kdetrees Manual

Grady Weyenberg

June 2, 2015

1 Introduction

KDETrees is a tool for finding discordant phylogenetic trees. It takes as input a collection of trees (as a `ape::multiPhylo` object), and produces a unnormalized density *score* for each tree. High scores mean the tree is relatively similar to other trees in the sample, while low scores indicate that the tree in question may be discordant with the others. Low scoring trees are identified as putative *outliers*, and their contribution to the score calculation is removed. The result object contains scores for each tree, and a list of the putative outlier trees. If you use the program in your research, please cite our paper:

> G. Weyenberg, P. Huggins, C. Schardl, D.K. Howe, and R. Yoshida. kdetrees: Nonparametric estimation of phylogenetic tree distributions. *Bioinformatics*, 2014.

2 Using `kdetrees`

2.1 Importing Trees

Trees may be imported using any of the methods provided by the `ape` package. (See `?read.tree` and `?read.nexus` for examples.) In the following examples, many functions are a part of the `ape` package, and it is recommended that you import it. For example, to load the example `apicomplexa` dataset, I placed the Newick tree strings into a file named `apicompexa.tre` file and ran the commands:

```
> library(ape)
> apicomplexa <- read.tree("apicomplexa.tre")
```

NB: The apicomplexa dataset is already included in the kdetrees package as example data, you do not need to import it. This is purely an example showing how to import your own trees.

kdetrees also provides the `load.trees` function, which searches for files by directory and file extension.

2.2 Running kdetrees

The simplest way to run `kdetrees` is to call the function of the same name, with the list of trees as the first argument.

```
> result <- kdetrees(apicomplexa, outgroup="Tt")
> result

Call: kdetrees(trees = apicomplexa, outgroup = "Tt")
Density estimates:
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
-38.740   5.108   6.107   5.472   6.616   7.199 
Cutoff: 

Outliers detected:
 [1] 472.tre 478.tre 488.tre 505.tre 515.tre 553.tre 578.tre
 [8] 585.tre 588.tre 630.tre 641.tre 645.tre 662.tre 725.tre
[15] 745.tre 750.tre
```

There are 3 main settings which control the method used in the analysis: the outlier detection tuning parameter (`k`), the distance computation method (`distance`), and whether or not to include branch length information in the distance calculation (`topo.only`). The default options are `distance="geodesic"`, `topo.only=FALSE`, and `k=1.5`.

For example, this call uses topology-based dissimilarity map distance.

```
> kdetrees(apicomplexa, k=1.25, distance="diss", topo.only=TRUE)
```

One can `plot` or `hist` the result object to create diagnostic plots. The `plot` and `hist` methods use the `ggplot2` package, not base graphics, thus you can modify them as you see fit. See Figure 1 for example plots.

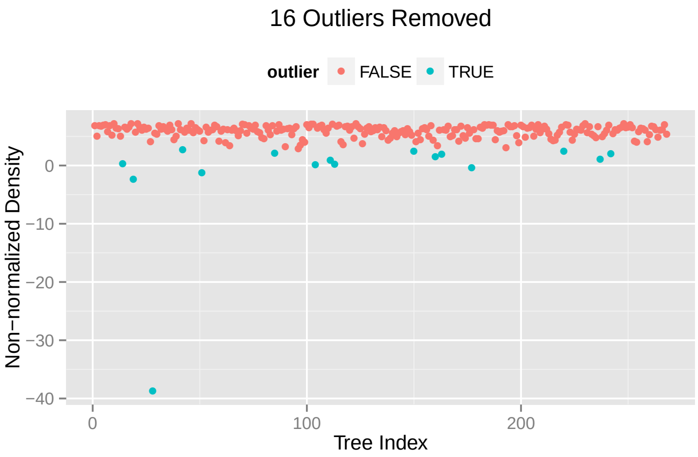
2.3 Results

The result object is a list with three components, as well as several attributes that are used internally. The first element, `density`, has the computed score for each tree in the input list. This is the variable displayed in the diagnostic plots. The second element, `i`, contains the indices of the low scoring trees which were not included in the calculations. Finally, the `outliers` element contains the trees which were identified as outliers.

One might then wish to look at a plot of the putative outlier trees. Here I plot the lowest scoring tree in the apicomplexa dataset. It appears that something bad happened during the reconstruction of this tree, causing one branch to be much longer than the others.

An `as.data.frame` method is provided for the result object, which can be used to export the results using the standard R methods. (Such as `write.csv`.) The list of outlier trees can be exported using `ape::write.tree`

```
> plot(result)
```



```
> hist(result)
```

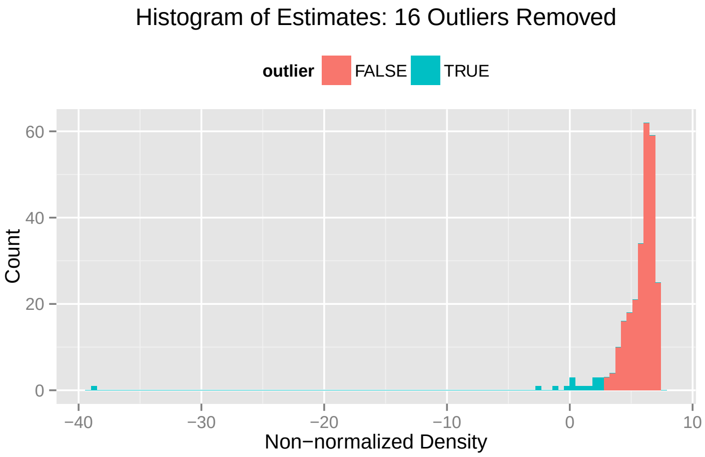


Figure 1: Diagnostic plots can be created with `plot` and `hist`.

```
> plot(result$outliers[[1]])
```

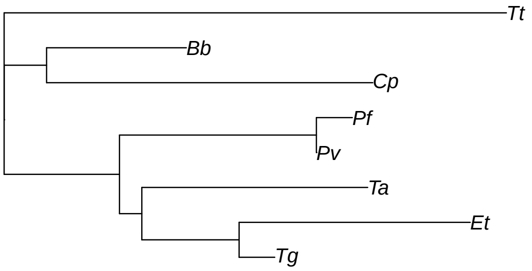


Figure 2: A plot of an outlying tree.

```
> write.tree(result$outliers,file="outliers.tre")
> result.df <- as.data.frame(result)
> write.csv(result.df, file="scores.csv")
```

3 Advanced Options

3.1 Distance Methods

There are currently two methods implemented for the `distance` option in `kdetrees`: The `"geodesic"` method, based on work by Billera et al. [2001], as well as that of Owen and Provan [2011]; and the `"dissimilarity"` map method, which utilizes pairwise tip-to-top distances.

Each of these distances can use either branch lengths supplied by the input trees, `topo.only=FALSE`, or the branch lengths can be ignored by setting `topo.only=TRUE`.

3.2 Bandwidth Selection

Currently, kdetrees uses an adaptive bandwidth method based on a nearest-neighbor calculation by default. It is possible to control the number of trees used to define the neighborhood, or disable the adaptive method entirely and provide a constant bandwidth, using the `bw` parameter.

If the `bw` parameter is supplied a list, the list is used as a set of parameters for a call to `bw.nn`. Currently, the only interesting setting that may be passed in this way is `prop`, which controls the proportion of the set of trees used to define the neighborhoods. For example, to change the neighborhood to include 50% of the sample we pass the following option.

```
> kdetrees(apicomplexa, bw=list(prop=0.5))
```

If bw is set to a single number, a constant bandwidth is used.

```
> kdetrees(apicomplexa, bw=6)
```

If bw is a vector, the values are used as bandwidths for the corresponding trees in the sample, possibly recycling the values if needed.

3.3 Command Line Interface

CLI use can be achieved by using the Rscript executable included with R. For example, this CLI command replicates the first example call in Section 2.

```
$ Rscript -e 'library(kdetrees); kdetrees.complete("trees.tre")'
```

The desired R commands can also be placed into a R script file (e.g. myscript.R), and run using

```
$ Rscript myscript.R
```

References

Louis J Billera, Susan P Holmes, and Karen Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

Megan Owen and J Scott Provan. A fast algorithm for computing geodesic distances in tree space. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 8(1):2–13, 2011.

G. Weyenberg, P. Huggins, C. Schardl, D.K. Howe, and R. Yoshida. kdetrees: Nonparametric estimation of phylogenetic tree distributions. *Bioinformatics*, 2014.