

# ggtree: a phylogenetic tree viewer for different types of tree annotations

Guangchuang Yu

[guangchuangyu@gmail.com](mailto:guangchuangyu@gmail.com)

School of Public Health, The University of Hong Kong

2015-01-06

## Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Tree visualization</b>	<b>2</b>
2.1	viewing tree with ggtree . . . . .	2
2.2	layout . . . . .	3
2.3	support multiple phylogenetic classes . . . . .	6
2.4	Display evolution distance . . . . .	6
2.5	display nodes/tips . . . . .	7
2.6	display labels . . . . .	8
2.7	theme . . . . .	9
2.8	update tree viewing with a new tree . . . . .	9
<b>3</b>	<b>Tree annotation</b>	<b>10</b>
3.1	annotating tree with BEAST output . . . . .	10
3.2	annotating tree with PAML output . . . . .	12
3.3	annotating tree with HYPHY output . . . . .	19
3.4	annotating tree with EPA and PPLACER output . . . . .	20
3.5	annotating tree using <i>ggplot2</i> layers . . . . .	21
3.6	user specific annotation . . . . .	22
3.7	jplace file format . . . . .	27
3.8	visualize tree and associated matrix . . . . .	29
<b>4</b>	<b>Session info</b>	<b>31</b>
<b>5</b>	<b>References</b>	<b>31</b>

You can't even begin to understand biology, you can't understand life, unless you understand what it's all there for, how it arose - and that means evolution. — Richard Dawkins

## 1 Introduction

---

This project came out with my needs to annotate nucleotide substitutions in the phylogenetic tree, and I found that there is no tree visualization software can do this easily. All the tree viewer are designed for displaying phylogenetic tree, but not annotating it. Although some tree viewers support displaying bootstrap values in the tree, it is hard/impossible to display other information in the tree. My first solution for displaying nucleotide substitution in the tree is to add these information in the node/tip names and use traditional tree viewer to show it. I displayed the information in the tree successfully, but I do believe this quick-and-dirty hack is ugly.

In the old day, we even don't have enough sequences to infer phylogenetic tree. At that time, as we almost don't have a need to annotate a tree, displaying the evolution relationships is mostly enough. Nowadays, we can obtain a lot of

data from different experiments, and we want to associate our data, for instance antigenic change, with the evolution relationship. Visualizing these associations in the phylogenetic tree can help us to identify evolution patterns. I believe we need a next generation tree viewer that can view a phylogenetic tree easily as we did with classical software and support adding annotation data in a layer one by one above the tree. This is the objective of developing the *ggtree*. Common tasks of annotating a phylogenetic tree should be easy and complicated tasks can be possible to achieve by adding multiple layers of annotation.

The *ggtree* is designed by extending the *ggplot2*<sup>1</sup> package. It based on grammar of graphics and takes all the good parts of *ggplot2*. There are other R packages that implemented tree viewer using *ggplot2*, including *OutbreakTools*, *phyloseq*<sup>2</sup> and *ggphylo*, but all of them only create complex tree view function for their specific needs. They are just classical tree viewers that only viewing the tree or annotating a specific data type. The good parts of *ggplot2* are not available in these packages. They lack of flexibilities of annotating phylogenetic tree by diverse user inputs.

## 2 Tree visualization

### 2.1 viewing tree with ggtree

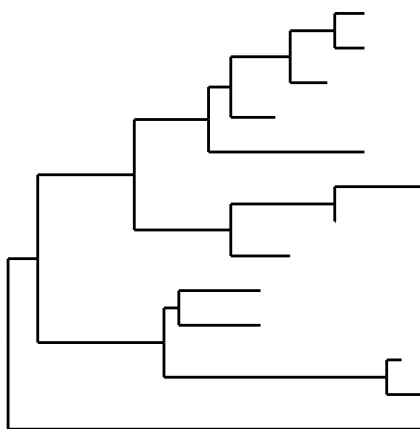
*ggtree* extend *ggplot* to support viewing phylogenetic tree. It implement *geom\_tree* layer for displaying phylogenetic tree, as shown below:

```
nwk <- system.file("extdata", "sample.nwk", package = "ggtree")
x <- readLines(nwk)
cat(substr(x, 1, 56), "\n", substr(x, 57, "\n"))

## ((((((A:4,B:4):6,C:5):8,D:6):3,E:21):10,((F:4,G:12):14,
## H:8):13):13,((I:5,J:2):30,(K:11,L:11):2):17):4,M:56);

library("ggtree")

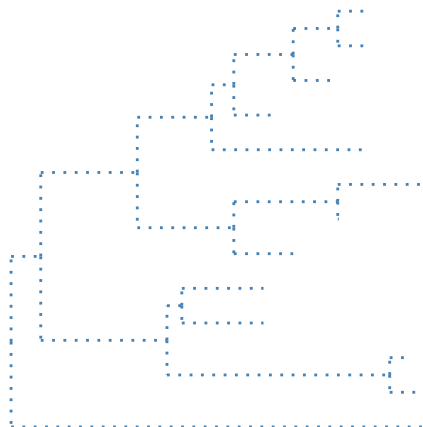
tree <- read.tree(nwk)
ggplot(tree, aes(x, y)) + geom_tree() + theme_tree() + xlab("") + ylab("")
```



This example tree was obtained from Chapter 34 of *Inferring Phylogenies*<sup>3</sup>. The function, *ggtree*, was implemented as a short cut to visualize a tree, and it works exactly the same as shown above.

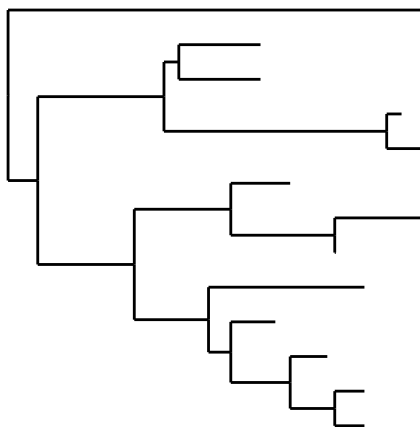
*ggtree* takes all the advantages of *ggplot2*. For example, we can change the color, size and type of the lines as we did with *ggplot2*.

```
ggtree(tree, color="steelblue", size=0.5, linetype="dotted")
```



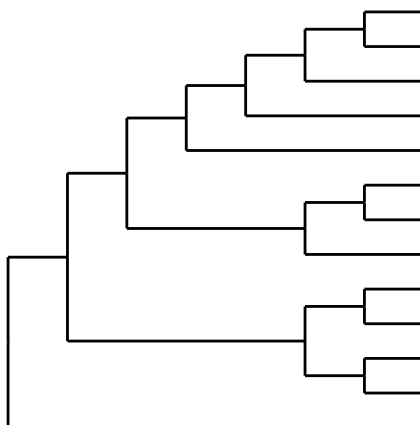
By default, the tree is viewing in ladderize form, user can set the parameter `ladderize = FALSE` to disable it.

```
ggtree(tree, ladderize=FALSE)
```



The `branch.length` is used to scale the edge, user can set the parameter `branch.length = "none"` to only viewing the tree topology.

```
ggtree(tree, branch.length="none")
```



## 2.2 layout

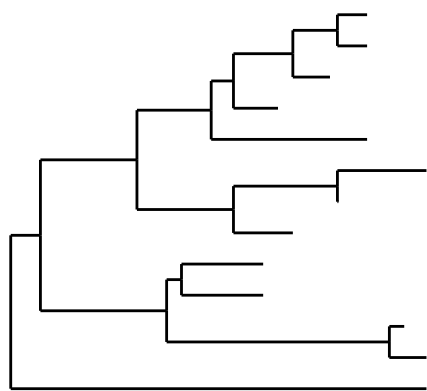
Currently, `ggtree` supports several layout, including:

- phylogram (by default)
- cladogram
- dendrogram
- fan
- unrooted.

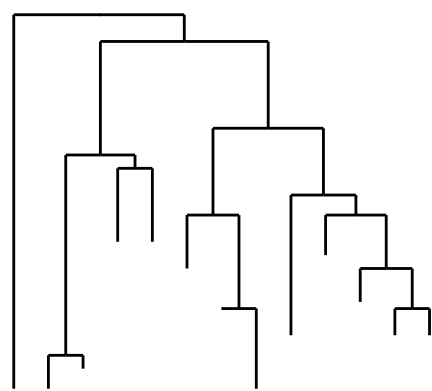
Unrooted layout was implemented by the *equal-angle algorithm* that described in *Inferring Phylogenies*<sup>3</sup>.

```
library("gridExtra")
grid.arrange(ggtree(tree) + ggtitle("phylogram layout"),
             ggtree(tree, layout="dendrogram") + ggtitle("dendrogram layout"),
             ggtree(tree, layout="cladogram") + ggtitle("cladogram layout"),
             ggtree(tree, layout="cladogram", branch.length="none") +
               scale_x_reverse()+coord_flip() + ggtitle("cladogram layout"),
             ggtree(tree, layout="fan") + ggtitle("fan layout"),
             ggtree(tree, layout="unrooted") + ggtitle("unrooted layout"),
             ncol=2)
```

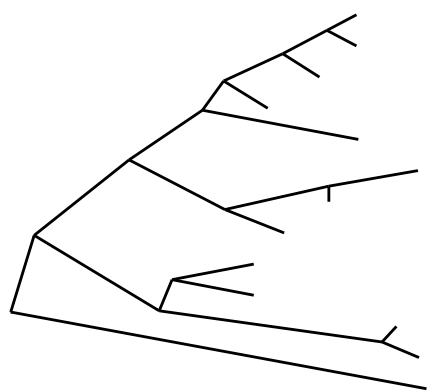
phylogram layout



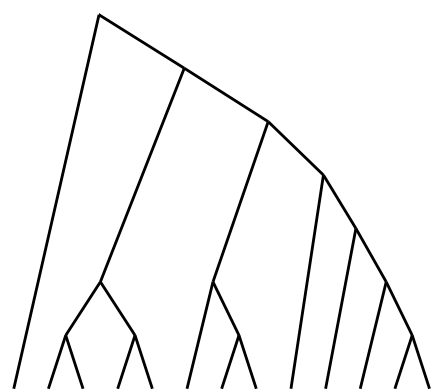
dendrogram layout



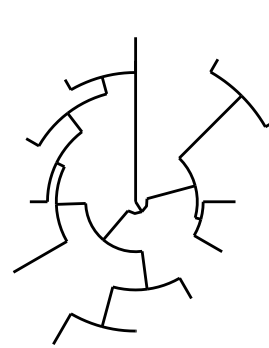
cladogram layout



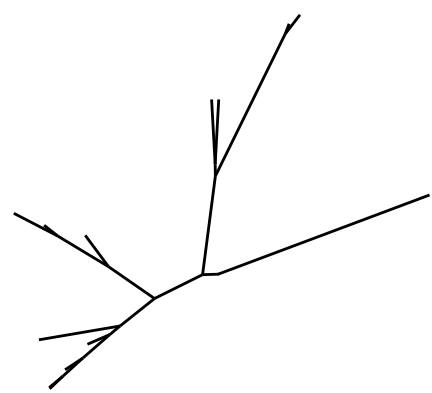
cladogram layout



fan layout



unrooted layout



## 2.3 support multiple phylogenetic classes

*ggtree* defined several S4 classes to store phylogenetic object and its associated annotation, including:

- jplace
- palm\_rst
- codeml\_mlc
- codeml
- hyphy
- beast

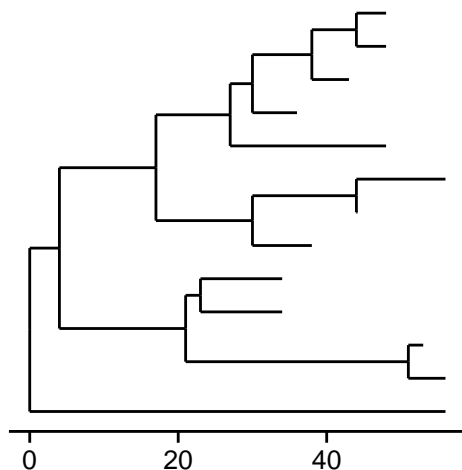
In addition, it also supports *phylo* (defined by [ape](#)<sup>4</sup>), and *phylo4* (defined by [phylobase](#))

User can use *ggtree(object)* command to view the phylogenetic tree directly, and annotation data stored in these objects can be added as demonstrated in [Tree annotation](#) session.

## 2.4 Display evolution distance

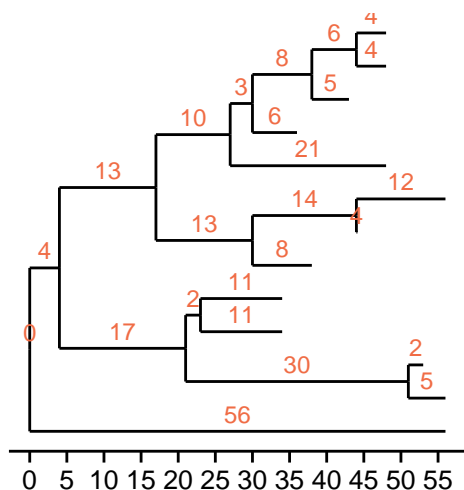
To show evolution distance, we can use *theme\_tree2()* or *ggtree(showDistance=TRUE)*

```
ggtree(tree) + theme_tree2()
```



Another way is to show the edge length of the tree. Besides, the scale of branch length can be specify via *scale\_x\_continuous()*.

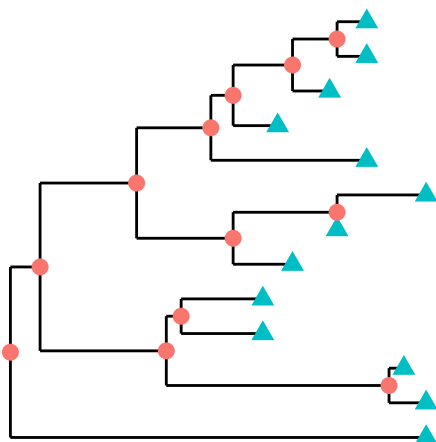
```
ggtree(tree, showDistance=TRUE) +  
  geom_text(aes(label=length, x=branch), size = 3,  
    vjust=-0.5, color="#F06C45") +  
  scale_x_continuous(breaks=seq(0, 60, 5))
```



## 2.5 display nodes/tips

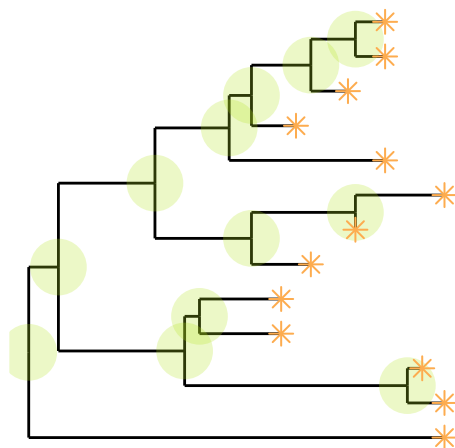
Show all the internal nodes and tips in the tree can be done by adding a layer of points using *geom\_point*.

```
ggtree(tree)+geom_point(aes(shape=isTip, color=isTip), size=3)
```



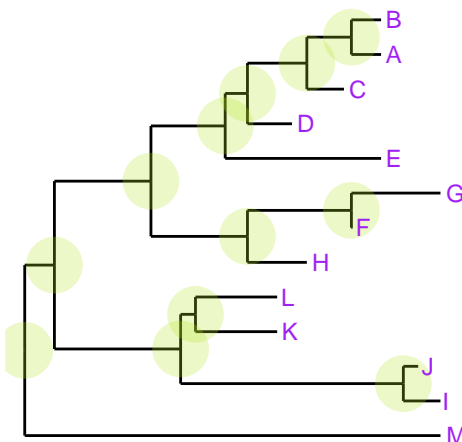
And of course, we can separate nodes and tips by using *subset*.

```
p <- ggtree(tree) + geom_point(subset=.(!isTip),
                              color="#b5e521", alpha=1/4, size=10)
p + geom_point(color="#FDAC4F", shape=8, size=3, subset=.(isTip))
```



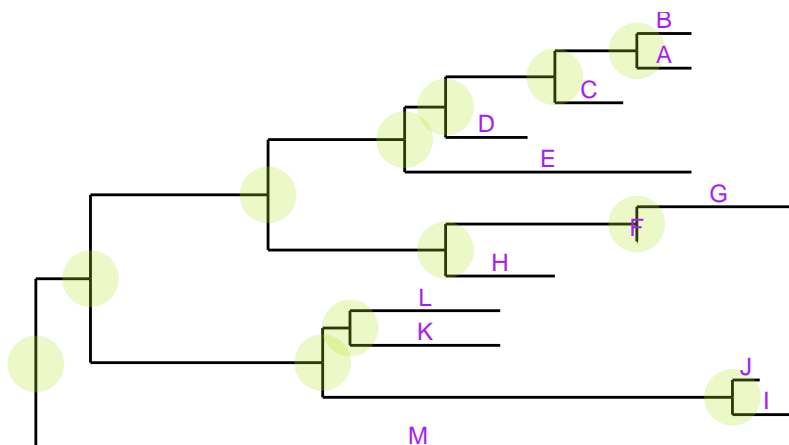
## 2.6 display labels

```
p + geom_text(aes(label=label), size=3, color="purple", hjust=-0.3)
```



By default, the positions are based on the node positions, we can change them to based on the middle of the branch/edge.

```
p + geom_text(aes(x=branch, label=label), size=3, color="purple", vjust=-0.3)
```



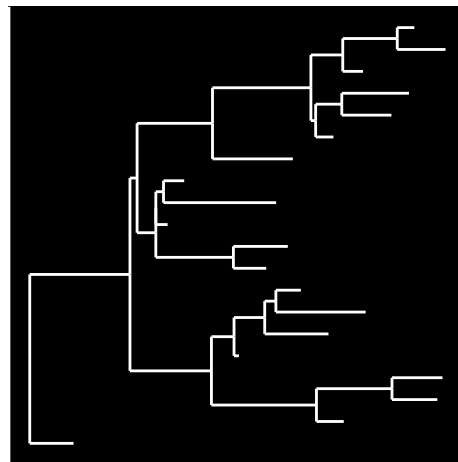
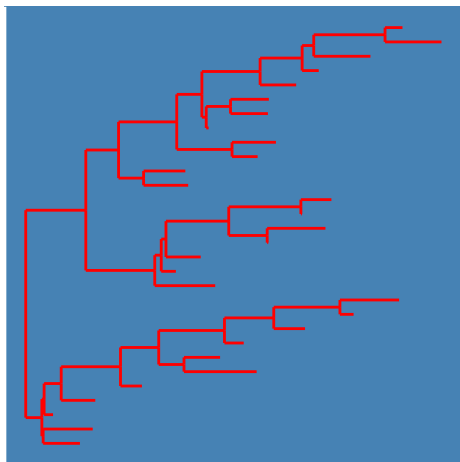
Based on the middle of branches is very useful when annotating transition from parent node to child node.



## 2.7 theme

`theme_tree()` defined a totally blank canvas, while `theme_tree2()` add phylogenetic distance legend. These two themes all accept a parameter of `bgcolor` that defined the background color.

```
grid.arrange(
  ggtree(rtree(30), color="red") + theme_tree("steelblue"),
  ggtree(rtree(20), color="white") + theme_tree("black"),
  ncol=2)
```

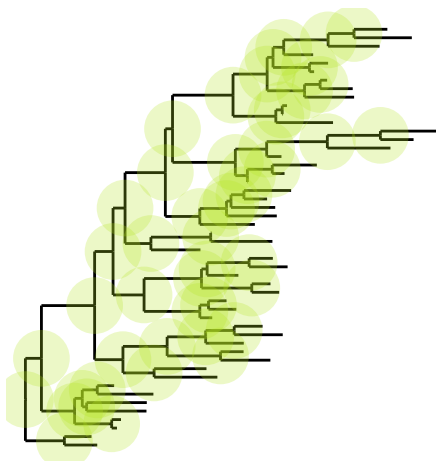


## 2.8 update tree viewing with a new tree

In the **Display nodes/tips** section, we have a `p` object that stored the tree viewing of 13 tips and internal nodes highlighted with specific colored big dots. If you want to applied this pattern, and we can imaging more complex one, to a new tree, we don't need to build the tree view step by step. `ggtree` provides an operator, `%<%`, for applying the visualization pattern to a new tree.

For example, the pattern in the `p` object will be applied to a new tree with 50 tips as shown below:

```
p %<% rtree(50)
```



Another example can be found in [CODEML](#) session.

### 3 Tree annotation

---

In *ggtree*, we implemented several functions to parse the output from [PAML](#)<sup>5</sup>, [HYPHY](#)<sup>6</sup>, [EPA](#)<sup>7</sup>, [PPLACER](#)<sup>8</sup> and [BEAST](#)<sup>9</sup> and defined several classes to store phylogenetic object and associated annotation.

Classes include:

- *palm\_rst* for *rst* file obtained by [PAML](#)<sup>5</sup>, including *BASEML* and *CODEML*.
- *codeml\_mlc* for *mlc* file obtained by *CODEML*.
- *codeml* for interpreting *rst* and *mlc* files obtained by *CODEML*.
- *hyphy* for [HYPHY](#)<sup>6</sup> output.
- *jplace* for [EPA](#)<sup>7</sup> and [PPLACER](#)<sup>8</sup> output.
- *beast* for [BEAST](#)<sup>9</sup>

*jplace* class is also designed to store user specific annotation data, and serves as a standard format for tree annotation within the *ggtree* package. Please refer to the [jplace file format](#) session.

For each classes, we defined *read.className* to parse input file and output a corresponding object, *get.fields* method to get the annotation features available in the object, access methods to get these features, and *plot* methods for quickly viewing these annotation features.

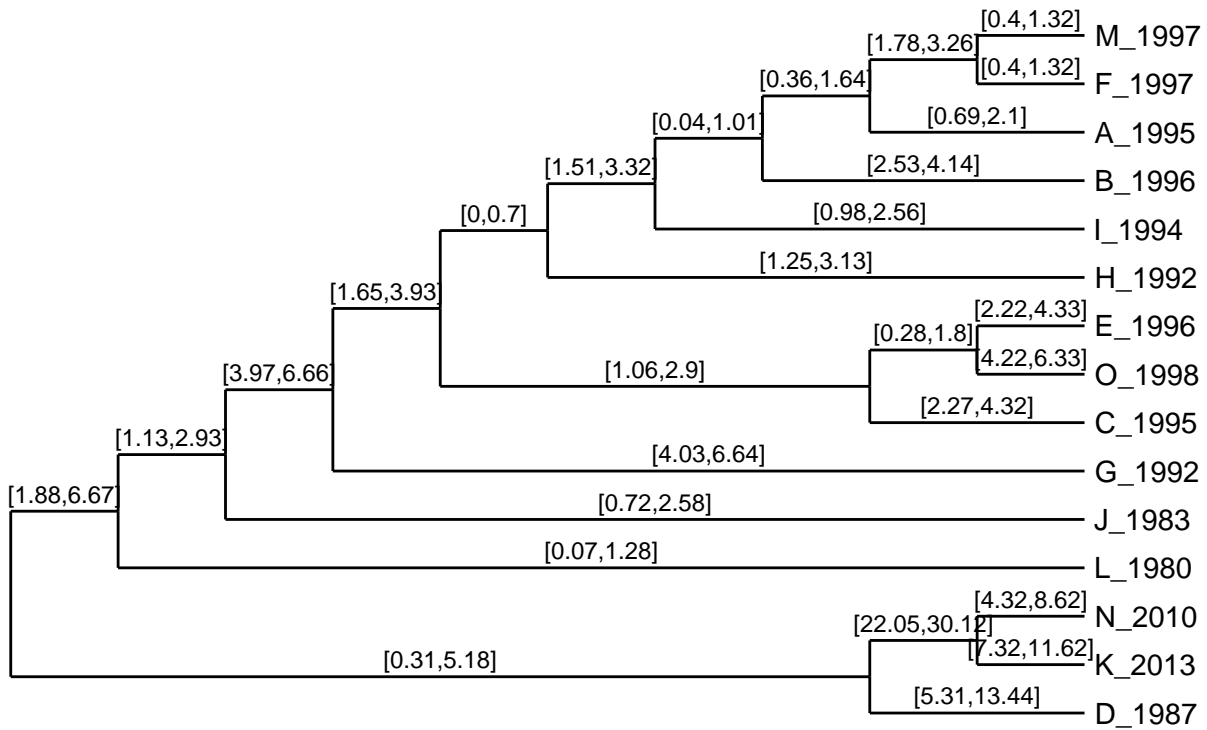
#### 3.1 annotating tree with BEAST output

```
file <- system.file("extdata/BEAST", "beast_mcc.tree", package="ggtree")
beast <- read.beast(file)
beast
```

```
## 'beast' S4 object that stored information of
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/BEAST/beast_mcc.tree'.
##
## ...@ tree:
## Phylogenetic tree with 15 tips and 14 internal nodes.
##
## Tip labels:
## A_1995, B_1996, C_1995, D_1987, E_1996, F_1997, ...
##
## Rooted; includes branch lengths.
##
## with the following features available:
##   'height', 'height_95._HPD', 'height_median', 'height_range', 'length',
##   'length_95._HPD', 'length_median', 'length_range', 'posterior', 'rate',
##   'rate_95._HPD', 'rate_median', 'rate_range'.
```

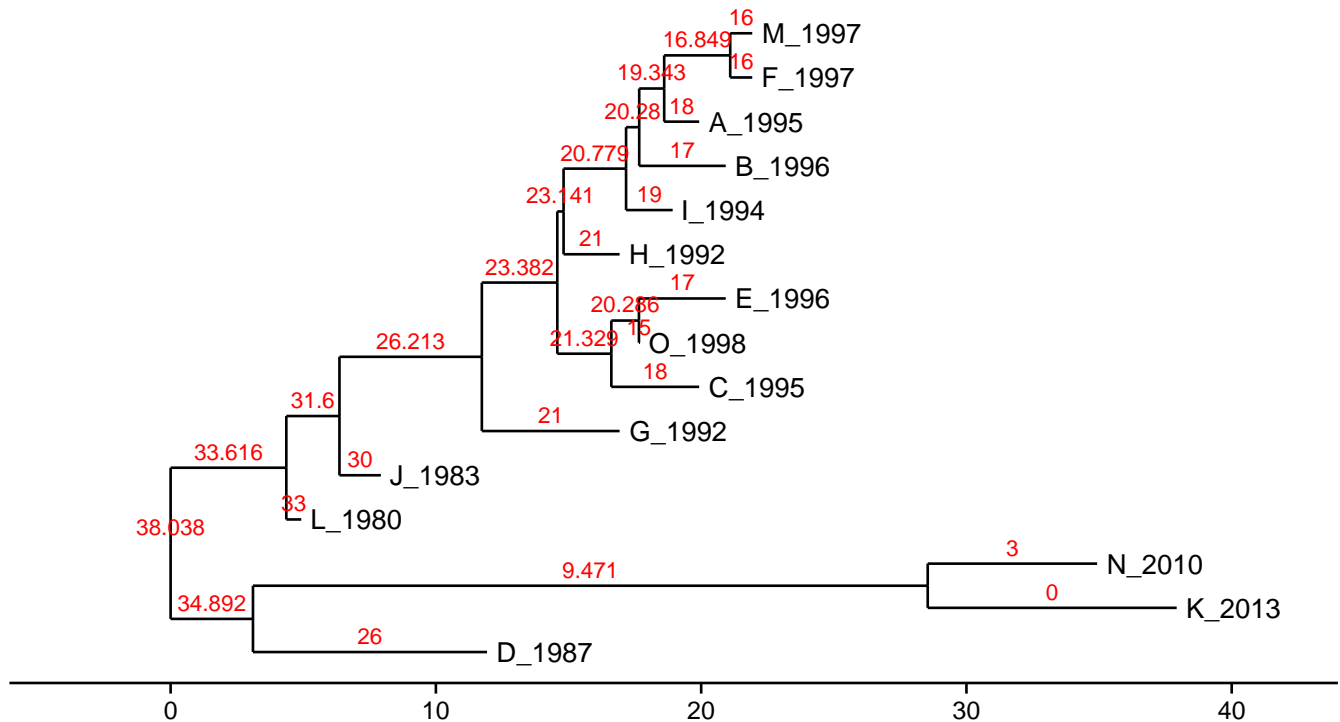
Since % is not a valid character in *names*, all the feature names that contain % will convert to .. For example, *length\_95%\_HPD* will be changed to *length\_95.\_HPD*.

```
plot(beast, annotation = "length_95._HPD", branch.length = "none") + theme_tree()
```



User can round the digits by setting the parameter *ndigits*. The default value is 2.

```
plot(beast, annotation = "height", ndigits = 3, annotation.color = "red")
```



## 3.2 annotating tree with PAML output

### 3.2.1 BASEML

**3.2.1.1 rst file** *rst* file from *baseml* is similar to *codeml* output. The only difference is the space in the sequences. For *baseml*, each ten bases are separated by one space, while for *codeml*, each three bases (triplet) are separated by one space. We defined a *read.paml\_rst* to parse *rst* file. It supports *baseml* and *codeml* output. The information will be stored in *paml\_rst* S4 object.

```
rstfile <- system.file("extdata/PAML_Baseml", "rst", package="ggtree")
tipfas <- system.file("extdata", "pa.fas", package="ggtree")
rst <- read.paml_rst(rstfile, tipfas)
rst

## 'paml_rst' S4 object that stored information of
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/PAML_Baseml/rst' and
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/pa.fas'.
##
## ...@ tree:
## Phylogenetic tree with 15 tips and 13 internal nodes.
##
## Tip labels:
##  A, B, C, D, E, F, ...
## Node labels:
##  16, 17, 18, 19, 20, 21, ...
##
## Unrooted; includes branch lengths.
##
## with the following features available:
##   'marginal_subs',   'joint_subs',   'marginal_AA_subs', 'joint_AA_subs'.
```

The function *read.paml\_rst* can accept only one parameter, *rstfile*, and the output can be used to view the phylogeny. But if we want to view the substitution annotation, we should provide tip sequence fasta file to parameter *tip.fasfile*, since *rstfile* only contain inferred ancestral sequences.

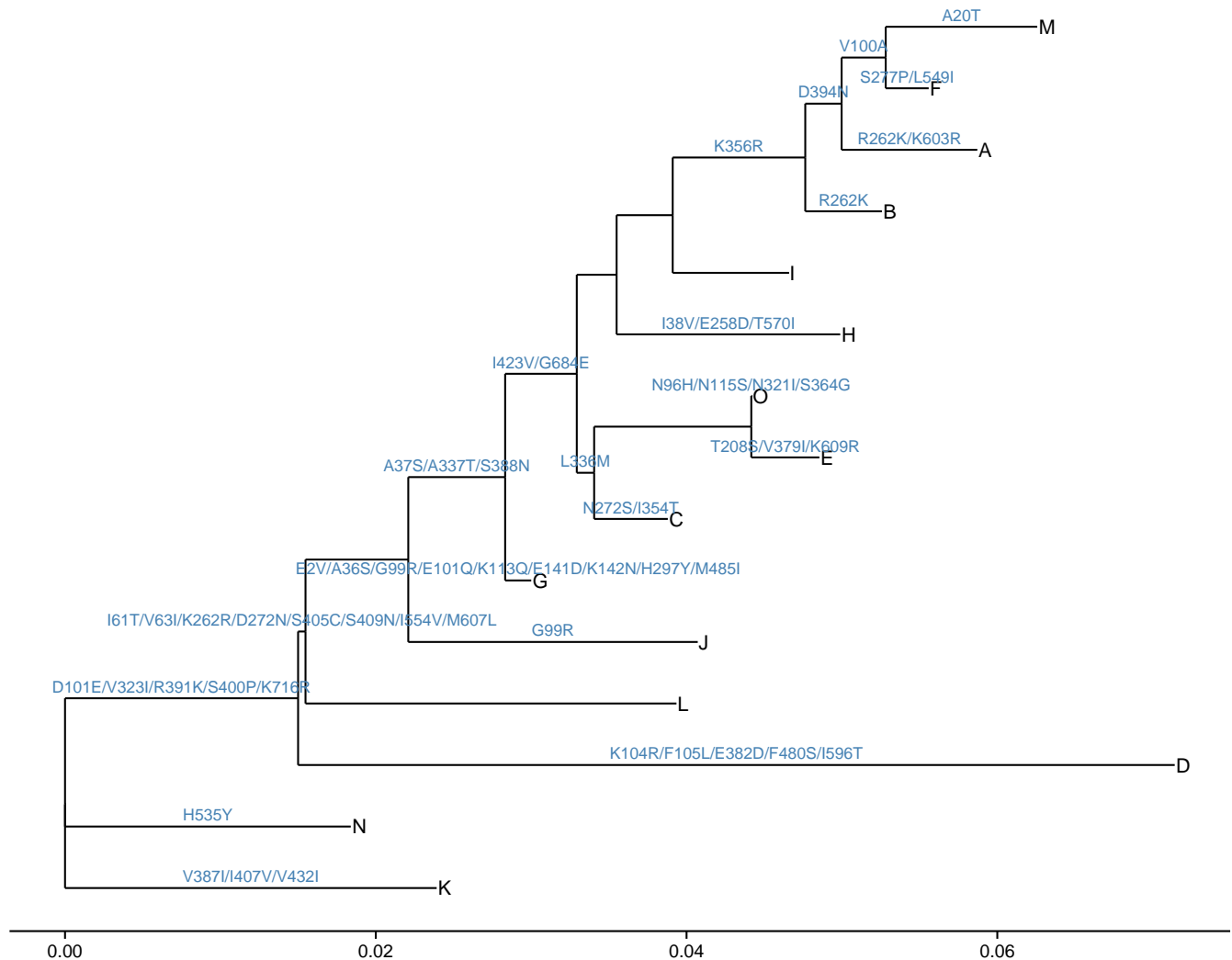
*mlb* file of *baseml* output do contain tip sequences, and *ggtree* provides another function *read.baseml* to parse *rstfile* and *mlbfile* simultaneously.

So for viewing substitution annotation, we can use *read.paml\_rst* with *rstfile* and *tip.fasfile* provided or use *read.baseml* with *rstfile* and *mlbfile* provided.

```
mlbfile <- system.file("extdata/PAML_Baseml", "mlb", package="ggtree")
baseml <- read.baseml(rstfile, mlbfile)
baseml

## 'paml_rst' S4 object that stored information of
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/PAML_Baseml/rst'.
##
## ...@ tree:
## Phylogenetic tree with 15 tips and 13 internal nodes.
##
## Tip labels:
##  A, B, C, D, E, F, ...
## Node labels:
##  16, 17, 18, 19, 20, 21, ...
##
## Unrooted; includes branch lengths.
```

```
##  
## with the following features available:  
## 'marginal_subs', 'joint_subs', 'marginal_AA_subs', 'joint_AA_subs'.  
p <- plot(rst, annotation = "marginal_AA_subs", annotation.color = "steelblue")  
print(p)
```



The following command will generate the same figure.

```
plot(baseml, annotation="marginal_AA_subs", annotation.color="steelblue")
```

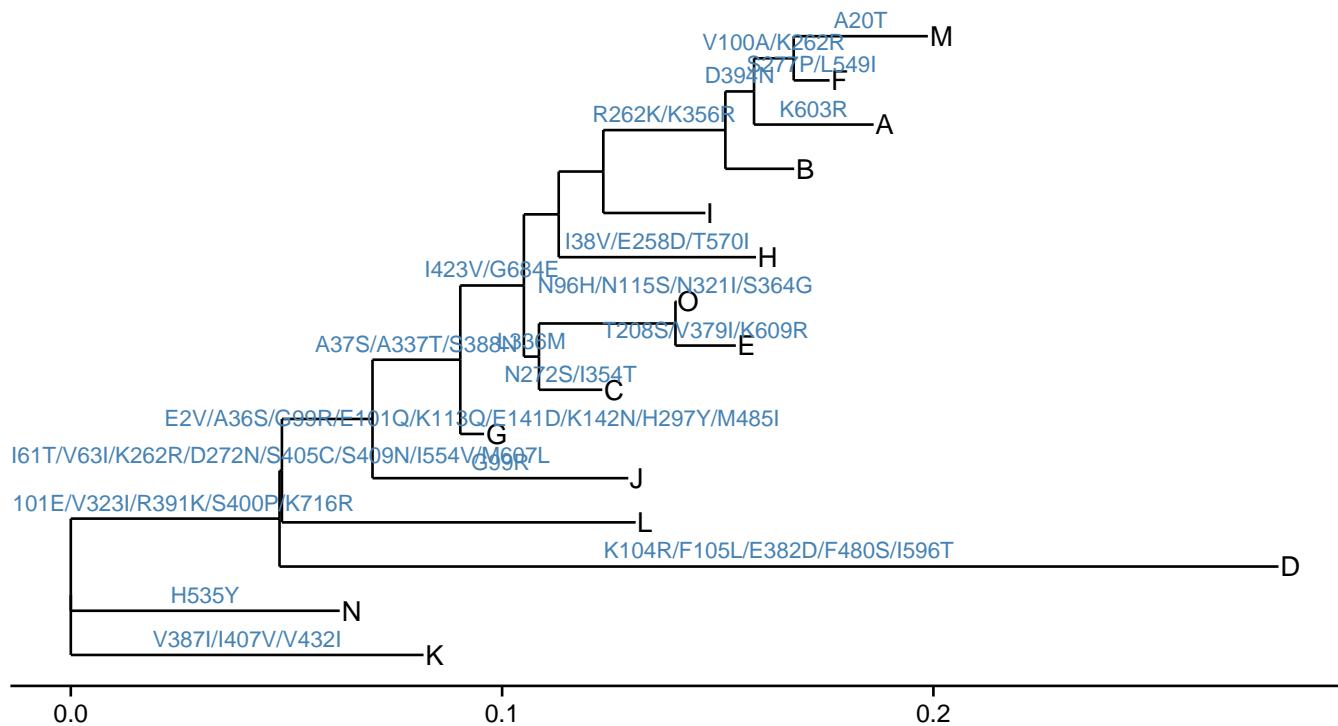
### 3.2.2 CODEML

**3.2.2.1 rst file** *rst* file from *CODEML* is similar to *BASEML*, and also parsed by *read.paml\_rst* function. The *plot* method works also in the same way.

If you remember the `%<` operator introduced in [update-tree-viewing-with-a-new-tree](#) session, you can use it to update a tree view with a new object.

In last session, we use *rstfile* of *BASEML* to build a tree view with amino acid substitution annotated. The following example use another *rstfile* from *CODEML* to update the tree view.

```
rstfile <- system.file("extdata/PAML_Codeml", "rst", package = "ggtree")
rst <- read.paml_rst(rstfile, tipfas)
p %<% rst
```



You can find that these two figures have different evolution distances, and substitutions inferred from *BASEML* and *CODEML* are slightly different.

```
mlcfile <- system.file("extdata/PAML_Codeml", "mlc", package = "ggtree")
mlc <- read.codeml_mlcm(mlcfile)
mlc
```

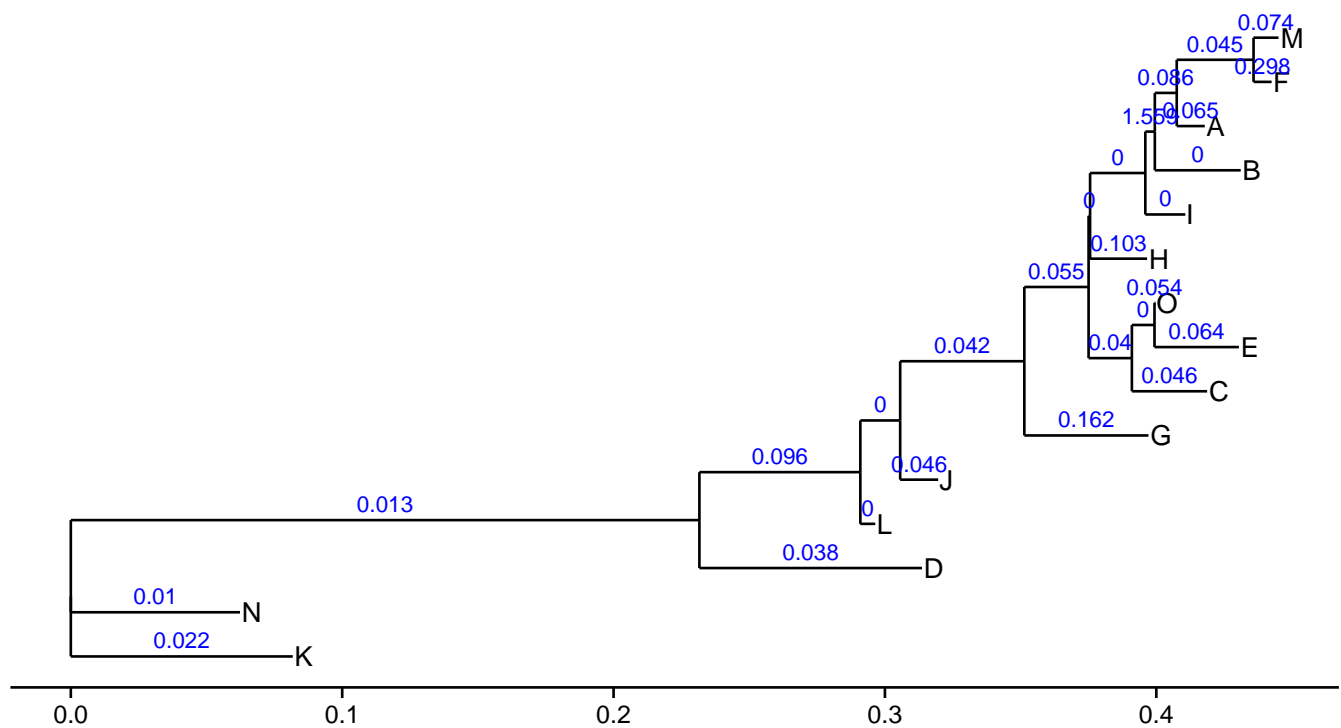
### 3.2.2.2 mlc file

```
## 'codeml_mlcm' S4 object that stored information of
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/PAML_Codeml/mlc'.
##
## ...@ tree:
## Phylogenetic tree with 15 tips and 13 internal nodes.
##
## Tip labels:
##   A, B, C, D, E, F, ...
## Node labels:
##   16, 17, 18, 19, 20, 21, ...
##
## Unrooted; includes branch lengths.
##
## with the following features available:
##   't', 'N', 'S', 'dN.dS', 'dN', 'dS', 'NxdN', 'SxdS'.
```

Please aware that / and \* are not valid characters in *names*, they were changed to . and x respectively.

So  $dN.dS$  is  $dN/dS$ ,  $NxdN$  is  $N*dN$ , and  $SxdS$  is  $S*dS$ .

```
plot(mlc, branch.length = "branch.length", annotation = "dN.dS", annotation.color = "blue",
      ndigits = 3)
```



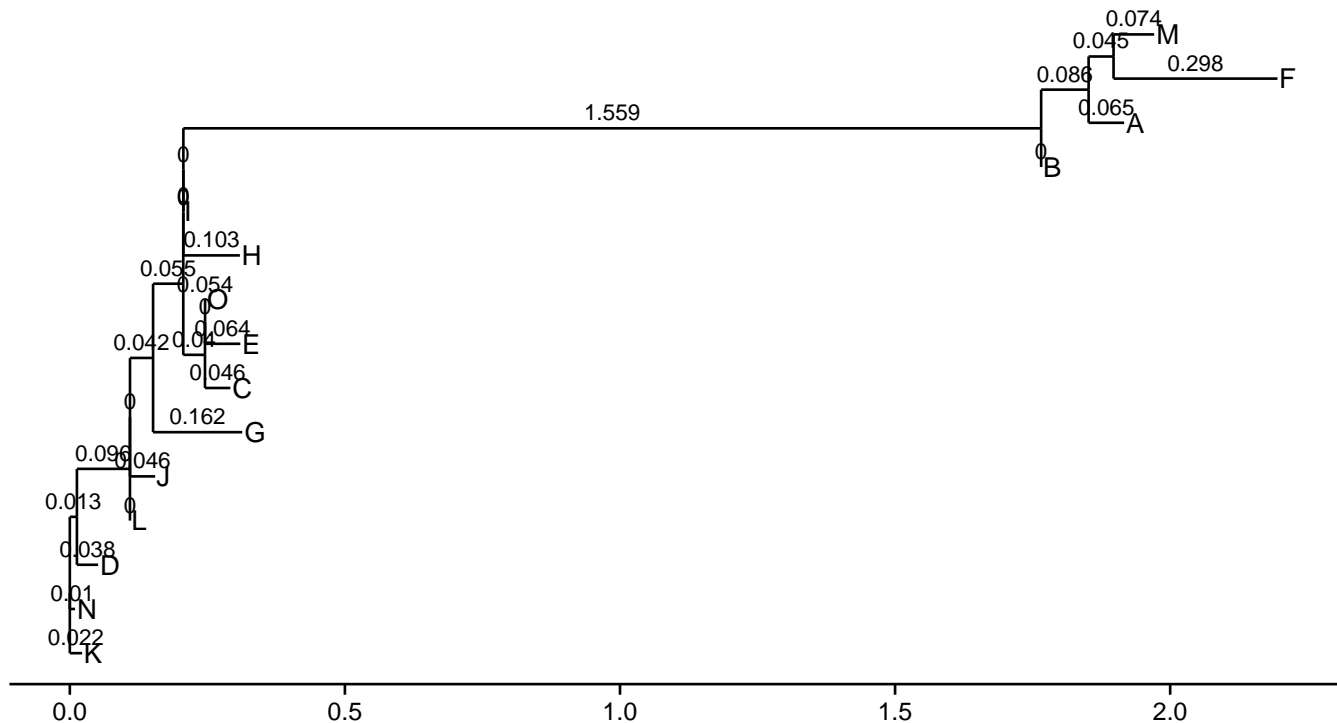
The parameter `branch.length` can be one of available annotations:

```
get.fields(mlc)
```

```
## [1] "t"      "N"      "S"      "dN.dS" "dN"     "dS"     "NxdN"  "SxdS"
```

For example, if we set `branch.length` to `dN.dS`, it will plot the  $\omega$  ( $dN/dS$ ) tree:

```
plot(mlc, branch.length = "dN.dS", annotation = "dN.dS", ndigits = 3)
```



We can also plot the  $dN$  or  $dS$  tree and others. The parameter *annotation* can also be one of the available annotations.

**3.2.2.3 CODEML output: mlc and rst files** We annotate the tree with information presented in *rstfile* and *mlcfile* separately as demonstrated in previous sessions.

We can also use both of them and it's highly recommended. User don't need to provide tip sequences, as it's already available in *mlcfile*. All the features in both files are available for annotation.

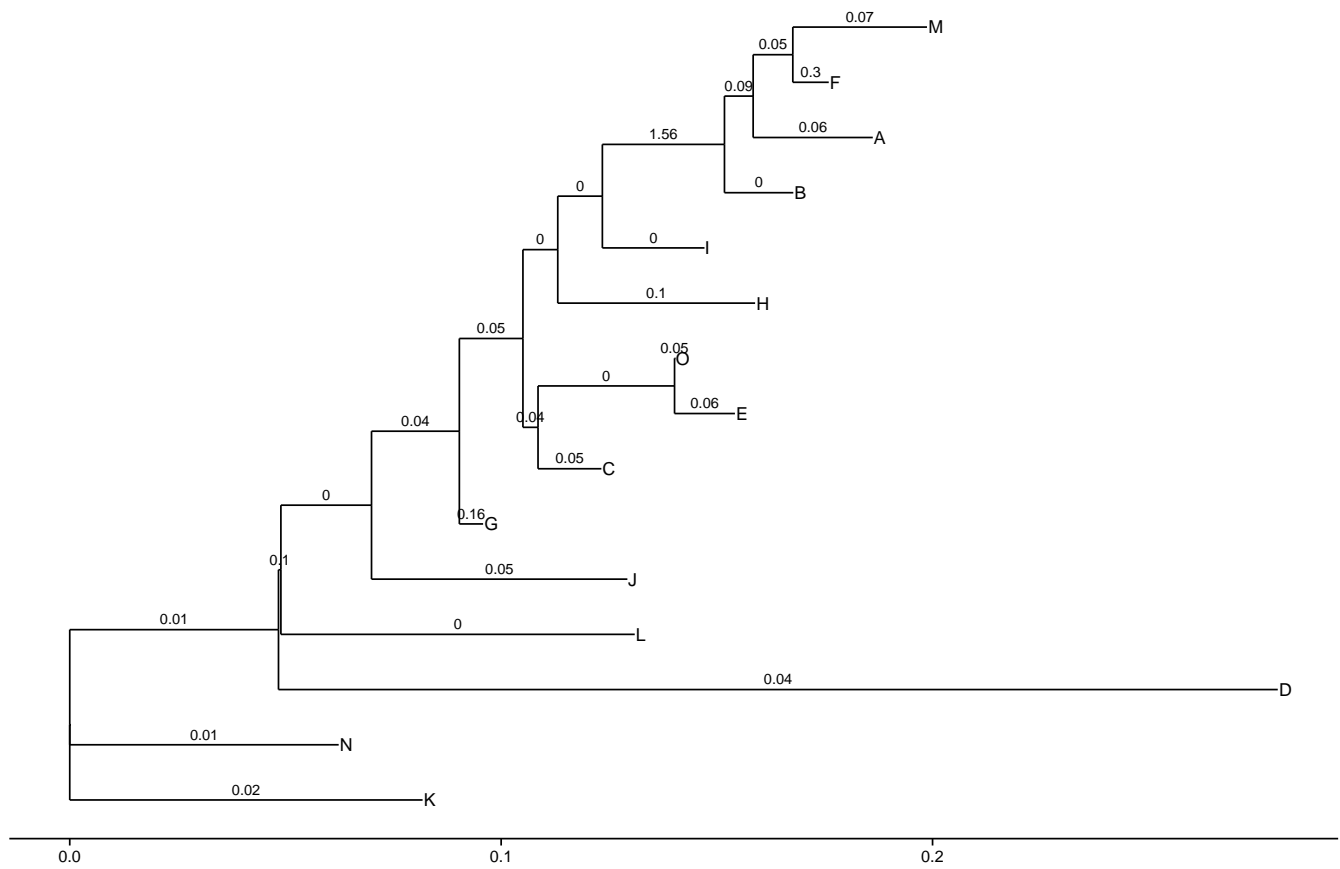
```
ml <- read.codeml(rstfile, mlcfile)
ml

## 'codeml' S4 object that stored information of
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/PAML_Codeml/rst' and
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/PAML_Codeml/mlc'.
##
## ...@ tree:
## Phylogenetic tree with 15 tips and 13 internal nodes.
##
## Tip labels:
## A, B, C, D, E, F, ...
## Node labels:
## 16, 17, 18, 19, 20, 21, ...
##
## Unrooted; includes branch lengths.
##
## with the following features available:
##   'marginal_subs', 'joint_subs', 'marginal_AA_subs', 'joint_AA_subs',
##   't', 'N', 'S', 'dN.dS',
##   'dN', 'dS', 'NxdN', 'SxdS',
## .
```

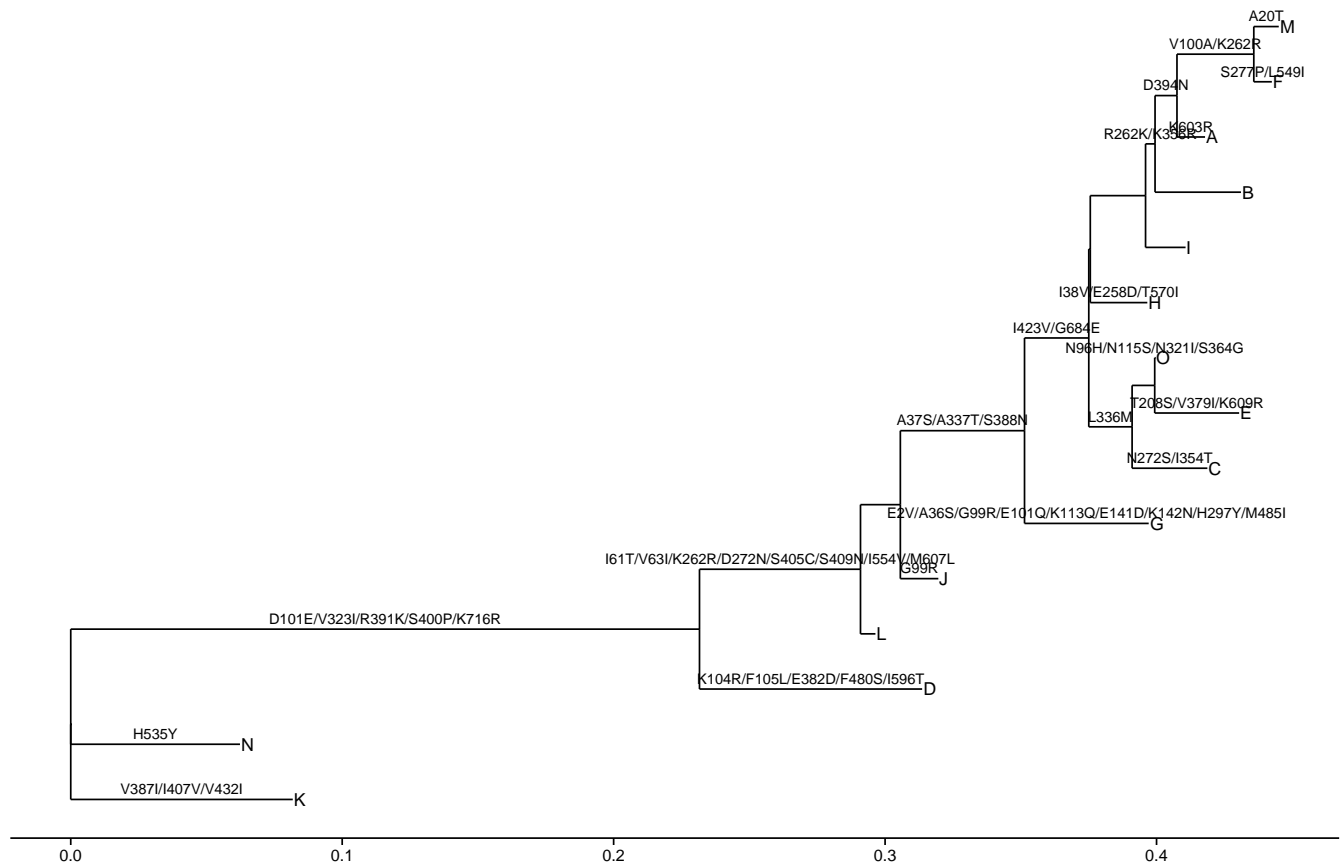
So we can annotate  $dN/dS$  with the tree in *rstfile* and amino acid substitutions with the tree in *mlcfile*.



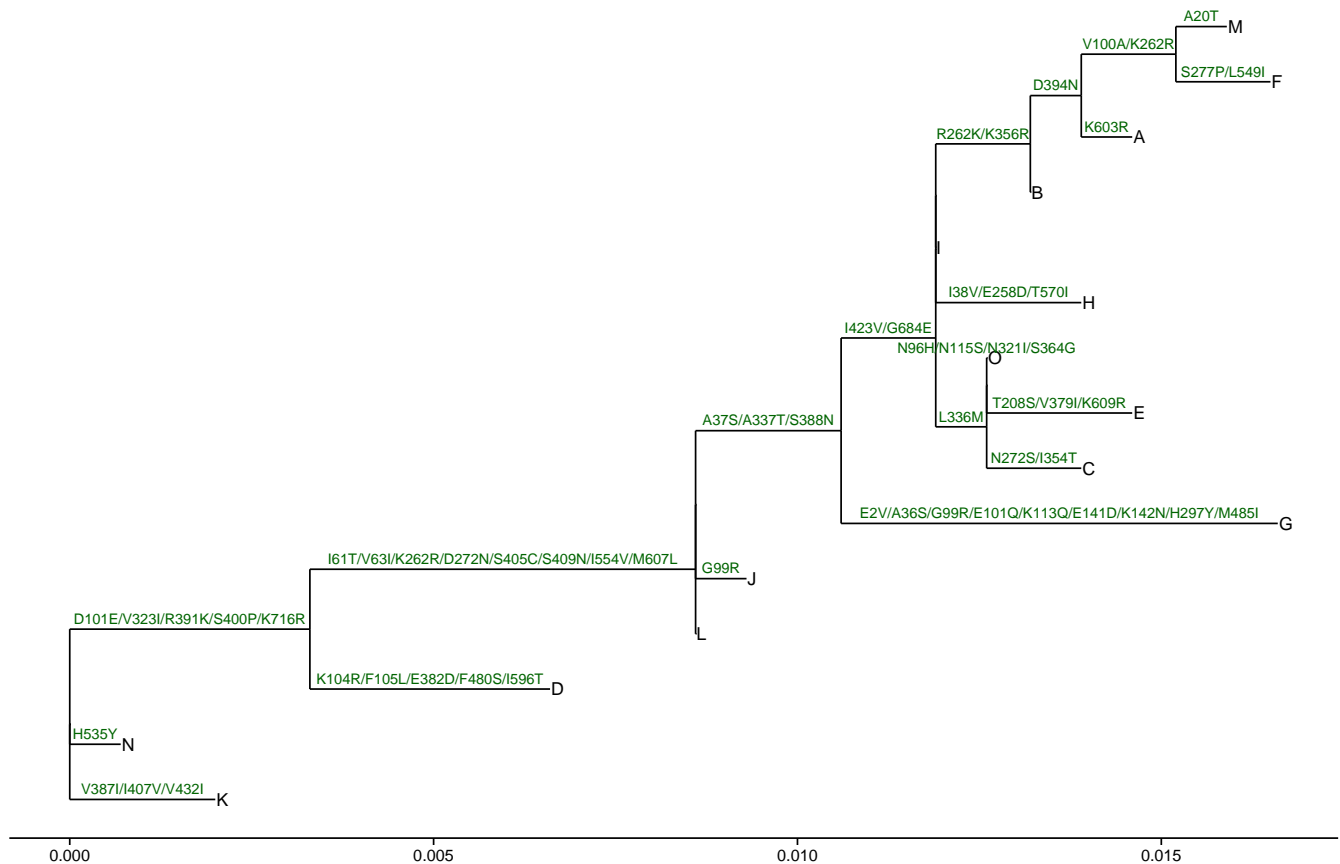
```
plot(ml, branch.length = "rst.branch.length", annotation = "dN.dS")
```



```
plot(ml, branch.length = "mlc.branch.length", annotation = "marginal_AA_subs")
```



```
plot(ml, branch.length = "dN", annotation = "joint_AA_subs", annotation.color = "darkgreen")
```

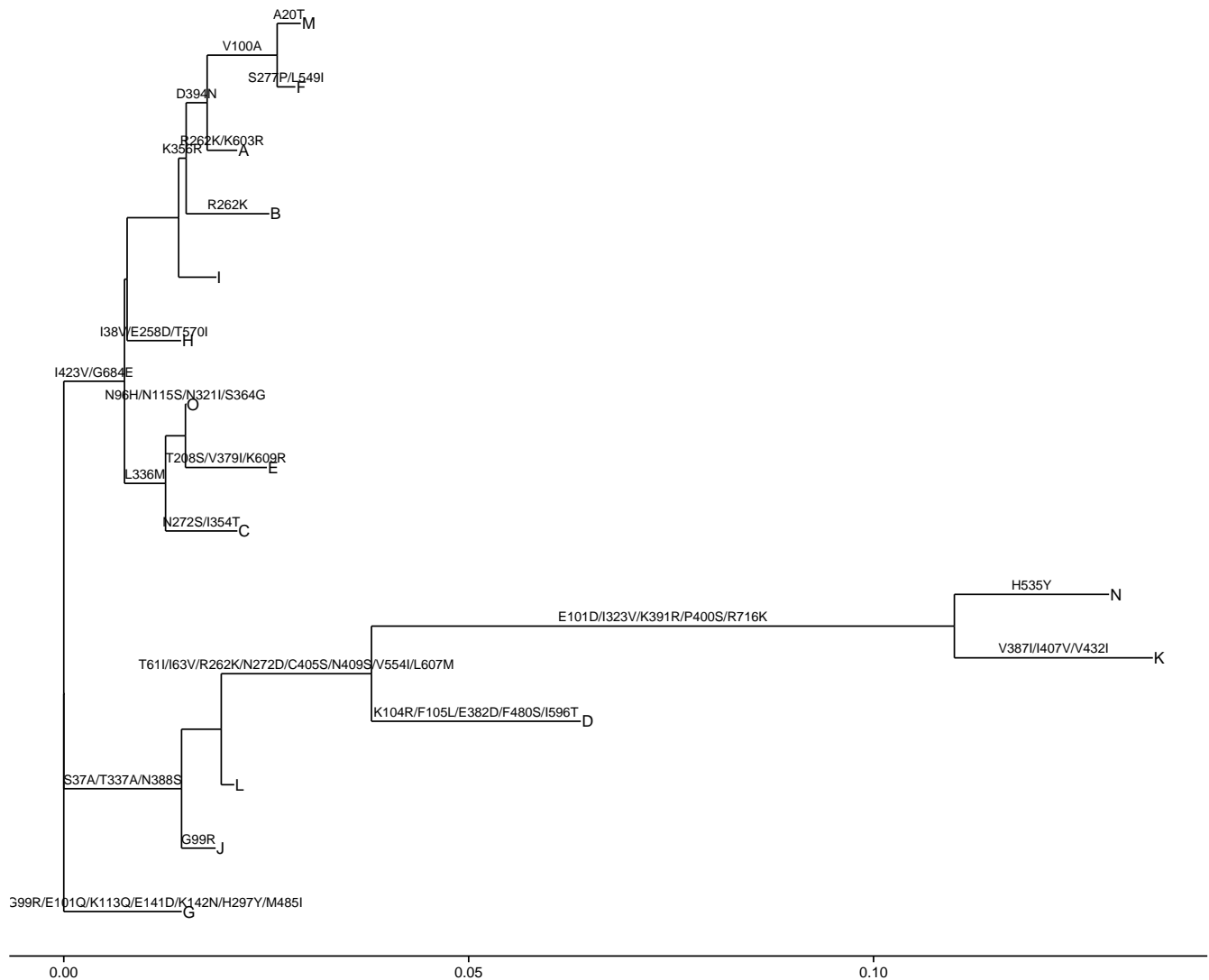


### 3.3 annotating tree with HYPHY output

```
nwk <- system.file("extdata/HYPHY", "labelledtree.tree", package="ggtree")
ancseq <- system.file("extdata/HYPHY", "ancseq.nex", package="ggtree")
hy <- read.hyphy(nwk, ancseq, tipfas)
hy
```

```
## 'hyphy' S4 object that stored information of
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/HYPHY/labelledtree.tree',
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/HYPHY/ancseq.nex' and
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/pa.fas'.
##
## ...@ tree:
## Phylogenetic tree with 15 tips and 13 internal nodes.
##
## Tip labels:
## K, N, D, L, J, G, ...
## Node labels:
## Node1, Node2, Node3, Node4, Node5, Node12, ...
##
## Unrooted; includes branch lengths.
##
## with the following features available:
##   'subs',    'AA_subs'.
```

```
plot(hy, annotation = "AA_subs")
```



### 3.4 annotating tree with EPA and PPLACER output

EPA<sup>7</sup> and PPLACER<sup>8</sup> have common output file format, jplace.

```
jpf <- system.file("extdata/sample.jplace", package="ggtree")
jp <- read.jplace(jpf)
print(jp)
```

```
## 'jplace' S4 object that stored information of
##   '/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/sample.jplace'.
##
## ...@ tree:
## Phylogenetic tree with 13 tips and 12 internal nodes.
##
## Tip labels:
## A, B, C, D, E, F, ...
##
```

```
## Rooted; includes branch lengths.
##
## with the following features available:
##   'edge_num',   'likelihood',   'like_weight_ratio',   'distal_length',   'pendant_length'.
```

This is only a tiny sample file. In reality, [EPA](#) and [PPLACER](#) may place thousands of short reads on a reference tree.

In [ggtree](#), we provide `get.placements` method to access the placement.

```
## get only best hit
get.placements(jp, by="best")

##   name edge_num likelihood like_weight_ratio distal_length pendant_length
## 1  AA      24  -61371.30      0.333344      3e-06      0.003887
## 2  BB       1  -61312.21      0.333335      1e-06      0.000003
## 3  CC       8  -61312.23      0.200011      1e-06      0.000003
```

```
## get all placement
get.placements(jp, by="all")

##   name edge_num likelihood like_weight_ratio distal_length pendant_length
## 1  AA      24  -61371.30      0.333344      0.000003      0.003887
## 2  BB       1  -61312.21      0.333335      0.000001      0.000003
## 3  BB       2  -61312.21      0.333322      0.000003      0.000003
## 4  BB      550  -61312.21      0.333322      0.000961      0.000003
## 5  CC       8  -61312.23      0.200011      0.000001      0.000003
## 6  CC       9  -61312.23      0.200000      0.000003      0.000003
## 7  CC      10  -61312.23      0.199992      0.000003      0.000003
```

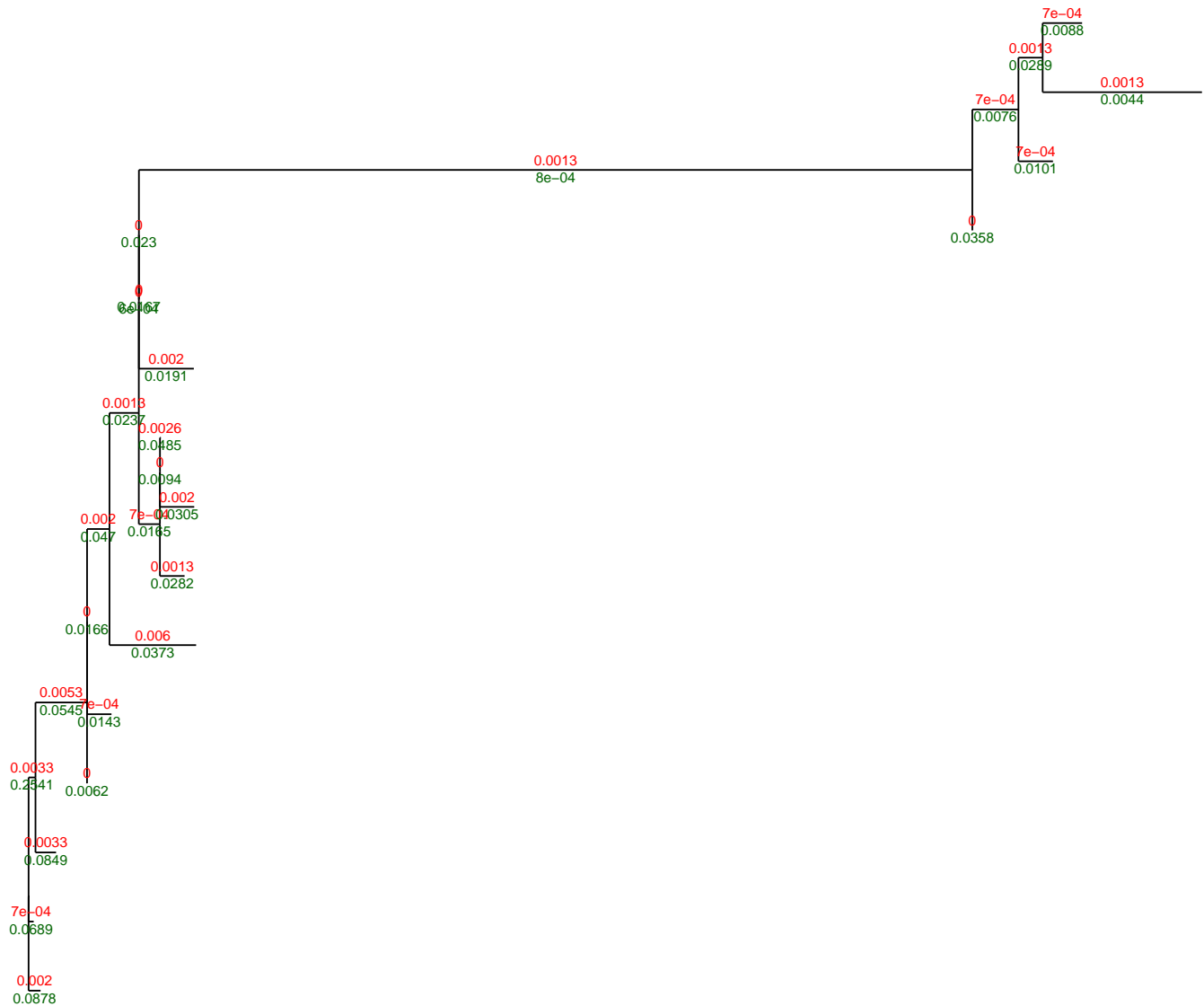
We may, for example, count the number of placement and annotate this information in the tree. We do not provide a `plot` method for `jplace` object, since we use this file format as a standard annotation format in [ggtree](#) package and have no assumption of information it may stored. Please refer to [jplace-file-format](#) session.

### 3.5 annotating tree using *ggplot2* layers

We implemented several `plot` methods for easily view annotation data. Users are not restricted to `plot` methods provided. They can use `geom_text` to add annotation layer. All annotation data are visible to `ggplot2`.

In the following example, we use the `codeml` object to visualize the  $\omega$  ( $dN/dS$ ) tree, and annotate the tree with  $dN$  and  $dS$ .

```
ggtree(ml, branch.length = "dN.dS") + geom_text(aes(x = branch, label = dN),
  size = 3, vjust = -0.5, color = "red") + geom_text(aes(x = branch, label = dS),
  size = 3, vjust = 1.2, color = "darkgreen")
```



### 3.6 user specific annotation

We provides several functions to parse and store information from common software output, and corresponding *plot* methods for visualizing annotation in the tree.

Here, we would like to demonstrate how to inject user specific annotation data in a tree.

Suppose we have the following data that associated with the tree and would like to attach the data in the tree.

```
nwk <- system.file("extdata", "sample.nwk", package="ggtree")
tree <- read.tree(nwk)
p <- ggtree(tree)

dd <- data.frame(taxa=LETTERS[1:13],
                 place=c(rep("GZ", 5), rep("HK", 3), rep("CZ", 4), NA),
                 value=round(abs(rnorm(13, mean=70, sd=10)), digits=1))
## you don't need to order the data
## data was reshuffled just for demonstration
```

```
dd <- dd[sample(1:13, 13), ]
row.names(dd) <- NULL

print(dd)
```

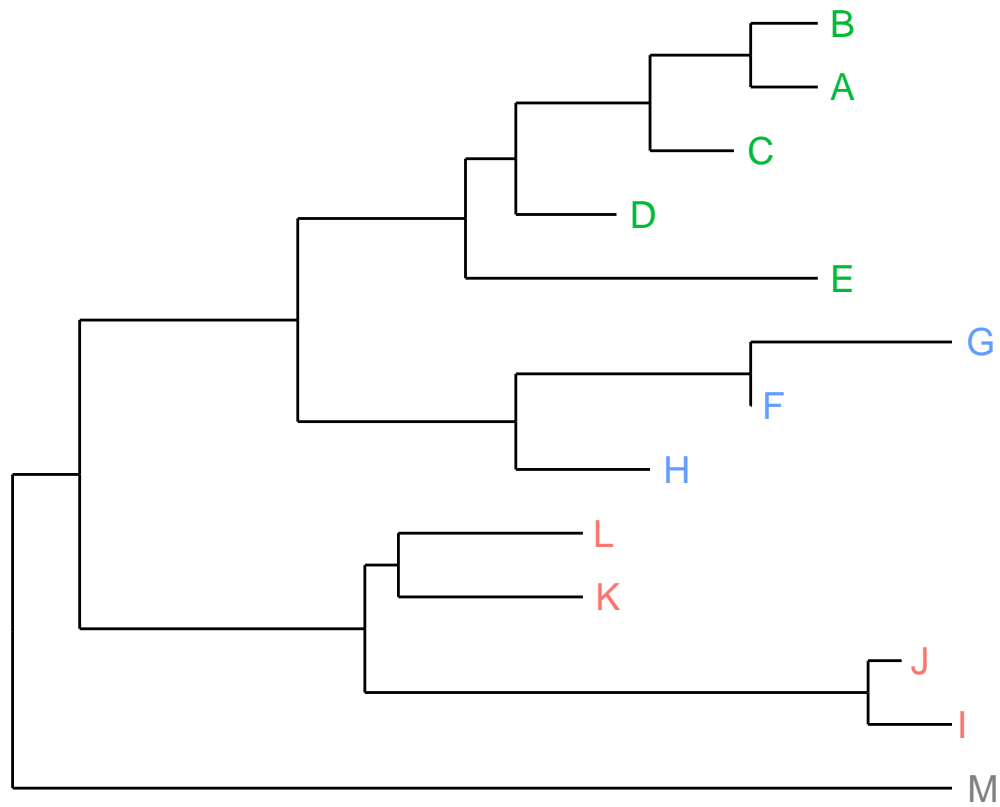
taxa	place	value
M	NA	71.5
L	CZ	69.5
K	CZ	72.1
G	HK	69.4
J	CZ	72.0
F	HK	51.7
E	GZ	82.6
D	GZ	70.7
H	HK	58.9
B	GZ	67.1
I	CZ	73.5
C	GZ	69.6
A	GZ	62.8

We can imagine that the *place* column is the place we isolated the species and *value* column stored numerical values for example bootstrap values.

We have shown using the operator, `%<%`, to update a tree view with a new tree. Here, we will introduce another operator, `%<+%`, that attach annotation data to a tree view. The only requirement of the input data is that its first column should be matched with the node labels of the tree.

After attaching the annotation data to the tree by `%<+%`, all the columns in the data are visible to *ggplot2*. As an example, here we attach the above annotation data to the tree view, *p*, and add a layer that showing the tip labels and colored by the isolation site stored in *place* column.

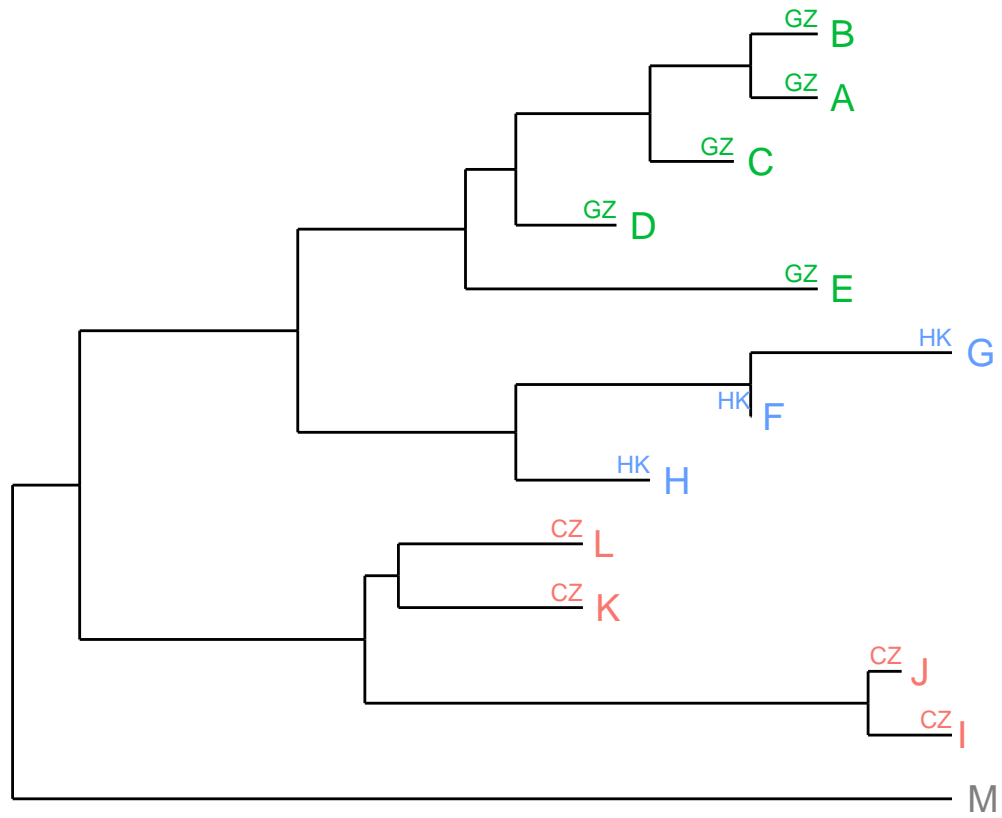
```
p <- p %<+% dd + geom_text(aes(color=place, label=label), hjust=-0.5)
print(p)
```



We can add another layer to display the isolation sites.

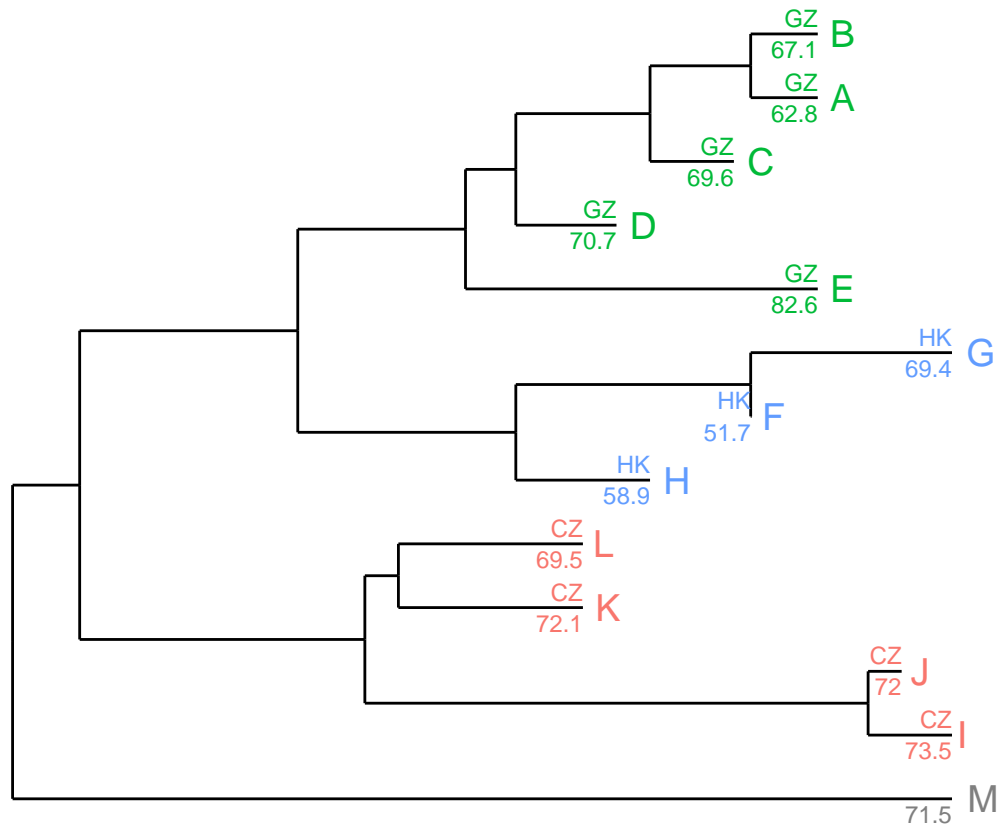
```
p <- p + geom_text(aes(color=place, label=place), hjust=1, vjust=-0.4, size=3)
print(p)
```



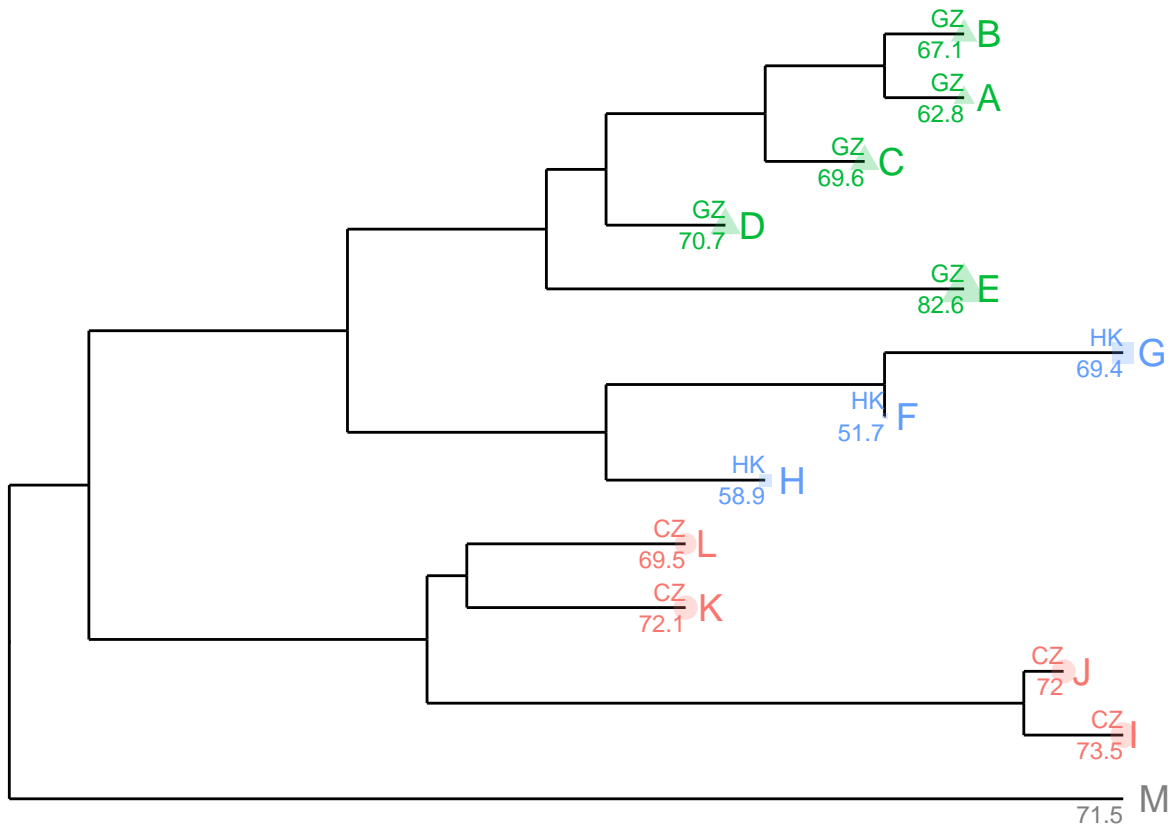


And another layer showing numerical values:

```
p <- p + geom_text(aes(color=place, label=value), hjust=1, vjust=1.4, size=3)
print(p)
```



```
p <- p + geom_point(aes(size=value, shape=place, color=place), alpha=0.25, subset=.(isTip))
print(p)
```



### 3.7 jplace file format

The *jplace* file format was defined by Masten<sup>10</sup> for phylogenetic placements. We employ this file format to store user specific annotation. Suppose we have a tree, and the associated data that indicate the amino acid substitutions from parent node to child node as shown below:

```
tree <- system.file("extdata", "pa.nwk", package="ggtree")
data <- read.csv(system.file("extdata", "pa_subs.csv", package="ggtree"), stringsAsFactor=FALSE)
print(tree)
```

```
## [1] "/home/ygc/R/x86_64-unknown-linux-gnu-library/3.1/ggtree/extdata/pa.nwk"
```

```
head(data)
```

```
##   label      subs      gc
## 1    A      R262K/K603R 0.444
## 2    B      R262K 0.442
## 3    C      N272S/I354T 0.439
## 4    D K104R/F105L/E382D/F480S/I596T 0.449
## 5    E      T208S/V379I/K609R 0.443
## 6    F      S277P/L549I 0.444
```

The *data* contains amino acid substitution and GC content. We can annotate the tree as demonstrated in user specific annotation session.

*ggtree* provides a function, *write.jplace*, to combine a tree and an associated data and store them to a single *jplace* file.

```
outfile <- tempfile()
write.jplace(tree, data, outfile)
```

Then user can use *read.jplace* function to parse the *jplace* file and store the information to a *jplace* object.

```
jp <- read.jplace(outfile)
print(jp)
```

```
## 'jplace' S4 object that stored information of
##   '/tmp/RtmpUK78my/file7b641bef19b0'.
##
## ...@ tree:
## Phylogenetic tree with 15 tips and 13 internal nodes.
##
## Tip labels:
##   K, N, D, L, J, G, ...
##
## Unrooted; includes branch lengths.
##
## with the following features availables:
##   'label',   'subs', 'gc'.
```

Now we know the *jp* object stored the tree and the associated amino acid substitution and GC content information, we can view the tree and display the associated annotation data on it directly by *ggtree*.

```
ggtree(jp, showDistance=TRUE) +
  geom_text(aes(x=branch, label=subs), color="purple", vjust=-1, size=3) +
  geom_text(aes(x=branch, label=gc), color="steelblue", vjust=1.5, size=3) +
  geom_text(aes(label=label), hjust=-.5)
```



### 3.8 visualize tree and associated matrix

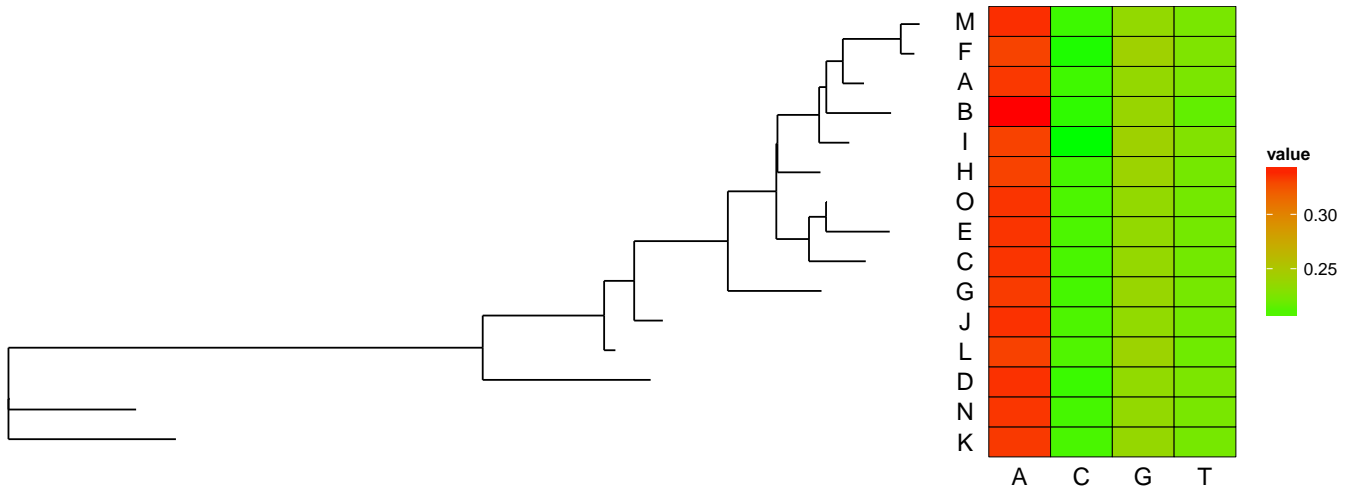
```
seqs <- ml@rst@tip_seq
library(Biostrings)
x <- DNAStringSet(seqs)

dd <- alphabetFrequency(x)[, 1:4]
dd <- dd/rowSums(dd)
row.names(dd) <- names(seqs)
```

```
head(dd)
```

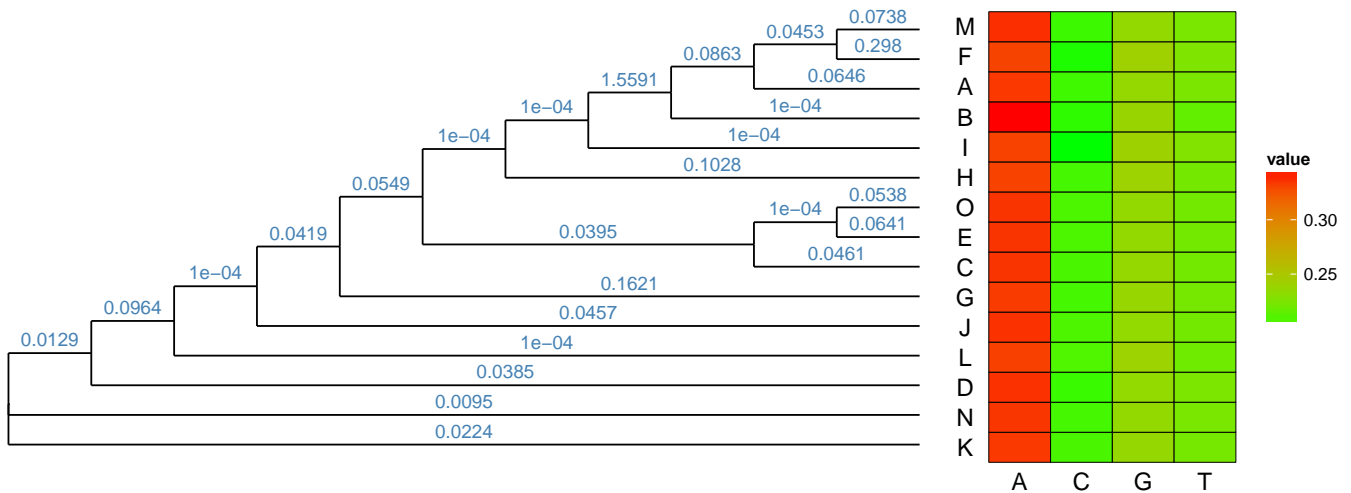
	A	C	G	T
A	0.3351955	0.2081006	0.2355680	0.2211359
B	0.3361266	0.2071695	0.2346369	0.2220670
C	0.3375233	0.2053073	0.2337058	0.2234637
D	0.3328678	0.2094972	0.2392924	0.2183426
E	0.3375233	0.2090317	0.2337058	0.2197393
F	0.3347300	0.2071695	0.2369646	0.2211359

```
p <- ggtree(ml)
gplot(p, dd, low="green", high="red", widths=c(.7, .3))
```



Of course, we can use an annotated tree.

```
p2 <- ggtree(ml, branch.length="none") +
  geom_text(aes(x=branch, label=dN.dS), vjust=-.5, color="steelblue", size=4)
gplot(p2, dd, low="green", high="red", widths=c(.7, .3))
```



## 4 Session info

---

Here is the output of `sessionInfo()` on the system on which this document was compiled:

```
## R version 3.1.2 (2014-10-31)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8      LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8  LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8     LC_NAME=C
##  [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] grid      stats4    parallel  stats      graphics  grDevices  utils
##  [8] datasets  methods  base
##
## other attached packages:
##  [1] gridExtra_0.9.1    Biostrings_2.34.1  XVector_0.6.0
##  [4] IRanges_2.0.1      S4Vectors_0.4.0    BiocGenerics_0.12.1
##  [7] ggtree_0.99.5      ggplot2_1.0.0      ape_3.2
## [10] BiocStyle_1.4.1     fortunes_1.5-2
##
## loaded via a namespace (and not attached):
##  [1] colorspace_1.2-4 digest_0.6.8     evaluate_0.5.5  formatR_1.0
##  [5] gtable_0.1.2      htmltools_0.2.6  jsonlite_0.9.14 knitr_1.8
##  [9] labeling_0.3      lattice_0.20-29  magrittr_1.5    MASS_7.3-35
## [13] munsell_0.4.2     nlme_3.1-118    plyr_1.8.1      proto_0.3-10
## [17] Rcpp_0.11.3       reshape2_1.4.1  rmarkdown_0.4.2 scales_0.2.4
## [21] stringr_0.6.2     tools_3.1.2     yaml_2.1.13     zlibbioc_1.12.0
```

## 5 References

---

1. Wickham, H. *et al.* *ggplot2: Elegant graphics for data analysis*. (Springer, 2009).
2. McMurdie, P. J. & Holmes, S. *et al.* phyloseq: An r package for reproducible interactive analysis and graphics of microbiome census data. *PLoS ONE* **8**, e61217 (2013).
3. Felsenstein, J. *et al.* *Inferring phylogenies*. (Sinauer Associates, 2003).
4. Paradis, E., Claude, J. & Strimmer, K. *et al.* APE: Analyses of phylogenetics and evolution in r language. *Bioinformatics* **20**, 289–290 (2004).
5. Yang, Z. *et al.* PAML 4: Phylogenetic analysis by maximum likelihood. *Molecular Biology and Evolution* **24**, 1586–1591 (2007).
6. Pond, S. L. K., Frost, S. D. W. & Muse, S. V. *et al.* HyPhy: hypothesis testing using phylogenies. *Bioinformatics* **21**, 676–679 (2005).
7. Berger, S. A., Krompass, D. & Stamatakis, A. *et al.* Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood. *Systematic Biology* **60**, 291–302 (2011).
8. Matsen, F. A., Kodner, R. B. & Armbrust, E. V. *et al.* pplacer: linear time maximum-likelihood and bayesian phylogenetic placement of sequences onto a fixed reference tree. *BMC Bioinformatics* **11**, 538 (2010).

9. Bouckaert, R. *et al.* BEAST 2: A software platform for bayesian evolutionary analysis. *PLoS Comput Biol* **10**, e1003537 (2014).
10. Matsen, F. A., Hoffman, N. G., Gallagher, A. & Stamatakis, A. *et al.* A format for phylogenetic placements. *PLoS ONE* **7**, e31009 (2012).