How to use package spatcart

Package installation

spatcart may be installed with the following command

```
library(devtools)
devtools::install_github("Servane-Gey/spatcart")
```

or directly from the source file spatcart_1.0.0.tar.gz or the binary file spatcart_1.0.0.tgz.

Load the package, and let's start.

```
library(spatcart)
```

Package description

spatcart provides classification trees adapted to bivariate marked spatial point processes by using a splitting criterion associated with the intertype K-function interaction between marks proposed by Ripley. The package also provides heatmap colored partitions induced by tree leaves with respect to interaction between marks.

The full description of the SpatCART algorithm can be found in the article *Spatial CART Classification Trees*, by A. Bar-Hen, S. Gey and J.-M. Poggi (2020).

Main features

The main features of **spatcart** are presented with respect to the lines followed in the article *Spatial CART Classification Trees*.

Spatial CART classification trees

- **spatcart** constructs the Spatial CART classification trees using Ripley's intertype K-function as impurity function to partition bivariate marked spatial point process, with pruning. Also provides optional graphical results on the obtained partitions. The output values of the function *spatcart()* are listed in the following objects:
  - *max.tree*: maximal tree, of class 'tree'.
  - *pruned.seq*: sequence of subtrees pruned from the maximal one, list of corresponding number of leaves, pruning complexities, and deviance.
  - *opt.tree.max*: largest optimal tree, selected among the pruned sequence with respect to the modified largest gap slope heuristic (see article). Object of class 'tree'.
  - *opt.tree.min*: smallest optimal tree, selected among the pruned sequence with respect to the largest plateau slope heuristic (see article). Object of class 'tree'.
  - *cp*: numeric vector of inside node scale resolutions in maximal tree.
  - K: numeric vector of inside node impurities in maximal tree.

Load the *ants* data set from package *spatstat*:

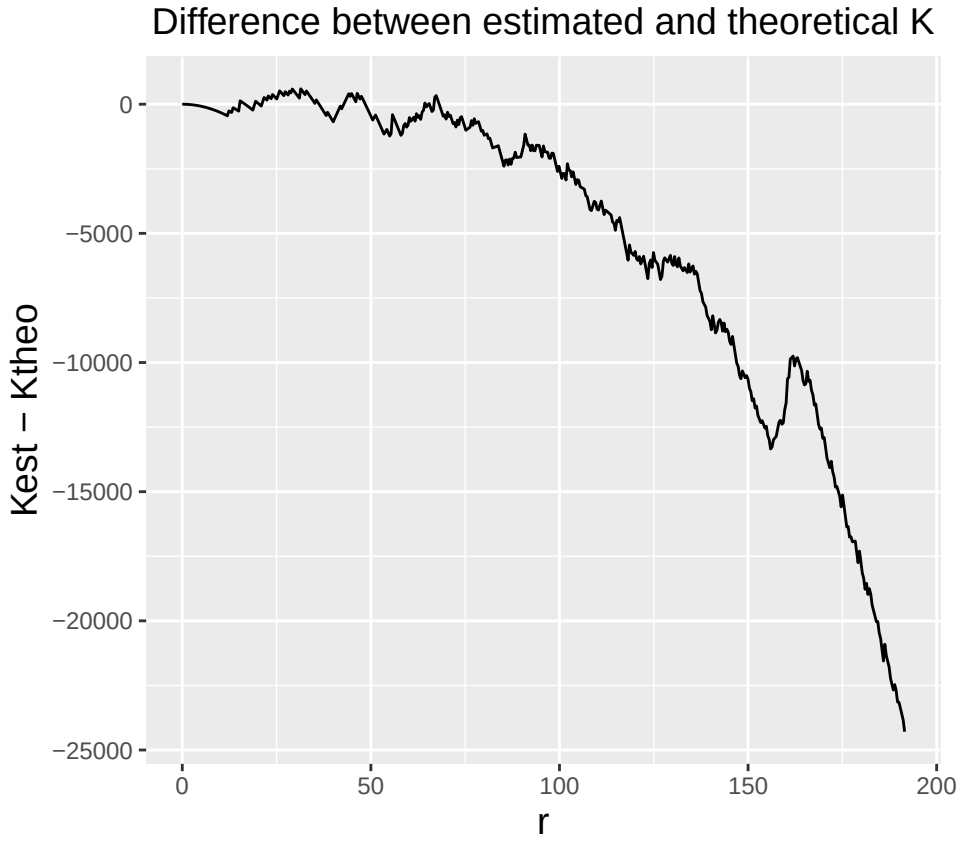
```
library(spatstat)
ypp = ants
ypp
```

```
#> Marked planar point pattern: 97 points
#> Multitype, with levels = Cataglyphis, Messor
#> window: polygonal boundary
#> enclosing rectangle: [-25, 803] x [-49, 717] units (one unit = 0.5 feet)
```

Look at the difference between estimated and theoretical uncorrected intertype K-functions to determine initial scale resolution to put into function *spatcart()*:

```
library(dplyr)
library(ggplot2)
major =  names(which.max(intensity(ypp)))
minor = names(which.min(intensity(ypp)))

K0 = Kcross(ypp, major, minor, correction="none")
K01 = tibble(
    scale = K0$r,
    difference = K0$un-K0$theo
    )
ggplot(K01, aes(x=scale, y=difference))+geom_line()+xlab("r")+
  ylab("Kest - Ktheo")+
  ggtitle("Difference between estimated and theoretical K")+
  theme(axis.title.x = element_text(size=14))+
  theme(axis.title.y = element_text(size=14))+
  theme(plot.title = element_text(hjust = 0.5, size=14))
```

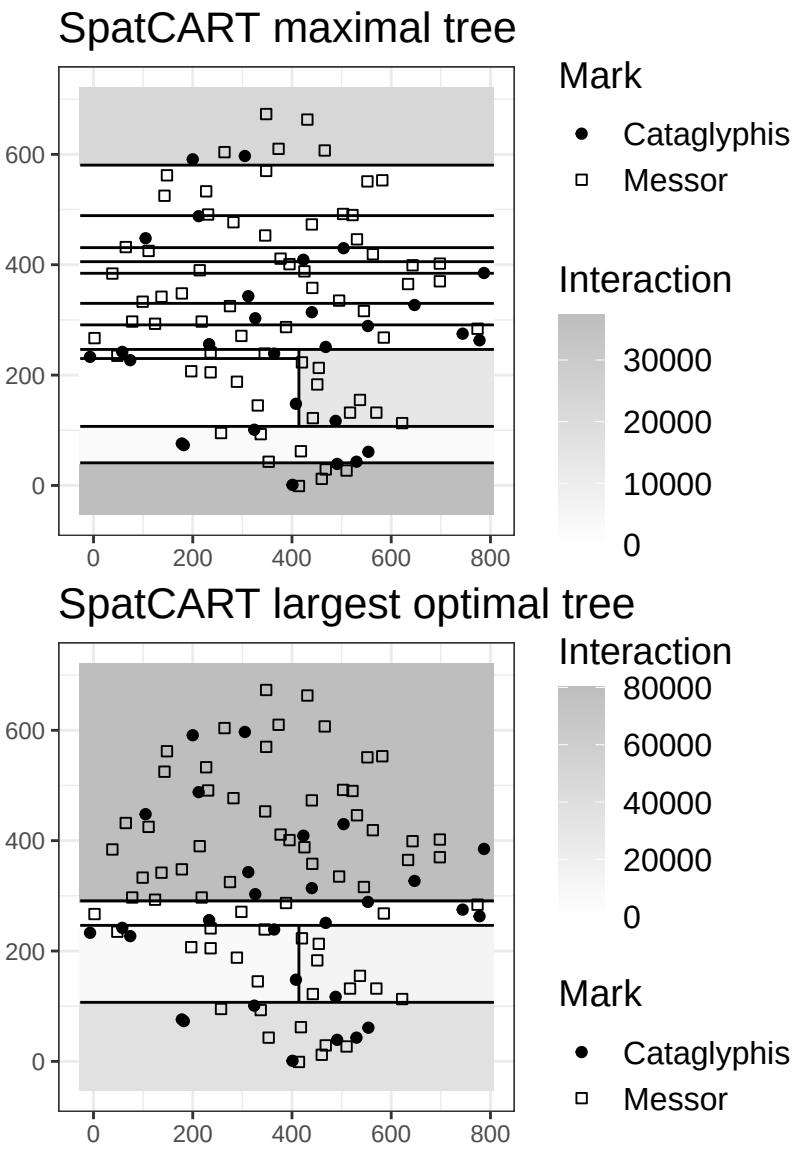


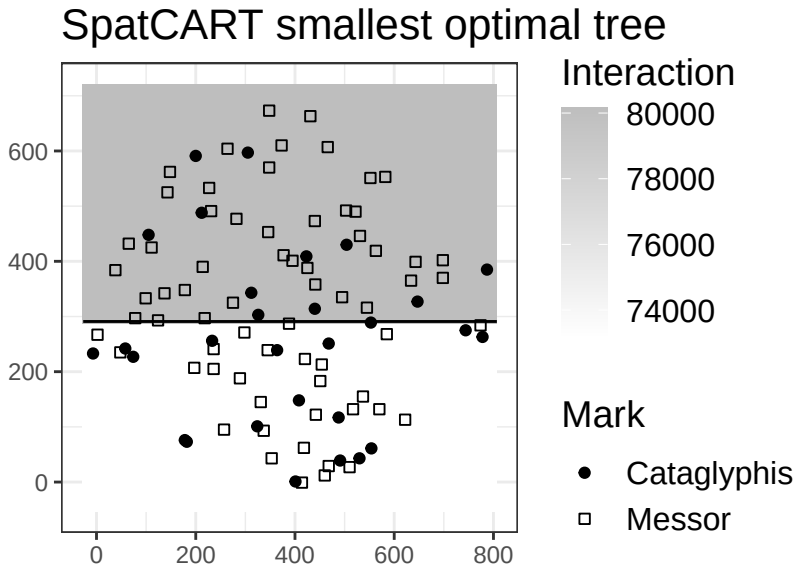
For example, select the value *r0* for initial resolution scale as the local minimum between r=150 and r=170:

```
zone = which(K0$r>=150 & K0$r<=170)
r1 = K0$r[zone]
r0 = r1[which.min((K0$un-K0$theo)[zone])]
r0
#> [1] 155.9678
```

Apply function *spatcart()* on the ants data set, with initial resolution *r0*:

```
t = spatcart(ypp, r0)
```





Here optional parameters are set as default: *ties = TRUE*, *method = "deviance"*, *minsplit = 10*, *minleaf = 5* and *graph = TRUE*.

This produces a maximal tree having at least *minleaf = 5* points in each leaf, and stopping splitting process if a node contains less than *minsplit = 10* points. Let us note that *minsplit* must be at least equal to 2x*minleaf*.

ties = TRUE does not optimize on tie splits, and take the lowest value for the corresponding split. To optimize on tie splits using scale adaptation, set as *ties = FALSE*.

method = "deviance" produces class probability trees using Gini's criterion to prune the maximal tree. To obtain classification trees using misclassification criterion, set *method = "misclass"*.

graph = TRUE produces heatmap colored partitions induced by respectively the maximal, largest optimal and smallest optimal trees' leaves with respect to the interaction between marks.

- **spattree** constructs the maximal Spatial CART classification tree using Ripley's intertype K-function as impurity function to partition bivariate marked spatial point process. The output values of the function *spattree()* are listed in the following objects:
  - *max.tree*: maximal tree, of class 'tree'.
  - *cp*: numeric vector of inside node scale resolutions in maximal tree.
  - *K*: numeric vector of inside node impurities in maximal tree.

Construct the maximal Spatial CART classification tree at initial scale resolution *r0* on the *ants* data set:

```
tmax = spattree(ypp, r0)
```

Here optional parameters are set as default: *ties = TRUE*, *minsplit = 10* and *minleaf = 5*.

This produces a maximal tree having at least *minleaf = 5* points in each leaf, and stopping splitting process if a node contains less than *minsplit = 10* points. Let us note that *minsplit* must be at least equal to 2x*minleaf*.

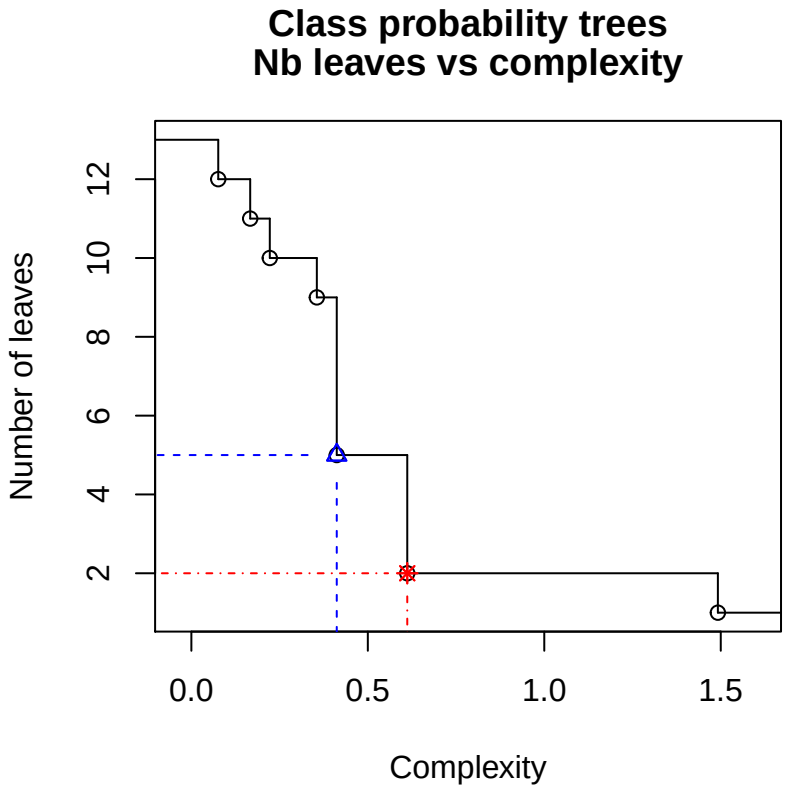
ties = TRUE does not optimize on tie splits, and take the lowest value for the corresponding split. To optimize on tie splits using scale adaptation, set as *ties = FALSE*.

- **spatprune** prunes a tree. The output values of the function *spatprune()* are listed in the following objects:
  - *pruned.seq*: sequence of subtrees pruned from the maximal one, list of corresponding number of leaves, pruning complexities, and deviance.
  - *opt.tree.max*: largest optimal tree, selected among the pruned sequence with respect to the modified largest gap slope heuristic (see article). Object of class 'tree'.

– *opt.tree.min*: smallest optimal tree, selected among the pruned sequence with respect to the largest plateau slope heuristic (see article). Object of class 'tree'.

Prune the maximal tree to obtain the pruned subtrees sequence, and the largest and smallest optimal trees:

```
seq = spatprune(tmax$tree)
```



Here optional parameters are set as default: *method = "deviance"* and *graph = TRUE*.

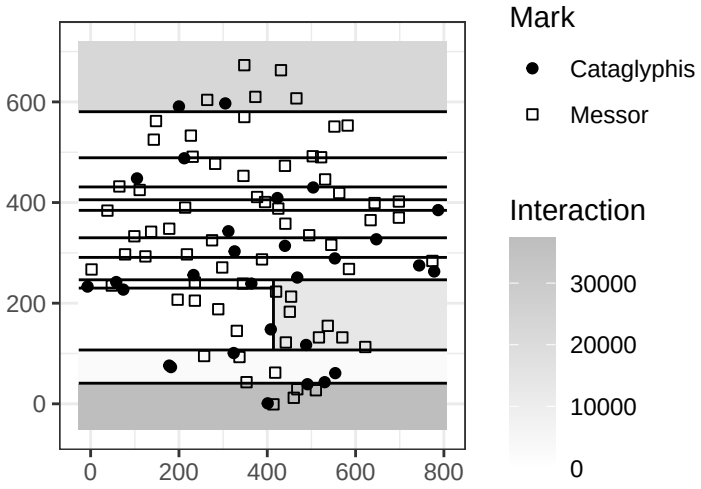
method = "deviance" produces class probability trees using Gini's criterion to prune the maximal tree. To obtain classification trees using misclassification criterion, set *method = "misclass"*.

graph = TRUE produces the graphics of the number of leaves of the pruned subtrees sequence with respect to the complexity parameter used in the pruning algorithm. The triangle symbol represents the tree selected via the modified largest jump method, while the diamond symbol represents the tree selected via the largest plateau method.

- **partition.spattree** plots the heatmap colored partitions induced by a tree's leaves with respect to the interaction between marks of a bivariate marked spatial point process. The output value of the function *partition.spattree()* is the corresponding 'ggplot' object. Needs also the input values of impurities inside tree leaves.

Plot the partition induced by the maximal Spatial CART classification tree constructed on the *ants* data set:

```
K = tmax$K[rownames(tmax$tree$frame)][tmax$tree$frame$var=="<leaf>"]
partition.spattree(tmax$tree, ypp, K)
```



Here the optional parameter d is set as default $d = 100$. d sets the resolution of the grid used to construct the heatmap at d^2.

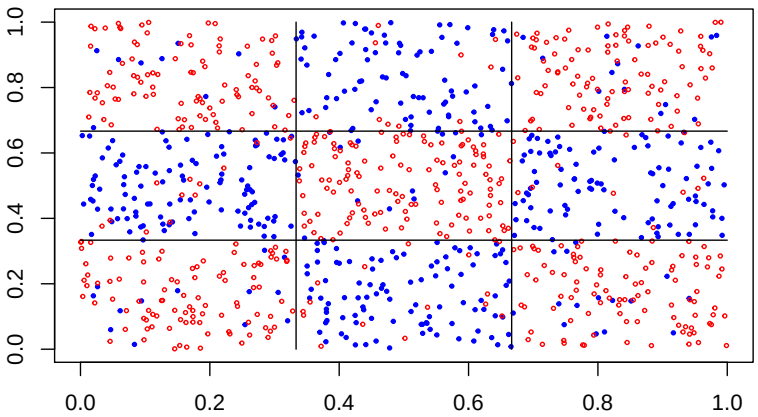
Simulations

The features for simulations of package **spatcart** are presented to cope with the simulation study proposed in R script *Simulation.study.R*, which can be found on the github repository https://github.com/Servane-Gey/Spatial-classification-trees. For more details about simualtions, please see the article *Spatial CART Classification Trees*, by A. Bar-Hen, S. Gey and J.-M. Poggi (2020).

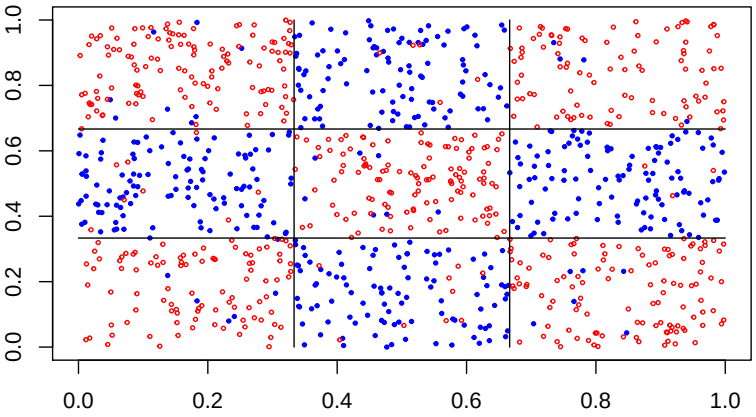
- **damier, repulsion** respectively simulates the *chess* and *locally repulsive* data sets presented in the article. The optional parameter *model* of the function *damier()* allows to simulate points as uniformly distributed on the unit square if *model = "unif"* (default), or as a Poisson point process on the unit square if *model = "Poisson"*.
  For both functions *damier()* and *repulsion()*,
  - the optional margin parameter $h = 0.4$ (default) defines the mixing proportion of marks inside windows,
  - the optional parameter *graph = TRUE* (default) produces the scatter plot of the resulting simulated points.

Simulate 1000 points from functions *damier()* and *repulsion()*, with repulsive scale equal to 0.05 for *repulsion()*:

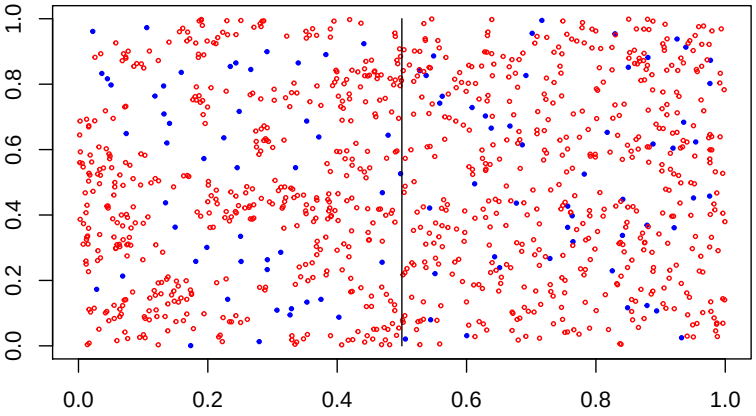
```
set.seed(12)
chess = damier(1000)
```




```
chess = damier(1000, h=0.45,  model = "Poisson")
```



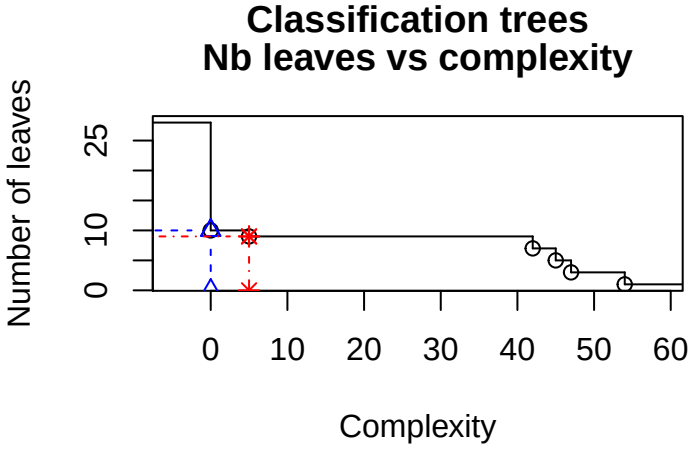

```
repuls = repulsion(1000, 0.05)
```



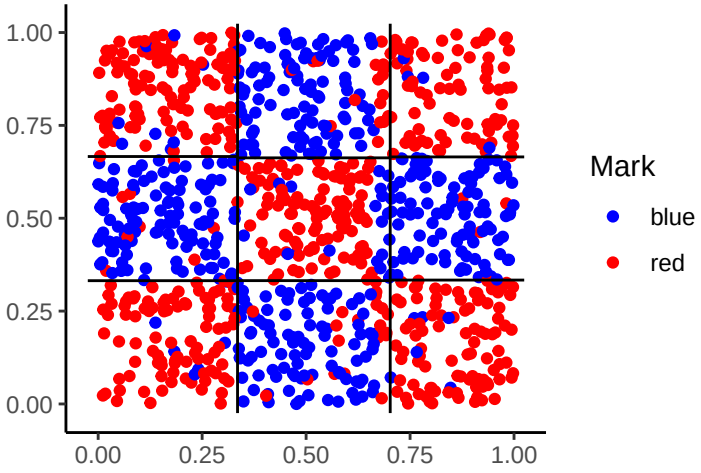
- **gg.partition.tree** plots the partition of a tree involving one or two variables.

Plot the partition induced by a classical CART classification tree on the *chess* data set:

```
nuage = data.frame(
    x = chess$data$x1,
    y = chess$data$x2,
    Mark = chess$data$label
    )
library(tree)
tmax = tree(Mark~.,data=nuage,split="gini",model=T, minsize=50,mincut=25)
seq = spatprune(tmax, method = "misclass")
```



```
t = seq$opt.tree.min # selects smallest optimal subtree
nuage %>% ggplot(aes(x,y, color = Mark))+geom_point()+
  scale_color_manual(values = c("blue", "red"))+
  gg.partition.tree(t)+
  labs(x = "", y = "")+
  theme_classic()
```



See function *partition.tree()* of package *tree* for optional arguments.

Note *gg.partition.tree()* does not assign tree labels on the partition.

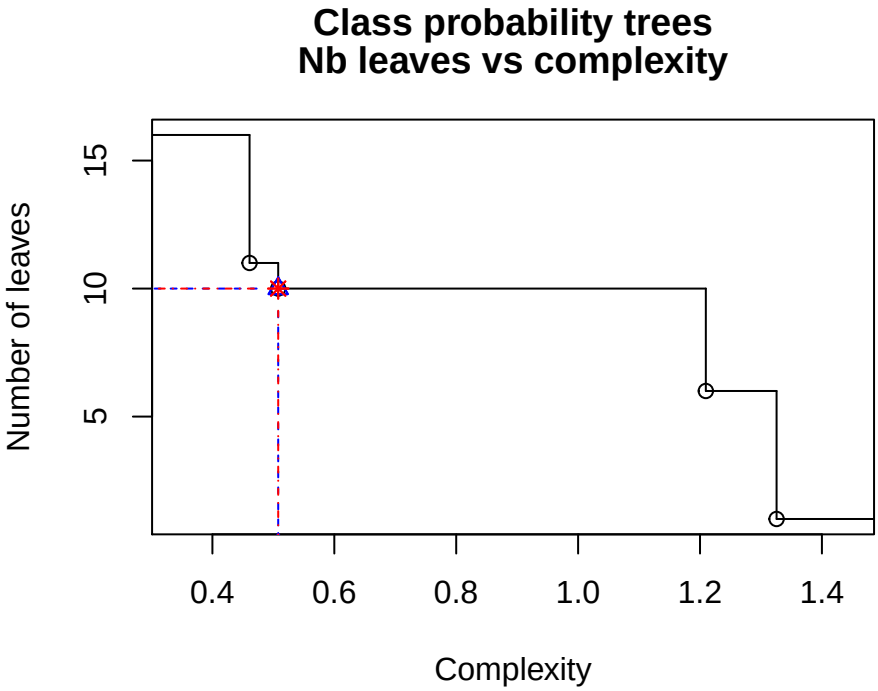
Paracou data set

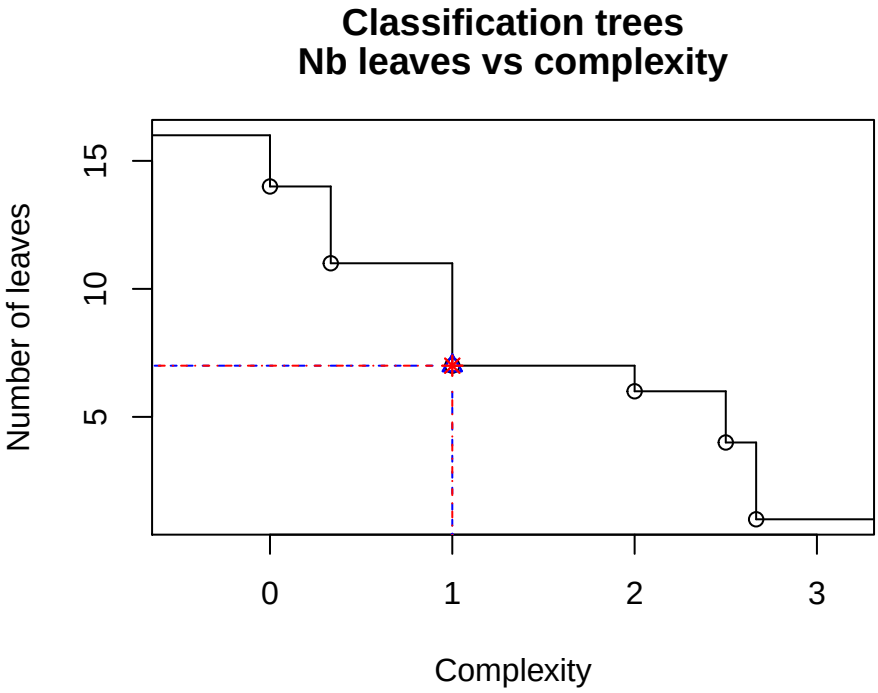
The features for the Paracou data set proposed in package **spatcart** are presented to cope with the experiments done in the article *Spatial CART Classification Trees*, by A. Bar-Hen, S. Gey and J.-M. Poggi (2020). The original data set can be freely downloaded from https://paracou.cirad.fr/.

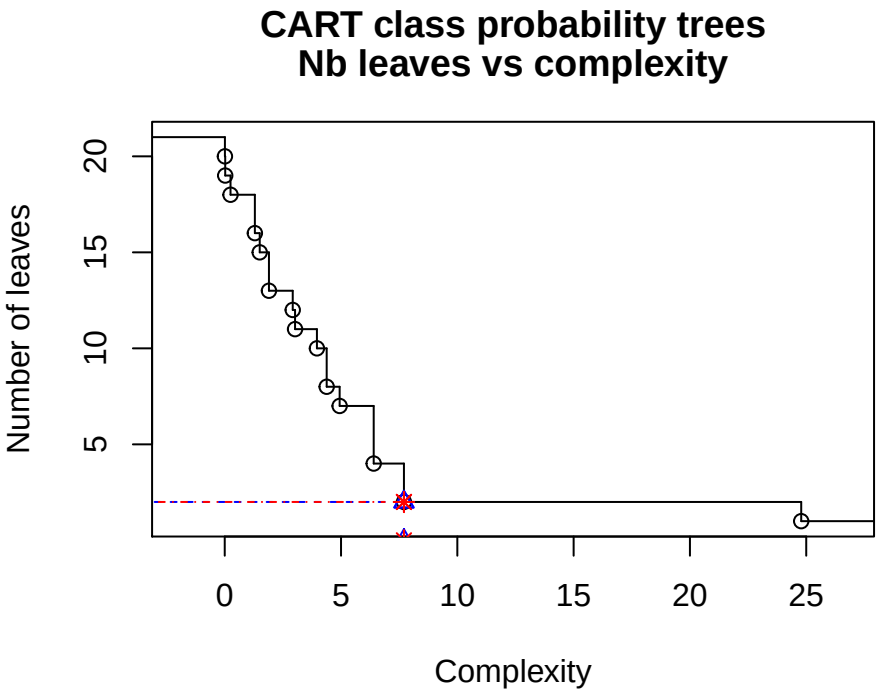
- **Paracou** produces graphical results of CART and SpatCART algorithms at fixed initial resolution for the Paracou data set. The output value of the function *Paracou()* is the following list of 'ggplot' objects:
  - *DiffK*: graphic of difference between estimated and theoretical intertype K-functions with respect to scale resolution, with a dashed line at the value of initial resolution.
  - *SC.maxtree*: graphic of the heatmap colored partition induced by the maximal SpatCART classification tree's leaves with respect to the interaction between marks.
  - *SC.Classprobtree*: graphic of the heatmap colored partition induced by the optimal SpatCART class probability tree's leaves with respect to the interaction between marks.
  - *SC.Classtree*: graphic of the heatmap colored partition induced by the optimal SpatCART classification tree's leaves with respect to the interaction between marks.
  - *C.maxtree*: graphic of the partition induced by the maximal CART classification tree's leaves.
  - *C.Classprobtree*: graphic of the partition induced by the optimal CART class probability tree's leaves.
  - *C.Classtree*: graphic of the partition induced by the optimal CART classification tree's leaves.

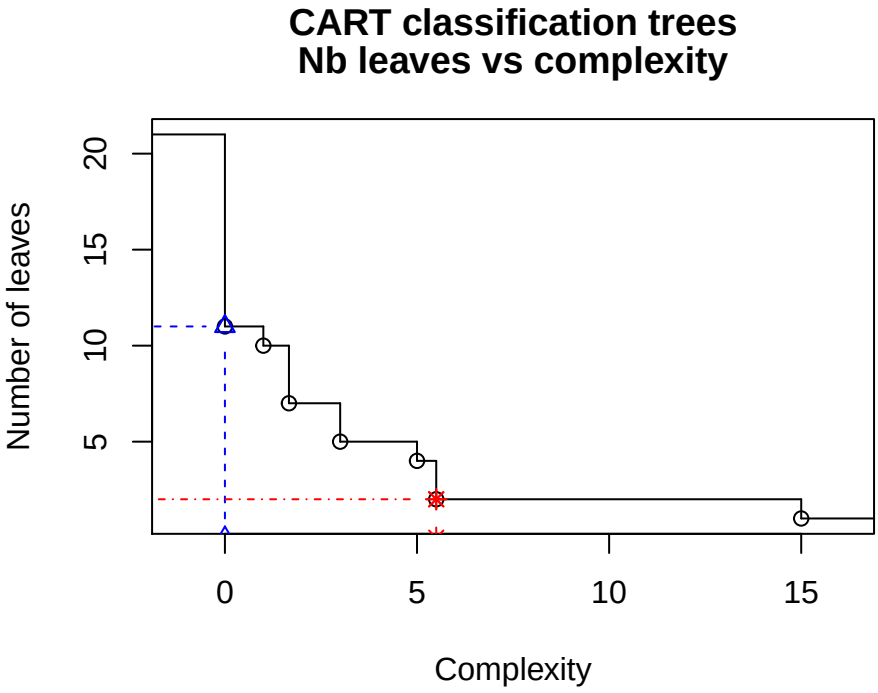
Produce graphics for initial scale resolution $r0 = 15$:

```
Paracou(15)
```

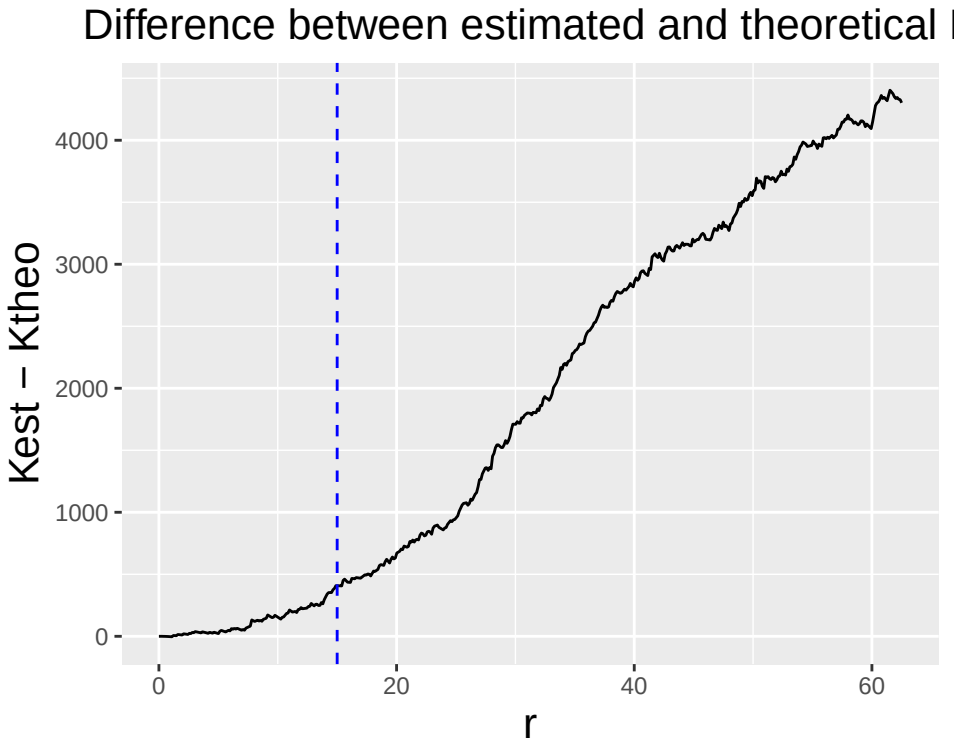

Class probability trees
Nb leaves vs complexity
Number of leaves
5
10
15
0.4
0.6
0.8
1.0
1.2
1.4
Complexity


Classification trees
Nb leaves vs complexity
Number of leaves
5
10
15
0
1
2
3
Complexity

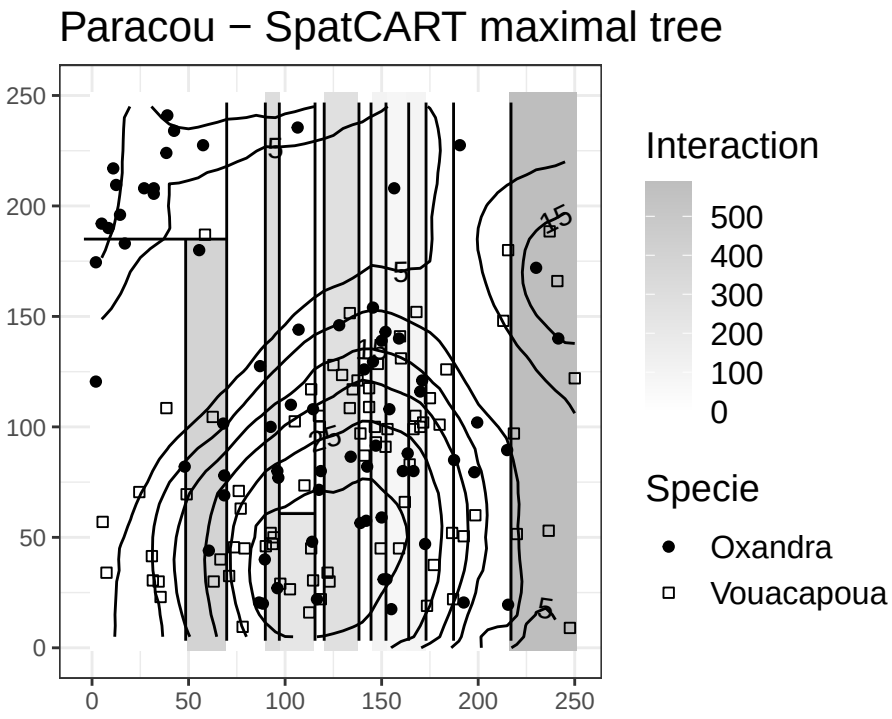

CART class probability trees
Nb leaves vs complexity
Number of leaves
5
10
15
20
0
5
10
15
20
25
Complexity


CART classification trees
Nb leaves vs complexity
Number of leaves
5
10
15
20
0
5
10
15
Complexity

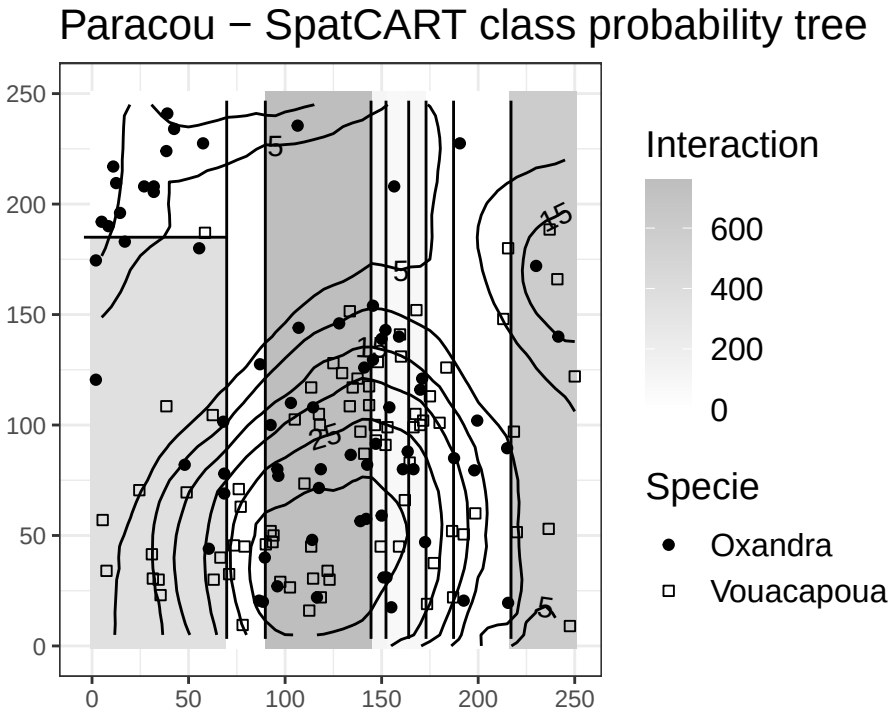
```
#> $DiffK
```



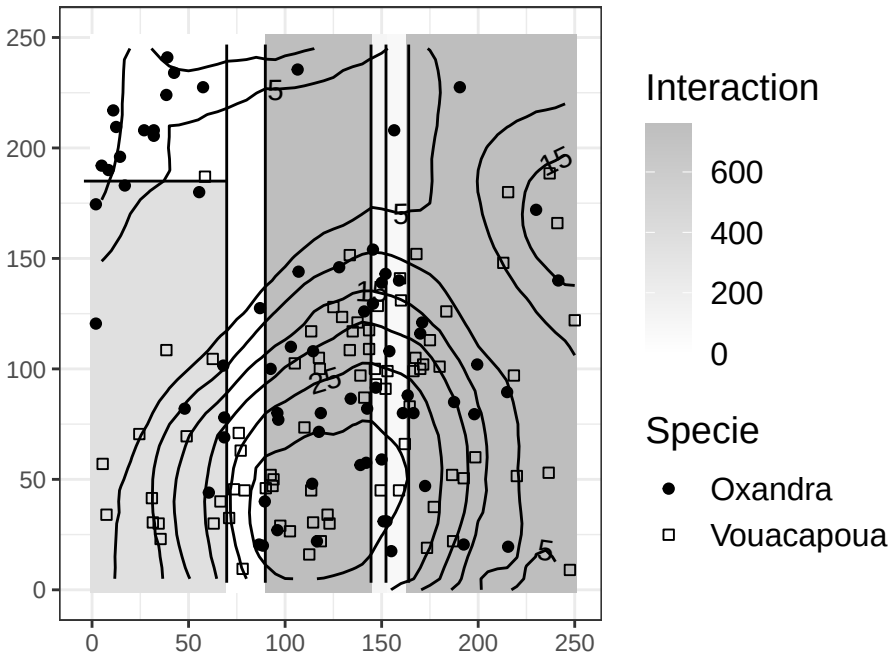

```
#>
#> $SC.maxtree
```



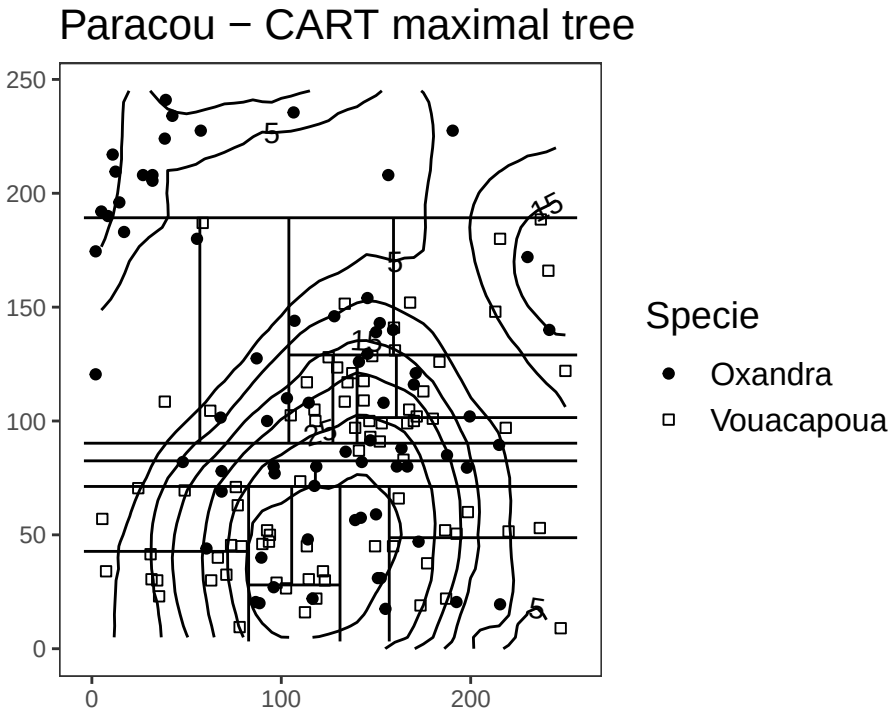

```
#>
#> $SC.Classprobtree
```



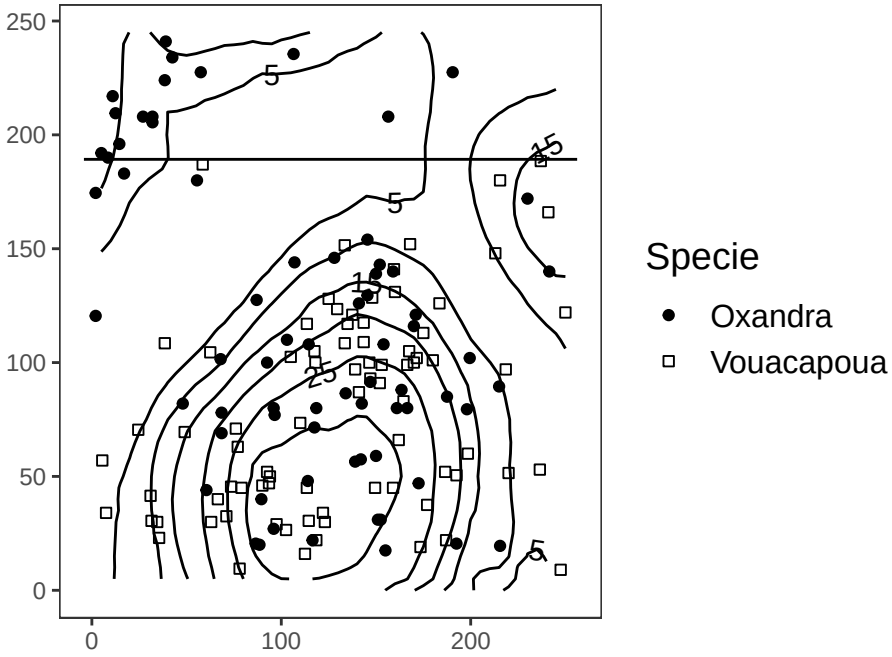

```
#>
#> $SC.Classtree
```



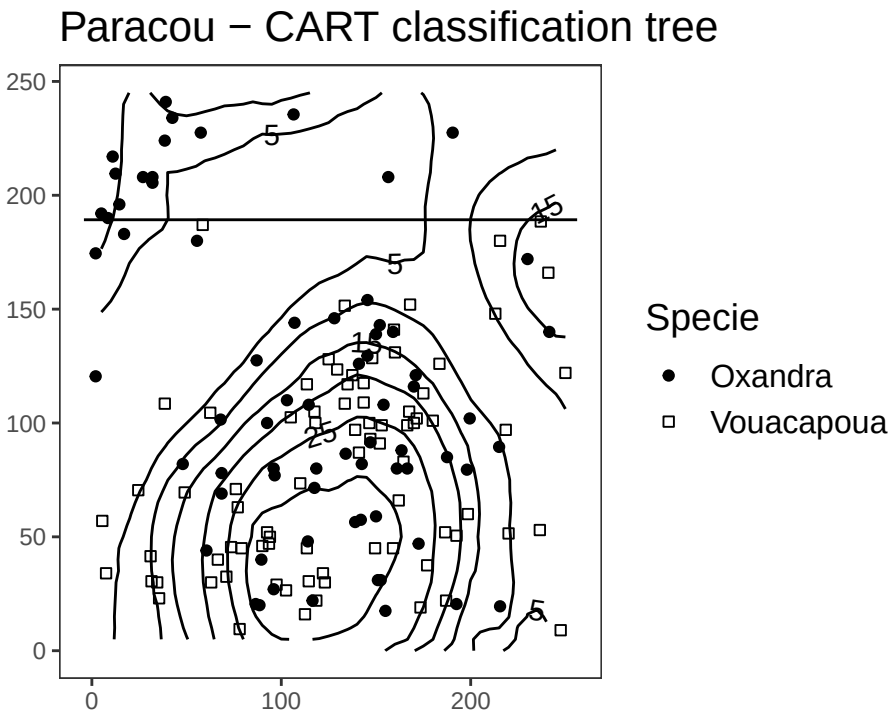

```
#>
#> $C.maxtree
```




```
#>
#> $C.Classprobtree
```




```
#>
#> $C.Classtree
```



- **Paracoudata, Paracou.plot** respectively loads the Paracou data set as a bivariate marked spatial point process of class 'ppp', and plots the heatmap colored partition induced by a SpatCART classification tree's leaves with respect to the interaction between marks of the Paracou data set. The output value of the function *Paracou.plot()* is the corresponding 'ggplot' object. Needs also the input values of impurities inside tree leaves.

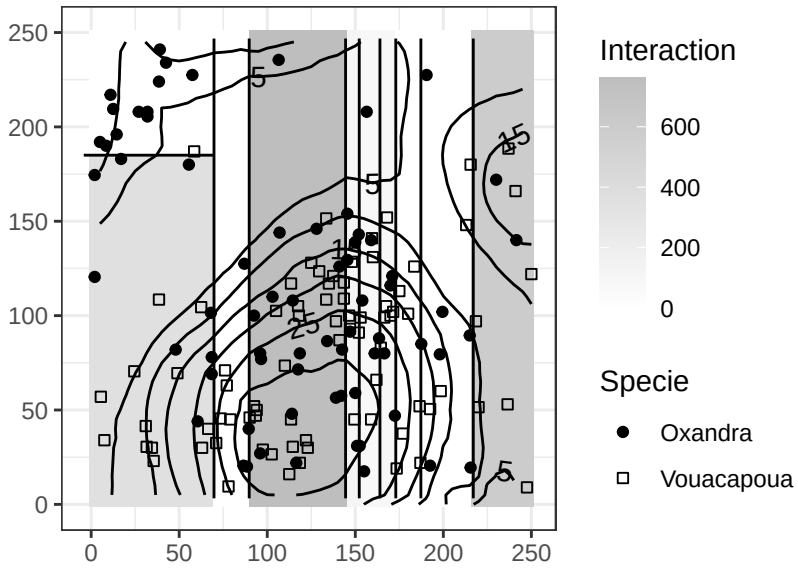
Plot the graphic of heatmap colored partition induced by the SpatCART class probability tree's leaves:

```
ypp = Paracoudata()

# SpatCART trees
t = spatcart(ypp, 15, graph = FALSE)

a = t$opt.tree.min
K = t$K[rownames(a$frame)][a$frame$var=="<leaf>"]

Paracou.plot(a,ypp,K)
```


250
200
150
100
50
0
0
50
100
150
200
250
5
5
15
10
25
5
Interaction
600
400
200
0
Specie
Oxandra
Vouacapoua