Using `ggphylo` and `ggplot` to visualize phylogenetic trees and alignments

Gregory Jordan

February 14, 2012

`ggphylo` is a package that provides a useful PI for manipulating `phylo` and alignment objects from R and plotting them using `ggplot`. Annotations can be loaded from NHX formatted trees or from associated CSV files and plotted in a simple, flexible way.

1 For the impatient

1.1 Getting started

To get started, simply input a `phylo` object (or a list of `phylo` objects) and call the `ggphylo` function:

```
> tree.list <- list()
> for (i in 1:3) {
+   x <- rtree(20)  # Random trees 20 leaves.
+   tree.list[[i]] <- x
+ }
> ggphylo(tree.list) # Plot the list of tres.
```

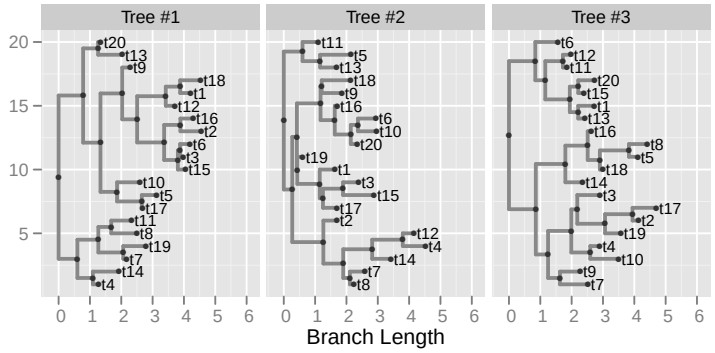

Figure 1: A few trees.

This is largely similar to the standard `plot.phylo` function. It also scales well to several trees:

```
> n <- 10
> sizes <- sample(2:20, n, replace=T)
> for (i in 1:n) {
+   tree.list[[i]] <- rtree(sizes[i])
+ }
> ggphylo(tree.list, label.size=2) # Plot the list of trees.
```

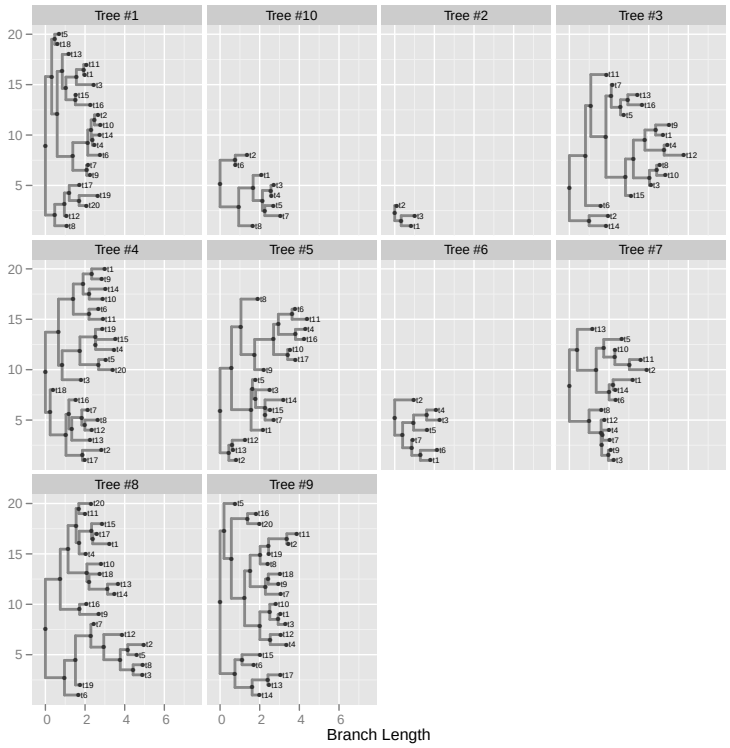

Figure 2: Many trees.

1.2 Plotting data along trees

An important aspect of `ggplot` is its expressive *grammar of graphics*, which allows data to be mapped easily and flexibly to any visual property. The `ggphylo` package hooks into `ggplot` by defining three visual entities (**lines, nodes, and labels**) and three visual properties (**color, alpha, size**). Any combination of entity and property can be mapped to a value from the tree. For example, we can map **bootstrap** values to **line color**, and **population size** values to **node size**:

```
> n <- 40;  x <- rtree(n); n.nodes <- length(nodes(x))
> bootstraps <- 100 - rexp(n.nodes, rate=5) * 100
> pop.sizes <- pmax(0, rnorm(n.nodes, mean=50000, sd=50000))
> for (i in nodes(x)) {
+   x <- tree.set.tag(x, i, 'bootstrap', bootstraps[i])
+   x <- tree.set.tag(x, i, 'pop.size', pop.sizes[i])
+ }
> plot.args <- list(
+   x,
+   line.color.by='bootstrap',
+   line.color.scale=scale_colour_gradient(limits=c(50, 100), low='red', high='black'),
+   node.size.by='pop.size',
+   node.size.scale = scale_size_continuous(limits=c(0, 100000), range=c(1, 5)),
+   label.size=2
+ )
> do.call(ggphylo, plot.args)
```

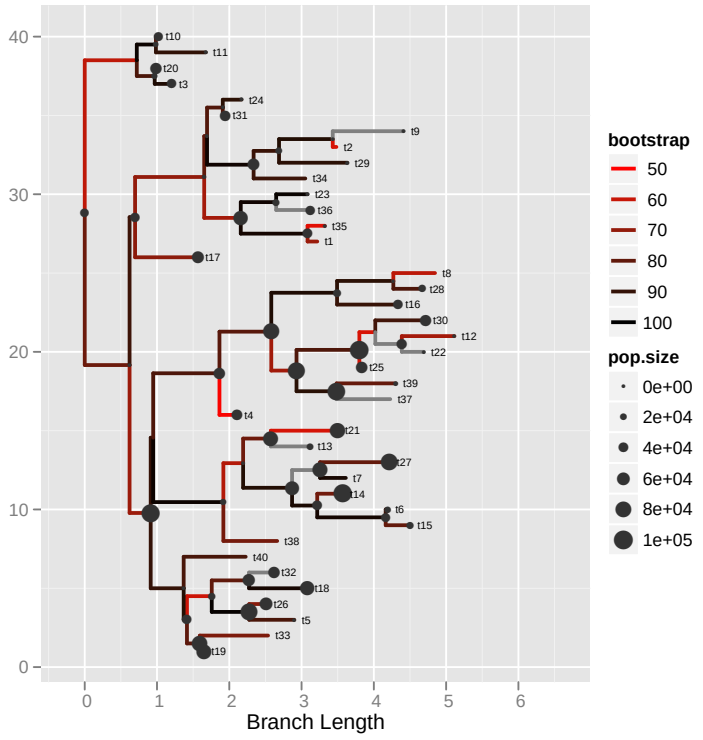

Figure 3: A tree with scalar values mapped to branch color and node size.

The previous example covers the main functionality provided by `ggphylo`: mapping data to visual elements along a tree. Note that, in order to store data along the tree, `ggphylo` introduces the concept of *tags*, which are stored internally as a list attached to each node of the `phylo` object. See `??` for more details.

It should be stressed that `ggplot` is an extremely versatile plotting system, capable of transforming data in myriad ways. For example, creating a radial view of the tree is handled internally by asking `ggplot` to transform the representation from cartesian to polar coordinates:

```
> radial.args <- plot.args
> radial.args[['layout']] <- 'radial'
> do.call(ggphylo, radial.args)


> unrooted.args <- plot.args
> unrooted.args[['layout']] <- 'unrooted'
> do.call(ggphylo, unrooted.args)
```

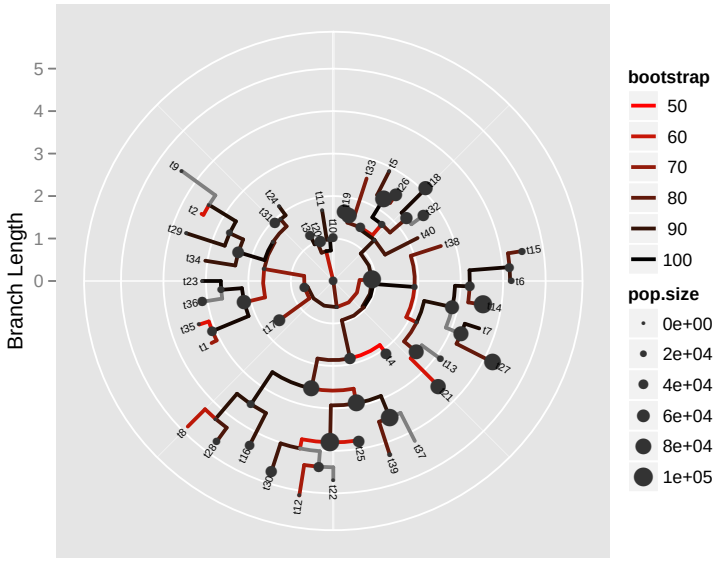

Figure 4: A "radial" tree drawn in polar coordinates by `ggphylo`.

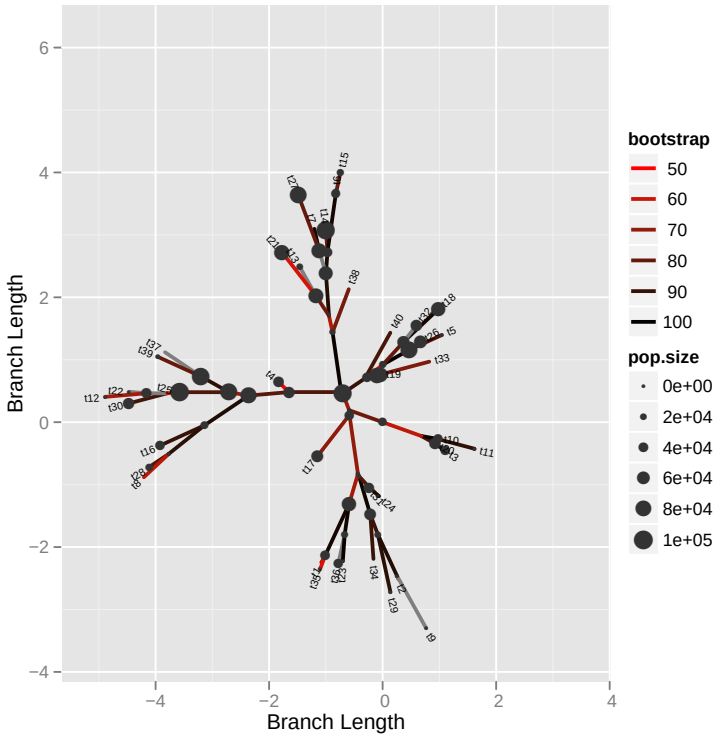

Figure 5: A tree drawn in an "unrooted" layout by `ggphylo`.