compInd package vignette

John L. Godlee

1st August 2024

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.

```r
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:

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

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

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

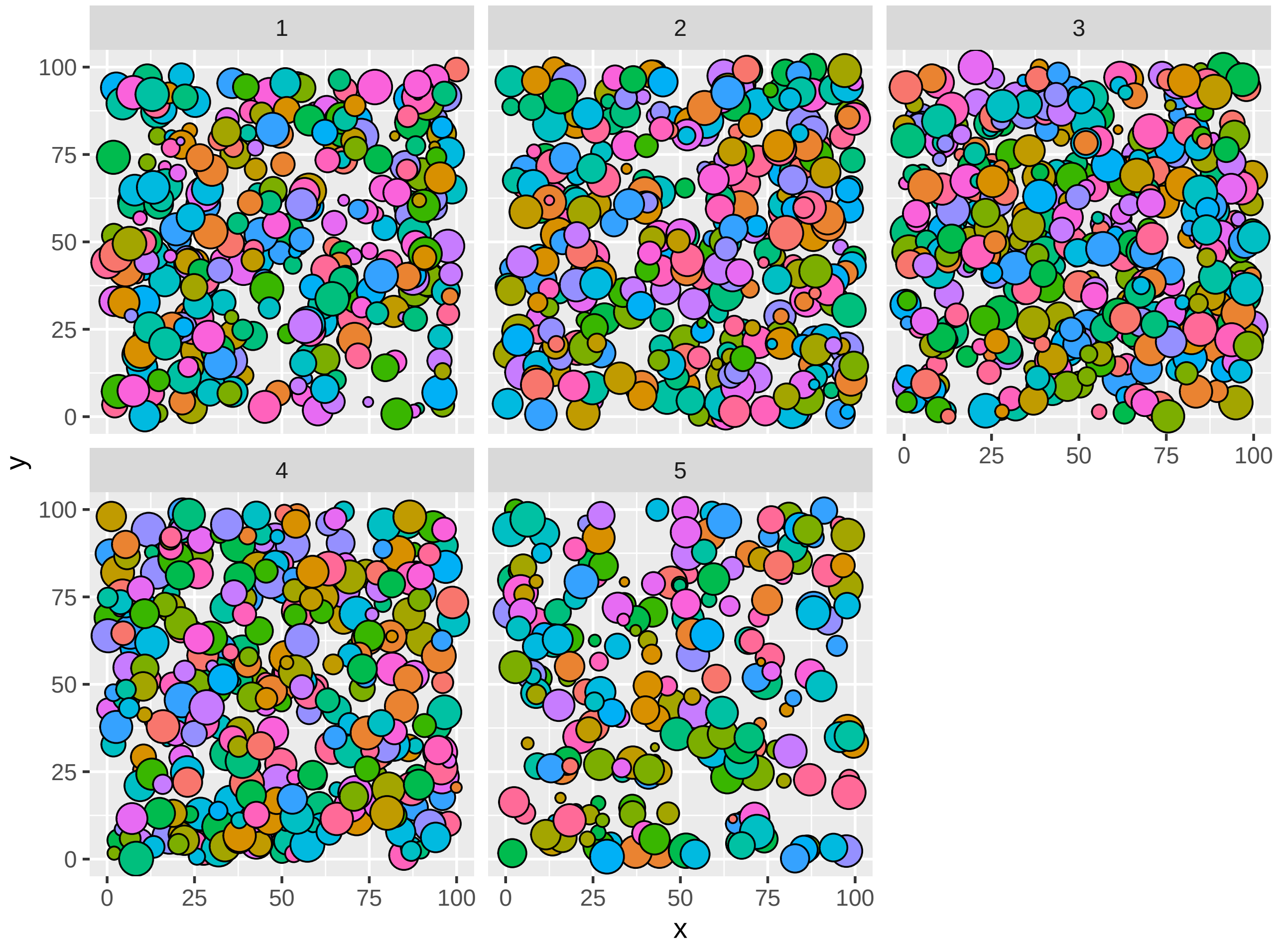
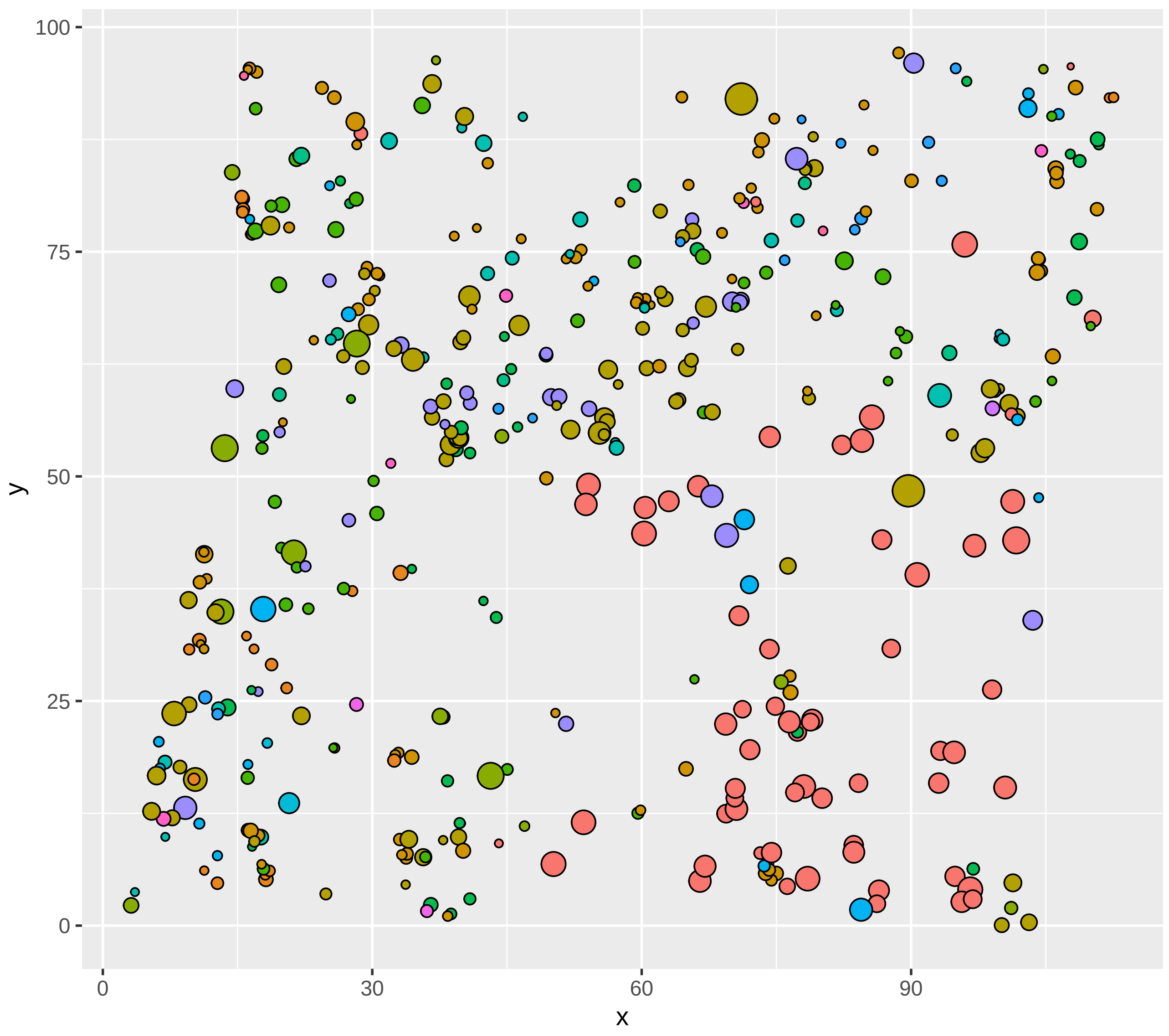


Figure 1: Five plots with randomly located stems of different species.

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, y = y, fill = species, size = diam),
    shape = 21) +
  theme(legend.position = "none") +
  coord_equal()
```



Generate some extra columns, like basal area from stem diameter:

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

Many of the functions in `compInd` are calculated for each individual in the structural unit (normally a plot), 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.

```r
# Split data into list of plots
dat_split <- split(dat, dat$plot_id)

# For each plot
dat_neighb <- lapply(dat_split, function(x) {
  # Calculate nearest neighbours
  neighb <- nearNeighb(x[,c("x", "y")], k = 4)

  # Add auxiliary information
  out <- lapply(neighb, function(y) {
    # Add data for focal tree
    y_f <- x[y$focal, c("diam", "species", "x", "y")]
    names(y_f) <- paste0("focal_", names(y_f))

    # Add data for neighbouring trees
    y_nb <- x[y$nb, c("diam", "species", "x", "y")]
    names(y_nb) <- paste0("nb_", names(y_nb))

    # Combine dataframes
    cbind(y, y_f, y_nb)
  })
  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()`
- `domConInd()`

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

```r
# For each plot
alemdag_list <- lapply(dat_neighb, function(x) {
  # For each individual
  unlist(lapply(x, function(y) {
    # Calculate competition index
    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 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. It can also be informative to look at the distribution of values by calculating the standard deviation.

Other functions for competition indices, those which don't rely on a zone of competition, act on each structural unit (normally a 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:

```
# For each plot
clarkEvans_list <- lapply(dat_split, function(x) {
  # Calculate competition index
  clarkEvans(x[,c("x", "y")], 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 explained above.

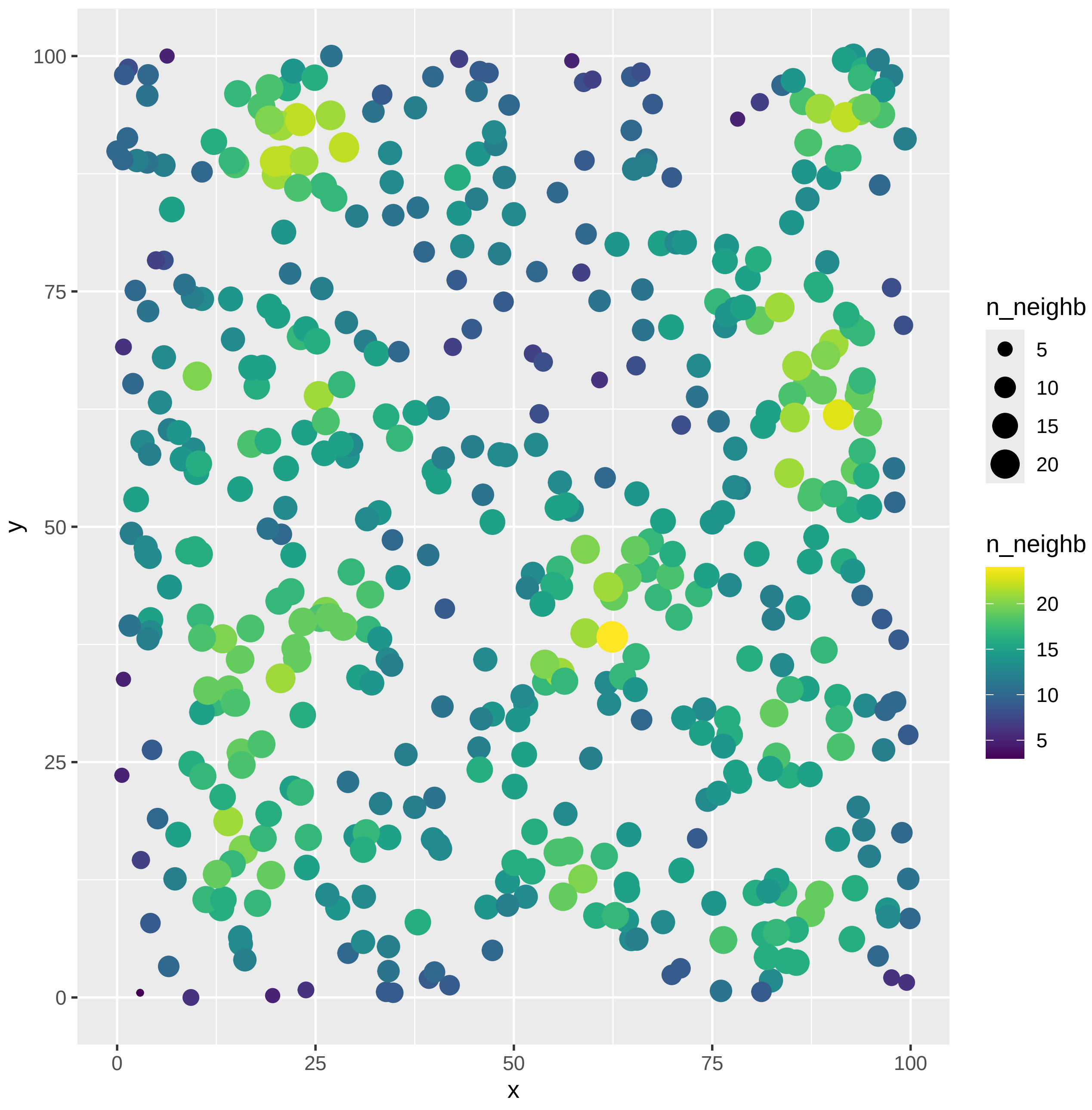
`lorimerCZR()` can be used to estimate the competition zone radius for a structural unit based on the number of individuals in the unit. The output of `lorimerCZR()` could then be fed to either `nearNeighb()` as the `radius` argument, or to `lorimerComp()`, which uses this value to normalise the result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

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


## [1] 4.663
```

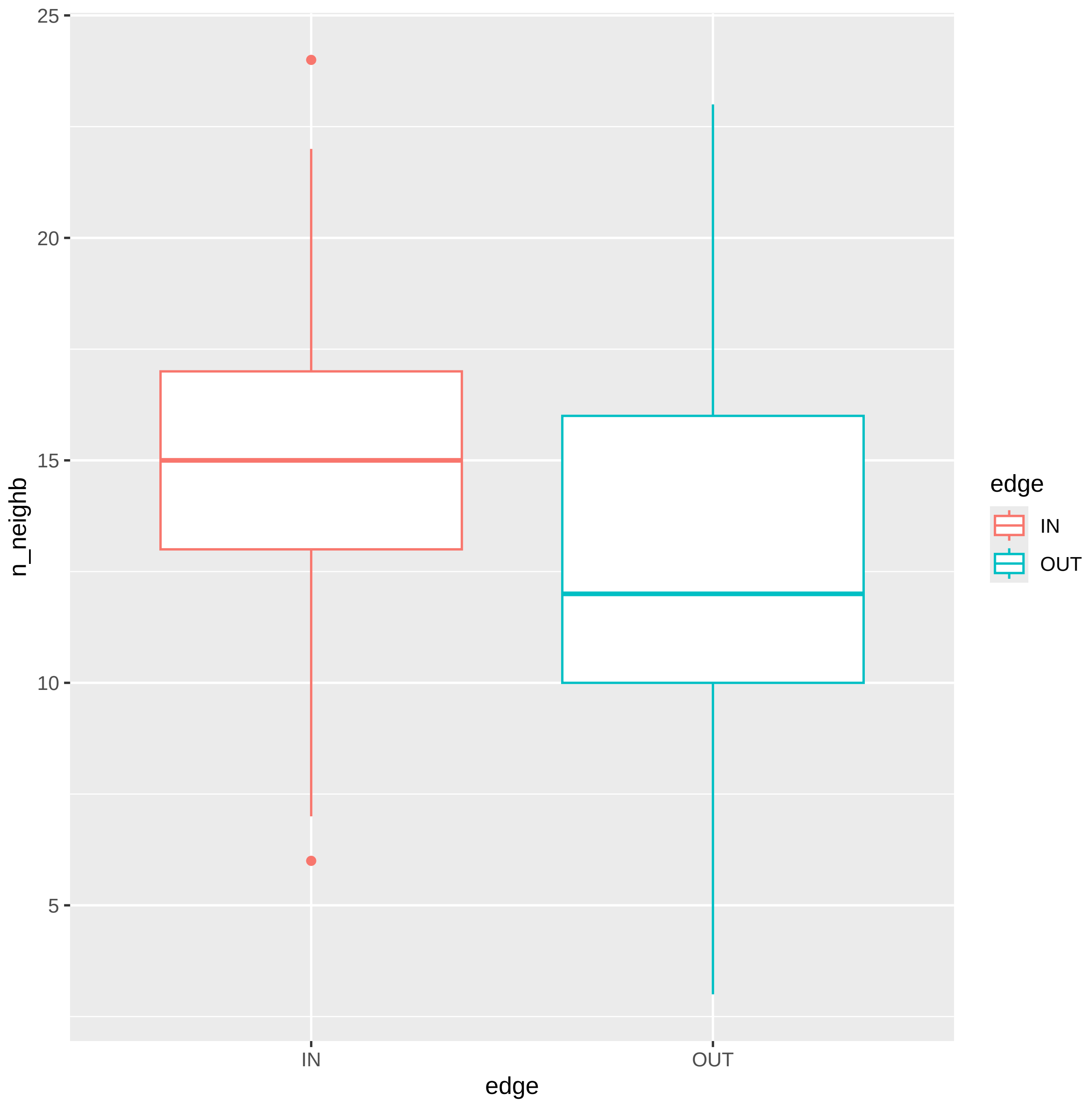
`edgeExclude()` can be used to find individuals that are sufficiently distant from the edge of a plot. For competition indices which rely on a competition zone radius such as `lorimerComp()`, `alemdag()` etc, individuals near the edges of the plot will erroneously under-estimate competition compared to other individuals in the plot interior, because individuals outside the plot are not recorded. It is common practice to exclude focal individuals if they are located a distance from the 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()`:

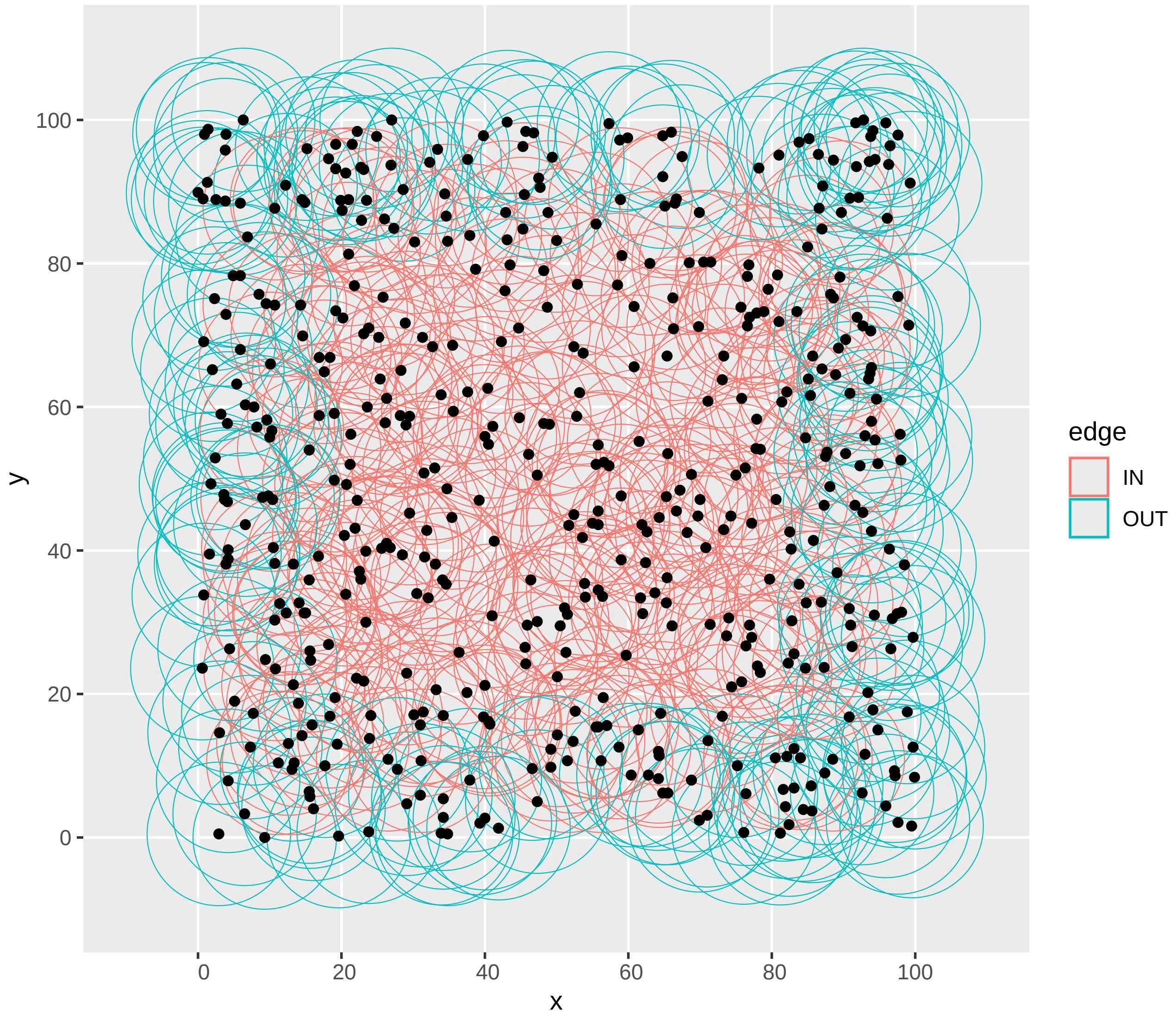
```r
# Generate data for one plot, 500 stems
dat <- dataGen(nplots = 1, min_stems = 500, max_stems = 500)

# Find nearest neighbours for each individual
neighb_rad10 <- nearNeighb(dat[,c("x", "y")], 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_point(data = dat,
    aes(x = x, y = y, size = n_neighb, colour = n_neighb)) +
  scale_colour_viridis()
```



```
# Find if an individual is in the edge of a plot
dat$edge <- "OUT"
dat$edge[edgeExclude(dat[,c("x", "y")], 10, 0, 100, 0, 100)] <- "IN"

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



```
# Create buffers around individuals to illustrate competition zone radius
dat_buffer <- st_as_sf(dat, coords = c("x", "y")) %>%
  st_buffer(., 10)

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



edge
IN
OUT
y
x
0
20
40
60
80
100