An Introduction to treePlotArea

Andreas Dominik Cullmann

July 10, 2022

Contents

1 What treePlotArea Does

The German national forest inventory uses angle count sampling, a sampling method invented by Bitterlich in 1947 ([1]) and extended by Grosenbaugh [2] as probability proportional to size sampling. When plots are located near stand boundaries, their sizes and hence their probabilities need to be corrected.

1.1 Why corrections?

As you can see in Figure 1, a screenshot from `bwi2022de`, the software used for the field surveys of the German national forest inventory in 2022, the tree plot area of tree 7 of corner 2 of tract 166 gets intersected by two stand boundaries. So its plot area needs to be corrected.

1.2 What did we do before?

Until now, the German national forest inventory used a program called `grenzkreis` that was written by Bernhard Bösch at the Forest Research Institute of Baden-Württemberg. It is written in C++, poorly documented and, to our poor understanding, hard to maintain. The last version dates from March 2011. We felt the need to re-write the library in R. Gerald Kändler used `grenzkreis` to provide reference data to this package, which we use for testing.

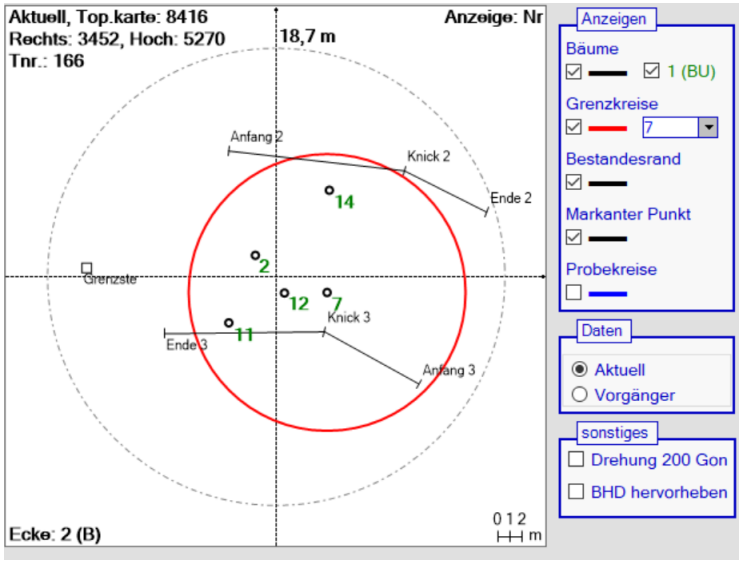
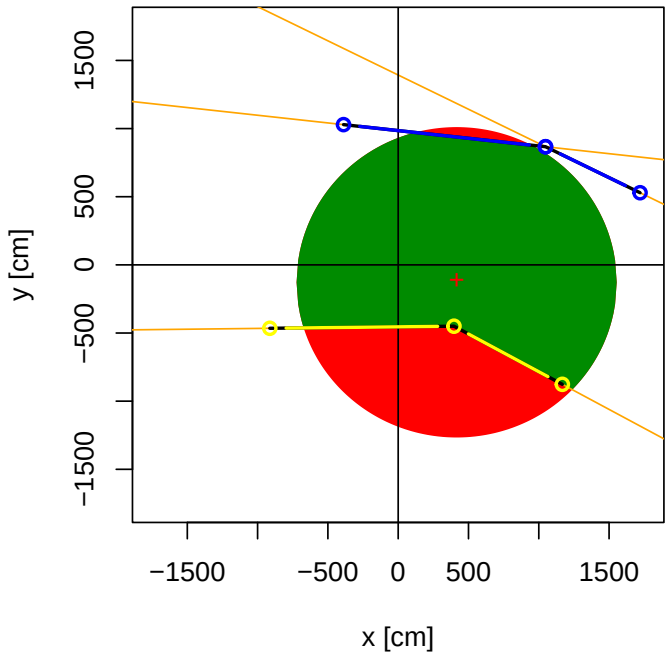


Figure 1: Tree plot from `bwi2022de`.

1.3 Single Tree Plot Areas

If we give a tract, corner and tree id, we can plot the effective tree plot area and output the ratio of the theoretical (red) to the effective (green) tree plot area:

```
> library("treePlotArea")
> tnr <- 166
> enr <- 2
> bnr <- 7
> angle_counts <- bw2bwi2022de(get(data("trees",
+                                       package = "treePlotArea")))
> boundaries <- get(data("boundaries", package = "treePlotArea"))
> cf <- plot_tree_plot_area(angle_counts = angle_counts,
+                           boundaries = boundaries,
+                           tnr =  tnr, enr = enr, bnr = bnr,
+                           frame_factor = 0.35)
```



Now we compare the correction factor to the reference value given by `grenzkreis`:

```
> print(cf)

[1] 1.357454

> angle_counts[angle_counts[["tnr"]] == tnr &
+              angle_counts[["enr"]] == enr &
+              angle_counts[["bnr"]] == bnr, "kf2"]

[1] 1.357323
```

This is rather close, the difference probably due to different approximations of circles by polygons in the two libraries/packages.

1.4 Tree Plot Areas for Forest Inventories

Of course we do not look at single trees, we want to get the correction factors for collections of trees.

```
> nrow(angle_counts)

[1] 1121

> nrow(boundaries)

[1] 148
```

```
> correction_factors <- get_correction_factors(angle_counts,
+                                              boundaries,
+                                              verbose = FALSE)
> print(subset(correction_factors, tnr == 166 & enr == 2))
```

```
    tnr enr bnr correction_factor
105 166   2  11          1.805366
106 166   2  12          1.152779
107 166   2  14          1.632166
108 166   2   2          1.023632
109 166   2   7          1.357454
```

We check that the correction factors do not differ too much from the reference values:

```
> m <- merge(angle_counts[TRUE, c("tnr", "enr", "bnr",
+                                 "kf2", "pk", "stp")],
+            correction_factors)
> m[["diff"]] <- m[["correction_factor"]] - m[["kf2"]]
> m <- select_valid_angle_count_trees(m)
> rdiff <- ifelse(m[["kf2"]] == 0,
+                 m[["diff"]] / (m[["kf2"]] + 1e-10),
+                 m[["diff"]] / m[["kf2"]])
> works <- RUnit::checkTrue(all(abs(rdiff) < 0.001))
> if (works) {
+     print("Concurs with tests in runit/")
+ } else {
+     stop("Break vignette.")
+ }
```

```
[1] "Concurs with tests in runit/"
```

2 Customizing Data

We have used data coming with package: one data frame containing angle counts (trees), and one containing boundaries.

The names of the data columns essential to this package are retrieved from the global option `treePlotArea`. We can set it to its default (which is done by the package wherever needed without overwriting an slots in that list already set) and then inspect it:

```
> set_options()
> str(getOption("treePlotArea"))
```

```
List of 2
 $ angle_counts:List of 6
  ..$ tract_id : chr "tnr"
  ..$ corner_id: chr "enr"
  ..$ tree_id  : chr "bnr"
  ..$ distance : chr "hori"
  ..$ azimuth  : chr "azi"
```

```
  ..$ dbh      : chr "bhd"
 $ boundaries  :List of 10
  ..$ tract_id        : chr "tnr"
  ..$ corner_id       : chr "enr"
  ..$ boundary_type   : chr "rart"
  ..$ boundary_status : chr "rk"
  ..$ distance_start  : chr "spa_m"
  ..$ distance_flexing: chr "spk_m"
  ..$ distance_end    : chr "spe_m"
  ..$ azimuth_start   : chr "spa_gon"
  ..$ azimuth_flexing : chr "spk_gon"
  ..$ azimuth_end     : chr "spe_gon"
 - attr(*, "package")= chr "treePlotArea"
```

The columns of our data frames could have different names (we just convert them to upper case here):

```
> names(angle_counts) <- toupper(names(angle_counts))
> names(boundaries) <- toupper(names(boundaries))
```

We would then have to set the options accordingly

```
> option_list <- sapply(get_defaults(), function(x) lapply(x, toupper))
> set_options(angle_counts = option_list[["angle_counts"]],
+             boundaries = option_list[["boundaries"]])
```

and the results stay the same:

```
> correction_factors_upper <- get_correction_factors(angle_counts,
+                                                    boundaries,
+                                                    verbose = FALSE)
> RUnit::checkEquals(correction_factors_upper, correction_factors,
+                    checkNames = FALSE)
```

```
[1] TRUE
```

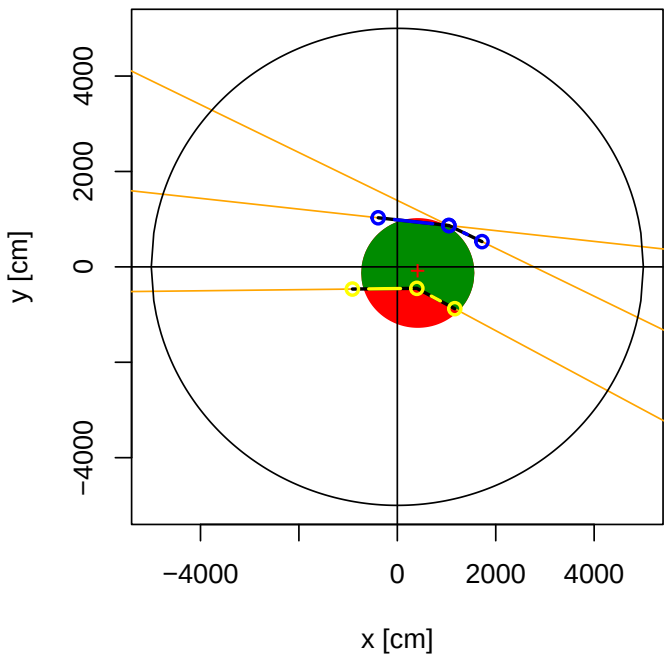
3 How treePlotArea Works

This is the rather technical part, you might not be interested ...

3.1 Flexed Boundaries

First, we decide whether a flexed boundary is flexed towards the center of the corner or away from it:

```
> cf <- plot_tree_plot_area(angle_counts = angle_counts,
+                           boundaries = boundaries,
+                           tnr =  tnr, enr = enr, bnr = bnr,
+                           frame_factor = 1)
```



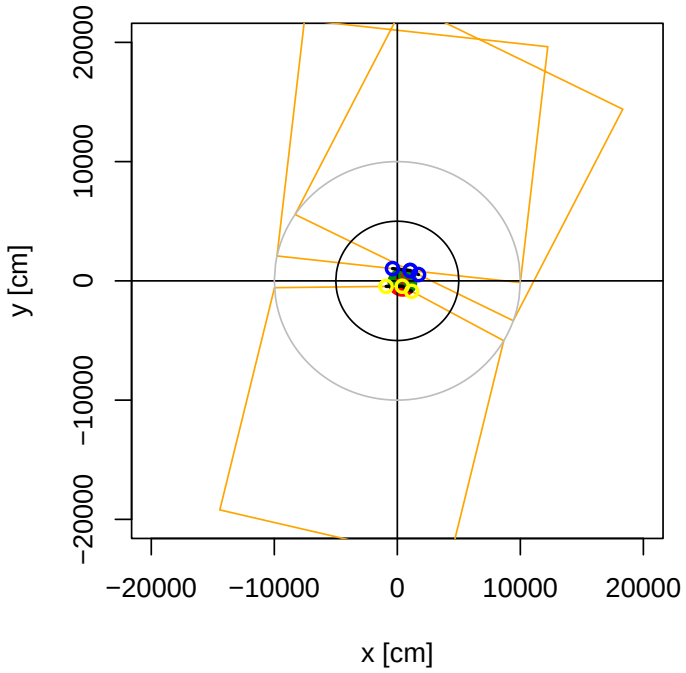
Yellow lines mark boundaries flexed away from the center, blue ones boundaries flexed towards it.

How do we decide whether a flexed boundary is 'flexed towards the center'? We build triangles from each flexed boundary that intersect a circle with a radius of 50 km. If the tree coordinates do fall into that triangle, the boundary is 'flexed towards the center'.

3.2 Boundary Polygons

We then build polygons (depending on the type of boundary):

```
> cf <- plot_tree_plot_area(angle_counts = angle_counts,
+                           boundaries = boundaries,
+                           tnr =  tnr, enr = enr, bnr = bnr,
+                           frame_factor = 4)
```



Boundaries flexed towards the center are split into two straight lines. For each straight line, a square 'away from the center' is defined. For boundaries flexed away, pentagon 'away from the center' are defined.

'Away from the center' means that the polygons are created such that their corner towards the center lie on a circle with radius 100 meter. We choose this value because for a tree with a breast height diameter of 2000 millimeter, the tree plot area for angle count factor 4 (which is used in the German national forest inventory) will be a circle with a radius of 50m. We just double that value to make sure no tree plot area could possibly stick out of our polygons somewhere. We then use package `sf` to intersect the tree plot area with the boundary polygons. That's it.

References

[1] W. Bitterlich. Die Winkelzählmessung. *Allgemeine Forst- und Holzwirtschaftliche Zeitung*, 58, 1947.

[2] L. R. Grosenbaugh. Plotless timber estimates – new, fast, easy. *Journal of Forestry*, 1952.