Building a tree inventory

Roberto Molowny-Horas

2022-12-15

Initialize

Creating a sf object containing the individual tree diameters of an inventory is straightforward. We will exemplify for the case of Spain. We start by creating an initial sf object. At this stage, only labels (unique for every tree plot/stand/station) and x, y coordinates for the locations are required. In addition, we may also add information about the reference system, though this can be done later by hand. Thus:

```r
a <- fpm::start_stand("ID1", 5, 45, "EPSG:4326")
```

is equivalent to this:

```r
a <- fpm::start_stand("ID1", 5, 45)
sf::st_crs(a) <- "EPSG:4326"
```

The easiest way to initialize at once an sf object containing many empty plots is to give vectors as inputs (in this example, coordinates are random latitudes and longitudes):

```r
a <- fpm::start_stand(paste0("ID", 1:10), runif(10, 0, 5), runif(10, 35, 45))
```

Later on, new plots can be also added by using the merge_stand function. However, for this function to work we must set in advance the reference system of both sf objects.

```r
sf::st_crs(a) <- "EPSG:4326"
b <- fpm::start_stand(paste0("ID", 25:27), runif(3, 0, 5), runif(3, 35, 45), "EPSG:4326")
a <- fpm::merge_stands(a, b)
```

Include tree and/or sapling data

The sf object initalized in the previous calculations does not yet contain seedling, sapling or tree data. Therefore, our next step will consist of adding those data to the plots. For that purpose we can use the function build_stand. Notice that, at this stage, the attribute *country* should have been set previously in *a* to any of the available options (at this moment, only “spain”, “france” or “usa” are allowed). Moreover, that attribute and the *country* input in build_stand must match. Otherwise, build_stand will complain.

For the sake of illustration, we will add trees and saplings to two different sf objects and will then merge them to create a new, larger sf object. Some of the plots (those from *a*) will have tree information, whereas some others (those from *b*) will not. Nevertheless, they can all be merged together to build a larger sf.

```r
# Creating random trees and adding them to each stand.
a <- fpm::set_attributes(a, country = "spain")
fd <- c(127.324, 31.83099)
for (i in a$idplot) {
  df <- data.frame(species = c(sample(c("Pnigra","Phalep"),5,replace=T)),
                   dbh1 = 7.5+runif(5)*20, factor_diam1 = sample(fd,5,replace=T))
  a <- fpm::build_stand(a, i, df,
                        data_type = "trees",
                        stand_type = "individual",
                        date = 2000,
                        country = "spain")
}
b <- fpm::start_stand(paste0("ME", 450:500), runif(51, 0, 5),
                      runif(51, 35, 45), "EPSG:4326")
b <- fpm::set_attributes(b, country = "spain")
a <- fpm::merge_stands(a, b)
```

Similarly, seedling and/or sapling data can be added to some plots.

```
# Creating random trees and adding them to each stand.
a <- fpm::set_attributes(a, country = "spain")
for (i in a$idplot) {
  df <- data.frame(species = c("Pnigra","Phalep"),
                   N = rpois(2,30))
  a <- fpm::build_stand(a, i, df,
                        data_type = "saplings",
                        stand_type = "individual",
                        date = 2000,
                        country = "spain")
}
```

Smoothing of discrete tree data

So far we have dealt with discrete tree data that had been collected from field studies. If we wish to apply the IPM methodology to our data, we must describe the tree population at each station in terms of a continuous distribution of number of trees as a function of e.g. diameter at breast height. Forest inventories represent a collection of discrete data which are not well suited for IPMs and, as a previous step, they must be converted into continuous distribution of e.g. size. For that purpose, function $\texttt{smooth}_{\texttt{s}}\texttt{tand}$ can be used to smooth the discrete data to obtain a continuous distribution, for each species.

Before $\texttt{smooth}_{\texttt{s}}\texttt{tand}$ can be applied, we must determine in advance the abscissas that will be employed in all future calculations and store them as an attribute in the `sf` object. Again for the sake of illustration, let us assume that diameters will range from a minimum of 7.5 cm to a maximum that will depend on each species. Then, the resulting `data.frame` will be stored as an attribute labeled “integvars”.

```
x <- data.frame(Pnigra = seq(7.5,200,length=1000), Phalep = seq(7.5,250,length=1000))
b <- a
b <- fpm::set_attributes(b, integvars = x)
```

Next, we give $\texttt{smooth}_{\texttt{s}}\texttt{tand}$ the `sf` object with the discrete tree data and the function will smooth/convolve the data with a smooth distribution (default is a gaussian curve) to obtain a continuous distribution of sizes. In the case of Spain, to conserve the total number of trees the results will be multiplied by the corresponding *factor_diam1* factor to account for variables stand radius.

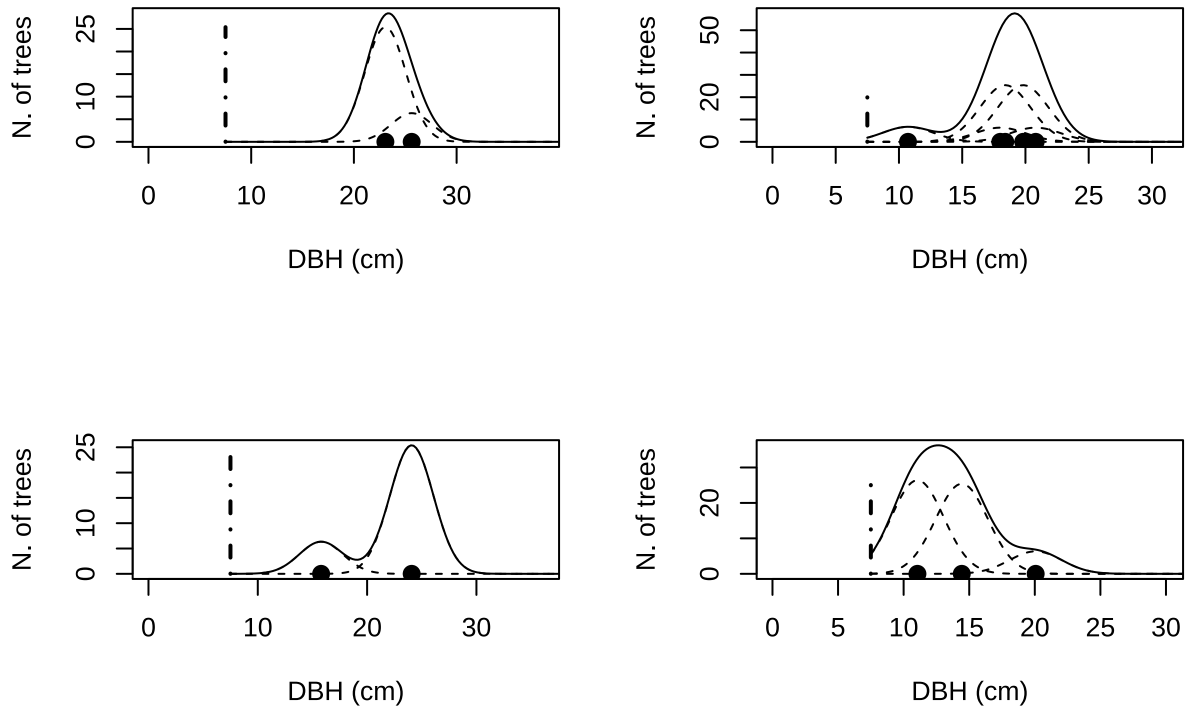
```
b <- fpm::smooth_stand(b)
```

Notice that, although the number of trees is preserved, the basal area is not due to the square-diameter factor inside the integral. However, if the width of the smoothing window is not very large, the basal area computed with the discrete values and that obtained with the smoothed distribution are not very different.

An example of the smoothing internal algorithm is shown below, where the resulting tree size distributions for one species at four tree stand are shown together with the contributions from each individual tree. The thick vertical dashed-dotted line on the left indicates the minimum diameter in the inventory (7.5 cm).

```
par(mfcol=c(2,2))
for (i in 1:4) {
  sp <- a[i, ]$trees[[1]]$species[1]
  xx <- x[, sp]
  y <- subset(a[i, ]$trees[[1]], species == sp)
  z <- sapply(1:nrow(y), function(j) {
    MiscStat::fast_kernsmooth(xx, y$dbh1[j] , width = 2) * y$factor_diam1[j]})

  plot(x[, sp], b[i, ]$trees[[1]][, sp], xlab = "DBH (cm)", ylab = "N. of trees",
       xlim = c(0, max(y$dbh1)*1.5), type = "l")
  for (j in 1:nrow(y)) points(xx, z[, j], type = "l", lty = 2)
  points(y$dbh1,rep(0, nrow(y)), pch = 16, cex = 1.5)
  points(c(7.5, 7.5), c(0, max(z)), type = "l", lty = 4, lwd = 2)
}
```



```
par(mfcol=c(1,1))
```