compInd package vignette

John L. Godlee

10th May 2021

compInd is a collection of spatially explicit competition and diversity indices taken primarily from the forestry literature. The aim of the package is to provide these functions in a consistent framework and to provide convenience functions to prepare data for analysis. This document provides some basic usage of compInd, and describes the behaviour of some of the competition indices included as functions in the package. See the manual for the package for more information on each function, including references and equations.

```
library(compInd)

library(sf)
library(dplyr)
library(tidyr)
library(ggplot2)
library(patchwork)
library(viridis)
library(parallel)
```

Generate example plot data suited to spatial competition indices:

```
dat <- dataGen()
```

Randomly located individuals within 1 ha (100x100 m) plots:

```
ggplot() +
  geom_point(data = dat,
    aes(x = x_grid, y = y_grid, fill = species, size = diam),
    shape = 21) +
  facet_wrap(~plot_id) +
  theme(legend.position = "none") +
  coord_equal()
```

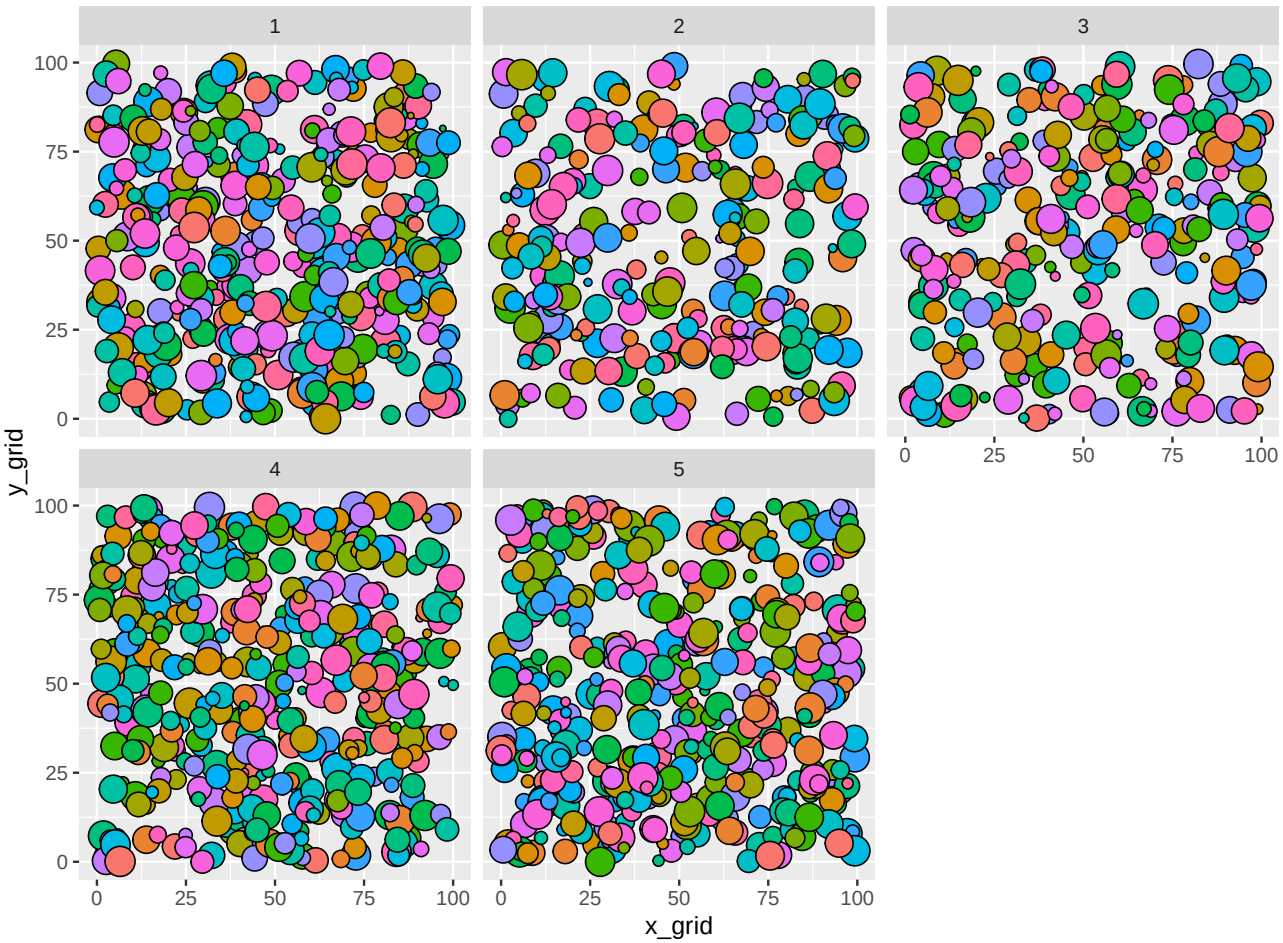

Figure 1: Five plots with randomly located stems of different species.

There is also a real-world example of a 1 ha woodland plot in Bicuar National Park, southwest Angola:

```
data(bicuar)

ggplot() +
  geom_point(data = bicuar,
    aes(x = x_grid, y = y_grid, fill = species, size = diam),
    shape = 21) +
  theme(legend.position = "none") +
  coord_equal()
```

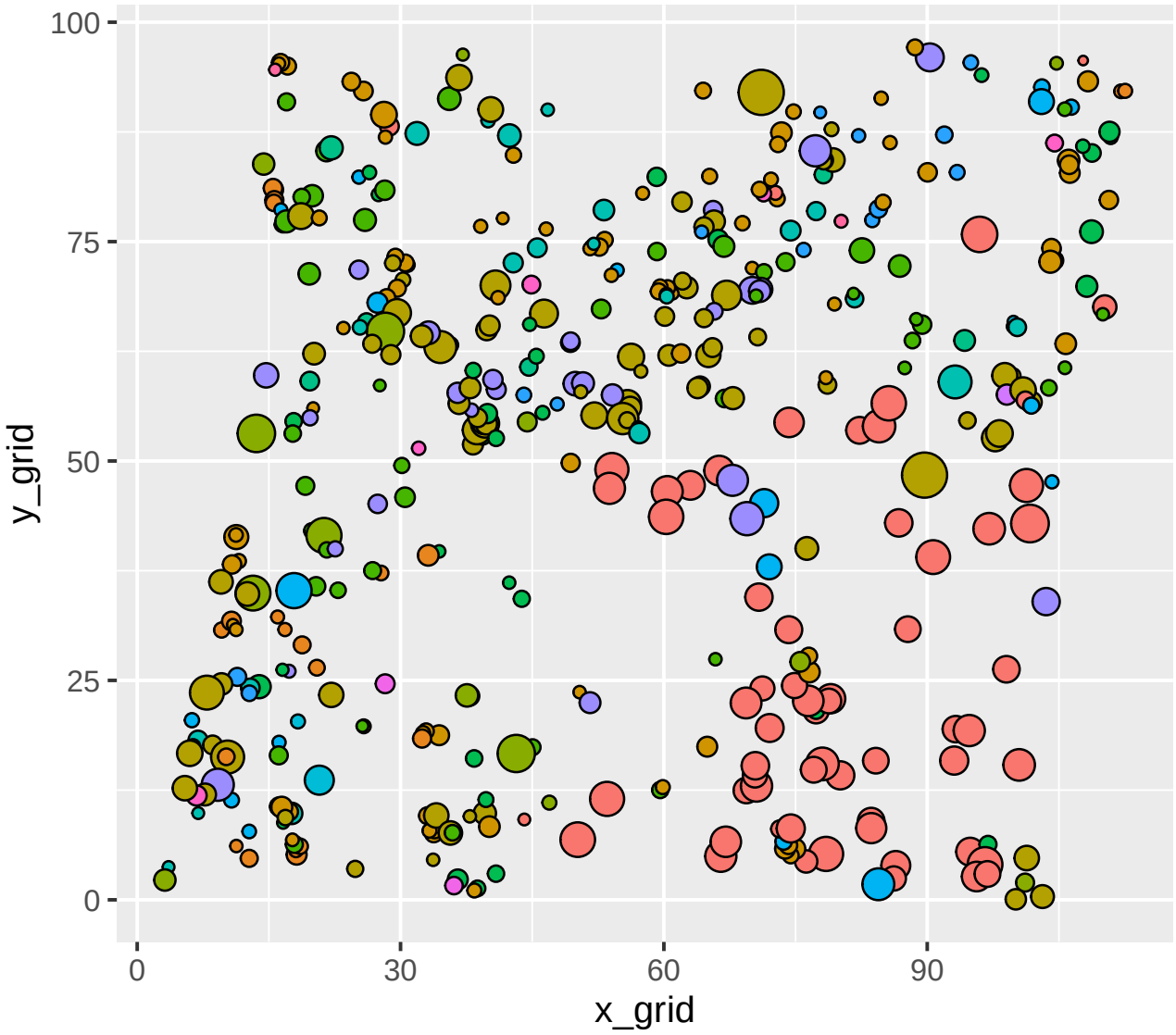

Figure 2: Bicuar National Park data object.

Generate some extra columns:

```
dat$ba <- basalArea(dat$diam)
```

Many of the functions in `compInd` are calculated for each individual in the structural unit, based on the individuals in their local neighbourhood. `nearNeighb()` provides multiple methods of identifying neighbours around focal individuals, returning a list of dataframes, one per focal individual, with the spatial relationships between the focal individual and each neighbour. The example below first splits the example data by plot ID, then feeds the data plot by plot to `nearNeighb()`, extracting the 4 nearest neighbours to each individual in the plot. After that, it adds some extra information to each focal and neighbour individual such as the species, diameter and spatial coordinates.

```
dat_split <- split(dat, dat$plot_id)

dat_neighb <- lapply(dat_split, function(x) {
  neighb <- nearNeighb(x$x_grid, x$y_grid, x$stem_id, k = 4)
  out <- lapply(neighb, function(y) {
    left_join(y, x, by = c("focal" = "stem_id")) %>%
      rename(focal_diam = diam, focal_species = species,
        focal_x_grid = x_grid, focal_y_grid = y_grid, focal_ba = ba) %>%
      left_join(., x[,-which(names(x) == "plot_id")],
        by = c("nb" = "stem_id")) %>%
      rename(nb_diam = diam, nb_species = species,
        nb_x_grid = x_grid, nb_y_grid = y_grid, nb_ba = ba)
  })
  return(out)
})
```

Now it's possible to test various functions in `compInd`. The following functions act on each focal individual separately:

- `alemdag()`
- `baLarger()`
- `baLocal()`
- `crowdInd()`
- `dbhCorr()`
- `dbhDiff()`
- `dbhDom()`
- `hegyi()`
- `lorimerComp()`
- `martinEk()`
- `pointDens()`

Using `alemdag()` as an example, calculate the value of the Alemdag's tree competition index for each focal individual in each plot, using four neighbours as per convention:

```
alemdag_list <- lapply(dat_neighb, function(x) {
        unlist(lapply(x, function(y) {
                alemdag(y$nb_diam, y$nb_dist, unique(y$focal_diam))
        }))
})
```

To use the other functions listed above, just switch the function name and particular arguments out in place of `alemdag()`. To scale up to a whole plot, either the sum or mean of values from focal individuals is commonly used, depending on the competition index.

Other functions for certain competition indices, those which don't rely on a zone of competition, act on each structural unit, i.e. plot, separately. These functions are:

- clarkEvans()
- pielou()
- spatialMingling()
- winkelmass()

Using clarkEvans() as an example, calculate the value of the Clark-Evans index for each individual in the plot:

```
clarkEvans_list <- lapply(dat_split, function(x) {
        clarkEvans(x$x_grid, x$y_grid, area = 10000)
})
```

There are some extra convenience functions in compInd to make it easier to prepare data for the competition index calculations. The first, nearNeighb() has already been showcased above.

lorimerCZR() can be used to estimate the competition zone radius for a structural unit based on the number of individuals in the unit. The output of lorimerCZR() could then be fed to either nearNeighb() to identify neighbours or to lorimerComp(), which uses this value to normalise the result of the index for comparison among units with different individual densities.

```
lorimerCZR(k = 1, nrow(bicuar))
```

edgeExclude() can be used to find individuals in the edge or buffer zone of a plot. For competition indices which rely on a radius of expected neighbourhood effects such as lorimerComp(), alemdag() etc, individuals near the edges of the plot will have an erroneous under-estimation of competition compared to other individuals in the plot interior, because there is a lack of data outside the plot. It is common practice to exclude focal individuals if they are located a distance from plot edge that is less than the radius of the competition zone. These individuals could still be used as competitors for other focal individuals, but not as focal individuals themselves.

Consider the plots below, which show the effect of a buffer of 10 m radius in causing under-estimations of competition calculated by crowdInd():

```
# Generate data for one plot, 500 stems
dat <- dataGen(nplots = 1, min_stems = 500, max_stems = 500)

# Create a polygon plot outline
dat_poly <- st_polygon(list(
                cbind(c(0,0,100,100,0), c(0,100,100,0,0))))

# Find nearest neighbours for each individual
neighb_rad10 <- nearNeighb(dat$x_grid, dat$y_grid, radius = 10)

# Count number of neighbours identified
dat$n_neighb <- unlist(lapply(neighb_rad10, nrow))

# Create a map of individuals, coloured and sized by number of neighbours

ggplot() +
  geom_sf(data = dat_poly, fill = NA, colour = "red") +
  geom_point(data = dat,
          aes(x = x_grid, y = y_grid, size = n_neighb, colour = n_neighb)) +
  scale_colour_viridis()
```

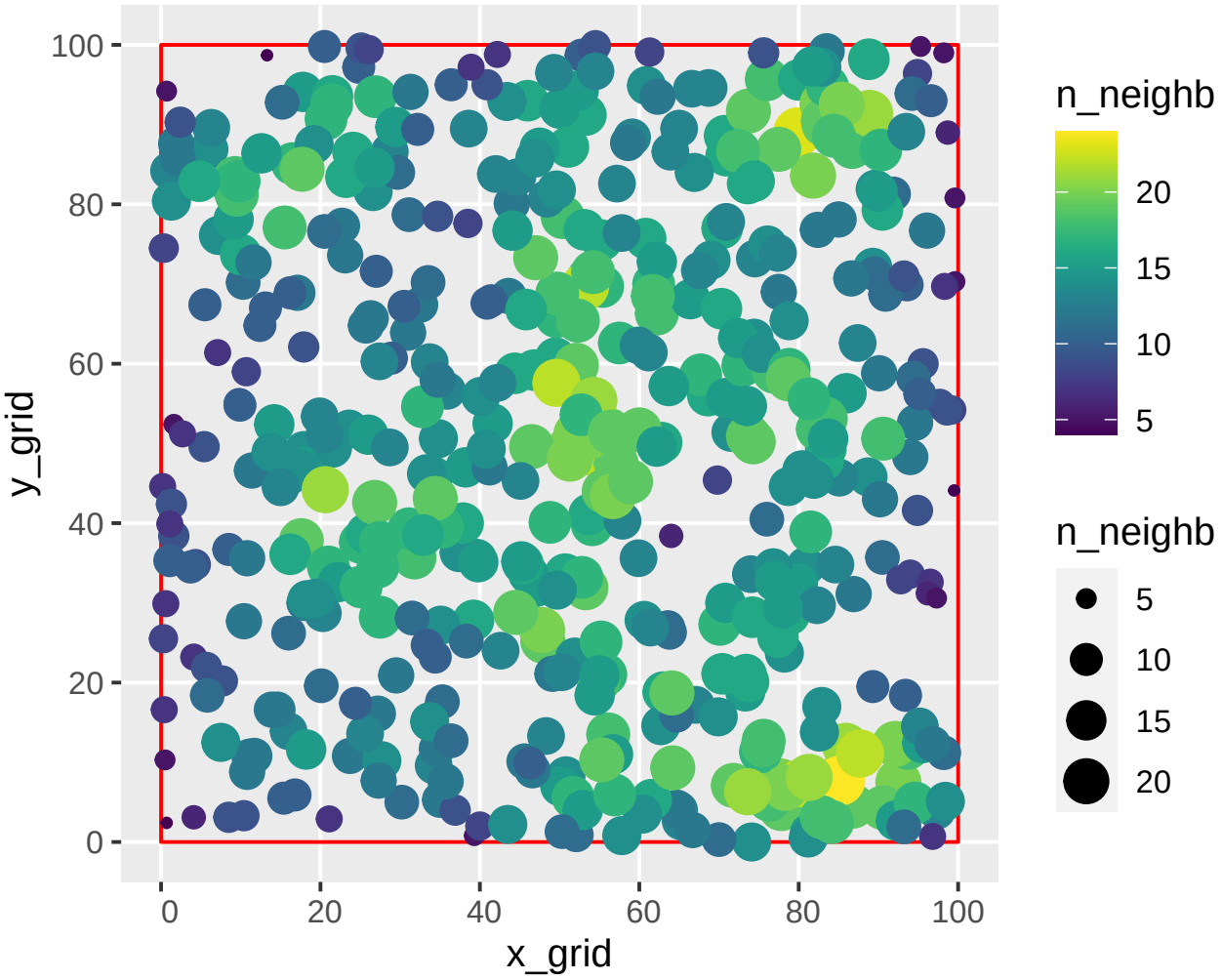

Figure 3: Visualising the number of neighbours within a 10 m radius in a 1 ha plot.

```
# Find if an individual is in the edge of a plot
dat$edge <- ifelse(rownames(dat) %in%
        edgeExclude(dat_poly, 10, dat$x_grid, dat$y_grid), "OUT", "IN")

# Create boxplot
ggplot() +
        geom_boxplot(data = dat,
                aes(x = edge, y = n_neighb, colour = edge))
```

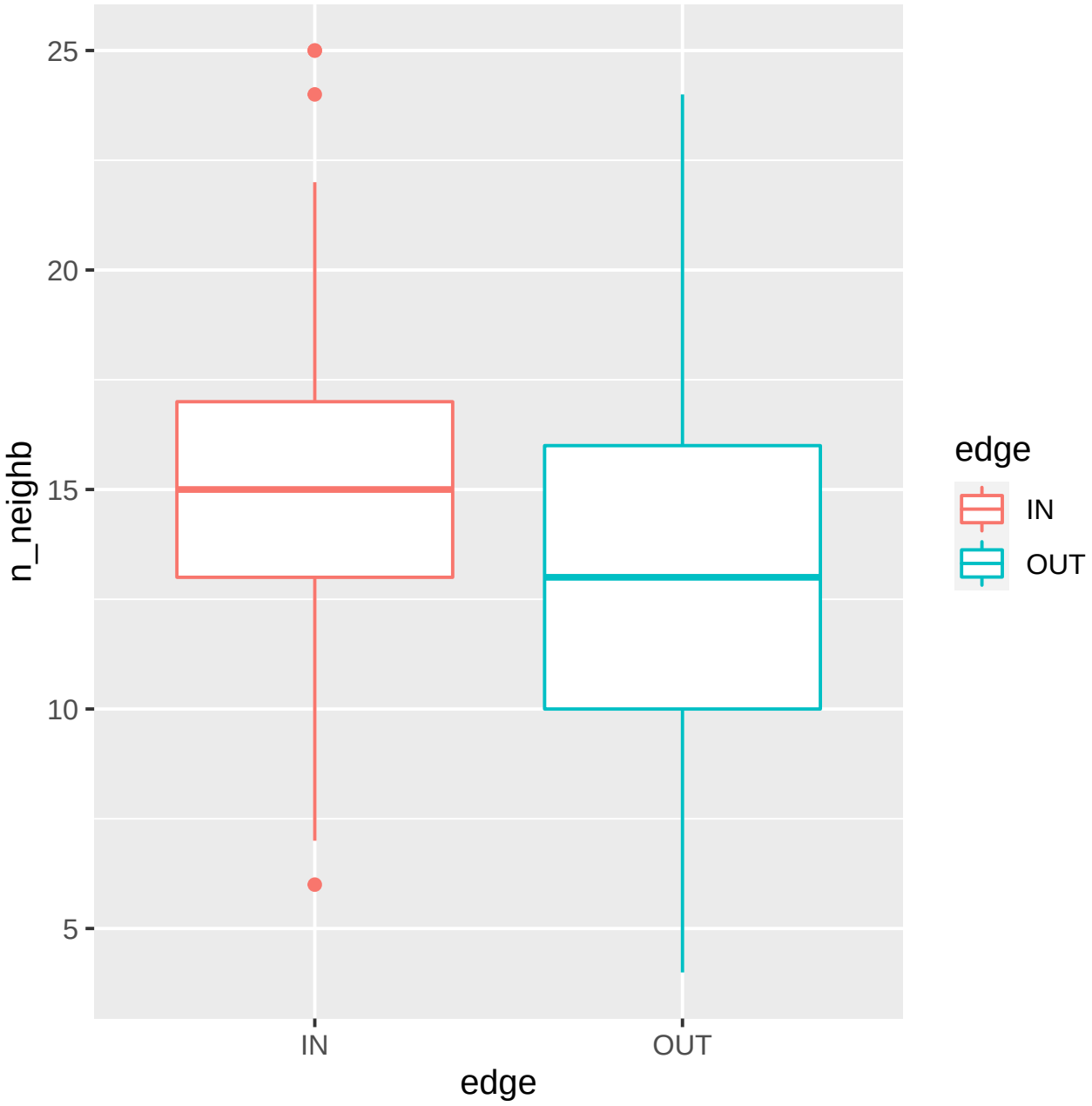

Figure 4: Boxplot of number of neighbours within a 10 m radius in a 1 ha plot.

```
# Create a buffer around each plot
dat_buffer <- st_as_sf(dat, coords = c("x_grid", "y_grid")) %>%
        st_buffer(., 10)

dat_poly_buffer <- sf::st_buffer(dat_poly, -10, endCapStyle = "FLAT")

# Map of buffer radiuses, coloured by if individual is in edge of plot
ggplot() +
        geom_sf(data = dat_buffer, aes(colour = edge), fill = NA) +
        geom_point(data = dat,
          aes(x = x_grid, y = y_grid)) +
```

```
        geom_sf(data = dat_poly, fill = NA, colour = "red") +
        geom_sf(data = dat_poly_buffer, fill = NA, colour = "green")
```

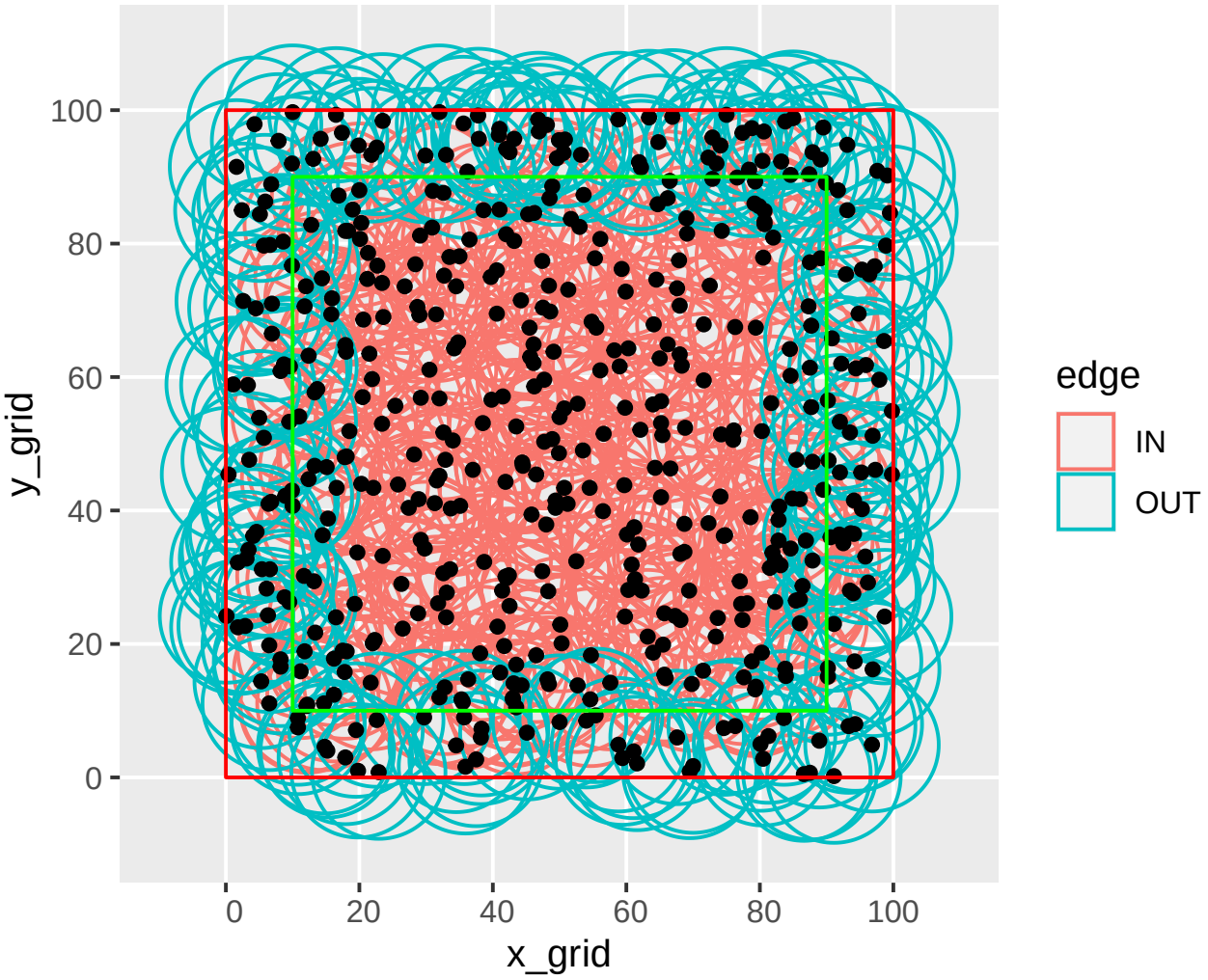

Figure 5: Results of `edgeExclude()`. Each individual is given a buffer of the radius of the competition zone (10 m). These radii are coloured according to whether they are identified for exclusion by `edgeExclude()`. The red squares indicate the boundaries of the plot, and the green square indicates the 10 m buffer than `edgeExclude()` uses to identify points at the edge of the plot.

The various competition indices in `compInd` behave in subtly different ways as the number of individuals, species, neighbours or their spatial distribution changes. Below we produce visualisations of various indices under different conditions.

Generate data with increasing abundance:

```
ab_vec <- rep(seq(50,1000, 100), 10)
ab_list <- lapply(ab_vec, function(x) {
  out <- dataGen(nplots = 1, min_stems = x, max_stems = x, species = LETTERS[1:10])
  out$plot_id <- as.character(x)
  out$ba <- basalArea(out$diam)
  return(out)
})

ggplot() +
  geom_point(data = do.call(rbind, ab_list[seq(1,20,2)]),
    aes(x = x_grid, y = y_grid, fill = species),
    shape = 21) +
  facet_wrap(~as.numeric(plot_id)) +
  theme_bw() +
  theme(legend.position = "none") +
  coord_equal()
```

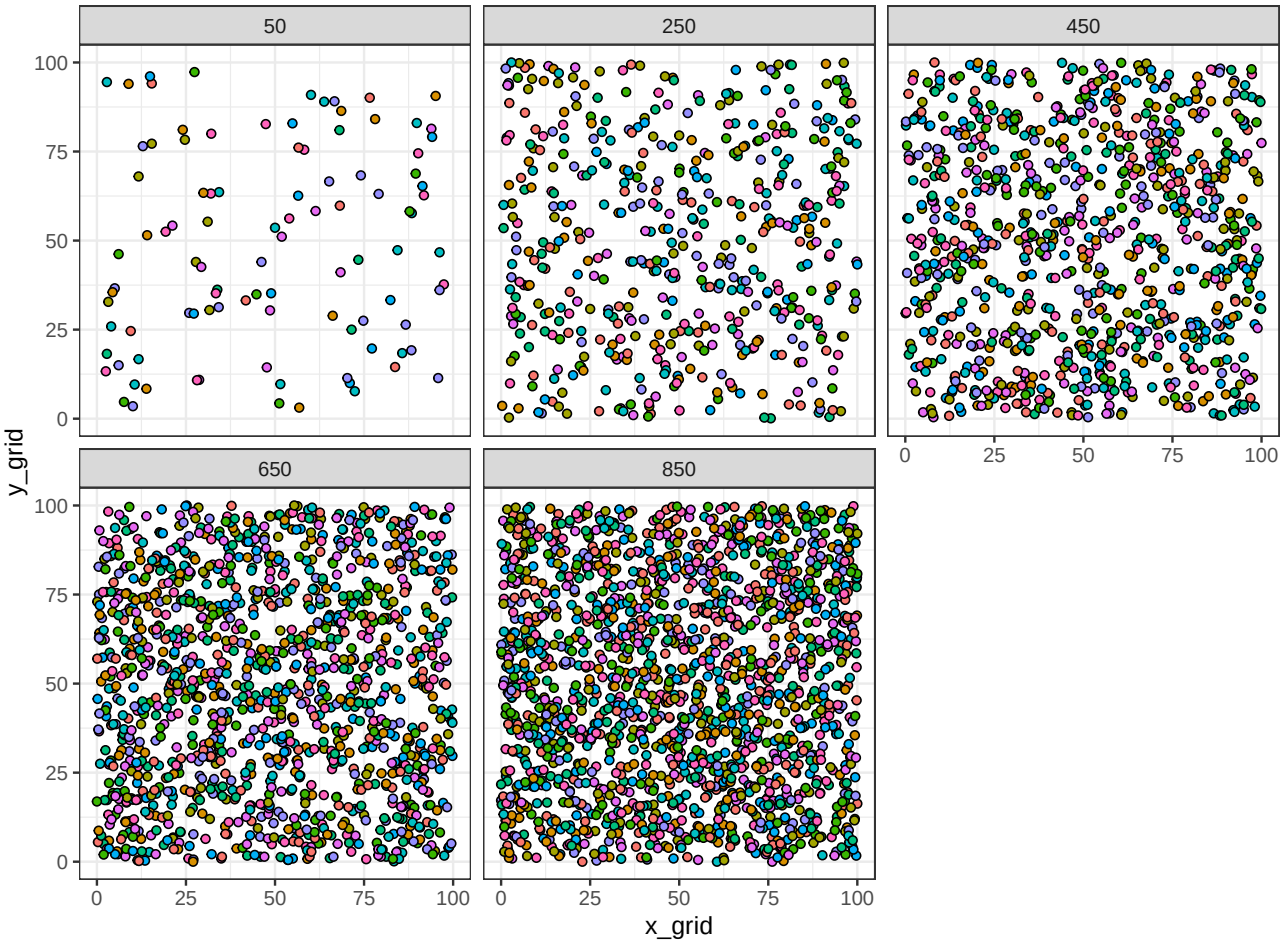

Figure 6: Visualising variation in individual density.

Calculate nearest neighbours for each sample:

```
ab_neighb <- mclapply(ab_list, function(x) {
  neighb <- nearNeighb(x$x_grid, x$y_grid, x$stem_id, k = 4)
  out <- lapply(neighb, function(y) {
    left_join(y, x, by = c("focal" = "stem_id")) %>%
      rename(focal_diam = diam, focal_species = species,
        focal_x_grid = x_grid, focal_y_grid = y_grid, focal_ba = ba) %>%
      left_join(., x[,-which(names(x) == "plot_id")],
        by = c("nb" = "stem_id")) %>%
      rename(nb_diam = diam, nb_species = species,
        nb_x_grid = x_grid, nb_y_grid = y_grid, nb_ba = ba)
  })
  return(out)
}, mc.cores = 4)
```

Calculate plot level values for various indices:

```
ab_summ <- do.call(rbind, mclapply(seq_along(ab_neighb), function(x) {
  plot_id <- unique(ab_list[[x]]$plot_id)
  ab_lorimerCZR <- lorimerCZR(1, nrow(ab_list[[x]]))
  ab_clarkEvans <- clarkEvans(ab_list[[x]]$x_grid, ab_list[[x]]$y_grid, 10000)
  ab_pielou_mean <- mean(pielou(ab_list[[x]]$x_grid, ab_list[[x]]$y_grid, 0, 100, 0, 100, 4))
  ab_spatialMingling_mean <- mean(spatialMingling(
                  ab_list[[x]]$x_grid, ab_list[[x]]$y_grid, ab_list[[x]]$species))
  ab_winkelmass_mean <- mean(winkelmass(ab_list[[x]]$x_grid, ab_list[[x]]$y_grid, 4))

  out <- do.call(rbind, lapply(ab_neighb[[x]], function(y) {
    data.frame(
      ab_alemdag = alemdag(y$nb_diam, y$nb_dist, unique(y$focal_diam)),
      ab_baLarger = baLarger(y$nb_ba, unique(y$focal_ba)),
      ab_baLocal = baLocal(y$nb_ba),
      ab_crowdInd = crowdInd(y$nb_diam),
      ab_dbhCorr = dbhCorr(y$nb_diam, unique(y$focal_diam)),
      ab_dbhDiff = dbhDiff(y$nb_diam, unique(y$focal_diam)),
      ab_dbhDom = dbhDom(y$nb_diam, unique(y$focal_diam)),
      ab_hegyi = hegyi(y$nb_diam, y$nb_dist, unique(y$focal_diam)),
      ab_lorimerComp = lorimerComp(y$nb_diam, y$nb_dist, unique(y$focal_diam), ab_lorimerCZR),
      ab_martinEk = martinEk(y$nb_diam, y$nb_dist, unique(y$focal_diam)),
      ab_pointDens = pointDens(y$nb_diam, y$nb_dist)
    )
  }))

  out_summ <- out %>%
    summarise(
      ab_alemdag_mean = mean(ab_alemdag),
      ab_baLarger_sum = sum(ab_baLarger),
      ab_baLocal_sum = sum(ab_baLocal),
      ab_crowdInd_mean = mean(ab_crowdInd),
      ab_dbhCorr_mean = mean(ab_dbhCorr),
      ab_dbhDiff_mean = mean(ab_dbhDiff),
      ab_dbhDom_mean = mean(ab_dbhDom),
      ab_hegyi_mean = mean(ab_hegyi),
      ab_lorimerComp_mean = mean(ab_lorimerComp),
      ab_martinEk_mean = mean(ab_martinEk),
      ab_pointDens_mean = mean(ab_pointDens)
    ) %>%
```

```
    mutate(
      plot_id = plot_id,
      ab_lorimerCZR = ab_lorimerCZR,
      ab_clarkEvans = ab_clarkEvans,
      ab_pielou_mean = ab_pielou_mean,
      ab_spatialMingling_mean = ab_spatialMingling_mean,
      ab_winkelmass_mean = ab_winkelmass_mean
    )

  return(out_summ)

}, mc.cores = 4))
```

Plot variation in indices with varying abundance:

```
ab_summ_gather <- ab_summ %>%
  gather(key, value, -plot_id)

ggplot() +
  geom_point(data = ab_summ_gather, aes(x = as.numeric(plot_id), y = value),
    fill = "darkgrey", shape = 21) +
  geom_smooth(data = ab_summ_gather, aes(x = as.numeric(plot_id), y = value),
    method = "loess") +
  facet_wrap(~key, scales = "free") +
  theme_bw() +
  labs(x = "N stems", y = "")
```

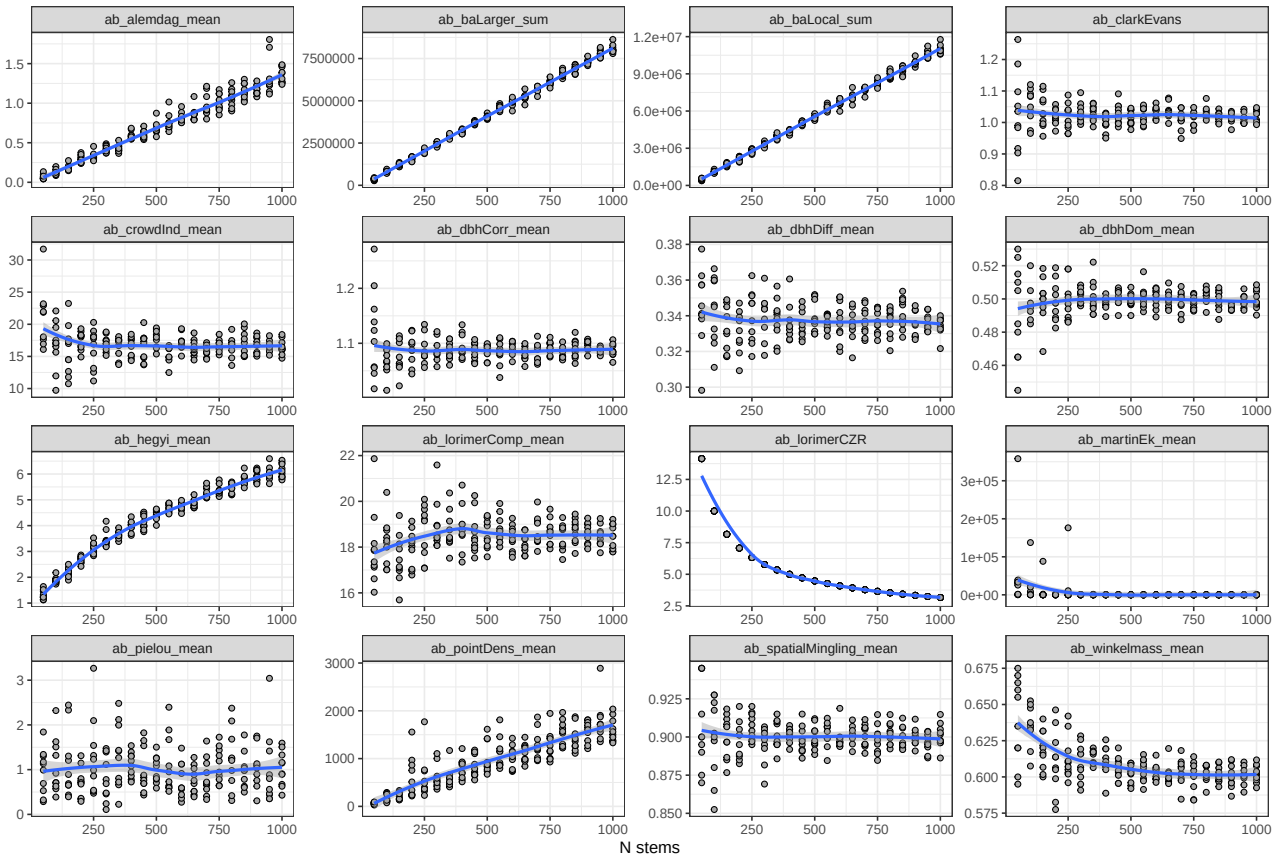

Figure 7: Variation in various competition indices with increasing individual abundance.

The number of neighbours considered by `nearNeighb()` also affects the behaviour of the indices:

```
# Using datasets with 100 individuals
k_vec <- seq(4,12,1)

k_summ <- do.call(rbind, mclapply(k_vec, function(x) {
  do.call(rbind, lapply(ab_neighb12[seq(2,182,20)], function(y) {
    out <- do.call(rbind, lapply(y, function(z) {
      data.frame(
        ab_alemdag = alemdag(z$nb_diam[1:x], z$nb_dist[1:x], unique(z$focal_diam)),
        ab_baLarger = baLarger(z$nb_ba[1:x], unique(z$focal_ba)),
        ab_baLocal = baLocal(z$nb_ba[1:x]),
        ab_crowdInd = crowdInd(z$nb_diam[1:x]),
        ab_dbhCorr = dbhCorr(z$nb_diam[1:x], unique(z$focal_diam)),
        ab_dbhDiff = dbhDiff(z$nb_diam[1:x], unique(z$focal_diam)),
        ab_dbhDom = dbhDom(z$nb_diam[1:x], unique(z$focal_diam)),
        ab_hegyi = hegyi(z$nb_diam[1:x], z$nb_dist[1:x], unique(z$focal_diam)),
        ab_martinEk = martinEk(z$nb_diam[1:x], z$nb_dist[1:x], unique(z$focal_diam)),
        ab_pointDens = pointDens(z$nb_diam[1:x], z$nb_dist[1:x])
      )
    }))

    out_summ <- out %>%
      summarise(
        ab_alemdag_mean = mean(ab_alemdag),
```

```
        ab_baLarger_sum = sum(ab_baLarger),
        ab_baLocal_sum = sum(ab_baLocal),
        ab_crowdInd_mean = mean(ab_crowdInd),
        ab_dbhCorr_mean = mean(ab_dbhCorr),
        ab_dbhDiff_mean = mean(ab_dbhDiff),
        ab_dbhDom_mean = mean(ab_dbhDom),
        ab_hegyi_mean = mean(ab_hegyi),
        ab_martinEk_mean = mean(ab_martinEk),
        ab_pointDens_mean = mean(ab_pointDens)
      ) %>%
      mutate( k = x )

    return(out_summ)
  }))

}, mc.cores = 4))
```

```
k_summ_gather <- k_summ %>%
  gather(key, value, -k)

ggplot() +
  geom_point(data = k_summ_gather, aes(x = as.numeric(k), y = value),
    fill = "darkgrey", shape = 21) +
  geom_smooth(data = k_summ_gather, aes(x = as.numeric(k), y = value),
    method = "loess") +
  facet_wrap(~key, scales = "free") +
  theme_bw() +
  labs(x = "k", y = "")
```

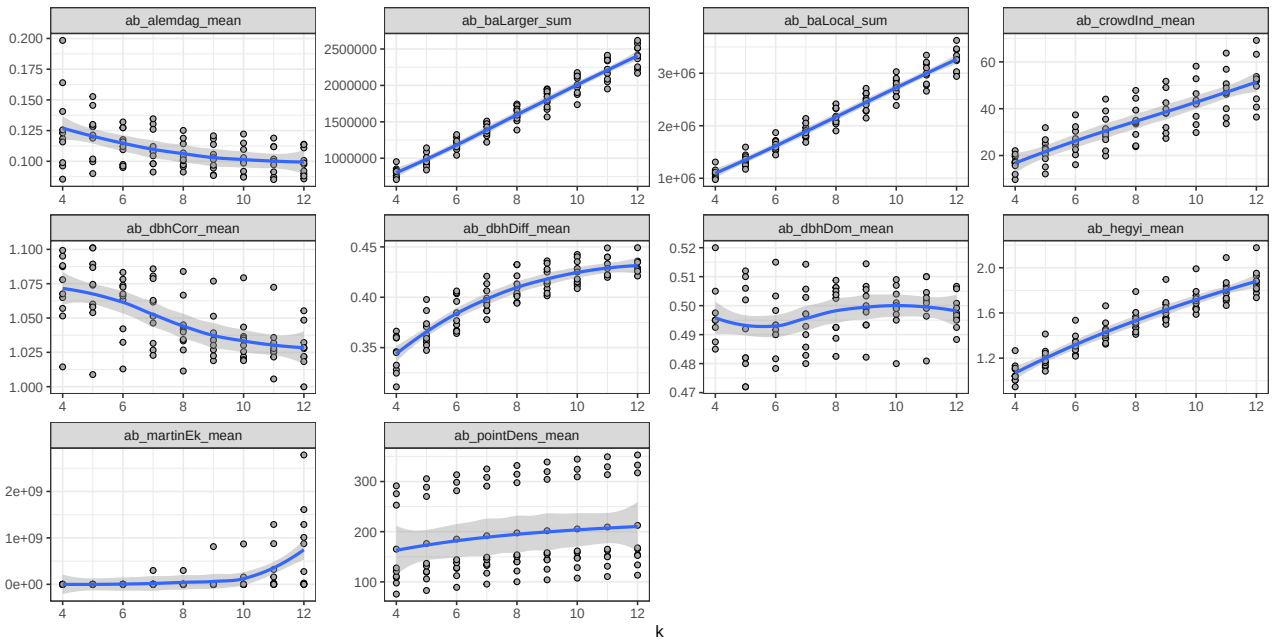

Figure 8: Variation in various competition indices with increasing number of neighbours considered.

Variation in the number of species affects `spatialMingling()`:

```r
# Variation in number of species
sp <- paste0(
  rep(LETTERS, each = 26),
  rep(letters, times = 26))
sp_vec <- rep(seq(1,50,1), 10)
sp_list <- lapply(sp_vec, function(x) {
  out <- dataGen(nplots = 1, min_stems = 100, max_stems = 100, species = sp[1:x])
  out$plot_id <- as.character(x)
  out$ba <- basalArea(out$diam)
  return(out)
})

sp_spatialMingling_list <- unlist(lapply(sp_list, function(x) {
  mean(spatialMingling(x$x_grid, x$y_grid, x$species))
}))

ggplot() +
  geom_point(aes(x = sp_vec, y = sp_spatialMingling_list),
    shape = 21, fill = "darkgrey") +
  theme_bw() +
  labs(x = "N species", y = expression(bar(M[i])))
```

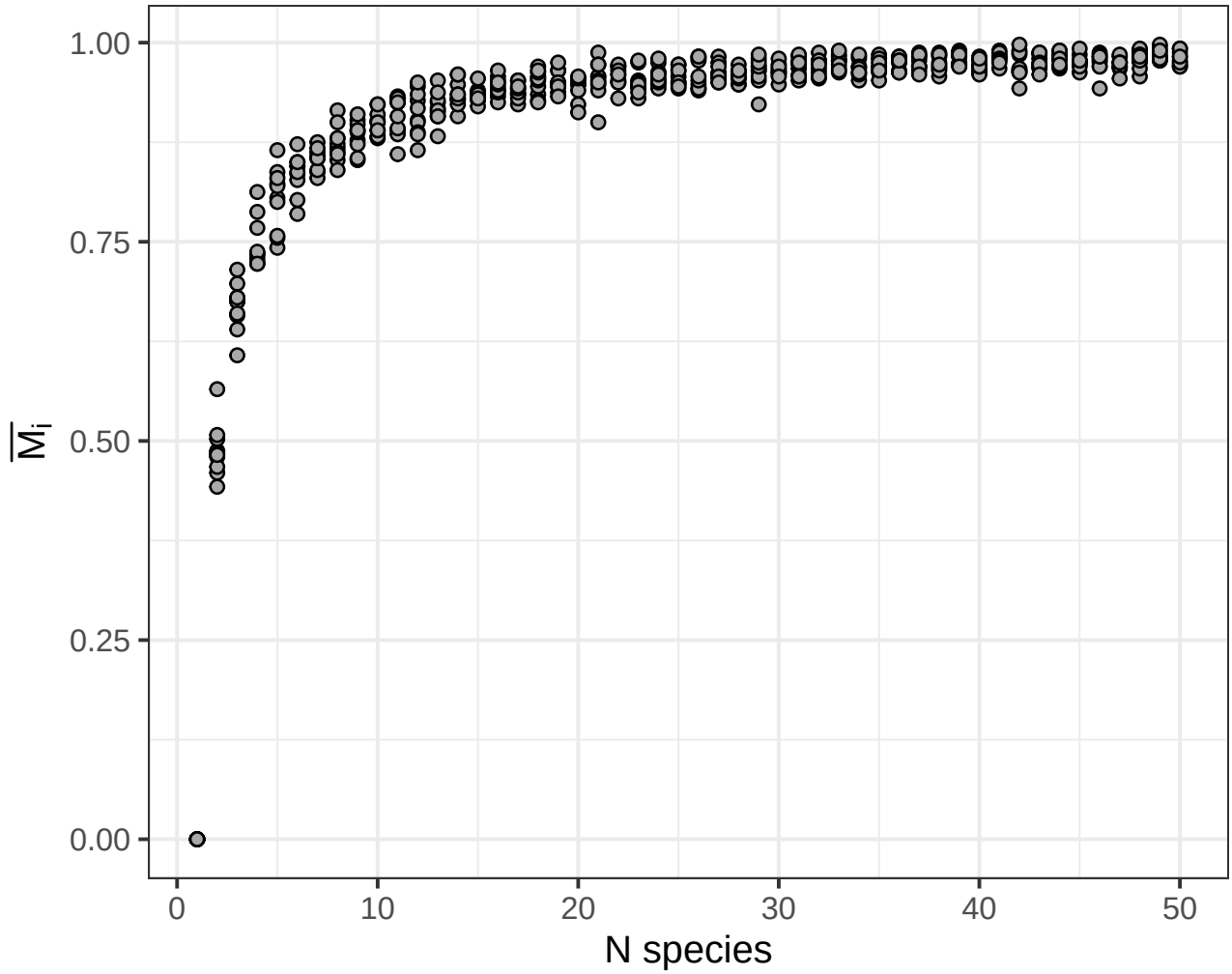

Figure 9: Variation in spatial mingling with increasing number of randomly distributed species.

As does variation in the spatial mixing of species, which is what `spatialMingling()` is designed to detect:

```
# Increasing mixing of species

# Choose 9 species
mi_n_sp <- 9
sp_vec <- LETTERS[1:mi_n_sp]

# Create squares
xy_vec <- seq(0.5,3, 0.5)
dat <- expand.grid(xy_vec, xy_vec)

dat_list <- lapply(sp_vec, function(x) {
  dat$sp <- x
  return(dat)
    })

adj_list <- expand.grid(c(0,3,6), c(0,3,6))

grid_df <- do.call(rbind, lapply(seq_along(dat_list), function(x) {
```

```
  dat_list[[x]]$Var1 <- dat_list[[x]]$Var1 + adj_list[x,1]
  dat_list[[x]]$Var2 <- dat_list[[x]]$Var2 + adj_list[x,2]

  return(dat_list[[x]])
    }))

mi_n_reps <- 100
repl_list <- replicate(mi_n_reps, grid_df, simplify = FALSE)
repl_list <- lapply(repl_list, function(x) {
  x$adj <- 0
  x
    })
repl_list <- list(repl_list)

for (i in seq_len(1000)) {
  repl_list[[i + 1]] <- repl_list[[i]]
  repl_list[[i + 1]] <- lapply(repl_list[[i + 1]], function(x) {
    repls <- sample(seq_len(nrow(grid_df)), 2)
    repl_a <- x$sp[repls[1]]
    repl_b <- x$sp[repls[2]]
    x$sp[repls[1]] <- repl_b
    x$sp[repls[2]] <- repl_a
    x$adj <- i
    x
  })
}

mi_repl_df <- do.call(rbind, mclapply(repl_list, function(x) {
  do.call(rbind, lapply(x, function(y) {
    data.frame(
      adj = y$adj[1],
      spm = mean(spatialMingling(y$Var1,
          y$Var2, y$sp, k = 4, adj = TRUE))
      )
  }))
  }, mc.cores = 4))

mi_repl_df_g <- mi_repl_df %>%
  group_by(adj) %>%
  mutate(run = as.character(row_number()))

ggplot() +
  geom_line(data = mi_repl_df_g,
    aes(x = adj, y = spm, group = run), alpha = 0.5) +
  theme_bw() +
  labs(x = "N substitutions", y = expression(bar(M[i])))
```

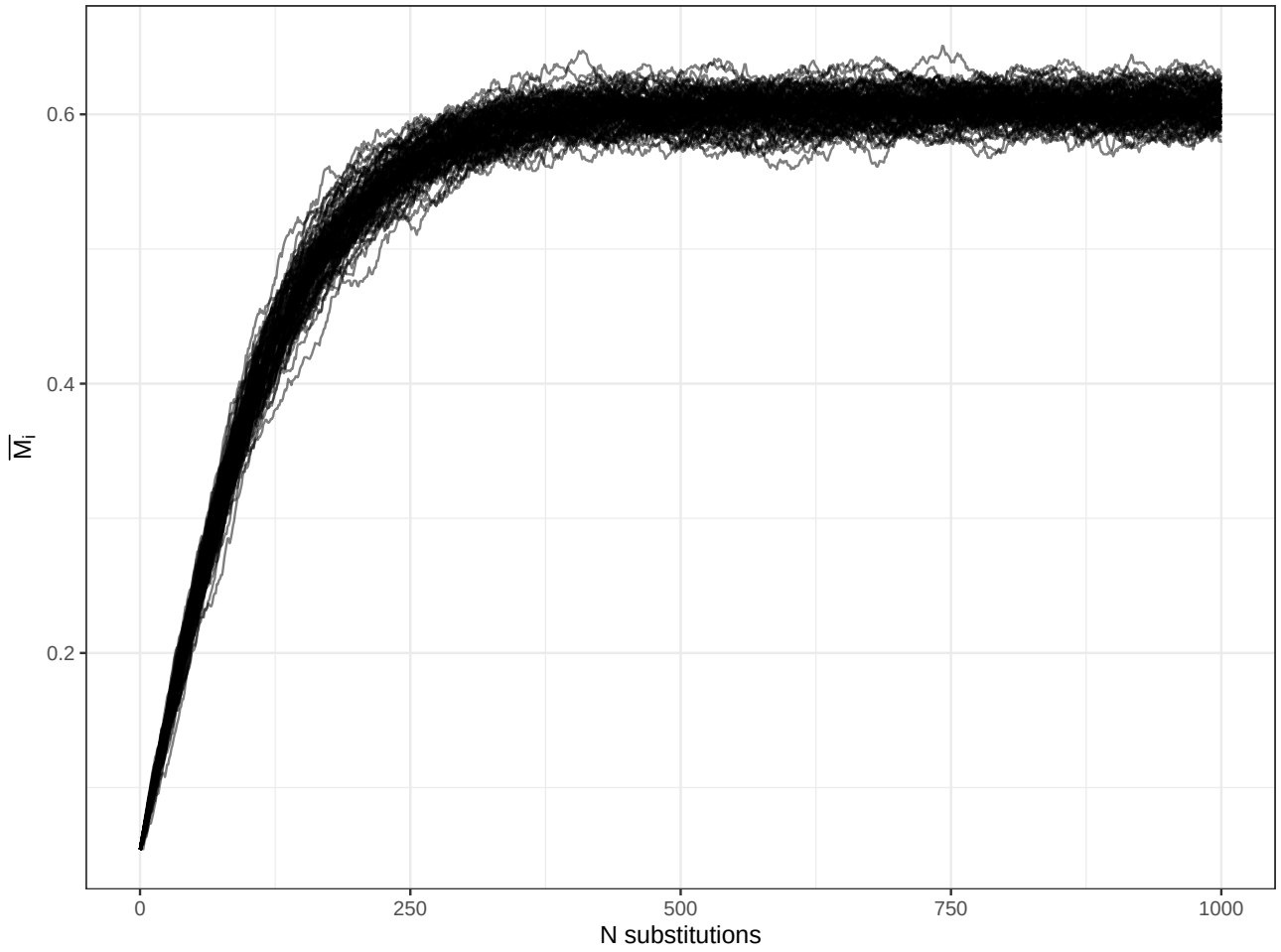

Figure 10: Variation in spatial mingling with increasing mingling of species.

The winkelmass measures the spatial regularity of individuals. The plots below show how the winkelmass varies with increasingly random individual location:

```
xy_vec <- seq(2,50, 4)
dat <- expand.grid(xy_vec, xy_vec)
names(dat) <- c("x", "y")

wi_reps <- 20
wi_list <- replicate(20, dat, simplify = FALSE)
wi_list <- lapply(wi_list, function(x) {
  x$adj <- 0
  x
    })
wi_list <- list(wi_list)

coord_repls <- seq(0,50,0.1)

for (i in seq_len(200)) {
  wi_list[[i + 1]] <- wi_list[[i]]
  wi_list[[i + 1]] <- lapply(wi_list[[i + 1]], function(x) {
    x[sample(nrow(x), 1),c(1,2)] <- sample(coord_repls, 2)
    x$adj <- i
```

```
    x
  })
}
```

```
wi_df <- do.call(rbind, mclapply(wi_list, function(x) {
  do.call(rbind, lapply(x, function(y) {
    data.frame(
      adj = y$adj[1],
      wi = mean(winkelmass(y$x, y$y, k = 4))
      )
    }))
  }, mc.cores = 4))

wi_df_clean <- wi_df %>%
  group_by(adj) %>%
  mutate(run = row_number())

wi_samples <- c(0,50,100,150,200)

wi_plot <- ggplot() +
  geom_line(data = wi_df_clean,
    aes(x = adj, y = wi, group = run)) +
  geom_vline(xintercept = wi_samples,
    colour = "red", linetype = 2) +
  theme_bw() +
  labs(x = "N substitutions", y = expression(bar(W[i]))) +
  theme(legend.position = "bottom")

wi_df_fil <- do.call(rbind,
  lapply(wi_list[wi_samples + 1], "[[", 1)) %>%
  mutate(adj = paste0("N = ", adj)) %>%
  mutate(adj = factor(adj, levels = paste0("N = ", wi_samples)))

wi_map_plot <- ggplot() +
  geom_point(data = wi_df_fil,
    aes(x = x, y = y),
    fill = "darkgrey", shape = 21) +
  facet_wrap(~adj, nrow = 1) +
  theme_bw() +
  theme(
    axis.title = element_blank(),
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    legend.position = "none") +
  labs(x = "X", y = "Y") +
  coord_equal()

wi_plot + wi_map_plot +
  plot_layout(ncol = 1, heights = c(2,1))
```

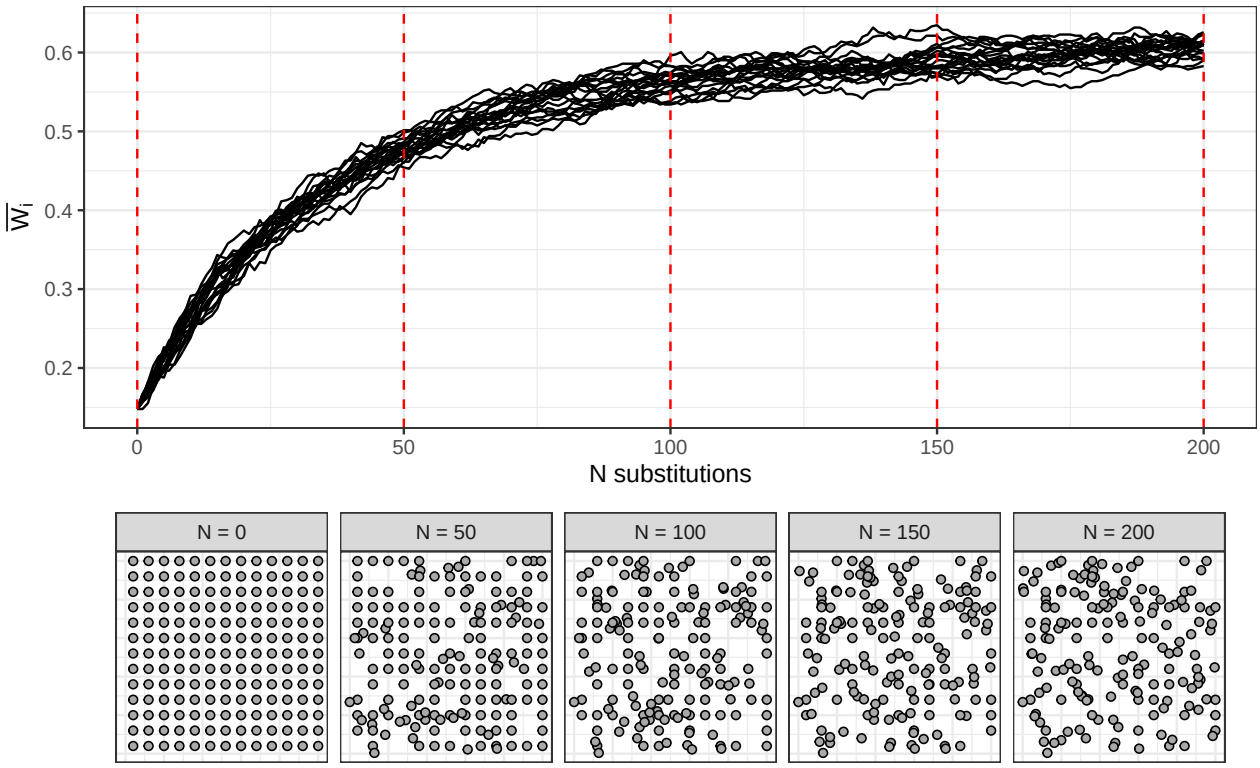

Figure 11: Variation in the winkelmass with increasing spatial randomness of individual location.