -2.70e+00  3.810058 4831.9 1.000
z.glacier2     7.71e-02 1.42e+00 -2.45e+00  3.089253 3428.4 0.999
z.grassland    1.08e+00 5.64e-01  9.25e-03  2.191884 1216.8 1.001
z.grassland2  -4.58e-01 2.07e-01 -8.80e-01 -0.081429 2495.2 1.001
z.lakes       -1.12e+00 1.07e+00 -3.60e+00  0.518774 1863.4 1.000
z.lakes2       2.11e-01 4.27e-01 -6.56e-01  1.044259 2169.4 1.000
z.nitrogen    -2.57e-01 4.30e-01 -1.11e+00  0.559878 1302.5 1.001
z.nitrogen2   -4.97e-01 4.56e-01 -1.39e+00  0.365299 2918.5 1.001
z.northness   -7.00e-01 1.33e-01 -9.73e-01 -0.448607 2331.2 1.002
z.northness2  -3.23e-01 1.27e-01 -5.74e-01 -0.072778 2548.7 1.000
z.rivers      -2.12e-02 1.57e-01 -3.34e-01  0.273732 2367.0 1.000
z.rivers2     -2.40e-01 1.41e-01 -5.26e-01  0.028960 2463.5 1.002
z.roads        1.45e-02 2.97e-01 -5.79e-01  0.605644 2192.7 1.000
z.roads2       5.61e-01 2.15e-01  1.37e-01  0.989910 2330.7 1.001
z.rocks       -8.49e-01 1.17e+00 -3.43e+00  1.201773 1834.4 1.000
z.rocks2       2.52e-01 5.40e-01 -9.10e-01  1.218817 2052.8 1.000
z.shoreline   -1.76e-01 3.39e-01 -9.30e-01  0.417396 1696.2 1.001
z.shoreline2  -5.15e-01 3.48e-01 -1.32e+00  0.011826 1950.1 1.001
z.slope        1.12e+00 4.20e-01  3.09e-01  1.966786 1874.1 1.000
z.slope2      -2.33e-01 2.20e-01 -6.63e-01  0.204667 2047.0 1.000
z.structures   3.76e-01 2.41e-01 -8.79e-02  0.838372 1139.2 1.001
z.structures2 -4.51e-02 7.98e-02 -2.06e-01  0.110617 2714.4 1.000
z.wetlands    -6.93e-01 1.03e+00 -2.97e+00  0.982944 1812.7 1.000
z.wetlands2   -5.61e-01 8.09e-01 -2.26e+00  0.840076 1954.7 1.000
z.kfrivers    -2.40e-01 1.33e-01 -5.02e-01  0.014947 2187.6 1.001
z.kfrivers2    2.57e-01 9.62e-02  7.37e-02  0.449384 1989.0 1.001
RSR [tau]      9.02e-05 4.31e-05  4.03e-05  0.000196   53.7 1.078

Detection (logit-scale):
            Estimate     SD   2.5%  97.5% n_eff  Rhat
(Intercept)  -0.4517 0.0786 -0.604 -0.292  3173 0.999
date1        -0.2865 0.0745 -0.433 -0.147  4260 1.000
date2         0.0183 0.0789 -0.135  0.170  2953 1.000

LOOIC: 3210.774
Runtime: 30.301 min
```

We note that the LOOIC model selection criterion for the nonspatial version of this model was 3238.539.

```
fm2
```

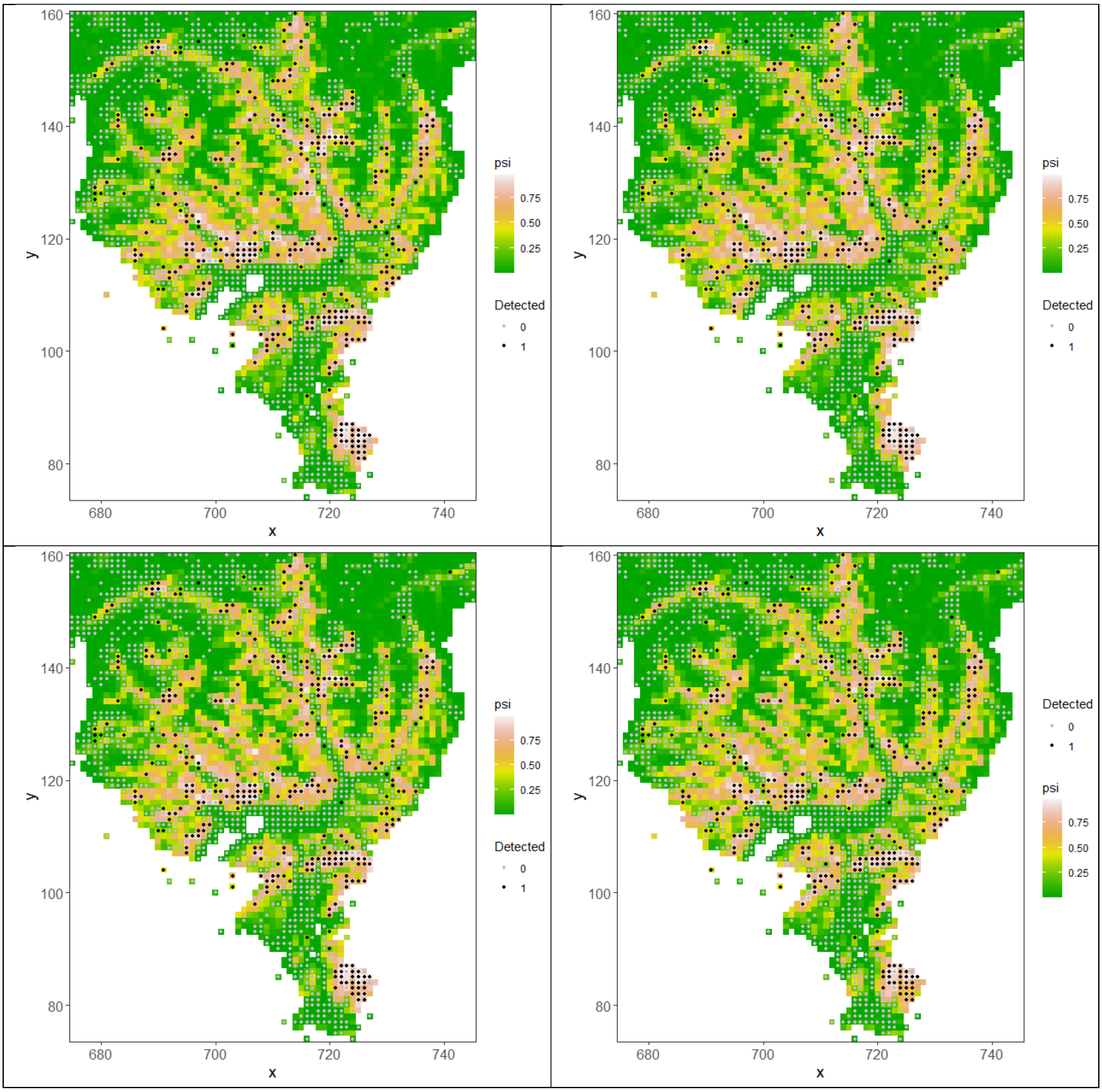
All spatial models have a lower LOOIC, showing they have a better predictive performance that does the nonspatial variant of the model.

We go on and make and then plot the predictions from both the nonspatial model (where we repeat some of what we did in Section 3.3) and of all four spatial models.

```
# Repeat predictions from nonspatial model fm2X
cbind(names(CovarData))      # not shown
newdata <- data.frame(CovarData[20:53])
pred.ubms <- ubms::predict(fm2, newdata = newdata, submodel = 'state')
str(pred.ubms)
head(pred.ubms)
```

To plot predictions from a spatial model in `ubms`, there is the handy function `plot_spatial()`. This produces predictions from the spatial model and overlays the observed, site-level detection/nondetection data. This yields a good visual impression of the fit of the model.

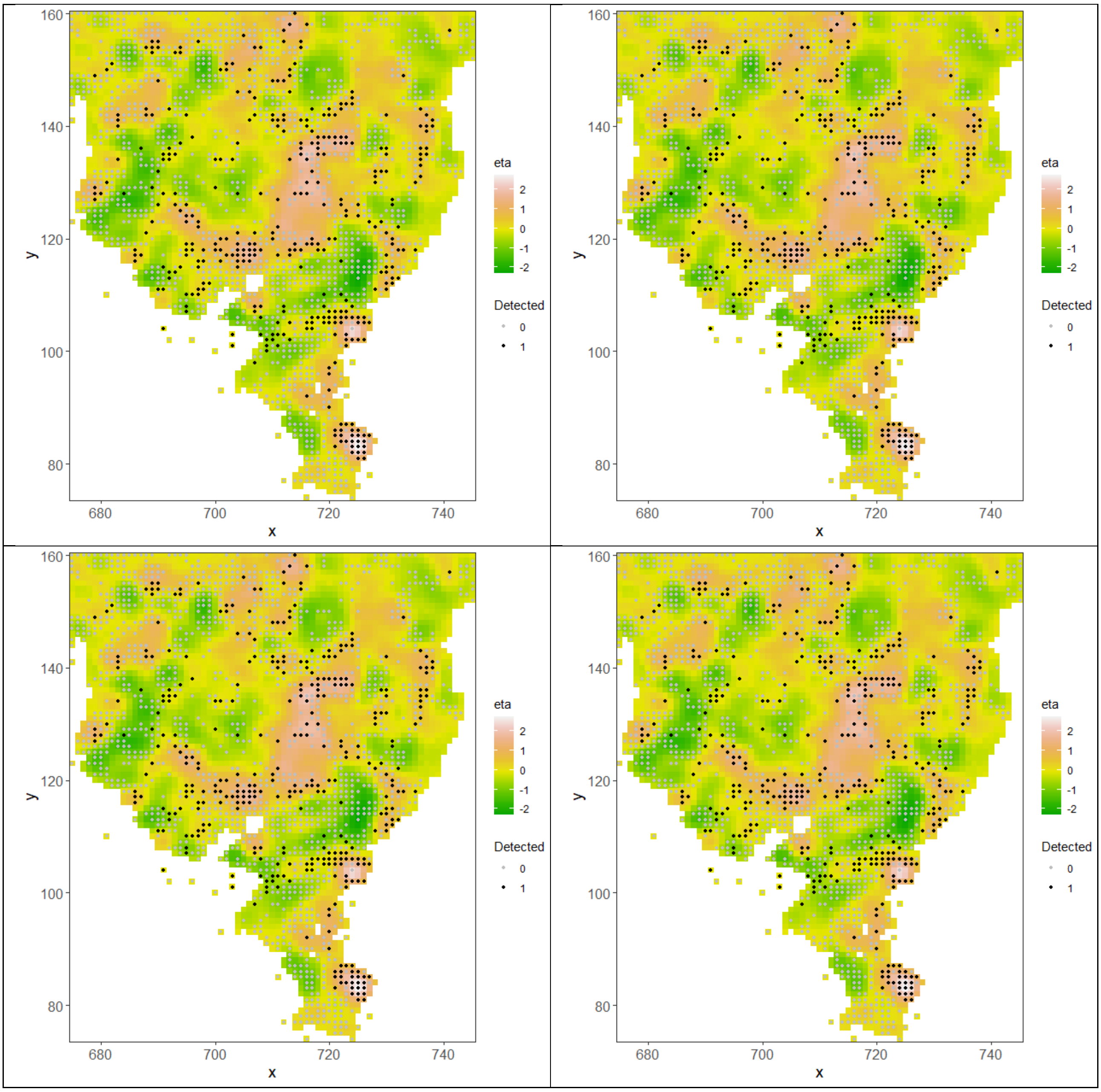
```
# Use plot_spatial()
# par(mfrow = c(2, 2))     # Does not work
plot_spatial(fm51)        # top left in 2x2 plot: 1.5
plot_spatial(fm52)        # top right: 5
plot_spatial(fm53)        # bottom left: 10
plot_spatial(fm54)        # bottom right: 20
```



Do same with the spatial effects.

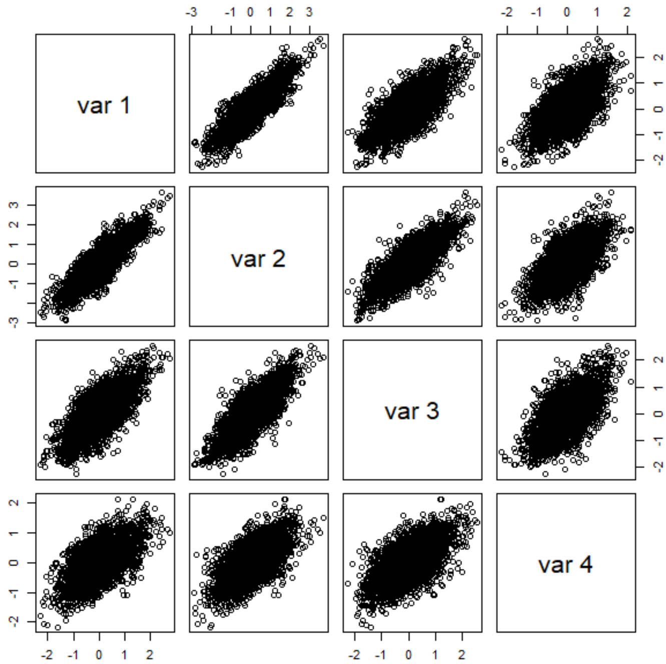
```
# Use plot_spatial() for the spatial random field in the models
tmp1 <- plot_spatial(fm51, "eta") # top left in 2x2 plot: 1.5
tmp2 <- plot_spatial(fm52, "eta")        # top right: 5
```

```
tmp3 <- plot_spatial(fm53, "eta")       # bottom left: 10
tmp4 <- plot_spatial(fm54, "eta")       # bottom right: 20
```



Are these any different at all ???

```
# Predictions from spatial models fm51 - fm54
pairs(cbind(tmp1$data$est, tmp2$data$est, tmp3$data$est, tmp4$data$est))
# OK, they are…
```



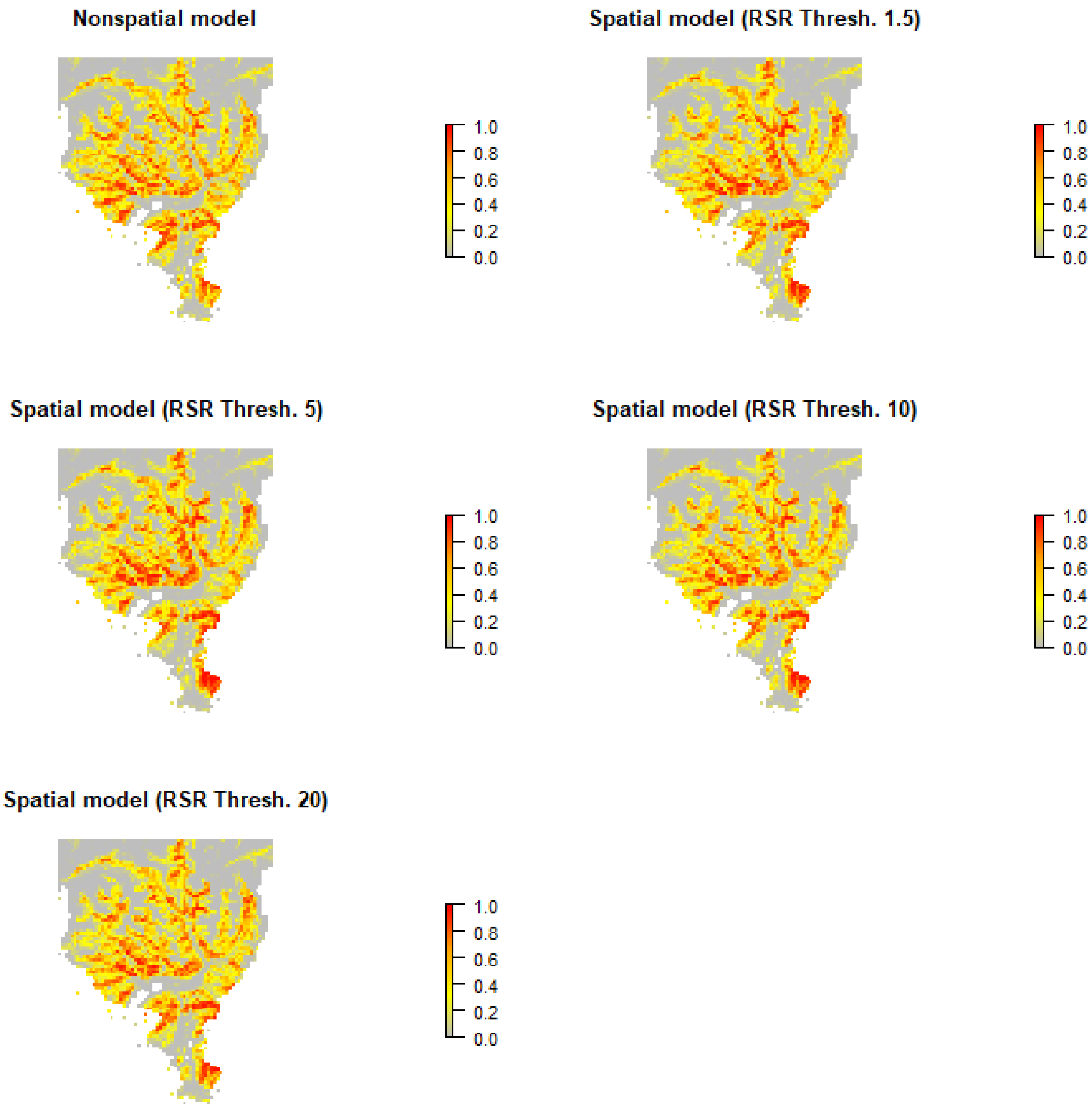
However, we also want to obtain the same predictions directly and then make our own plots, to compare directly with plots of predictions from `spOccupancy`.

```
# Predictions from spatial models fm51 - fm54
pred.fm51 <- ubms::predict(fm51, submodel = 'state')
pred.fm52 <- ubms::predict(fm52, submodel = 'state')
pred.fm53 <- ubms::predict(fm53, submodel = 'state')
pred.fm54 <- ubms::predict(fm54, submodel = 'state')
```

Rock Bunting species distribution map in SE Switzerland from `ubms`:

```
par(mfrow = c(3, 2), mar = c(2,1,4,5), cex.main = 1.3)
# Nonspatial model
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.ubms[,1]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Nonspatial model", zlim = c(0, 1))

# Spatial model with RSR threshold 1.5
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.fm51[,1]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial model (RSR Thresh. 1.5)", zlim = c(0, 1))

# Spatial model with RSR threshold 5
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.fm52[,1]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial model (RSR Thresh. 5)", zlim = c(0, 1))

# Spatial model with RSR threshold 10
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.fm53[,1]))
```

```
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial model (RSR Thresh. 10)", zlim = c(0, 1))

# Spatial model with RSR threshold 20
r1 <- rasterFromXYZ(data.frame(x = CovarData$x, y = CovarData$y,
  z = pred.fm54[,1]))
plot(r1, col = mapPalette1(100), axes = FALSE, box = FALSE, main =
"Spatial model (RSR Thresh. 20)", zlim = c(0, 1))
```



These look fairly similar.