Exploration of landscapes of phylogenetic trees

Thibaut Jombart, Michelle Kendall

2015-12-16

treescape: exploration of landscapes of phylogenetic trees

treescape implements new methods for the exploration and analysis of distributions of phylogenetic trees for a given set of taxa.

Installing *treescape*

To install the development version from github:

```r
library(devtools)
install_github("thibautjombart/treescape")
```

The stable version can be installed from CRAN using:

```r
install.packages("treescape")
```

Then, to load the package, use:

```r
library("treescape")
```

Content overview

The main functions implemented in *treescape* are:

`treescape`: explore landscapes of phylogenetic trees

`treescapeServer`: open up an application in a web browser for an interactive exploration of the diversity in a set of trees

`findGroves`: identify clusters of similar trees

`plotGroves`: scatterplot of groups of trees

`medTree`: find geometric median tree(s) to summarise a group of trees

Other functions are central to the computations of distances between trees: **`treeVec`**: characterise a tree by a vector

`treeDist`: find the distance between two tree vectors

`multiDist`: find the pairwise distances of a list of trees

Distributed datasets include:

`woodmiceTrees`: illustrative set of 201 trees built using the neighbour-joining and bootstrapping example from the *woodmice* dataset in the *ape* documentation.

Exploring trees with *treescape*

We first load *treescape*, and packages required for graphics:

```
library("treescape")
library("ade4")
library("adegenet")
library("adegraphics")
library("ggplot2")
```

The function `treescape` defines typologies of phylogenetic trees using a two-step approach:

1. perform pairwise comparisons of trees using various (Euclidean) metrics; by default, the comparison uses the Kendall and Colijn metric (Kendall & Colijn, 2015) which is described in more detail below; other metrics rely on tips distances implemented in *adephylo* (Jombart *et al.* 2010).
2. use Metric Multidimensional Scaling (MDS, aka Principal Coordinates Analysis, PCoA) to summarise pairwise distances between the trees as well as possible into a few dimensions; the output of the MDS is typically visualised using scatterplots of the first few Principal Components (PCs); this step relies on the PCoA implemented in *ade4* (Dray & Dufour 2007).

The function `treescape` performs both tasks, returning both the matrix of pairwise tree comparisons (`$D`), and the PCoA (`$pco`). This can be illustrated using randomly generated trees:

```
## generate list of trees
set.seed(1)
x <- rmtree(10, 20)
names(x) <- paste("tree", 1:10, sep = "")

## use treescape
res <- treescape(x, nf=3)
names(res)
```

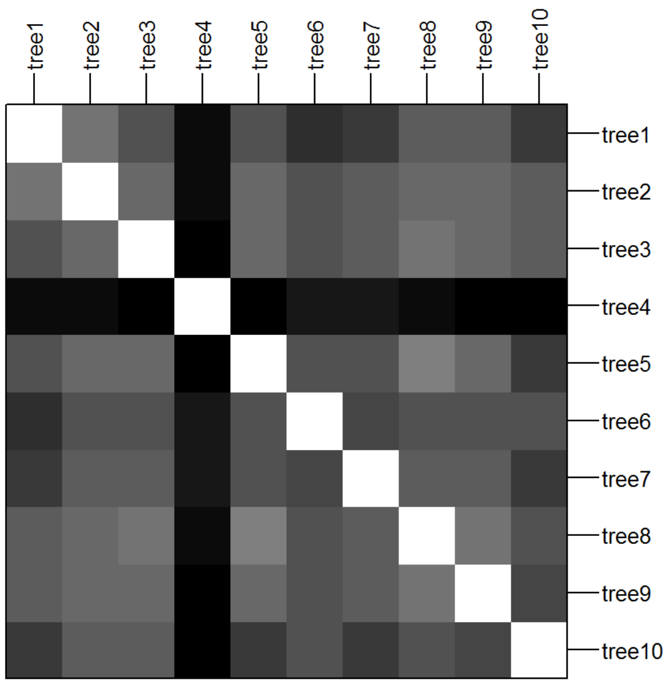
```
## [1] "D"   "pco"
```

```
res
```

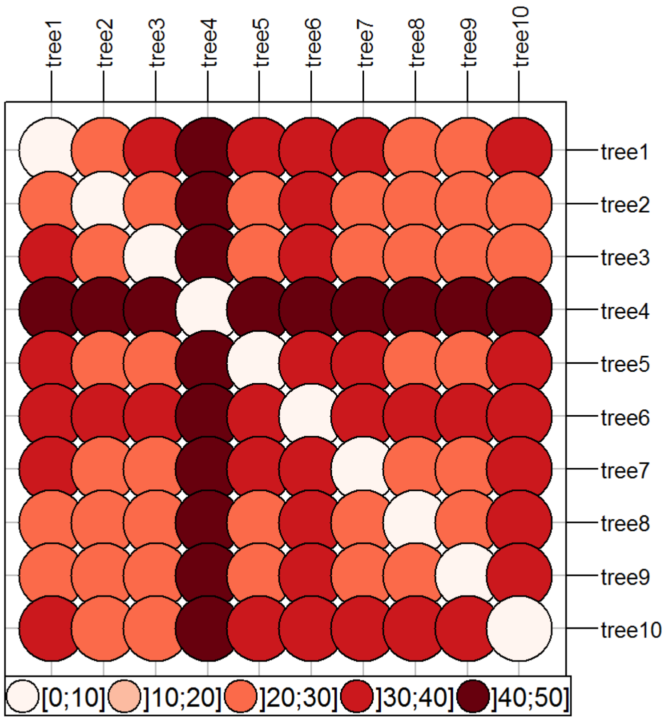
```
## $D
##        tree1 tree2 tree3 tree4 tree5 tree6 tree7 tree8 tree9
## tree2  26.00
## tree3  31.06 26.74
## tree4  42.85 42.12 44.44
## tree5  30.66 27.71 27.37 44.79
## tree6  36.50 31.18 30.18 41.81 31.59
## tree7  34.64 28.71 29.48 40.35 31.11 32.37
## tree8  28.97 26.29 24.45 43.74 23.47 30.41 29.00
## tree9  29.63 27.42 27.48 45.61 26.31 30.89 29.77 24.60
## tree10 34.87 30.00 29.44 44.97 34.06 31.05 34.41 31.54 32.59
##
## $pco
## Duality diagramm
## class: pco dudi
## $call: dudi.pco(d = D, scannf = is.null(nf), nf = nf)
##
## $nf: 3 axis-components saved
## $rank: 9
## eigen values: 142.1 76.52 62.69 49.88 41.07 ...
##   vector length mode    content
## 1 $cw    9      numeric column weights
## 2 $lw    10     numeric row weights
## 3 $eig   9      numeric eigen values
##
##   data.frame nrow ncol content
## 1 $tab       10   9    modified array
## 2 $li        10   3    row coordinates
## 3 $l1        10   3    row normed scores
## 4 $co        9    3    column coordinates
## 5 $c1        9    3    column normed scores
## other elements: NULL
```

Pairwise distances can be visualised using *adegraphics*:

```
## table.image
table.image(res$D, nclass=30)
```

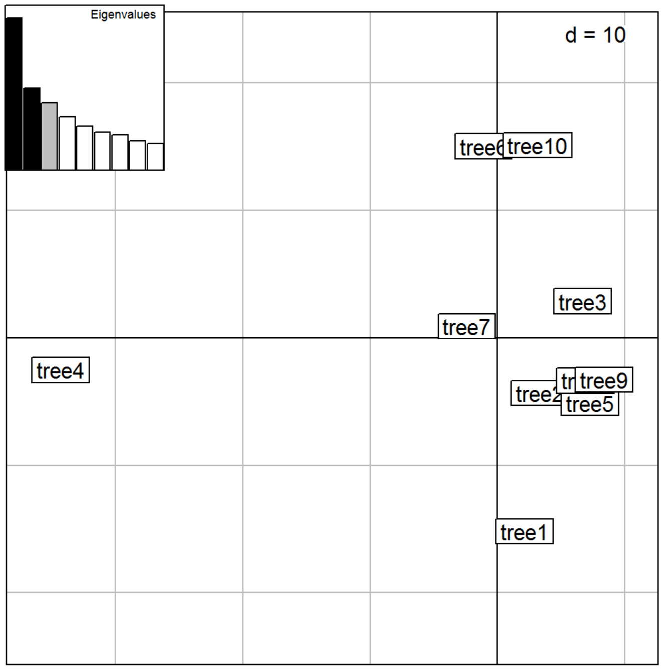


```
## table.value with some customization
table.value(res$D, nclass=5, method="color",
            symbol="circle", col=redpal(5))
```



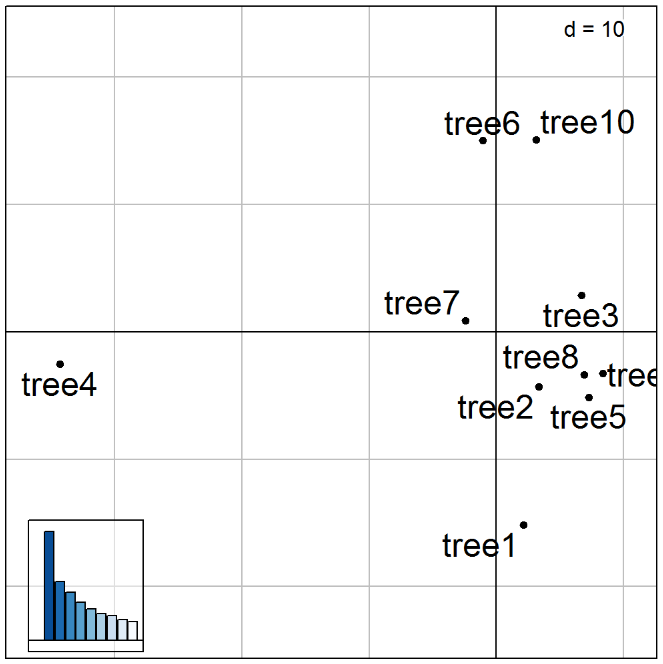
The best representation of these distances in a 2-dimensional space is given by the first 2 PCs of the MDS. These can be visualised using *adegraphics*'s function `scatter`:

```
scatter(res$pco)
```



Alternatively, the function `plotGroves` can be used:

```r
plotGroves(res$pco, lab.show=TRUE, lab.cex=1.5)
```



The functionality of `treecsape` can be further illustrated using *ape*'s dataset *woodmouse*, from which we built the 201 trees supplied in `woodmiceTrees` using the neighbour-joining and bootstrapping example from the *ape* documentation.

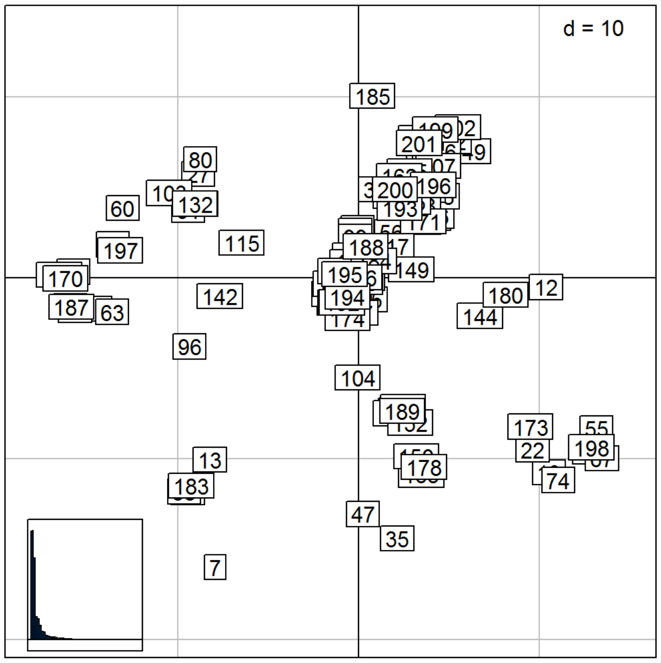
```r
data(woodmiceTrees)
wm.res <- treescape(woodmiceTrees,nf=3)

## this is the PCoA / MDS:
wm.res$pco
```

```
## Duality diagramm
## class: pco dudi
## $call: dudi.pco(d = D, scannf = is.null(nf), nf = nf)
##
## $nf: 3 axis-components saved
## $rank: 54
## eigen values: 32.69 24.41 6.952 6.348 4.363 ...
##   vector length mode    content
## 1 $cw    54     numeric column weights
## 2 $lw    201    numeric row weights
## 3 $eig   54     numeric eigen values
##
##   data.frame nrow ncol content
## 1 $tab       201  54   modified array
## 2 $li        201  3    row coordinates
## 3 $l1        201  3    row normed scores
## 4 $co        54   3    column coordinates
## 5 $c1        54   3    column normed scores
## other elements: NULL
```

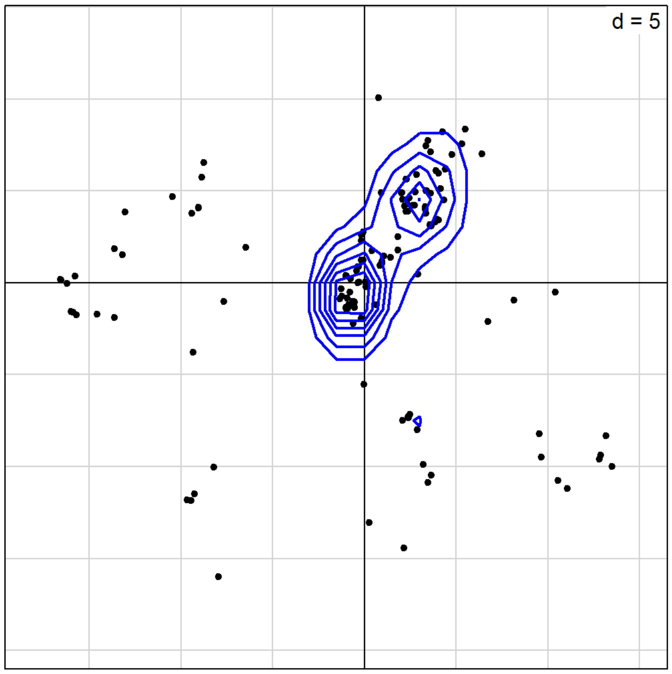
```r
## PCs are stored in:
head(wm.res$pco$li)
```

```
##          A1     A2      A3
## 1  -0.9949 -1.363 -0.7918
## 2  -0.6137 -1.014 -0.6798
## 3   2.6667  4.219 -2.9293
## 4 -13.6081  1.854  1.0947
## 5   2.1980  4.176 -3.1960
## 6   3.6013  4.865  2.9853
```

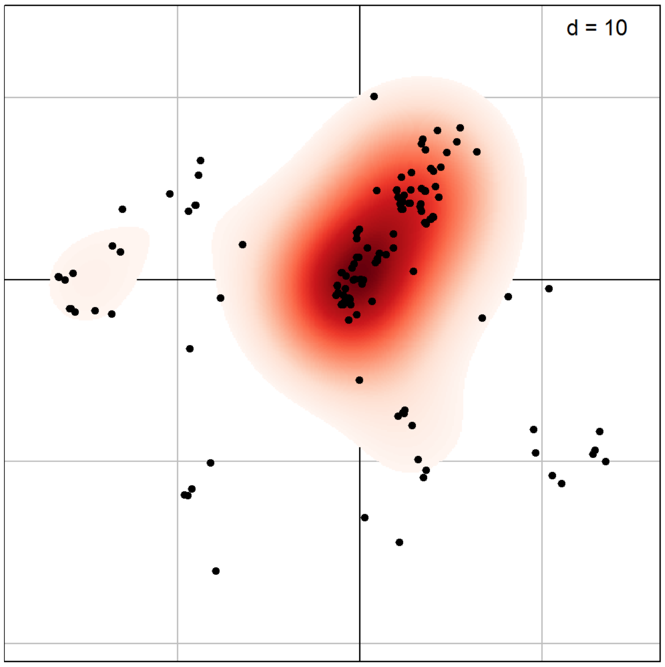
```r
## plot results
plotGroves(wm.res$pco, lab.show=TRUE, lab.optim=FALSE)
```



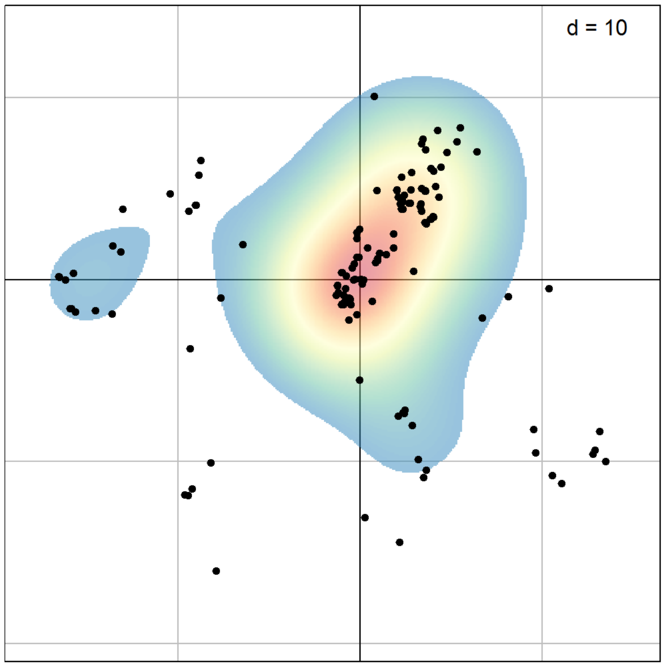
```r
## visualising density of points
s.kde2d(wm.res$pco$li)
```



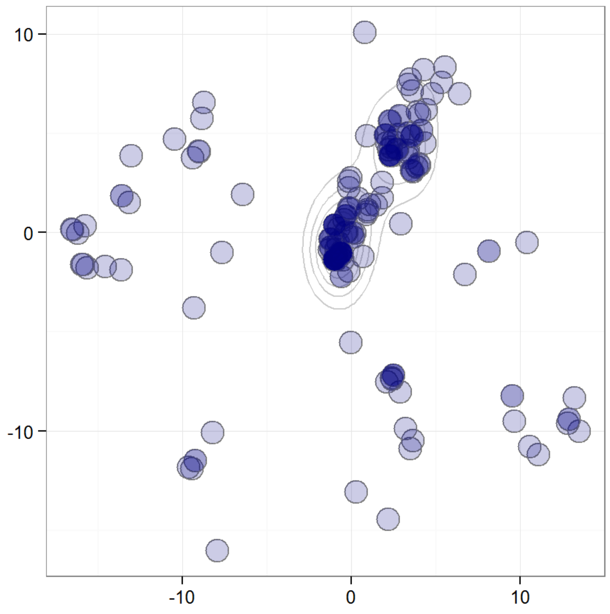
```
## alternative visualisation
s.density(wm.res$pco$li, col=redpal(100), bandwidth=3)
```



```
## same, other palette
s.density(wm.res$pco$li, col=rev(transp(spectral(100),.5)), bandwidth=3)
```



```
## alternative using ggplot2
woodmiceplot <- ggplot(wm.res$pco$li, aes(x=A1, y=A2)) # create plot
woodmiceplot + geom_density2d(colour="gray80") + # contour lines
geom_point(size=6, shape=1, colour="gray50") + # grey edges
geom_point(size=6, alpha=0.2, colour="navy") + # transparent blue points
xlab("") + ylab("") + theme_bw(base_family="") # remove axis labels and grey background
```



Note that alternatively, the function `multiDist` simply performs the pairwise comparison of trees and outputs a distance matrix. This function may be preferable for large datasets, and when principal co-ordinate analysis is not required. It includes an option to save memory at the expense of computation time.

Identifying clusters of trees

Once a typology of trees has been derived using the approach described above, one may want to formally identify clusters of similar trees. One simple approach is:

1. select a few first PCs of the MDS (retaining signal but getting rid of random noise)
2. derive pairwise Euclidean distances between trees based on these PCs
3. use hierarchical clustering to obtain a dendrogram of these trees
4. cut the dendrogram to obtain clusters

In *treescape*, the function `findGroves` implements this approach, offering various clustering options (see `?findGroves`). Here we supply the function with our `treescape` output `wm.res` since we have already calculated it, but it is also possible to skip the steps above and directly supply `findGroves` with a multiPhylo list of trees.

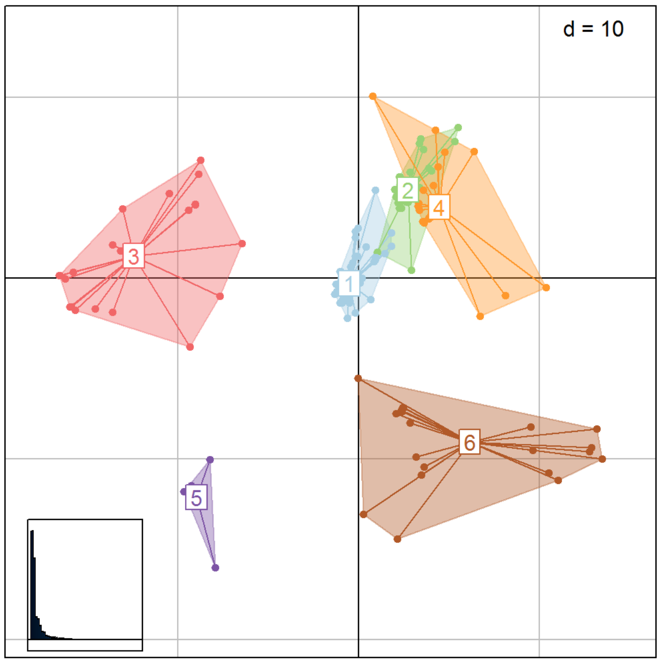
```
wm.groves <- findGroves(wm.res, nclust=6)
names(wm.groves)
```

```
## [1] "groups"    "treescape"
```

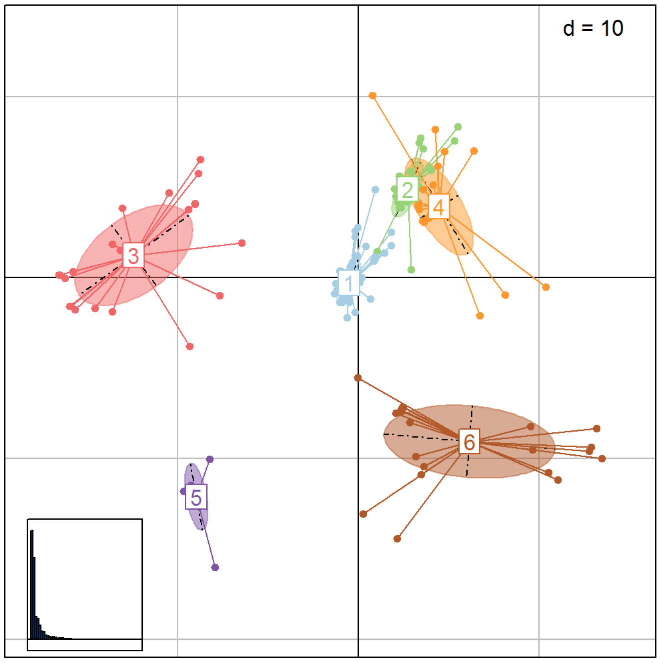
Note that when the number of clusters (`nclust`) is not provided, the function will display a dendrogram and ask for a cut-off height.

The results can be plotted directly using `plotGroves` (see `?plotGroves` for options):

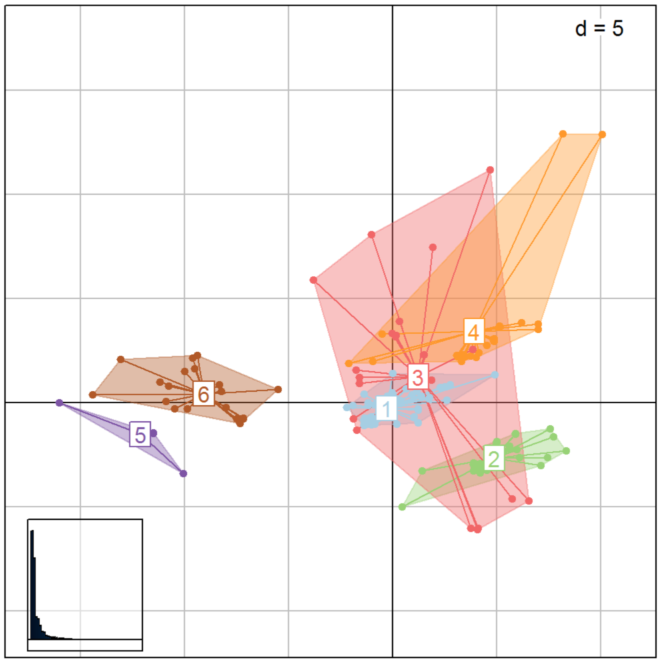
```
## basic plot
plotGroves(wm.groves)
```



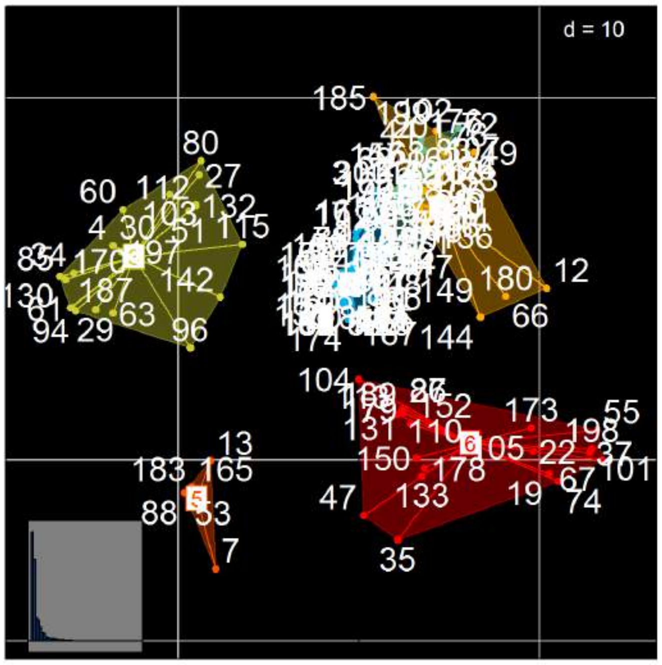
```
## alternative with inertia ellipses
plotGroves(wm.groves, type="ellipse")
```



```
## plot axes 2-3
plotGroves(wm.groves, xax=2, yax=3)
```

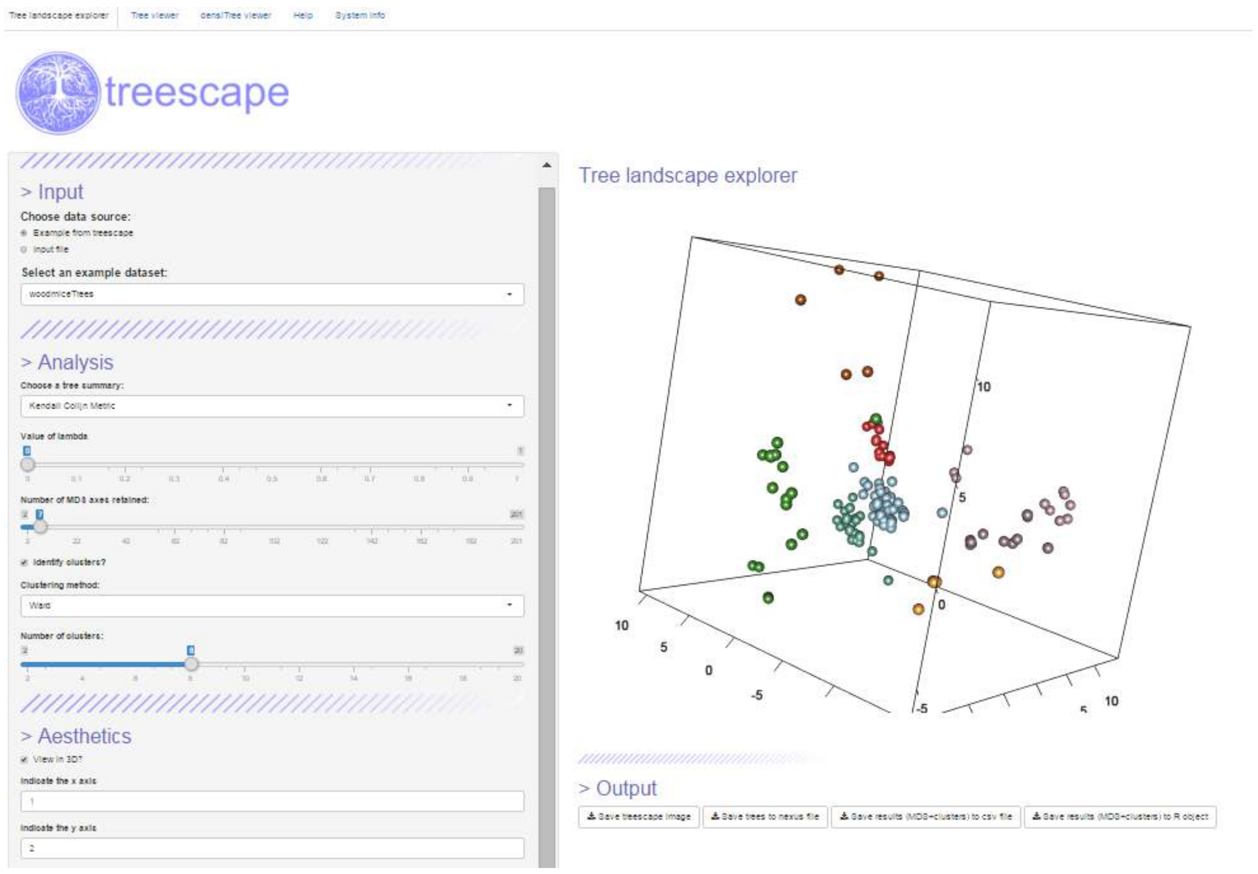


```
## customize graphics
plotGroves(wm.groves, bg="black", col.pal=lightseasun, lab.show=TRUE, lab.col="white", lab.cex=1.5)
```



`treescapeServer`: a web application for *treescape*

The functionalities of `treescape` are also available via a user-friendly web interface, running locally on the default web browser. It can be started by simply typing `treescapeServer()`. The interface allows you to import trees and run `treescape` to view and explore the tree space in 2 or 3 dimensions. It is then straightforward to analyse the tree space by varying lambda, looking for clusters using `findGroves` and saving results in various formats. Individual trees can be easily viewed including median trees per cluster, and collections of trees can be seen together using `densiTree` from the package `phangorn`. It is fully documented in the *help* tab.



> Input

Choose data source:

- [✓ Example from treescape]
- Input file

Select an example dataset: woodmiceTrees

> Analysis

Choose a tree summary: Kendall Colijn Metric

Value of lambda: 0

Number of MDS axes retained: 7

[✓] Identify clusters?

Clustering method: Ward

Number of clusters: 8

> Aesthetics

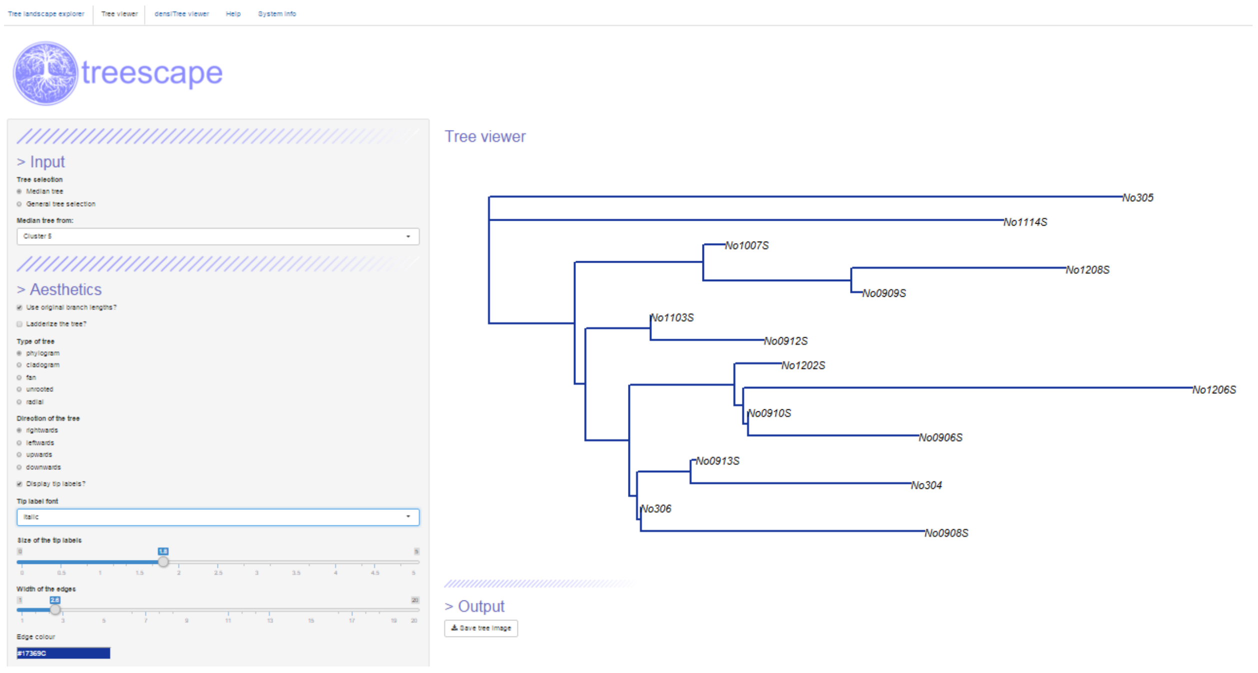
[✓] View in 3D?

Indicate the x axis: 1

Indicate the y axis: 2

Tree landscape explorer

> Output



> Input

Tree selection

- [✓ Median tree]
- General tree selection

Median tree from: Cluster 5

> Aesthetics

[✓] Use original branch lengths?

[ ] Ladderize the tree?

Type of tree

- [✓ phylogram]
- cladogram
- fan
- unrooted
- radial

Direction of the tree

- [✓ rightwards]
- leftwards
- upwards
- downwards

[✓] Display tip labels?

Tip label font: Italic

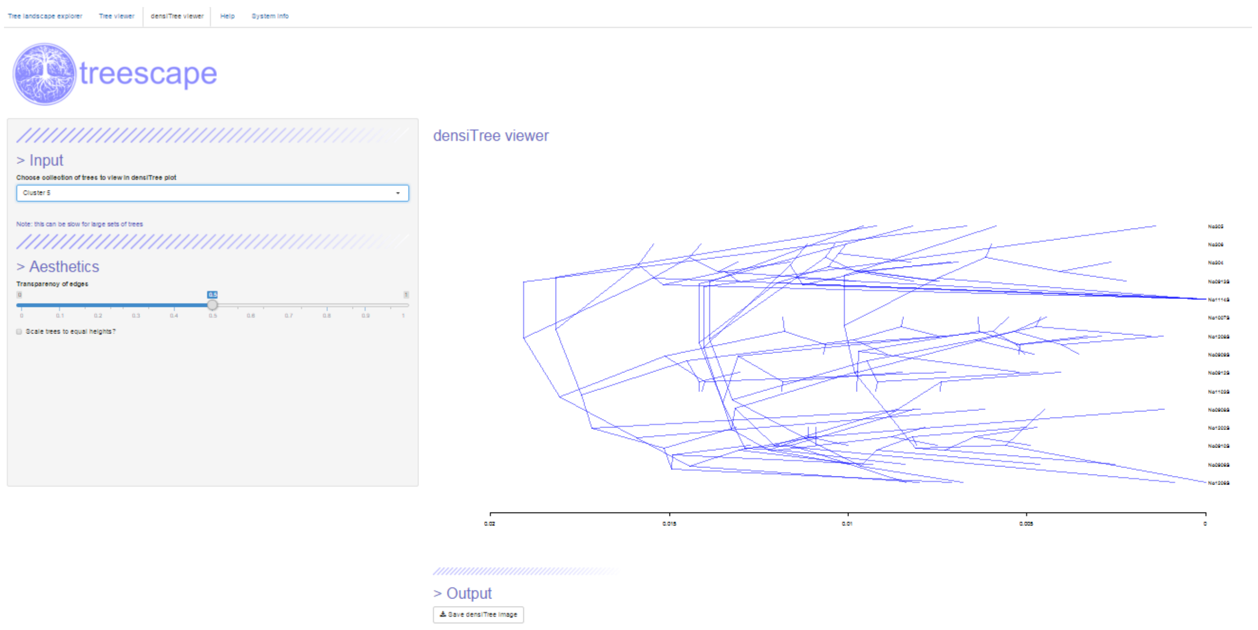
Size of the tip labels: 1.8

Width of the edges: 2.6

Edge colour: #17369C

Tree viewer

> Output



> Input

Choose collection of trees to view in densiTree plot: Cluster 5

Note: this can be slow for large sets of trees

> Aesthetics

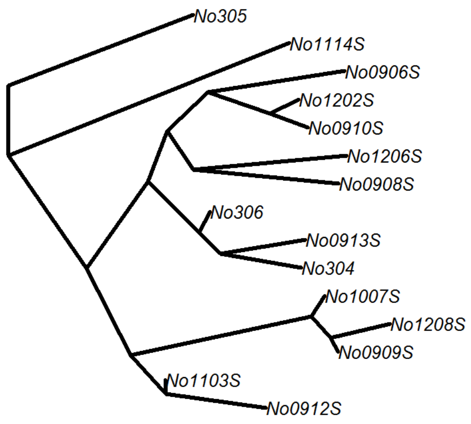
Transparency of edges: 0.5

Scale trees to equal heights?

Finding median trees

When a set of trees have very similar structures, it makes sense to summarize them into a single 'consensus' tree. In `treescape`, this is achieved by finding the *median tree* for a set of trees according to the Kendall and Colijn metric. That is, we find the tree which is closest to the centre of the set of trees in the tree landscape defined in `treescape`. This procedure is implemented by the function **`medTree`**:

```r
## get first median tree
tre <- medTree(woodmiceTrees)$trees[[1]]

## plot tree
plot(tre,type="cladogram",edge.width=3, cex=0.8)
```

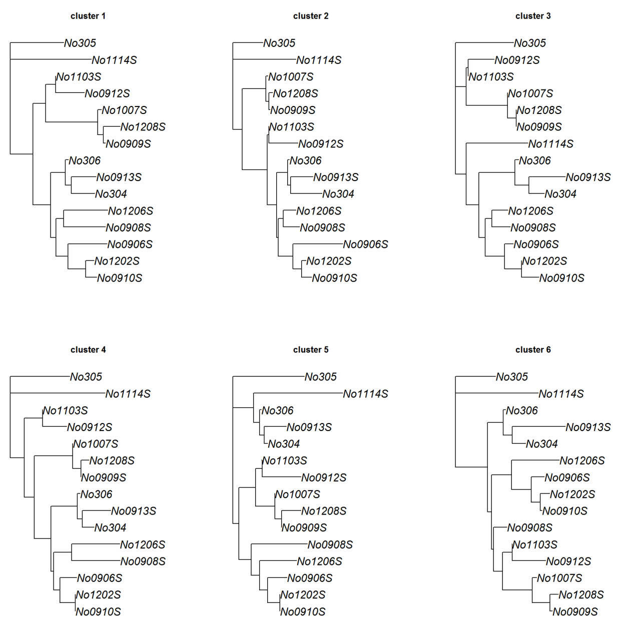


However, a more complete and accurate summary of the data can be given by finding a summary tree from each cluster. This is achieved using the `groups` argument of `medTree`:

```
## find median trees for the 6 clusters identified earlier:
res <- medTree(woodmiceTrees, wm.groves$groups)

## there is one output per cluster
names(res)
```

```
## [1] "1" "2" "3" "4" "5" "6"
```

```
## get the first median of each
med.trees <- lapply(res, function(e) ladderize(e$trees[[1]]))

## plot trees
par(mfrow=c(2,3))
for(i in 1:length(med.trees)) plot(med.trees[[i]], main=paste("cluster",i),cex=1.5)
```



These trees exhibit a number of topological differences, e.g. in the placement of the **(1007S,1208S,0909S)** clade. Performing this analysis enables the detection of distinct representative trees supported by data.

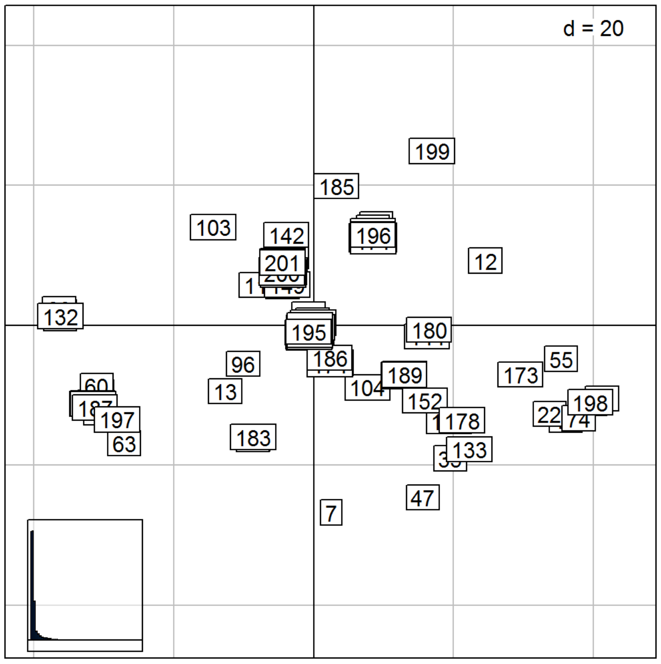
Note that we supplied the function `medTree` with the multiPhylo list of trees. A more computationally efficient process (at the expense of using more memory) is to use the option `return.tree.vectors` in the initial `treescape` call, and then supply these vectors directly to `medTree`. In this case, the tree indices are returned by `medTree` but the trees are not (since they were not supplied).

Emphasising the placement of certain tips or clades

In some analyses it may be informative to emphasise the placement of particular tips or clades within a set of trees. This can be particularly useful in large trees where the study is focused on a smaller clade. Priority can be given to a list of tips using the argument `emphasise.tips`, whose corresponding values in the vector comparison will be given a weight of `emphasise.weight` times the others (the default is 2, i.e. twice the weight).

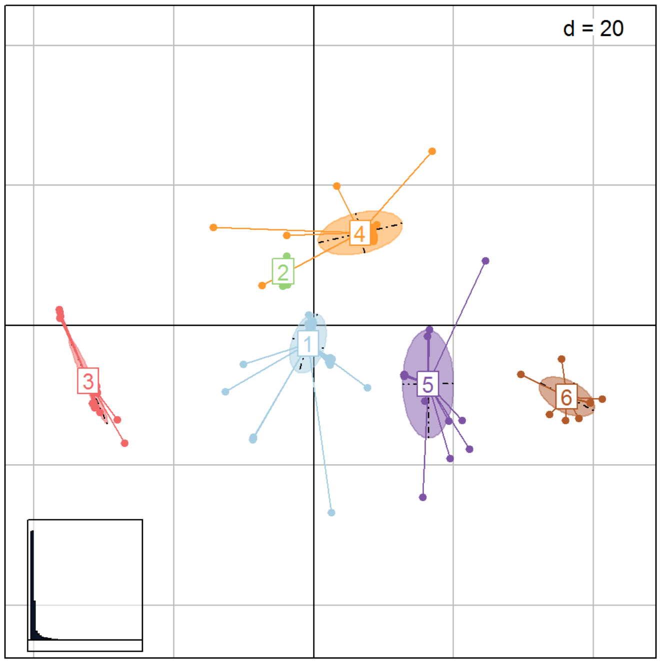
For example, if we wanted to emphasise where the woodmice trees agree and disagree on the placement of the **(1007S,1208S,0909S)** clade, we can simply emphasise that clade as follows:

```r
wm3.res <- treescape(woodmiceTrees, nf=2, emphasise.tips=c("No1007S","No1208S","No0909S"),
                     emphasise.weight=3)

## plot results
plotGroves(wm3.res$pco, lab.show=TRUE, lab.optim=FALSE)
```



It can be seen from the scale of the plot and the density of clustering that the trees are now separated into more distinct clusters.

```r
wm3.groves <- findGroves(woodmiceTrees, nf=3, nclust=6,
                         emphasise.tips=c("No1007S","No1208S","No0909S"),
                         emphasise.weight=3)
plotGroves(wm3.groves, type="ellipse")
```



Conversely, where the structure of a particular clade is not of interest (for example, lineages within an outgroup which was only included for rooting purposes), those tips can be given a weight less than 1 so as to give them less emphasis in the comparison. We note that although it is possible to give tips a weighting of 0, we advise caution with this as the underlying function will no longer be guaranteed to be a metric. That is, a distance of 0 between two trees will no longer necessarily imply that the trees are identical. In most cases it would be wiser to assign a very small weighting to tips which are not of interest.

Method: characterising a tree by a vector

Kendall and Colijn proposed a metric for comparing rooted phylogenetic trees. Each tree is characterised by a vector which notes the placement of the most recent common ancestor (MRCA) of each pair of tips. Specifically, it records the distance between the MRCA of a pair of tips (i, j) and the root in two ways: the number of edges $m_{i,j}$, and the path length $M_{i,j}$. It also records the length p_i of each 'pendant' edge between a tip i and its immediate ancestor. This procedure results in two vectors for a tree T:

$$m(T) = (m_{1,2}, m_{1,3}, \ldots, m_{k-1,k}, \underbrace{1, \ldots, 1}_{k \text{ times}})$$

and

$$M(T) = (M_{1,2}, M_{1,3}, \ldots, M_{k-1,k}, p_1, \ldots, p_k).$$

In $m(T)$ we record the pendant lengths as 1, as each tip is 1 step from its immediate ancestor. We combine m and M with a parameter lambda between zero and one to weight the contribution of branch lengths, characterising each tree with a vector

$$v_\lambda(T) = (1 - \lambda)m(T) + \lambda M(T).$$

This is implemented as the function **treeVec**. For example,

```
## generate a random tree:
tree <- rtree(6)
## topological vector of mrca distances from root:
treeVec(tree)
```

```
##  [1] 1 0 2 0 1 0 1 0 2 0 1 0 0 1 0 1 1 1 1 1 1
```

```
## vector of mrca distances from root when lambda=0.5:
treeVec(tree,0.5)
```

```
##  [1] 0.5616 0.0000 1.3923 0.0000 0.5616 0.0000 0.5616 0.0000 1.0878 0.0000
## [11] 0.8762 0.0000 0.0000 0.5616 0.0000 0.5621 0.7501 0.5160 0.5793 0.8971
## [21] 0.7319
```

```
## vector of mrca distances as a function of lambda:
vecAsFunction <- treeVec(tree,return.lambda.function=TRUE)
## evaluate the vector at lambda=0.5:
vecAsFunction(0.5)
```

```
##  [1] 0.5616 0.0000 1.3923 0.0000 0.5616 0.0000 0.5616 0.0000 1.0878 0.0000
## [11] 0.8762 0.0000 0.0000 0.5616 0.0000 0.5621 0.7501 0.5160 0.5793 0.8971
## [21] 0.7319
```

The metric – the distance between two trees – is the Euclidean distance between these vectors:

$$d_\lambda(T_a, T_b) = \|v_\lambda(T_a) - v_\lambda(T_b)\|.$$

This can be found using **`treeDist`**:

```
## generate random trees
tree_a <- rtree(6)
tree_b <- rtree(6)

## topological (lambda=0) distance:
treeDist(tree_a,tree_b)
```

```
## [1] 5.477
```

```
## branch-length focused (lambda=1) distance:
treeDist(tree_a,tree_b,1)
```

```
## [1] 3.058
```

References

- Dray S & Dufour AB (2007): The ade4 package: implementing the duality diagram for ecologists. Journal of Statistical Software 22(4): 1-20.
- Jombart R, Balloux F & Dray S (2010) adephylo: new tools for investigating the phylogenetic signal in biological traits. Bioinformatics 26: 1907-1909. Doi: 10.1093/bioinformatics/btq292
- Kendall M & Colijn C (Preprint 2015) A tree metric using structure and length to capture distinct phylogenetic signals. arXiv 1507.05211

Authors / Contributors

Authors: Thibaut Jombart, Michelle Kendall, Jacob Almagro-Garcia

Contributors: Caroline Colijn

Maintainer of the CRAN version: Michelle Kendall