The Forestecology R Package for Fitting and Assessing Neighborhood Models of the Effect of Interspecific Competition on the Growth of Trees

Albert Y. Kim
*Smith College*, akim04@smith.edu

David N. Allen
*Middlebury College*

Simon P. Couch
*Johns Hopkins University*

Follow this and additional works at: https://scholarworks.smith.edu/sds_facpubs

Part of the Data Science Commons, Other Computer Sciences Commons, and the Statistics and Probability Commons

Recommended Citation

This Article has been accepted for inclusion in Statistical and Data Sciences: Faculty Publications by an authorized administrator of Smith ScholarWorks. For more information, please contact scholarworks@smith.edu
The forestecology R package for fitting and assessing neighborhood models of the effect of interspecific competition on the growth of trees

Albert Y. Kim¹ | David N. Allen² | Simon P. Couch³

¹Program in Statistical & Data Sciences, Smith College, Northampton, MA, USA
²Biology Department, Middlebury College, Middlebury, VT, USA
³Department of Biostatistics, Johns Hopkins University, Baltimore, MD, USA

Abstract

1. Neighborhood competition models are powerful tools to measure the effect of interspecific competition. Statistical methods to ease the application of these models are currently lacking.
2. We present the forestecology package providing methods to (a) specify neighborhood competition models, (b) evaluate the effect of competitor species identity using permutation tests, and (c) measure model performance using spatial cross-validation. Following Allen and Kim (PLoS One, 15, 2020, e0229930), we implement a Bayesian linear regression neighborhood competition model.
3. We demonstrate the package’s functionality using data from the Smithsonian Conservation Biology Institute’s large forest dynamics plot, part of the ForestGEO global network of research sites. Given ForestGEO’s data collection protocols and data formatting standards, the package was designed with cross-site compatibility in mind. We highlight the importance of spatial cross-validation when interpreting model results.
4. The package features (a) tidyverse-like structure whereby verb-named functions can be modularly "piped" in sequence, (b) functions with standardized inputs/outputs of simple features sf package class, and (c) an S3 object-oriented implementation of the Bayesian linear regression model. These three facts allow for clear articulation of all the steps in the sequence of analysis and easy wrangling and visualization of the geospatial data. Furthermore, while the package only has Bayesian linear regression implemented, the package was designed with extensibility to other methods in mind.

KEYWORDS

forest ecology, ForestGEO, interspecific competition, neighborhood competition, R, spatial cross-validation, tree growth
1 | INTRODUCTION

Repeat-censused forest plots offer excellent opportunities to test neighborhood models of the effect of competition on the growth of trees (Canham et al., 2004). Neighborhood models of competition have been used to test whether the species identity of a competitor matters Uriarte et al. (2004); measure species-specific competition coefficients (Das, 2012; Tatsumi et al., 2016); test competing models to see what structures competitive interactions, for example, traits or phylogeny (Allen & Kim, 2020; Uriarte et al., 2010); and inform selective logging practices (Canham et al., 2006). Although these are well-described methods, few methods are currently available for easy application.

We address this shortcoming with the forestecology R package providing methods and data for forest ecology model fitting and assessment, available on CRAN (https://cran.r-project.org/package=forestecology) and on GitHub (https://github.com/rudeboyber/forestecology). The package is written to model stem diameter growth between two censuses based on neighborhood competition, largely following the methods in Allen and Kim (2020).

Let \( i = 1, \ldots, n \) index all \( n \) trees of “focal” species; let \( j = 1, \ldots, J \) index all \( J \) focal species; and let \( k = 1, \ldots, K \) index all \( K \) “competitor” species. The average annual growth in diameter at breast height (DBH) \( y_{ij} \) (in centimeters/year) of the \( j \)th tree of focal species \( i \) is modeled as

\[
y_{ij} = \beta_{ij0} + \beta_{ij\text{dbh}} \cdot \text{dbh}_{ij} + \sum_{k=1}^{K} \lambda_{jk} \cdot x_{ijk} + \epsilon_{ij}
\]

(1)

where \( \beta_{ij0} \) is the diameter-independent growth rate of species \( j \); \( \text{dbh}_{ij} \) is the DBH of the focal tree at the earlier census and \( \beta_{ij\text{dbh}} \) is the slope of that species’s diameter–growth relationship; \( x_{ijk} \) is the sum of some numerical explanatory variable of all trees of competitor species \( k \), and \( \lambda_{jk} \) quantifies the corresponding change in growth for individuals of species \( j \) from these competitors; and \( \epsilon_{ij} \) is a random error term distributed Normal \( (0, \sigma^2) \).

Allen and Kim (2020) use the sum of the basal area of all trees of competitor species \( k \) as \( x_{ijk} \). Furthermore, they estimate all parameters via Bayesian linear regression, while exploiting Normal/Inverse Gamma conjugacy to derive closed-form solutions to all posterior distributions. These closed-form solutions are not as computationally expensive as approximations from Markov chain Monte Carlo algorithms.

To evaluate whether competitor species identity matters, Allen and Kim (2020) run a permutation test where a null hypothesis of no species grouping-specific effects of competition is assumed; thus, the species identity of all competitors can be permuted:

\[
H_0: \lambda_{jk} = \lambda_j \quad \text{for all} \quad k = 1, \ldots, K
\]

vs.

\[
H_1: \text{at least one } \lambda_{jk} \text{ is different}
\]

(2)

Furthermore, to account for the spatial autocorrelation in their estimates of out-of-sample model error, Allen and Kim (2020) use spatial cross-validation. Estimates of model error that do not account for this dependence tend to underestimate the true model error (Roberts et al., 2017).

The package is designed with “tidy” design principles in mind (Wickham et al., 2019). Much like all tidyverse packages, forestecology has verb-named functions that can be modularly composed using the pipe \( %>% \) operator to sequentially complete all necessary analysis steps (Bache & Wickham, 2020).

Furthermore, the inputs and outputs of most functions use the same “simple features for R” data structures for spatial data from the sf package (Bivand et al., 2013). Previously, sp package classes were commonly used for storing spatial data and interfacing with geospatial libraries (Bivand et al., 2013); the sf package aims to improve on the sp package by:

1. Using simple feature access as the base standard for representing and encoding spatial data, rather than shapefiles (Herring, 2011).
2. Leveraging improvements in external libraries for reading and writing spatial data (GDAL) and for geometrical operations (GEOS) (GEOS Development Team, 2017; Warmerdam, 2008).
3. Integrating closely with the popular tidyverse suite of packages for data science (Wickham et al., 2019).

By using the sf package classes to represent spatial data rather than the sp package, the implementation and use of the forestecology package’s spatial algorithms was greatly simplified.

2 | forestecology WORKFLOW: A CASE STUDY

We present a case study of forestecology’s functionality on data from the Smithsonian Conservation Biology Institute (SCBI) large forest dynamics plot in Front Royal, VA, USA, part of the ForestGEO global network of research sites (Anderson-Teixeira et al., 2015; Bourg et al., 2013; Davies et al., 2021). The 25.6-ha (640 × 400 m) plot is located at the intersection of three of the major physiographic provinces of the eastern United States—the Blue Ridge, Ridge and Valley, and Piedmont provinces—and is adjacent to the northern end of Shenandoah National Park.

The package has the following goals: to evaluate (a) the effect of competitor species identity using permutation tests and (b) model performance using spatial cross-validation. We outline the four-step basic analysis sequence:

1. Compute the growth of stems based on two censuses.
2. Add spatial information:
   a. Define a buffer region of trees.
   b. Add spatial cross-validation block information.
3. Identify all focal trees and their competitors.
4. Apply model, which includes:
   a. Fit model.
   b. Compute predicted values.
   c. Visualize posterior distributions.
We start by loading all packages (all code in this paper can be copied from here https://github.com/rudeboybert/forestecology/blob/master/paper/paper.R).

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

# Resolve conflicting functions
filter <- dplyr::filter
select <- dplyr::select

## 2.1 Step 1: Compute the growth of trees based on census data

We first compute the growth of trees using data from two censuses. `compute_growth()` computes the average annual growth based on census data that roughly follows ForestGEO standards. Despite such standards, minor variations will still exist between sites, thereby necessitating some data wrangling. For example, the SCBI site records all DBH values in millimeters (Bourg et al., 2013), whereas the Michigan Big Woods site used in Allen and Kim (2020) records them in centimeters (Allen et al., 2020).

We load both 2008 and 2014 SCBI census.csv files as they existed on GitHub on 2021/08/02 and perform minor data wrangling (Gonzalez-Akre, McGregor, et al., 2020). We then only consider a 9-ha subsection of the 25.6 ha of the site to speed up computation for this example: gx from 0–300 instead of 0–400 and gy from 300–600 instead of 0–640.

```r
census_2013_scbi <- read_csv("scbi_2013_scbi.csv") %>%
  select(stemID, sp, date = as.Date(date, format="Ym"), gx, gy, dbh, codes, status) %>%
  mutate(
    # Convert date from character to date
    date = mdy(date),
    # Convert dbh to be in cm
    dbh = as.numeric(dbh) / 10
  )

filter(gx < 300, between(gy, 300, 600))
census_2014_scbi <- read_csv("scbi_2014_scbi.csv") %>%
  select(stemID, sp, date = as.Date(date, format = "Ym"), gx, gy, dbh, codes, status) %>%
  mutate(
    # Convert date from character to date
    date = mdy(date),
    # Convert dbh to be in cm
    dbh = as.numeric(dbh) / 10
  )

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

These two data frames are then used as inputs to `compute_growth()`, along with id specifying the variable that uniquely identifies each tree-stem. We also discard all resprouts with `code == R` in the later census, since we are only interested in the growth of surviving, and not resprouted, stems.

```r
growth_scbi <-
  compute_growth(
    census_1 = census_2013_scbi,
    census_2 = census_2014_scbi
  )
```

The output `growth_scbi` is a data frame of class `sf` that includes among other variables the species variable `sp` converted to a factor, the average annual growth in DBH (cm·y\(^{-1}\)) for all stems that were alive at both time points, and the `sf` package's encoding of geolocations of geometry type `<POINT>`. Given that `growth_scbi` is of class `sf`, it can be easily plotted in ggplot2 using `geom_sf()` as seen in Figure 1.

```r
sp_info <- read_csv("SCBI_FORESTEO_sp_ecology.csv") %>%
  select(sp = spcode, family, genus, species, canopy_position, drought_tolerance)

left_join(sp_info, by = "sp") %>%
mutate(sp = as.factor(sp))
```

We also load species information as it existed on GitHub on 2021/08/02, which includes family, genus, and species information, and resprouts with `code == R` in the later census, since we are only interested in the growth of surviving, and not resprouted, stems.

![FIGURE 1](image-url) Step 1—Compute growth of trees based on census data. A map of the growth of a random sample of 500 trees from a 9-ha subsection of the Smithsonian Conservation Biology Institute (SCBI) forest plot.
Furthermore, we compute two potential competitor explanatory variables $x_{\text{comp}}$ from Equation 1. First, the basal area of each tree as a function of its DBH in the earlier census. Second, the aboveground biomass as estimated by allometric equations encoded in the get\_biomass() function from the allodb package (Gonzalez-Akre et al., 2020); this function has DBH, species, and geographic coordinates as arguments.

2.2 Step 2: Add spatial information

We then add spatial information to growth\_scbi. We first add a "buffer region" to the periphery of the study region. Since some of our model’s explanatory variables are cumulative, we must ensure that all trees being modeled are not biased to have different neighbor structures. This is of concern for trees at the boundary of the study region who will not have all their neighbors included in the census stems. To account for such edge effects, only trees that are not part of this buffer region, that is, are part of the interior of the study region, will have their growth modeled (Waller & Gotway, 2004).

Our model of interspecific competition relies on a spatial definition of who competitor trees are: all trees within a distance $\text{comp\_dist}$ of a focal tree. Here, we set $\text{comp\_dist}$ to 7.5m, a value informed by other studies (Canham et al., 2004; Canham et al., 2006; Uriarte et al., 2004), but the package could also be used to compare multiple distances and see which is best supported (see Appendix 1). We use $\text{comp\_dist}$ and a manually constructed sf representation of the study region’s boundary as inputs to add\_buffer\_variable() to add a buffer Boolean variable to growth\_scbi. All trees with buffer equal to FALSE will be our focal trees whose growth will be modeled, whereas those with TRUE will only act as competitor trees.

The second element of spatial information we add are blocks corresponding to folds of a spatial cross-validation algorithm. Conventional cross-validation algorithms assign individual observations to folds by randomly resampling them all while assuming they are statistically independent. In the case of forest census data however, observations exhibit spatial autocorrelation. We therefore incorporate this dependence into the cross-validation algorithm by resampling spatial blocks of trees (Pohjankukka et al., 2017; Roberts et al., 2017).

We first manually define an sf object defining four folds that partition the study region. We then use the output of the spatial\_Block() function from the blockCV package to associate each tree in growth\_scbi to the correct fold ID (Valavi et al., 2019). This foldID variable will be used in Section 2.6.

Figure 2 illustrates the net effect of adding these two elements of spatial information to growth\_scbi.
2.3 | Step 3: Identify all focal and corresponding competitor trees

We then identify all focal trees and their corresponding competitor trees and, more specifically, identify all trees that are not part of the buffer region, have a valid growth measurement, and have at least one neighbor within 7.5 m. We do this using `create_focal_vs_comp()`, which takes the previously detailed `comp_dist` and `id` arguments, the sf representation of the spatial cross-validation blocks `blocks_scbi`, and a specification `comp_x_var` of the `basal_area` variable we use as the competitor explanatory variable `x_{comp}` from Equation 1. This function returns a new data frame `focal_vs_comp_scbi`.

```
focal_vs_comp_scbi <- growth_scbi %>%
  create_focal_vs_comp(
    comp_dist, blocks = blocks_scbi, id = "stemID",
    comp_x_var = "basal_area"
  )
```

The resulting `focal_vs_comp_scbi` has 6,296 rows, representing the subset of the 7,954 trees in `growth_scbi` that will be considered as focal trees. The variables `focal_ID` and `focal_sp` relate to tree-stem identification and species information. Most notably however is the variable `comp`, which contains information on all competitor trees saved in `tidyr` package list-column format (Wickham, 2020). To inspect this information, we flatten the comp list-column for the tree with `focal_ID` 4 in the first row, here a tibble [20 x 4], into regular columns using `unnest()` from the `tidyr` package.

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

We observe 4 variables describing 20 competitor trees: the unique tree-stem ID, the distance to the focal tree (all ≤7.5 m), the species, and the basal area (in $m^2$) calculated as $\frac{\pi \times DBH^2}{4 \times 10000}$ for the DBH in cm from the earlier census. Saving competitor information in list-column format minimizes redundancy since we do not need to repeat information on the focal tree 20 times. We visualize the spatial distribution of these trees in Figure 3.

Here, we use basal area as the continuous competitor explanatory variable, but the package is flexible to allow the user to specify any competitor explanatory variable (basal area, biomass, tree height, a soil nutrient value). The package can also be used to compare competitor explanatory variables and see which best explains tree growth, see Appendix 2 for an example comparing basal area and above-ground biomass. Similarly, the package can use any categorical variable as an explanatory variable and compare between different categorical variables. For example, in Allen and Kim (2020), we compare grouping individuals based on species, family, and based on trait-based groups. In Appendix 3, we give another example and compare grouping individuals by species or by potential canopy position (canopy, understory, shrub layer).
2.4 | Step 4: Fit model

Lastly, we fit the competition Bayesian linear regression model for tree growth outlined in Equation 1 using `comp_bayes_lm()`. This function has an option to specify prior distributions of all parameters, chosen here to be the defaults detailed in `?comp_bayes_lm`.

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

The resulting `comp_bayes_lm_scbi` is an object of S3 class containing the posterior values of all parameters. Furthermore, this class includes generics for three methods. First, the generic for `print()` displays the names of all prior and posterior parameters and the model formula:

```r
comp_bayes_lm_scbi
#> Bayesian linear regression model parameters with a multivariate Normal
#> likelihood. See ?comp_bayes_lm for details:
#>
#> Parameter Type  Prior posterior
#> 1 Inverse-Gamma on sigma2 \sigma_0^2 \sigma_0^2
#> 2 Inverse-Gamma on sigma2 \sigma_0^2 \sigma_0^2
#> 3 Multivariate t on nu \nu_0 \nu_0
#> 4 Multivariate t on beta \beta_0 \beta_0
#>
#> Model formula:
#> growth \sim sp + dbh + astra + sp + acro + sp + amar + sp + astr
#> * sp + caca + * sp + caca + sp + caca + caca + caca + caca + caca + caca
#> * sp + chiv + sp + chiv + sp + chiv + sp + chiv + sp + chiv
#> * sp + divi + sp + divi + sp + divi + sp + divi + sp + divi
#> * sp + havi + sp + havi + sp + havi + sp + havi + sp + havi
#> * sp + juli + sp + juli + sp + juli + sp + juli + sp + juli
#> * sp + lico + sp + lico + sp + lico + sp + lico + sp + lico
#> * sp + quvo + sp + quvo + sp + quvo + sp + quvo + sp + quvo
#> * sp + quvo + sp + quvo + sp + quvo + sp + quvo + sp + quvo
#> * sp + ulam + sp + ulam + sp + ulam + sp + ulam + sp + ulam
#>
#> # Model: focal_species ~ focal_leaf + focal_biomass + focal_growth + focal_growth
```

Next, the generic for `predict()` takes the posterior parameter values in `comp_bayes_lm_scbi` and a new data frame and outputs a vector `growth_hat` of predicted DBH values $\hat{y}$ computed from the posterior predictive distribution.

```r
focal_vs_comp_scbi <- focal_vs_comp_scbi %>%
predict(comp_bayes_lm_scbi, newdata = focal_vs_comp_scbi)
```

```r
focal_vs_comp_scbi %>%
select(focal_leaf, focal_biomass, focal_growth, growth_hat)
#> # A tibble: 6,296 x 5
#> focal_leaf focal_biomass focal_growth growth_hat
#> <fct>          <fct>          <fct>          <dbl>
#> 1 4_nsy         13.6           0.201         0.0089
#> 2 5_havi        8.8            0.150         0.112
#> 3 79_tilom      47.7           -0.161        0.229
#> 4 88_caca       5.13           0.253         0.321
#> 5 96_lite       2.3            0.262         0.342
```

We can now compare the observed and predicted growths to compute the root mean squared error (RMSE) of our model:

```r
model_rese <- focal_vs_comp_scbi %>%
rmse(truth = growth, estimate = growth_hat) %>%
pull(estimate)
```

```r
[1] 0.128
```

Lastly, the generic for `ggplot2::autoplot()` allows us to visualize all posterior distributions, as seen in Figure 4. Setting type to “intercepts” and “dbh_slopes” returns species-specific
posterior distributions for $\beta_0$ and $\beta_{dbh,j}$ respectively, while setting type = "competitition" returns competition coefficients $\lambda_{jk}$.

```r
# Plot posterior for only a subset of species
sp_to_plot <- c("litu", "cagl")
plot1 <- autoplot(
  comp_bayes_ln_schi,
  type = "intercepts",
  sp_to_plot = sp_to_plot
)
plot2 <- autoplot(
  comp_bayes_ln_schi,
  type = "dbh_slopes",
  sp_to_plot = sp_to_plot
)
plot3 <- autoplot(
  comp_bayes_ln_schi,
  type = "competitition",
  sp_to_plot = sp_to_plot
)
# Combine plots using the patchwork package
(plot1 + plot2 + plot3)
```

For many users, the visualizations of $\lambda_{jk}$ will be of particular interest as they provide insight into species-specific competitive interactions, where negative values indicate a competitor species which slows the growth of a focal species. Here, for example, we see that tulip poplars (litu) have a strong negative effect on the growth of conspecifics but relatively lesser effect on pignut hickory (cagl) and red oak (quru) neighbors.

Currently, the forestecology package can only fit the competition Bayesian linear regression model in Equation 1. However, it can be extended to any model as long as it is implemented in a function similar to `comp_bayes_lm()`.

### 2.5 Evaluate the effect of competitor species identity using permutation tests

To evaluate the effect of competitor species identity, we use the above four steps along with the permutation test in Equation 2. Under a null hypothesis where competitor species identity does not matter, we can permute the competitor species identities within each focal tree, compute the RMSE test statistic, repeat this process several times to construct a null distribution, and compare it to the observed RMSE to assess significance. Going back to our example in Section 2.3 of focal tree with `focal_ID = 4` and its 20 competitors, the permutation test only randomly resamples the `comp_sp` variable without replacement, leaving all other variables intact. This resampling is nested within each focal tree in order to preserve neighborhood structure. We perform this permutation test once again using `comp_bayes_lm()` but by setting `run_shuffle = TRUE`.

```r
focal_vs_comp_scbi <- focal_vs_comp_scbi %>%
  run_cv(comp_dist = comp_dist, blocks = blocks_scbi)
model_rmse_cv <- focal_vs_comp_scbi %>%
mse(truth = growth, estimate = growth_hat) %>%
pull(estimate)
model_rmse_cv
## [1] 0.134
```

The resulting permutation test RMSE of 0.131 is larger than the earlier RMSE of 0.128, suggesting that models that do incorporate competitor species identity better fit the data.

### 2.6 Evaluate model performance using spatial cross-validation

To evaluate model performance, we use spatial cross-validation. The model fit in Section 2.4 uses the same data to both fit and assess model performance. Given the spatial-autocorrelation of our data, this can potentially lead to overfit models (Roberts et al., 2017). To mitigate this risk, we use the spatial cross-validation blocking scheme encoded in the `foldID` variable from Section 2.2 and visualized in Figure 2.

At each iteration of the cross-validation, one fold acts as the test set and the remaining three act as the training set. We fit the model to all focal trees in the training set, apply the model to all focal trees in the test set, compute predicted values, and compute the RMSE. Furthermore, to maintain spatial independence between the test and training sets, a “fold buffer” that extends 7.5 m outward from the boundary of the test set is considered; all trees within this “fold buffer” are excluded from the training set (see Figure 5).

This process is repeated for each of the four folds acting as the test set, and then, the four RMSE’s are averaged to provide a single estimate of model error. This algorithm is implemented in `run_cv()`, which acts as a wrapper function to both `comp_bayes_lm()` that fits the model and `predict()` that returns predicted values.

```r
FIGURE 5 Schematic of spatial cross-validation. Using the k = 1 fold (bottom-left) as the test set, k = 2 through 4 as the training set, along with a fold buffer extending outward from the test set to maintain spatial independence between it and the training set.
```

```r
fold
  test
  fold buffer
  training
```

---
The resulting RMSE of 0.14 computed using cross-validation is larger than the earlier RMSE of 0.128, suggesting that models that do not account for spatial autocorrelation generate model error estimates that are overly optimistic, that is, RMSE values that are too low.

3 | IMPORTANCE OF SPATIAL CROSS-VALIDATION

run_cv() also accepts the run_shuffle argument in order to permute competitor species identity as described in Section 2.5. Figure 6 compares model performance for 49 permutations of competitor species and RMSE calculations, both with and without cross-validation. Without cross-validation, competitor species identity does matter as the observed RMSE was significantly lower than the permutation null distribution of RMSE. However, once we incorporate spatial cross-validation, this improvement disappears. These results suggest that in this 9 ha subplot of the SCBI plot, competitive interactions do not depend on the identity of the competitor, which is the opposite of what has been observed in other locations (Allen & Kim, 2020; Uriarte et al., 2004). This provides a striking example of the importance of cross-validation, as without it the over-fit model gives rise to an incorrect conclusion.

4 | CONCLUSION AND FUTURE WORK

The forestecology package provides an accessible way to fit and test models of neighborhood competition. The package follows the tidy data design principles, leverages the sf package for spatial data, and S3 open-oriented model implementation structure (Pebesma, 2018). We hope that the package will increase the use of neighborhood competition models to better understand what structures plant competition.

While the package is designed with ForestGEO plot data in mind, we envision that it can be modified to work on any single large, mapped forest plot in which at least two measurements of each individual have been taken. Furthermore, we hope that future versions of the package will be flexible to other plot layouts, for example, inventory plot-structure with many spatially separated plots like the US Forest Service Forest Inventory and Analysis plots (Smith, 2002).

We also hope to extend the forestecology package’s functionality to account for a larger variety of models for tree growth. One clear future direction would be to allow competition based on species trait values rather than species identity. There is evidence that traits predict competitive outcomes (Kunstler et al., 2012; Lasky et al., 2014; Uriarte et al., 2010). Thus, an extension of the model would allow \( \lambda \) values from Equation 1 to be a function of the traits of competing species.

Lastly, the forestecology package current uses the blockCV package behind the scenes to create the spatial blocks acting as folds for our spatial cross-validation algorithm detailed in Sections 2.2 and 2.6 (Valavi et al., 2019). This back-end functionality could be substituted with the spatialsample package for spatial resampling infrastructure; a tidymodels package under active development as of 2021 (Kuhn & Wickham, 2020; Silge, 2021).

Lastly, currently, the package only implements the Bayesian linear regression model detailed in Equation 1. As we demonstrate in Section 2.4 however, the fitting of this model is self-contained in a single function comp_bayes_lm() which returns an object of S3 class type comp_bayes_lm(). This class has generic methods implemented to print, make predictions, and plot all results. Therefore, the package can be modularly extended to fit other models as long as they are coded similarly to comp_bayes_lm() and have equivalent generic methods implemented.

ACKNOWLEDGMENT
We thank Sophie Li for their feedback on the package interface.

CONFLICT OF INTEREST
We declare no competing interests.

AUTHOR CONTRIBUTIONS
Albert Y. Kim: Conceptualization (equal); Methodology (equal); Software (equal); Writing-original draft (lead); Writing-review & editing (equal). David N. Allen: Conceptualization (equal); Methodology...
OPEN RESEARCH BADGES

This article has been awarded <Open Materials, Open Data> Badges. All materials and data are publicly accessible via the Open Science Framework at [https://zenodo.org/record/4070038; https://github.com/rudeboybert/forestedology/tree/master/paper].

DATA AVAILABILITY STATEMENT

All data and source code for this manuscript on GitHub at https://github.com/rudeboybert/forestedology/tree/master/paper and are archived on Zenodo at https://doi.org/10.5281/zenodo.2649301 (Gonzalez-Akre, McGregor, et al., 2020). All data and formation data are available on GitHub at https://github.com/SCBI-ForestGEO/SCBI-ForestGEO-Data/tree/master and are archived on Zenodo at https://doi.org/10.5281/zenodo.2649301 (Gonzalez-Akre, Mcgegor, et al., 2020).

ORCID

Albert Y. Kim https://orcid.org/0000-0001-7824-306X
David N. Allen https://orcid.org/0000-0002-0712-9603
Simon P. Couch https://orcid.org/0000-0001-5676-5107

ENDNOTE

1 See Appendix 1-4 of Allen and Kim (2020), available at https://doi.org/10.1371/journal.pone.0229930.s004.

REFERENCES


APPENDIX 1

Compare different competitive distances

For all the above analyses, we set the cutoff distance (\texttt{comp\_dist}) for two stems to compete as 7.5m. This distance has been estimated in previous neighborhood competition studies in forests (Canham et al., 2004; Canham et al., 2006; Uriarte et al., 2004). We used 7.5 m in Allen and Kim (2020) as an average of the values estimated in other studies, but our package can be used to find which distance is best supported by the data. Here, we provide an example using another section of the SCBI plot to provide an additional example of the cross-validation block layout. To speed computation, we do not consider species differences in competitive effects and treat all species as the same.

We observe in Figure 7, that a cutoff distance of approximately 6m minimizes the cross-validation estimated RMSE.

census\_2013\_scbi <- read\_csv(\"scbi\_stem2\_csv\")
select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status)
mute(
  date = mdy(date),
  dbh = as\_numeric(dbh) / 10
)
filter(gx < 400, gy < 400)
census\_2018\_scbi <- read\_csv(\"scbi\_stem3\_csv\")
select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status)
mute(
  date = mdy(date),
  dbh = as\_numeric(dbh) / 10
)
filter(gx < 400, gy < 400)
growth\_scbi <-
  compute\_growth(
    census\_1 = census\_2013\_scbi,
    census\_2 = census\_2018\_scbi)
  filter(!str\_detect(codes, "R"),
    id = "stemID"
)
# make all species the same, needs to be factor with at least 2 levels
mutate(sp = factor("A", levels = c("A", "B")))
# Compute basal area:
mutate(basal\_area = 0.0081 * pi * (dbh\_1 / 2)^2)

study\_region\_scbi <- tibble(
  x = c(0, 400, 400, 0, 0),
  y = c(0, 0, 400, 400, 0))
sf\_polygon()

n\_fold <- 4
fold1 <- cbind(c(0, 200, 200, 0), c(0, 0, 200, 200))
fold2 <- cbind(c(200, 400, 400, 200), c(0, 0, 200, 200))
fold3 <- cbind(c(0, 200, 200, 0), c(200, 400, 400, 400))
fold4 <- cbind(c(200, 400, 400, 200), c(200, 200, 400, 400))

blocks\_scbi <- bind\_rows(
  sf\_polygon(fold1), sf\_polygon(fold2), sf\_polygon(fold3), sf\_polygon(fold4))
mutate(folds = c(1:n\_fold))
# Associate each observation to a fold

spatial\_block\_scbi <-
  spatial\_block(
    species\_data = growth\_scbi, k = n\_fold,
    selection = "systematic", blocks = blocks\_scbi,
    show\_blocks = FALSE, verbose = FALSE
  )

growth\_scbi <- growth\_scbi
mutate(fold\_ID = spatial\_block\_scbi$fold\_ID)

mult\_dist\_comp <- tibble(
  dist = c(0, 6.25, 7.5, 8.75, 10),
  rmse = 0
)

for (i in 1:length(mult\_dist\_comp$dist)) {
  comp\_dist <- mult\_dist\_comp$dist[i]
  growth\_scbi <- growth\_scbi
  add\_buffer\_variable(size = comp\_dist, region = study\_region\_scbi)
  focal\_vs\_comp\_scbi <- growth\_scbi
  create\_focal\_vs\_comp(
    comp\_dist = comp\_dist, blocks = blocks\_scbi, id = "stemID",
    comp\_x\_var = "basal\_area"
  )
  run\_cv(comp\_dist = comp\_dist, blocks = blocks\_scbi)
  mult\_dist\_comp$rmse[i] <- focal\_vs\_comp\_scbi
  rmse(truth = growth, estimate = growth\_hat)
  pull(.estimate)
FIGURE 7  Cross-validated RMSE estimates for 5 competitive distances
APPENDIX 2

Compare competitor explanatory variables

In the above code, we use the basal area of an individual as a continuous competitor explanatory variable, but the package allows the user to specify any competitor explanatory variable in the `comp_x_var` argument of `create_focal_vs_comp` function. Here, we use the cross-validated model comparison to see which of two possible competitor explanatory variables computed in Section 2.1, basal area or aboveground biomass, best explains growth.

census_2013_scbi <- read_csv("scbi.sten2.csv")
select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status) mutate(
  date = mdy(date),
  dbh = as.numeric(dbh) / 10)
filter(gx < 300, between(gy, 300, 600))
census_2018_scbi <- read_csv("scbi.sten3.csv")
select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status) mutate(
  date = mdy(date),
  dbh = as.numeric(dbh) / 10)
filter(gx < 300, between(gy, 300, 600))

growth_scbi <-
  compute_growth(
    census_1 = census_2013_scbi,
    census_2 = census_2018_scbi
  )
filter(!str_detect(codes, "R"), id = "stemID")

left_join(sp_info, by = "sp")

sp = as.factor(sp),
basal_area = 0.0001 * pi * (dbh1 / 2)^2,
agb = get_biomass(dbh = dbh1,
genus = genus,
species = species,
coors = c(-78.2, 38.9))

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

sf_polygon()

n_fold = 4
fold1 <- cbind(c(0, 150, 150, 0), c(300, 300, 450, 450))
fold2 <- cbind(c(150, 300, 300, 150), c(300, 300, 450, 450))
fold3 <- cbind(c(0, 150, 150, 0), c(450, 450, 600, 600))
fold4 <- cbind(c(150, 300, 300, 150), c(450, 450, 600, 600))
blocks_scbi <- bind_rows(sf_polygon(fold1), sf_polygon(fold2), sf_polygon(fold3), sf_polygon(fold4))

mutate(foldID = c(1:n_fold))

# Associate each observation to a fold
spatial_block_scbi <-
  spatial_block(
    speciesData = growth_scbi,
    k = n_fold,
    selection = "systematic",
    blocks = blocks_scbi,
    showblocks = FALSE, verbose = FALSE)

growth_scbi <- growth_scbi
mutate(foldID = spatial_block_scbi$foldID)

comp_dist <- 7.5

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

focal_vs_comp.ba <-
  growth_scbi
create_focal_vs_comp(
  comp_dist = comp_dist,
  blocks = blocks_scbi,
  id = "stemID",
  comp_x_var = "basal_area"
)

run_cv(comp_dist = comp_dist, blocks = blocks_scbi)

focal_vs_comp.agb <-
  growth_scbi
create_focal_vs_comp(
  comp_dist = comp_dist,
  blocks = blocks_scbi,
  id = "stemID",
  comp_x_var = "agb"
)

run_cv(comp_dist = comp_dist, blocks = blocks_scbi)

focal_vs_comp.ba

mse(truth = growth, estimate = growth_hat)
## [1] 0.14

focal_vs_comp_agb

mse(truth = growth, estimate = growth_hat)
## [1] 2.88
Here, we observe that basal area is a better competitor explanatory variable than aboveground biomass as suggested by the lower estimated RMSE.

APPENDIX 3

Compare grouping variables

The package also allows the user to specify the categorical explanatory grouping variable. Here, we compare two different such variables: species and the potential canopy position of that species. If we had individual-level crown classes (Smith (1986) dominant, codominant, intermediate, and suppressed) that could also be used.

We find that species identity has a lower RMSE, so does a better job. We still however plot the competition posteriors for the canopy position groupings in Figure 8. Unsurprisingly, we see that canopy and...
canopy emergent competitors generally have negative effects on their neighbors, while shrubs and understory competitors have neutral or even positive effects.

```r
run_cv(comp_dist = comp_dist, blocks = blocks_scbi)

focal_vs_comp_can_pos <- growth_scbi_can_pos
create_focal_vs_comp(comp_dist = comp_dist, blocks = blocks_scbi, id = "stemID",
  comp_vs_var = "basal_area")
run_cv(comp_dist = comp_dist, blocks = blocks_scbi)

focal_vs_comp_sp
rmse(truth = growth, estimate = growth_hat)

focal_vs_comp_can_pos
rmse(truth = growth, estimate = growth_hat)
pull(estimate)

fit_mod_can_pos <- growth_scbi_can_pos
create_focal_vs_comp(comp_dist = comp_dist, blocks = blocks_scbi, id = "stemID",
  comp_vs_var = "basal_area")
comp_bayes_lm(prior_paran = NULL)
```

**Figure 8** Posterior distributions of all competition parameters
APPENDIX 4

Replicate RMSE comparison
This code replicates Figure 6: A comparison of root mean squared error of models for standard, permuted, and spatial cross-validated error estimates.

```r
library(tidyverse)
library(lubridate)
library(here)
library(sf)
library(viridis)
library(forestmetrics)
library(blockCV)
library(tictoc)

# Compute growth of trees based on census data ---------------------------------

census_2013_scbl <- here("paper/scbl/stem2.csv")
read_csv()$X
select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status)$X
mutate(
    date = as.numeric(date) / 10
) %>%
filter(gx < 300, between(gy, 300, 600))
census_2018_scbl <- here("paper/scbl/stem3.csv")$X
read_csv()$X
select(stemID, sp, date = ExactDate, gx, gy, dbh, codes, status)$X
mutate(
    date = as.numeric(date) / 10
) %>%
filter(gx < 300, between(gy, 300, 600))
growth_scbl <-
compute_growth(
    census_1 = census_2013_scbl,
    census_2 = census_2018_scbl
) %>%
filter(!str_detect(codes, "N"),
    id = "stemID"
)$X
# Compute basal area:
mutate(basal_area = 0.0081 * pi * (dbh1 / 2)^2)

# Add spatial information -----------------------------------------------

# Define buffer region using competitive distance range
comp_dist <- 7.5
study_region_scbl <- tibble(
x = c(0, 300, 300, 0, 0),
y = c(300, 300, 600, 600, 300),
) %>%
sf::polygon()
growth_scbl <- growth_scbl %>%
add_buffer_variable(size = comp_dist, region = study_region_scbl)

# Manually define spatial blocks to act as folds
fold1 <- rbbox(c(0, 300, 150, 300), c(150, 300), c(150, 450), c(0, 450))
fold2 <- rbbox(c(150, 300), c(300, 300), c(300, 600), c(150, 600))
fold3 <- rbbox(c(0, 450), c(150, 450), c(150, 600), c(0, 600))
fold4 <- rbbox(c(150, 450), c(300, 450), c(300, 600), c(150, 600))

blocks_scbl <- bind_rows(
    sf::polygon(fold1), sf::polygon(fold2),
    sf::polygon(fold3), sf::polygon(fold4)
) %>%
mutate(folds = c(1:n_fold) %>% factor())

# Associate each observation to a fold
SpatialBlock_scbl <- spatialBlock(
    species_data = growth_scbl,
    k = n_fold, selection = "systematic",
    blocks = blocks_scbl, showblocks = FALSE, verbose = FALSE
)
growth_scbl <- growth_scbl %>%
mutate(foldID = SpatialBlock_scbl$foldID %>% factor())

# Compute focal versus competitor tree information ------------------------

focal_vs_comp_scbl <- growth_scbl %>%
create_focal_vs_comp(
    comp_dist, blocks = blocks_scbl, id = "stemID",
    comp_x_var = "basal_area"
)

# Run all simulations
# 1. Setup simulation for this species type ----- Start clock tic()

# 1. Compute observed test statistic: RMSE with no cross-validation ----

observed_RMSE <- focal_vs_comp_scbl %>%
mutate(
    growth_hat = predict(comp_bayes_im_scbl, focal_vs_comp_scbl)
)$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# 2. Compute observed test statistic: RMSE with cross-validation -------

observed_RMSE Cv <- focal_vs_comp_scbl %>%
run_cv(comp_dist = comp_dist, blocks = blocks_scbl)$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# 3. Permutation distribution: RMSE with no cross-validation -------

# Compute num_shuffle permutation test statistics
shuffle_RMSE <- numeric(length = num_shuffle)
for (j in 1:num_shuffle) {
    # Fit model (compute posterior parameters with shuffling
    comp_bayes_im_scbl <- focal_vs_comp_scbl %>%
    comp_bayes_im(prior_param = NULL, run_shuffle = TRUE)
}

# Make predictions and compute RMSE
shuffle_RMSE[] <- focal_vs_comp_scbl %>%
mutate(
    growth_hat = predict(comp_bayes_im_scbl, focal_vs_comp_scbl)
)$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# 4. Permutation distribution: RMSE with cross-validation -------

# Compute num_shuffle permutation test statistics
shuffle_RMSE Cv <- numeric(length = num_shuffle)
for (j in 1:num_shuffle) {
    # Compute and save RMSE
    shuffle_RMSE Cv[j] <- focal_vs_comp_scbl %>%
    run_cv(comp_dist = comp_dist, blocks = blocks_scbl, run_shuffle = TRUE)
}$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# Number of permutation shuffles:
num_shuffle <- 49

# Save results here
run_time <- 0
observed_RMSE <- 0
observed_RMSE Cv <- 0
shuffle_RMSE <- vector("list", 1)
shuffle_RMSE Cv <- vector("list", 1)

# Fit model and make predictions
# 1. Compute test statistic: RMSE with no cross-validation ----

# 2. Compute test statistic: RMSE with cross-validation -------

# 3. Permutation distribution: RMSE with no cross-validation -------

# Compute num_shuffle permutation test statistics
shuffle_RMSE <- numeric(length = num_shuffle)
for (j in 1:num_shuffle) {
    # Fit model (compute posterior parameters with shuffling
    comp_bayes_im_scbl <- focal_vs_comp_scbl %>%
    comp_bayes_im(prior_param = NULL, run_shuffle = TRUE)
}

# Make predictions and compute RMSE
shuffle_RMSE[] <- focal_vs_comp_scbl %>%
mutate(
    growth_hat = predict(comp_bayes_im_scbl, focal_vs_comp_scbl)
)$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# 4. Permutation distribution: RMSE with cross-validation -------

# Compute num_shuffle permutation test statistics
shuffle_RMSE Cv <- numeric(length = num_shuffle)
for (j in 1:num_shuffle) {
    # Compute and save RMSE
    shuffle_RMSE Cv[j] <- focal_vs_comp_scbl %>%
    run_cv(comp_dist = comp_dist, blocks = blocks_scbl, run_shuffle = TRUE)
}$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)
```

# Save results here
run_time <- 0
observed_RMSE <- 0
observed_RMSE Cv <- 0
shuffle_RMSE <- vector("list", 1)
shuffle_RMSE Cv <- vector("list", 1)

time() <- here("paper/simulation_results/")$X
str_c("2021-03-31", num_shuffle, _shuffles")

# Run all simulations
# 1. Setup simulation for this species type ----- Start clock tic()

# 1. Compute observed test statistic: RMSE with no cross-validation ----

observed_RMSE <- focal_vs_comp_scbl %>%
mutate(
    growth_hat =
    predict(comp_bayes_im_scbl, focal_vs_comp_scbl)
)$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# 2. Compute observed test statistic: RMSE with cross-validation -------

observed_RMSE Cv <- focal_vs_comp_scbl %>%
run_cv(comp_dist = comp_dist, blocks = blocks_scbl)$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# 3. Permutation distribution: RMSE with no cross-validation -------

# Compute num_shuffle permutation test statistics
shuffle_RMSE <- numeric(length = num_shuffle)
for (j in 1:num_shuffle) {
    # Fit model (compute posterior parameters with shuffling
    comp_bayes_im_scbl <- focal_vs_comp_scbl %>%
    comp_bayes_im(prior_param = NULL, run_shuffle = TRUE)
}

# Make predictions and compute RMSE
shuffle_RMSE[] <- focal_vs_comp_scbl %>%
mutate(
    growth_hat = predict(comp_bayes_im_scbl, focal_vs_comp_scbl)
)$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)

# 4. Permutation distribution: RMSE with cross-validation -------

# Compute num_shuffle permutation test statistics
shuffle_RMSE Cv <- numeric(length = num_shuffle)
for (j in 1:num_shuffle) {
    # Compute and save RMSE
    shuffle_RMSE Cv[j] <- focal_vs_comp_scbl %>%
    run_cv(comp_dist = comp_dist, blocks = blocks_scbl, run_shuffle = TRUE)
}$X
rmse(truth = growth, estimate = growth_hat)$X
pvalue(estimate)
```
# Status update
str_c("shuffle with permutation ", j, " at ", Sys.time()) %>%
print()

# 5. Save results ----
clock <- toc(quiet = TRUE)
run_time <- clock %>% clock()

model_comp_tbl <- tibble(
  run_time = run_time,
  observed_RMSE = observed_RMSE,
  observed_RMSE.CV = observed_RMSE.CV,
  shuffle_RMSE = shuffle_RMSE,
  shuffle_RMSE.CV = shuffle_RMSE.CV,
)
save(model_comp_tbl, file = filename %>% str_c(".RData"))

# Visualize results ---------------------------------

model_comp <- bind_rows(
  model_comp_tbl %>%
  select(run_time, observed = observed_RMSE, shuffle = shuffle_RMSE)
) %>%
names <- c(CV = FALSE),
model_comp_tbl %>%
  select(run_time, observed = observed_RMSE.CV, shuffle = shuffle_RMSE.CV)
) %>%
names <- c(CV = TRUE)
)

names <- c(type = "observed"),
unnest(cols = c(RMSE))
model_comp_shuffle <- model_comp %>%
  filter(type == "shuffle") %>%
  unnest(cols = c(RMSE))

cv_plot <- ggplot() +
geom_vline(
  data = model_comp_observed,
  aes(xintercept = RMSE, col = CV),
  linetype = "dashed", show.legend = F
) +
geom_histogram(
  data = model_comp_shuffle,
  aes(x = RMSE, fill = CV), bins = 50
) +
labs(
  fill = "Cross-validated?",
  x = expression(paste("RMSE (\(c\), \(y\)\{1\}), \(\)\))
) +
scale_color_viridis(discrete = TRUE, option = "D") +
scale_fill_viridis(discrete = TRUE) +
theme_light()

cv_plot

filename %>%
str_c(".pdf") %>%
ggsave(plot = cv_plot, width = 16 / 2, height = 9 / 2)