Using KDETrees

Grady Weyenberg

February 8, 2013

1 Introduction

KDETrees is a tool for finding discordant phylogenetic trees. It takes as input an `ape::multiPhylo` object, which contains a set of trees, and produces a 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. If desired, a number of low scoring trees may be designated as "outliers", which removes their contribution from the calculation. By default the lowest scoring 5% of the sample is removed.

2 Basic Use

2.1 Importing Trees

Trees may be imported using any of the methods provided by `ape`. See `?read.tree` and `?read.nexus` for examples. To import the `apicomplexa` dataset, for example, I placed the Newick tree stringsinto the `apicompexa.tre` file and used the following command:

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

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)
```

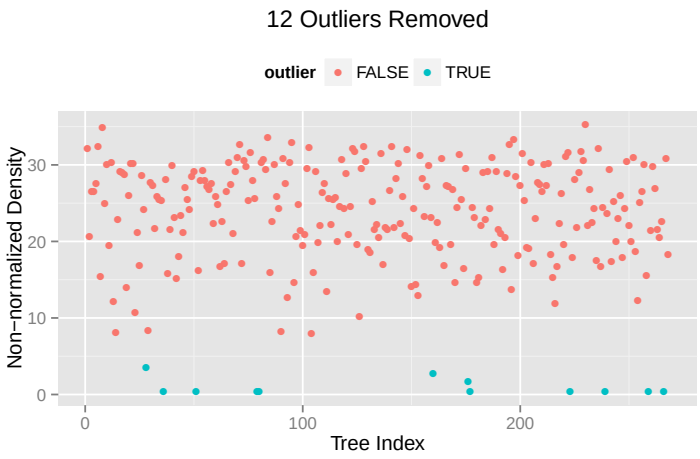
By default branch lengths are ignored, if you would like to use the branch length information in the calculation, set the `use.blen` option to true.

```
> result <- kdetrees(apicomplexa,use.blen=TRUE)
```

Another important option is the number of "outlier" trees which should be removed from the calculation. This is controlled by the `n` parameter.

```
> result <- kdetrees(apicomplexa,n=12,use.blen=TRUE)
```

```
> plot(result)
```




```
> hist(result)
```

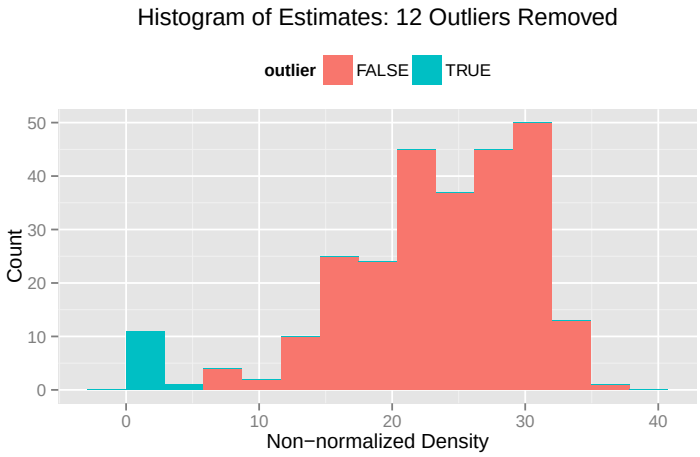
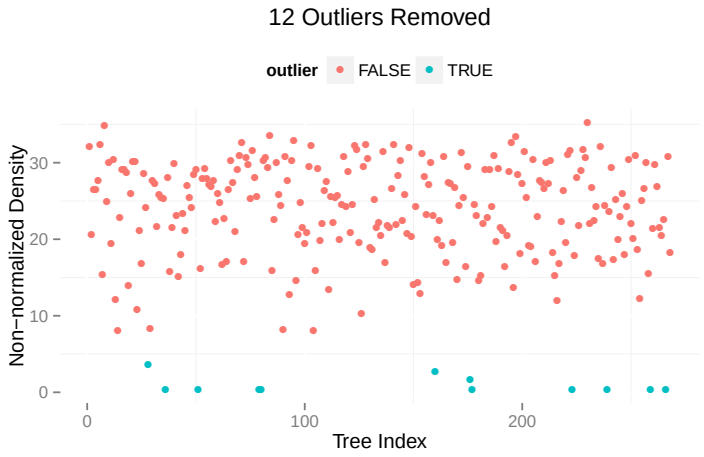


Figure 1: Diagnostic plots can be created with `plot` and `hist`. These methods use `ggplot2`, instead of base graphics.

It should be noted that the `plot` and `hist` methods use the `ggplot2` package, not base graphics. Thus, you can modify them as you see fit. For example,

```
> library(ggplot2)
> plot(result) + theme(panel.background=element_blank())
```



2.3 Results

The result object is a list with three components.

```
> str(result,strict.width="wrap")
```

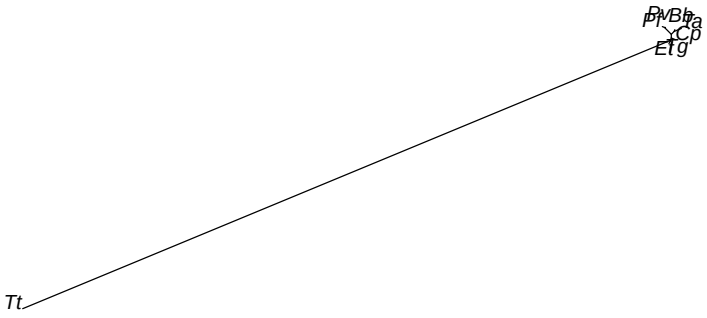
```
List of 3
$ density : Named num [1:268] 32.1 20.7 26.5 26.5 27.6 ...
..- attr(*, "names")= chr [1:268] "457.tre" "458.tre"
   "459.tre" "460.tre" ...
$ outliers : Named int [1:12] 51 80 266 36 177 79 259 223
   239 176 ...
..- attr(*, "names")= chr [1:12] "515.tre" "547.tre"
   "780.tre" "497.tre" ...
$ bandwidth: Named num [1:268] 1.5 2.85 1.61 1.65 1.63 ...
..- attr(*, "names")= chr [1:268] "457.tre" "458.tre"
   "459.tre" "460.tre" ...
- attr(*, "class")= chr "kdetrees"
```

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 `outliers` contains the indices of the low scoring trees which were not included in the calculations. We can use this to extract the outlier trees from the input list.

```
> outlier.trees <- apicomplexa[result$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.

```
> plot(outlier.trees[[1]],"u",no.margin=TRUE)
```



If you would like to export the outlier trees to a file, you may do something like the following.

```
> write.tree(outlier.trees,file="outliers.tre")
```

3 Shell Script

The KDETrees package also comes with a bash shell script that calls `kdetrees` using the Rscript executable. This is a convenience wrapper for cli users. If you copy this script to a working directory you can run it as follows.

```
$ ./kdetrees -h
Usage: ./kdetrees [options] file [...]
Options:
        -b, --use-branch-lengths
        -n NUM-OUTLIERS, --num-outliers=NUM-OUTLIERS
        -o OUTPUT-PREFIX, --output-prefix=OUTPUT-PREFIX
        -h, --help, Show this help message and exit
```

This script will read any trees in the file[s] provided as positional arguments, run kdetrees on them using the options provided, and write several output files to the current directory. The `outliers.tre` file will contain newick strings of

the outlier trees, these trees are also rendered in the `outliers.pdf` file. The scatterplot and histogram pdf filess contain the result of calling `plot` and `hist`, respectively, and the `results.csv` file contains the name, Newick string, and computed score for each tree found.

The final element of the result list `bandwidth` contains the bandwidths calculated by the nearest-neighbor algorithm. This is discussed further in the next section.

4 Advanced Options

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 `bw` is passed as a list, the list is used as a set of parameters for a call to `bw.nn`. For example, to change the neighborhood to include 50% of the sample, instead of the default 20%, we would do the following.

```
> kdetrees(apicomplexa,n=12,bw=list(prop=0.5),use.blen=TRUE)
```

If we wanted to set a constant bandwidth, we simply pass it directly to `bw`.

```
> kdetrees(apicomplexa,n=12,bw=6,use.blen=TRUE)
```

The `kdetrees` function is a fairly simple wrapper of a few component functions.

```
> kdetrees
```

```
function (trees, n = ceiling(0.05 * length(trees)), bw = list(),
    ...)
{
    dm <- dist.diss(trees, ...)
    if (is.list(bw))
        bw <- do.call(bw.nn, c(list(dm), bw))
    km <- normkern(dm, bw)
    i <- which.min(estimate(km))
    while (length(i) < n) {
        j <- which.min(estimate(km[-i, -i]))
        j[1] <- match(names(j), rownames(km))
        i <- c(i, j)
    }
    est <- estimate(km, i)
    out <- list(density = est, outliers = i, bandwidth = bw)
    class(out) <- "kdetrees"
    out
}
<environment: namespace:kdetrees>
```

Additional control over the method can be achieved by calling the `dist.diss`, `normkern`, and `estimate` functions separately, although this is not recommended unless you know what you are doing.