The `forestecology` R package for modeling interspecies competition between trees

Albert Y. Kim *
Program in Statistical & Data Sciences, Smith College
and
David Allen
Biology Department, Middlebury College
and
Simon P. Couch
Mathematics Department, Reed College

February 16, 2021

Abstract

Move abstract below here after completed.

Keywords: forest ecology, competition, R, Rstats, tidyverse, sf, cross-validation,

*Albert Y. Kim is Assistant Professor, Statistical & Data Sciences, Smith College, Northampton, MA 01063 (e-mail: `akim04@smith.edu`).

Abstract (350 words)

1. When modeling growth of trees forest ecologists often incorporate the effect of interspecies competition. Many such models are based on a neighborhood effect assumption whereby all trees within a fixed distance of all focal trees are considered competitors. Methods are needed to evaluate the effect of interspecies competition and to assess their quality.
2. We present the `forestecology` package providing methods for both 1) evaluating the out-of-sample performance of our model using spatial-crossvaliation and 2) testing a null hypothesis of of no impact of competitor species' identity on the growth of trees using a permutation test. We implement a class and methods using R's S3 object-oriented system, for a specific linear, Bayesian neighborhood competition model of tree growth.
3. We demonstrate the package's functions using data from the Smithsonian Conservation Biology Institute's large forest dynamics plot, part of the ForestGEO network of reseach sites. Given ForestGEO's data collection protocols and data formatting standards, the package cross-compatibility of code. We show both that 1) competitor species identity matters and 2) that not spatially cross-validating leads to error estimates that are overly optimistic.
4. The package follows `tidyverse`-like structure whereby verb-named functions can be modularly "piped" in sequence to intuitively display the sequence of steps of analysis from start to finish. Additionally, most inputs/outputs of functions assume an are of `sf` class from the simple features package, thereby facilitating all wrangling and visualization of geospatial data. Lastly, even though our package is currently limited to one specific model, the package is setup such that it can be easily extended to other models.

1 Introduction

Repeat-censused forest plots offer excellent data to test neighborhood models of tree competition Allen & Kim (2020) Canham et al. (2006) Uriarte et al. (2004). Here we describe

an R package, `forestecology`, to do that. This package implements the methods in Allen & Kim (2020). It provides: a convenient way to specify and fit models of tree growth based on neighborhood competition; a spatial cross validation method to test and compare model fits Roberts et al. (2017); and an ANOVA-like method to assess whether the competitor identity matters in these models. The model is written to work with ForestGEO plot data Anderson-Teixeira et al. (2015), but we envision that it could easily be modified to work with data from other forest plots, e.g. the US Forest Service Forest Inventory and Analysis plots Smith (2002).

The `forestecology` is designed with "tidy" data principles in mind as Wickham et al. (2019).

Given that our data is of geo-spatial nature, we represent our data using the "simple features" `sf` package class of objects Pebesma (2018) whereby. While previously the `sp` package serves such purposes Pebesma & Bivand (2005), the `sf` package is designed to interface with the `tidyverse` suite of packages.

1.1 Model

Describe model specifics.

2 Example

We demonstrate the `forestecology` package's features on the Smithsonian Conservation Biology Institute (SCBI) large forest dynamics plot, located at the Smithsonian's National Zoo and Conservation Biology Institute in Front Royal, VA, USA. The 25.6 ha (640 x 400 m) plot is located at the intersection of three of the major physiographic provinces of the eastern US: the Blue Ridge, Ridge and Valley, and Piedmont provinces and is adjacent to the northern end of Shenandoah National Park. The forest type is typical mature secondary eastern mixed deciduous forest, with a canopy dominated by tulip poplar (*Liriodendron tulipifera*), oaks (*Quercus* spp.), and hickories (*Carya* spp.), and an understory composed mainly of spicebush (*Lindera benzoin*), paw-paw (*Asimina triloba*), American hornbeam (*Carpinus caroliniana*), and witch hazel (*Hamamelis virginiana*) Bourg et al. (2013).

A high-level overview of the steps of our analysis pipeline is as follows:

1. Compute the growth of trees based on census data
2. Add spatial information:
    1. Define buffer region trees.
    2. Add spatial cross-validation block information.
3. Identify all focal and corresponding competitor trees.
4. Fit model and make predictions.
5. Additionally: Evaluate model performance using spatial cross-validation.
6. Additionally: Evaluate the effect of competitor species identity using permutation tests.

We load all necessary packages.

```
library(tidyverse)
library(lubridate)
library(sf)
library(forestecology)
library(blockCV)
```

2.1 Compute the growth of trees based on census data

The first step in the our analysis sequence is to compute the growth of trees using data from two censuses. The `compute_growth()` function computes growth assuming census data that follows ForestGEO standards. Despite such standards, minor variations will still exist between sites thereby necessitating some data wrangling and checking. For example, the SCBI site records all DBH's in millimeters, whereas the Michigan Big Woods site records them in centimeters Anderson-Teixeira et al. (2015) Allen et al. (2020). The data format of other sites may be such that our `compute_growth()` function doesn't work at all. However, in the end all that matters is that the growth of all trees is saved in a data frame of class `sf` whereby the geolocation of each tree is presented in a `geometry` variable of type <`POINT`> and at a minimum the data contains the following variables: a variable

uniquely identifying each tree-stem, `sp` of type `fct` factor identifying species, `dbh1` and `dbh2` of type `dbl` quantifying the DBH at earlier and later census, and `growth` of type `dbl` double quantifying the average annual growth in centimeters.

We load both 2008 and 2014 SCBI census data `.csv` files as they existed on GitHub on November 20, 2020. After selecting only relevant variables, we perform a few additional data wrangling steps: convert the variable with the date of measurement to be of type `date`, convert DBH to be in centimeters[1], convert the `sp` variable containing species information from type `chr` character to `fct` factor (we will discuss the need for this in Section 2.6). Furthermore, in order to speed up computation for purposes of this example, we only consider a 9 ha subsection of the 25.6 ha of the SCBI site: `gx` from 0–300 instead of 0–400 and `gy` from 300–600 instead of 0–640.

```
census_2013_scbi <- read_csv("scbi.stem2.csv") %>%
  select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status) %>%
  mutate(
    date = mdy(date),
    dbh = as.numeric(dbh)/10,
    sp = factor(sp)
  ) %>%
  filter(gx < 300, between(gy, 300, 600))

census_2018_scbi <- read_csv("scbi.stem3.csv") %>%
  select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status) %>%
  mutate(
    date = mdy(date),
    dbh = as.numeric(dbh)/10,
    sp = factor(sp)
  ) %>%
```

[1]A rule of thumb to determine the units of DBH is check if the smallest non-zero and non-missing measurement is 1 or 10. If the former, then centimeters. If the later, then millimeters. This is because ForestGEO protocols state that only trees with DBH greater or equal to 1cm should be included in censuses.

```
  filter(gx < 300, between(gy, 300, 600))
```

These two data frames are then used as the two primary arguments to the `compute_growth()` function, along with the `id` argument whereby the user specifies the name of the variable that uniquely identifies each tree-stem under consideration (note this does not include resprouts in the later census):

```
growth_scbi <-
  compute_growth(
    census_1 = census_2013_scbi,
    census_2 = census_2018_scbi %>% filter(!str_detect(codes, "R")),
    id = "stemID"
  )
growth_scbi
## Simple feature collection with 7954 features and 8 fields
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: 0.2 ymin: 300 xmax: 299.9 ymax: 600
## CRS:            NA
## # A tibble: 7,954 x 9
##   stemID sp     dbh1 codes1 status  dbh2 codes2 growth     geometry
##    <dbl> <fct> <dbl> <chr>  <chr>  <dbl> <chr>   <dbl>      <POINT>
## 1      4 nysy  13.6  M      A       14.2 M       0.103 (14.2 428.5)
## 2      5 havi   8.8  M      A        9.6 M;P     0.150  (9.4 436.4)
## 3      6 havi   3.25 NULL   A        4   M       0.140    (1.3 434)
## 4     77 qual  65.2  M      A       66   M       0.141 (34.7 307.2)
## 5     79 tiam  47.7  M      A       46.8 M      -0.161   (40 381.1)
## # ... with 7,949 more rows
```

The output `growth_scbi` is a single data frame of class `sf` that includes a numerical `growth` reflecting the average annual growth in DBH (in cm) for all trees that were alive at both time points as well a `geometry` variable encoding each tree's geolocation. Furthermore,

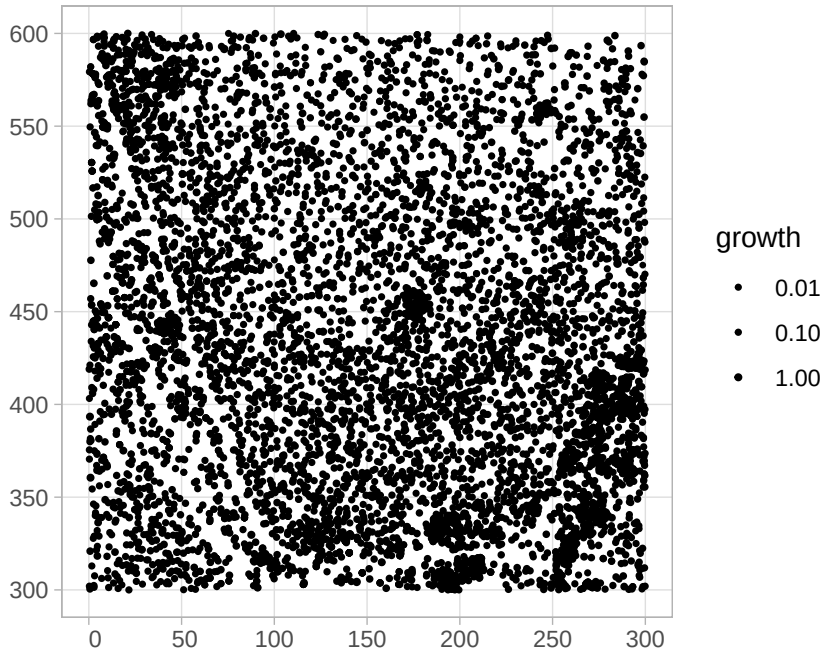

Figure 1: Growth of trees at SCBI.

variables that (in theory) remain unchanged between censuses appear only once, such as location variables `gx` and `gy`; as well as species-related variables. Variables that should change between censuses are suffixed with `1` and `2` indicating the earlier and later censuses, such as `dbh1/dbh2` and `codes1/codes2`.

Given that `growth_scbi` is of class `sf`, it can be easily plotted in `ggplot2` using the `geom_sf()` geometry as seen in Figure 1.

TODO: Rescale points in this plot:

```
ggplot() +
  geom_sf(data = growth_scbi, aes(size = growth)) +
  scale_size(breaks = c(0.01, 0.1, 1), range = c(0.1, 1))
```

2.2 Add spatial information

The next step in our analysis sequence is to add spatial information to our main `growth_scbi` data frame. The first element of spatial information we add is a "buffer region" to the periphery of the study region. Since some of our model's explanatory variables such as

competitor basal area are cumulative, we must ensure that all trees being modeled are not biased to have different neighbor structures. This is of particular concern for trees at the boundary of study regions, which will not have the same number of neighbors that act as competitors as trees in the internal part of the study region. In order to account for such edge effects only trees who are not part of this buffer region, i.e. are part of the interior of the study region, will have their growths modeled Waller & Gotway (2004).

Our model of interspecific competition relies on a spatial definition of who the competitor trees are for focal trees of interest: all trees within a distance `comp_dist` of a focal tree are considered its competitors (assuming the same units as the `gx` and `gy` location variables). In our case we set this value below at 7.5m, a value informed by Canham et al. (2004) Uriarte et al. (2004) Canham et al. (2006). Using this value along with a manually constructed `sf` object representation of the study region's boundary, we apply the `add_buffer_variable()` to our `growth_scbi` data frame to add a `buffer` boolean variable: all trees who have `buffer` set to `FALSE` will be our focal trees whose growths are modeled, whereas those with `buffer` set to `TRUE` will only be considered as competitor trees whose growth will not be modeled.

```
# Define buffer region using competitive distance range
comp_dist <- 7.5

study_region_scbi <- tibble(
  x = c(0, 300, 300, 0, 0),
  y = c(300, 300, 600, 600, 300)
) %>%
  sf_polygon()

growth_scbi <- growth_scbi %>%
  add_buffer_variable(size = comp_dist, region = study_region_scbi)
```

The second element of spatial information are blocks corresponding to folds of a spatial cross-validation algorithm used to estimate model error. Conventional cross-validation algorithms assign observations to folds by randomly resampling individual observations.

However, underlying this algorithm is an assumption that the observations are independent. In the case of forest census data, observations exhibit spatial autocorrelation. This spatial dependence is incorporated into the cross-validation algorithm by randomly resampling spatial blocks of trees Roberts et al. (2017) Pohjankukka et al. (2017). We therefore associate each observed tree to one of k spatial folds. In the example below, we first manually define two folds that partition the study region as an `sf` object. We then use the output of the `spatialBlock()` function from the `blockCV` package to associate each tree in `growth_scbi` to the correct fold (saved in the `foldID` variable) Valavi et al. (2019). \footnote{In the Appendix we present an example where the folds themselves are also created using the `spatialBlock()` function given a specified `cv_block_size`.}

```
# Manually define spatial blocks to act as folds
fold1 <- rbind(c(0, 300), c(150, 300), c(150, 600), c(0, 600))
fold2 <- rbind(c(150, 300), c(300, 300), c(300, 600), c(150, 600))

blocks_scbi <- bind_rows(sf_polygon(fold1), sf_polygon(fold2)) %>%
  mutate(folds = c(1, 2) %>% factor())

# Associate each observation to a fold
SpatialBlock_scbi <- spatialBlock(
  speciesData = growth_scbi, k = 2, selection = "systematic",
  blocks = blocks_scbi, showBlocks = FALSE, verbose = FALSE
)

growth_scbi <- growth_scbi %>%
  mutate(foldID = SpatialBlock_scbi$foldID %>% factor())
```

Figure 2 illustrates the net effect of adding these two elements of information to the `growth_scbi` data frame. The location of each tree is marked with an integer indicating which fold it belongs to, where the folds are marked with solid lines. The color of each digit indicates whether the tree is part of the buffer region (and thus will only be considered as a competitor tree in our model) or is part of the interior of the study region (and thus is a

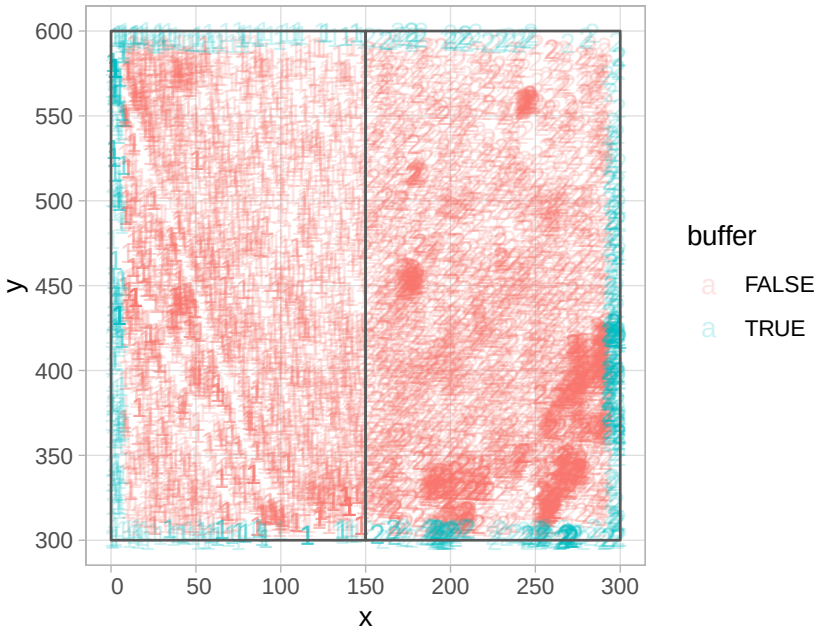

Figure 2: Buffer region and cross-validation block information for SCBI data.

focal tree whose growth is of modeled interest).

```
ggplot() +
  geom_sf_text(data = growth_scbi, aes(label = foldID, col = buffer),
               alpha = 0.2) +
  geom_sf(data = blocks_scbi, fill = "transparent")
```

TODO: Do we talk about `add_buffer_variable(direction = "out")` when each fold gets its turn being the training data?

2.3 Identify all focal and corresponding competitor trees

The next step in our analysis sequence is to identify all focal trees and their corresponding competitor trees. The `create_focal_vs_comp()` functions performs these tasks and returns a new data frame of type `sf` containing this information. On top of the previously discussed arguments `comp_dist` defining the competition neighborhood and `id` indicating which variable in the data frame uniquely identifies each tree-stem, this function also requires an `sf` object representation of the spatial cross-validation blocks/folds; in our case,

this was manually encoded in the `blocks_scbi` in Section 2.2 while in our Appendix we present an example where this was performed using `spatialBlock()` from the `blockCV` package. We present the resulting data frame below with the `foldID` variable omitted for compactness of presentation.

```
focal_vs_comp_scbi <- growth_scbi %>%
  create_focal_vs_comp(comp_dist, cv_grid_sf = blocks_scbi, id = "stemID")
focal_vs_comp_scbi %>%
  select(-foldID)
## # A tibble: 6,296 x 6
##   focal_ID focal_sp   dbh     geometry growth comp
##      <dbl> <fct>    <dbl>      <POINT>  <dbl> <list>
## 1        4 nysy     13.6  (14.2 428.5)  0.103 <tibble [20 x 4]>
## 2        5 havi      8.8   (9.4 436.4)  0.150 <tibble [32 x 4]>
## 3       79 tiam     47.7    (40 381.1) -0.161 <tibble [20 x 4]>
## 4       80 caca      5.15 (38.7 421.7)  0.253 <tibble [12 x 4]>
## 5       96 libe      2.3      (60 310)  0.262 <tibble [14 x 4]>
## # ... with 6,291 more rows
```

TODO: Below reconcile the number of rows as they off by one from `growth_scbi` pipe `filter(!is.na(growth) & !buffer)`. perhaps by removing NA's in the `growth_scbi` stage.

The resulting data frame `focal_vs_comp_scbi` has 6296 rows, representing the subset of the 7954 trees in `growth_scbi` that will be considered as focal trees and thus have their growths modeled. Recall from Section 2.2 this consists all trees that are not part of the buffer region in Figure 2. Two new variables `focal_ID` and `focal_sp` related to tree-stem identification and species information. Most notably however is a new variable `comp` which contains information on all competitor trees for a given focal tree saved in list-column format, a feature of the `tidyr` package Wickham (2020). For example, we drill-down on the tree with `focal_ID` equal to 4. It has 20 competitor trees each described by 4 variables as indicated by the fact that `comp` is a <`tibble [20 × 4]`>.

```
focal_vs_comp_scbi %>%
  filter(focal_ID == 4) %>%
  select(focal_ID, dbh, comp)
## # A tibble: 1 x 3
##   focal_ID   dbh comp
##      <dbl> <dbl> <list>
## 1        4  13.6 <tibble [20 x 4]>
```

Using the `unnest()` function from the `tidyr` package, we can flatten list-column into regular columns. We observe that for the same focal tree with DBH equal to 13.65cm, we have information on all 20 competitor trees whose `dist` distance to the focal tree is less than or equal to 7.5, including their unique tree-stem ID number, their species, and their basal area (in m^2) calculated as $\frac{\pi \times (\text{DBH}/2)^2}{10000}$ where DBH is the value from the earlier census in cm. Saving our focal versus competitor information in list-column minimizes redundancy since we do not repeat information on the focal tree 20 times.

```
focal_vs_comp_scbi %>%
  filter(focal_ID == 4) %>%
  select(focal_ID, dbh, comp) %>%
  unnest(cols = "comp")
```

```
## # A tibble: 20 x 6
##   focal_ID   dbh comp_ID  dist comp_sp comp_basal_area
##      <dbl> <dbl>   <dbl> <dbl> <fct>             <dbl>
## 1        4  13.6    1836  7.48 tiam            0.0176
## 2        4  13.6    1847  2.81 nysy            0.00332
## 3        4  13.6    1848  1.62 nysy            0.00396
## 4        4  13.6    1849  2.62 nysy            0.00535
## 5        4  13.6    1850  2.98 havi            0.00472
## # ... with 15 more rows
```

2.4 Fit model and make predictions

HEY BERT PICK IT UP HERE

Next we fit the following linear model to the DBH of each focal tree. Let $i = 1, \ldots, n_j$ index all n_j trees of "focal" species group j; let $j = 1, \ldots, J$ index all J focal species groups; and let $k = 1, \ldots, K$ index all K "competitor" species groups. We modeled the growth in diameter per year y_{ij} (in centimeters per year) of the i^{th} tree of focal species group j as a linear model f of the following covariates $\vec{x}_{ij}$

$$y_{ij} = f(\vec{x}_{ij}) + \epsilon_{ij} = \beta_{0,j} + \beta_{\text{DBH},j} \cdot \text{DBH}_{ij} + \sum_{k=1}^{K} \lambda_{jk} \cdot \text{BA}_{ijk} + \epsilon_{ij}$$

We estimate the model's parameters using Bayesian linear regression implemented in the `fit_bayesian_model()` function. TODO: define all parameters

For this linear model's case, there exists a closed form solution as described here. As such, the `fit_bayesian_model()` function using matrix algebra to obtain all parameter estimates, rather than computationally expensive Monte Carlo approximations. The inputs to this function are a `focal_vs_comp` data frame, `prior_param` a list of priors, and a boolean flag `run_shuffle` on whether or not to run competitor-species identity permutations which we will demonstrate below on the Michigan Big Woods data. This function returns the posterior means of all parameters.

Using these posterior means, we then use the posterior predictive distribution to obtain fitted/predicted values $\widehat{y}$ of the DBH for each focal tree using the `predict_bayesian_model()`. These $\widehat{y}$ can then be compared to the observed y DBH's to compute the root mean-square error, a measure of a model's predictive error which has the same units as the observed data y.

2.4.1 Big Woods

For the Michigan Big Woods data we present two use cases of the model fitting and prediction scheme. The first use case is the simplest where we assess the fit of the model using root mean squared error. The second use case then answers the question of whether species competitor identity matters using permutation test.

For the first use case, we fit the linear model specified in Equation XXX to our data frame of type `focal_vs_comp`. This input/outputs of the `fit_bayesian_model()` function are lists of the prior/posterior means of parameters of the linear regression specified in XXX. Generally speaking, there are two classes of regression parameters: β main effects and λ competitive effects. In the upcoming Section 2.5, we will present code visualizing this posterior distributions.

```
comp_bayes_lm_bw <- focal_vs_comp_bw %>%
  comp_bayes_lm(prior_param = NULL)
```

This output of posterior parameters for the specified competition model are then used along with the posterior predictive distribution encoded in `predict_bayesian_model()` to return predicted growths for each individual tree. We join these predicted growths to the original growth data frame.

```
focal_vs_comp_bw <- focal_vs_comp_bw %>%
  mutate(growth_hat = predict(comp_bayes_lm_bw, focal_vs_comp_bw))
```

We then use the `rmse()` function from the `yardstick` package to obtain the root mean squared error of the observed versus fitted values of growth.

```
focal_vs_comp_bw %>%
  rmse(truth = growth, estimate = growth_hat) %>%
  pull(.estimate)
```

The second use case is near identical to the first, but with a small change in the code to test whether the identity of the competitor matters. By adding a `run_shuffle = TRUE` argument to `fit_bayesian_model()`, for each focal tree its competitor trees' species identity will be "shuffled" randomly much like in a permutation test. By shuffling these species labels we are effectively fitting the model under a null model that competitor species identity does not matter. If the "shuffled" RMSE's are consistently lower than the unshuffled RMSE corresponding to the observed data, then we have evidence to suggest that competitor identity matters to competitive interactions.

```
comp_bayes_lm_bw_shuffle <- focal_vs_comp_bw %>%
  comp_bayes_lm(prior_param = NULL, run_shuffle = TRUE)
```

```
focal_vs_comp_bw <- focal_vs_comp_bw %>%
  mutate(growth_hat_shuffle = predict(comp_bayes_lm_bw_shuffle, focal_vs_comp_bw))
```

```
focal_vs_comp_bw %>%
  rmse(truth = growth, estimate = growth_hat_shuffle) %>%
  pull(.estimate)
```

The RMSE is fact lower for the non-shuffled version, indicative of a better model fit. This gives support for the idea that competitor identity does matter for competitive interactions. In Allen & Kim (2020) we run this shuffle a large number of times to construct a full permutation distribution to show that this difference is robust to resampling variation.

2.4.2 SCBI

In the case of the SCBI data, we once again perform the same model fitting and computing of fitted growths as with the Big Woods data, but this time we map the residuals of the observed minus fitted values to look for spatial patterns.

```
comp_bayes_lm_scbi <- focal_vs_comp_scbi %>%
  comp_bayes_lm(prior_param = NULL)
```

```
focal_vs_comp_scbi <- focal_vs_comp_scbi %>%
  mutate(growth_hat = predict(comp_bayes_lm_scbi, focal_vs_comp_scbi))
```

```
focal_vs_comp_scbi %>%
  rmse(truth = growth, estimate = growth_hat) %>%
  pull(.estimate)
```

In Figures **??** and **??** we present the residuals.

2.5 Visualize model results

Lastly, we return to the model fits from Section 2.4 and present tools to visually explore the posterior distributions of all parameters in our model. There are two main groups of parameters to consider. The β coefficients tell us about how fast each species grows and how this depends on DBH while the full matrix of λ values describe the competitive effects between pairs of species. There is a rich literature on this matrix (cite).

DO WE NEED TO DESCRIBE MECHANICS? Because of the structure of the `bw_fit_model` object we cannot simply draw these curves based on the posterior distribution. `bw_fit_model()` gives the parameters *compared* to a baseline. This is not of direct interest. So to display these parameters, as we care about them, we have to sample from the baseline distribution and from the comparison one to get the posterior distribution of interest.

2.5.1 Big Woods

Here we re-run the model fit to the Big Woods data from Section 2.4, but this time use "family" as the group for comparison which has. This makes the posterior distributions easier to follow. Also, surprisingly, grouping by family performed just as well as grouping by species Allen & Kim (2020). First we re-run `create_focal_vs_comp()` and `fit_bayesian_model()` with no permutation shuffling with the grouping variable as family.

```
focal_vs_comp_bw <- growth_bw %>%
  mutate(sp = family %>% factor()) %>%
  create_focal_vs_comp(comp_dist = comp_dist, cv_grid_sf = blocks_bw, id = "treeID")


comp_bayes_lm_bw <- focal_vs_comp_bw %>%
  comp_bayes_lm(prior_param = NULL)
```

Now the posterior parameter outputs of `fit_bayesian_model()` are passed to `plot_bayesian_model_pa` to generate visualizations of the posterior parameters. These visualizations are displayed in Figure 5 of Allen & Kim (2020). For simplicity we only plot a subset of the species families.

```
sp_to_plot <- c("cornaceae", "fagaceae", "hamamelidaceae", "juglandaceae",
                "lauraceae", "rosaceae", "sapindaceae", "ulmaceae")
```

The output is a list with three plots stored. Figure **??** The element `beta_0` gives the baseline growth intercept β_0, i.e., how fast an individual of each group grows independent of DBH).

```
plot1 <- autoplot(comp_bayes_lm_bw, type = "intercepts")
plot1
```

Figure **??** Next `beta_dbh` gives the slope for DBH slope $\beta_{dbh,i}$ for each group.

```
plot2 <- autoplot(comp_bayes_lm_bw, type = "dbh_slopes")
plot2
```

Finally Figure **??** `lambda` gives the competition coefficients λ.

```
plot3 <- autoplot(comp_bayes_lm_bw, type = "competition")
plot3
```

2.5.2 SCBI

We revisit the posterior parameters for the SCBI from Section {model-fit-predict}, but this time only focus on the λ competition coefficients.

```
sp_to_plot <- c("quru", "litu", "cagl", "cato")
```

```
plot3 <- autoplot(comp_bayes_lm_bw, type = "competition")
plot3
```

Add explanation here.

HEY BERT PICK IT UP HERE

2.6 Run spatial cross-validation

The model fits and predictions in Section 2.4 all suffer from a common failing: they use the same data to both fit the model and to assess the model's performance using the RMSE. As argued by Roberts et al. (2017), this can lead to overly optimistic assessments of model quality as the models can be overfit, in particular in situations where spatial-autocorrelation is present. To mitigate the effects of such overfitting, we use a spatially block cross-validation algorithm implemented in the `run_cv()`. This function at its core uses the same model fitting implemented in the `fit_bayesian_model()` function, however trains the model on $k - 1$ spatial folds of the train and returns fitted values for the test data. Recall that the spatial blocking scheme wass encoded in Section 2.2.

2.6.1 Big Woods

Applying this spatially cross-validated model fit yields an RMSE is higher than that when the model is fit without cross validation. In other words, our model fits in 2.4 were overly optimistic in the model's fitting power, whereas a cross-validated results yield an estimate that is closer to the truth. See Allen & Kim (2020) for more discussion of this.

```
focal_vs_comp_bw <- focal_vs_comp_bw %>%
  run_cv(comp_dist = comp_dist, cv_grid = blocks_bw)


focal_vs_comp_bw %>%
  rmse(truth = growth, estimate = growth_hat) %>%
  pull(.estimate)
```

2.6.2 SCBI

Observe once again that this RMSE is much higher than that for the above SCBI model fit without cross-validation.

```
focal_vs_comp_scbi <- focal_vs_comp_scbi %>%
  run_cv(comp_dist = comp_dist, cv_grid = blocks_scbi)
```

```
focal_vs_comp_scbi %>%
  rmse(truth = growth, estimate = growth_hat) %>%
  pull(.estimate)
```

3 Discussion

4 Acknowledgments

References

Allen, D., Dick, C., Burnham, R. J., Perfecto, I. & Vandermeer, J. (2020), 'The michigan big woods research plot at the edwin s. george, pinckney, mi, usa', *Miscellaneous Publications of the Museum of Zoology, University of Michigan* **207**.
URL: *http://hdl.handle.net/2027.42/156251*

Allen, D. & Kim, A. Y. (2020), 'A permutation test and spatial cross-validation approach to assess models of interspecific competition between trees', *PLOS ONE* **15**(3), e0229930. Publisher: Public Library of Science.
URL: *https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0229930*

Anderson-Teixeira, K. J., Davies, S. J., Bennett, A. C., Gonzalez-Akre, E. B., Muller-Landau, H. C., Wright, S. J., Salim, K. A., Zambrano, A. M. A., Alonso, A., Baltzer, J. L., Basset, Y., Bourg, N. A., Broadbent, E. N., Brockelman, W. Y., Bunyavejchewin, S., Burslem, D. F. R. P., Butt, N., Cao, M., Cardenas, D., Chuyong, G. B., Clay, K., Cordell, S., Dattaraja, H. S., Deng, X., Detto, M., Du, X., Duque, A., Erikson, D. L., Ewango, C. E. N., Fischer, G. A., Fletcher, C., Foster, R. B., Giardina, C. P., Gilbert, G. S., Gunatilleke, N., Gunatilleke, S., Hao, Z., Hargrove, W. W., Hart, T. B., Hau, B. C. H., He, F., Hoffman, F. M., Howe, R. W., Hubbell, S. P., Inman-Narahari, F. M., Jansen, P. A., Jiang, M., Johnson, D. J., Kanzaki, M., Kassim, A. R., Kenfack, D., Kibet, S., Kinnaird, M. F., Korte, L., Kral, K., Kumar, J., Larson, A. J., Li, Y., Li, X.,

Liu, S., Lum, S. K. Y., Lutz, J. A., Ma, K., Maddalena, D. M., Makana, J.-R., Malhi, Y., Marthews, T., Serudin, R. M., McMahon, S. M., McShea, W. J., Memiaghe, H. R., Mi, X., Mizuno, T., Morecroft, M., Myers, J. A., Novotny, V., Oliveira, A. A. d., Ong, P. S., Orwig, D. A., Ostertag, R., Ouden, J. d., Parker, G. G., Phillips, R. P., Sack, L., Sainge, M. N., Sang, W., Sri-ngernyuang, K., Sukumar, R., Sun, I.-F., Sungpalee, W., Suresh, H. S., Tan, S., Thomas, S. C., Thomas, D. W., Thompson, J., Turner, B. L., Uriarte, M., Valencia, R., Vallejo, M. I., Vicentini, A., Vrška, T., Wang, X., Wang, X., Weiblen, G., Wolf, A., Xu, H., Yap, S. & Zimmerman, J. (2015), 'CTFS-ForestGEO: a worldwide network monitoring forests in an era of global change', *Global Change Biology* **21**(2), 528–549.
URL: *http://onlinelibrary.wiley.com/doi/abs/10.1111/gcb.12712*

Bourg, N. A., McShea, W. J., Thompson, J. R., McGarvey, J. C. & Shen, X. (2013), 'Initial census, woody seedling, seed rain, and stand structure data for the SCBI SIGEO Large Forest Dynamics Plot', *Ecology* **94**(9), 2111–2112.
URL: *http://esajournals.onlinelibrary.wiley.com/doi/abs/10.1890/13-0010.1*

Canham, C. D., LePage, P. T. & Coates, K. D. (2004), 'A neighborhood analysis of canopy tree competition: effects of shading versus crowding', *Canadian Journal of Forest Research* **34**(4). Publisher: NRC Research Press Ottawa, Canada.
URL: *https://cdnsciencepub.com/doi/abs/10.1139/x03-232*

Canham, C. D., Papaik, M. J., Uriarte, M., McWilliams, W. H., Jenkins, J. C. & Twery, M. J. (2006), 'Neighborhood Analyses Of Canopy Tree Competition Along Environmental Gradients In New England Forests', *Ecological Applications* **16**(2), 540–554. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1890/1051-0761%282006%29016%5B0540%3ANAOCTC%5D2.0.CO%3B2.

Pebesma, E. (2018), 'Simple Features for R: Standardized Support for Spatial Vector Data', *The R Journal* **10**(1), 439–446.
URL: *https://journal.r-project.org/archive/2018/RJ-2018-009/index.html*

Pebesma, E. J. & Bivand, R. S. (2005), 'Classes and methods for spatial data in R', *R*

News **5**(2), 9–13.

URL: *https://CRAN.R-project.org/doc/Rnews/*

Pohjankukka, J., Pahikkala, T., Nevalainen, P. & Heikkonen, J. (2017), 'Estimating the prediction performance of spatial models via spatial k-fold cross validation', *International Journal of Geographical Information Science* **31**(10), 2001–2019.

Roberts, D. R., Bahn, V., Ciuti, S., Boyce, M. S., Elith, J., Guillera-Arroita, G., Hauenstein, S., Lahoz-Monfort, J. J., Schröder, B., Thuiller, W., Warton, D. I., Wintle, B. A., Hartig, F. & Dormann, C. F. (2017), 'Cross-validation strategies for data with temporal, spatial, hierarchical, or phylogenetic structure', *Ecography* **40**(8), 913–929.

URL: *http://onlinelibrary.wiley.com/doi/abs/10.1111/ecog.02881*

Smith, W. B. (2002), 'Forest inventory and analysis: a national inventory and monitoring program', *Environmental pollution* **116**, S233–S242.

Uriarte, M., Condit, R., Canham, C. D. & Hubbell, S. P. (2004), 'A spatially explicit model of sapling growth in a tropical forest: does the identity of neighbours matter?', *Journal of Ecology* **92**(2), 348–360. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.0022-0477.2004.00867.x.

URL: *http://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/j.0022-0477.2004.00867.x*

Valavi, R., Elith, J., Lahoz-Monfort, J. J. & Guillera-Arroita, G. (2019), 'blockCV: An r package for generating spatially or environmentally separated folds for k-fold cross-validation of species distribution models', *Methods in Ecology and Evolution* **10**(2), 225–232. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/2041-210X.13107.

URL: *http://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13107*

Waller, L. A. & Gotway, C. A. (2004), *Applied Spatial Statistics for Public Health Data*, John Wiley & Sons, Incorporated, Hoboken, UNITED STATES.

URL: *http://ebookcentral.proquest.com/lib/smith/detail.action?docID=214360*

Wickham, H. (2020), *tidyr: Tidy Messy Data*. R package version 1.1.2.

URL: *https://CRAN.R-project.org/package=tidyr*

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K. & Yutani, H. (2019), 'Welcome to the Tidyverse', *Journal of Open Source Software* **4**(43), 1686.

URL: *https://joss.theoj.org/papers/10.21105/joss.01686*