Ancestral State Reconstruction

based on (this tutorial)[http://www.phytools.org/eqg2015/asr.html].

Get and clean files

```
# read in tree and csv
fire.tree<-read.tree("fire.treefile")
fire.data<-read.csv("fire.csv", row.names = 1)
basics.tree<-read.tree("basics.treefile")
basics.data<-read.csv("basics.csv", row.names = 1)

# bookkeeping
## (checked for duplicate tips in python before creating .nex and .csv files!)

## make sure rows of character data are in same order as tip labels
fire.data = fire.data[order(match(row.names(fire.data), fire.tree$tip.label)), ]
basics.data = basics.data[order(match(row.names(basics.data), basics.tree$tip.label)), ]

## double check that labels are the same, in case some were missing or duplicated
if(! all( row.names(fire.data)==fire.tree$tip.label ) ){
  stop("fire.tree tip labels and dataframe rows not in same order!")
}
if(! all( row.names(basics.data)==basics.tree$tip.label ) ){
  stop("basics.tree tip labels and dataframe rows not in same order!")
}

## resolve polytomies
fire.tree <- multi2di(fire.tree)
basics.tree <- multi2di(basics.tree)

## root fire.tree by calculating midpoint
fire.tree <- midpoint(fire.tree)
## root basics.tree at mew
basics.tree <- root(basics.tree, "mew", resolve.root = TRUE)


# find and set important metadata abt each tree
## count number of tips
ntips_fire = length(fire.tree$tip.label)
ntips_basics = length(basics.tree$tip.label)

## get root node by finding node that isn't a child
root_fire = fire.tree$edge[(!fire.tree$edge[,1] %in% fire.tree$edge[,2]),1] %>% unique()
root_basics = basics.tree$edge[(!basics.tree$edge[,1] %in% basics.tree$edge[,2]),1] %>% unique()

## get max distance from root to another node (root age)
```

```r
age_fire = max(dist.nodes(fire.tree)[,root_fire])
age_basics = max(dist.nodes(basics.tree)[,root_basics])

## rescale tree so that age of root is 1
fire.tree$edge.length = fire.tree$edge.length / age_fire
basics.tree$edge.length = basics.tree$edge.length / age_basics

## recalculate root_age to make sure things look good
age_fire = max(dist.nodes(fire.tree)[,root_fire])
age_basics = max(dist.nodes(basics.tree)[,root_basics])

## figure out nice printing things
label_offset_fire = 0.05 * age_fire
label_offset_basics = 0.05 * age_basics

tree_width_fire = 1.5 * age_fire
tree_width_basics = 1.5 * age_basics

# make sure both trees are set to rooted
attr(fire.tree, "rooted") <- TRUE
attr(basics.tree, "rooted") <- TRUE

# print tree and data if want
# attributes(fire.tree)
# is.rooted(fire.tree)
# fire.tree
# fire.data
# basics.tree
# basics.data
```

Plotting the tree

```r
# create a bunch of pngs for basics trees which are otherwise too big to view (using plot.phylo())
png("basics_fan.png", width = 4000, height = 4000)
plot.phylo(basics.tree, type="fan", cex=1, edge.width = 5,
           x.lim=c(-4, 10), y.lim=c(-4, 7))
dev.off()
```

```
## pdf
##   2
```

```r
png("basics_radial.png", width = 1000, height = 1000)
plot.phylo(basics.tree, type="radial", cex=1.5, edge.width = 5, )
dev.off()
```

```
## pdf
##   2
```

```r
png("basics_cladogram.png", width = 1000, height = 1000)
plot.phylo(basics.tree, type="cladogram", cex=1.5, edge.width = 5, )
dev.off()
```

```
## pdf
##   2
```

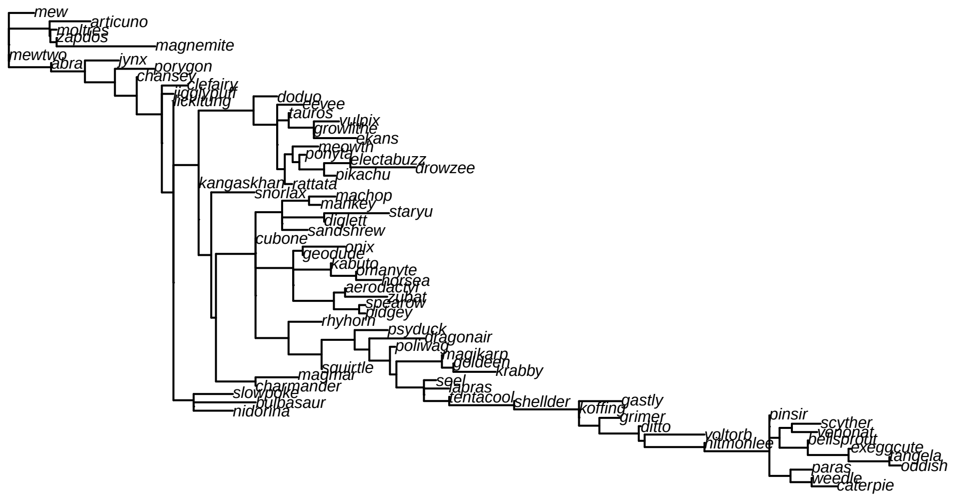
```r
png("basics_tidy.png", width = 1000, height = 1000)
plot.phylo(basics.tree, type="tidy", cex=1.5, edge.width = 5 )
dev.off()
```

```
## pdf
##   2
```

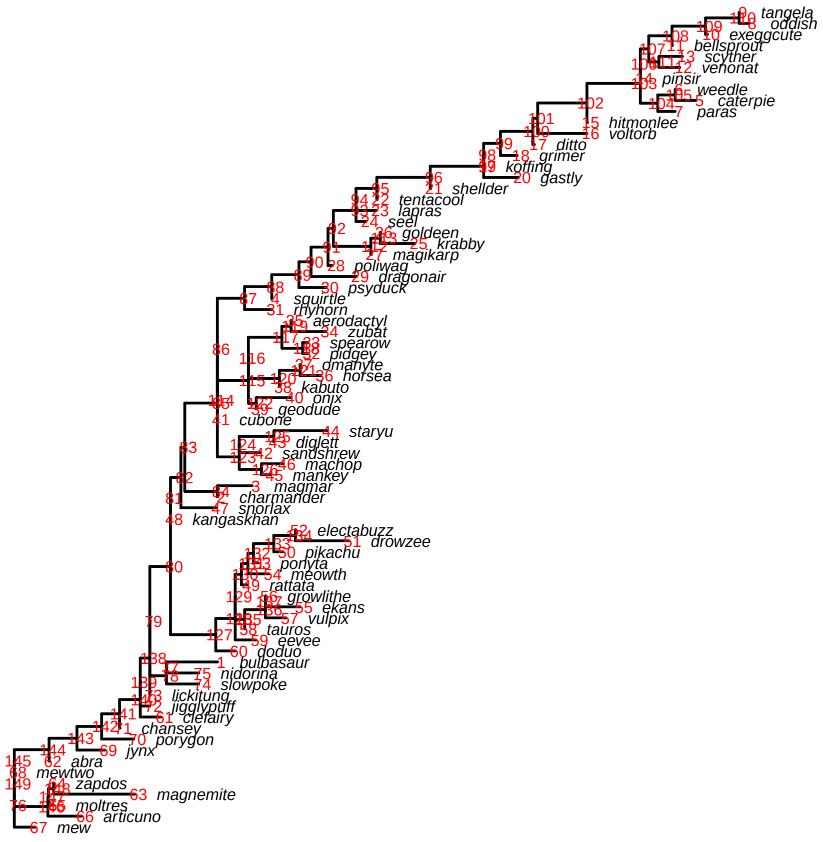
```r
png("basics_phylogram.png", width = 2000, height = 2000)
plot(basics.tree, cex=1.8)
nodelabels(basics.tree$node.label, cex=0.2)
dev.off()
```

```
## pdf
##   2
```

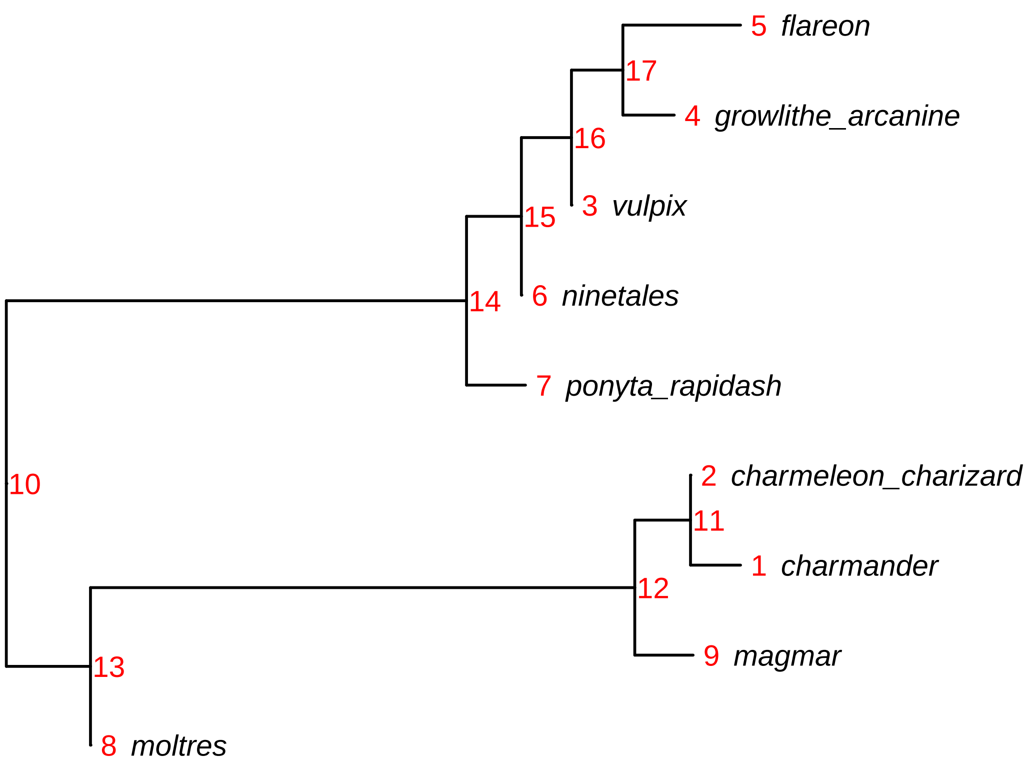
```r
# also show a small tidy basics tree in viewer
plot.phylo(basics.tree, type="tidy", cex=0.5, edge.width = 1 )
```



```r
# ggtree plot for basics.tree
basics_ggtree = ggtree(basics.tree) +
  geom_tiplab(fontface = "italic", size=2, offset = label_offset_basics/2) +
  xlim(0, tree_width_basics) +
  geom_text2(aes(label=node), col="red", size=2, nudge_x=0.005)
basics_ggtree
```



```
# ggtree plot for fire.tree
fire_ggtree = ggtree(fire.tree) + geom_tiplab(fontface = "italic",
  offset = label_offset_fire) +
  xlim(0, tree_width_fire) +
  geom_text2(aes(label=node), col="red", nudge_x=label_offset_fire/2)
fire_ggtree
```



Ancestral State Reconstruction

```
## ASR!
?ace

fire_hab_values <- fire.data$X.habitat.
basics_hab_values <- basics.data$X.habitat.
fire.tree$node.label
```

```
## [1] ""   "61" "55" "51" "51" "11" "6"  "17"
```

```
basics.tree$node.label
```

```
##  [1] "Root" ""     "32"   "24"   "27"   "20"   "35"   "42"   "50"   "35"
## [11] "8"    "32"   "34"   "48"   "48"   "44"   "43"   "45"   "30"   "49"
## [21] "53"   "50"   "52"   "52"   "54"   "51"   "54"   "62"   "49"   "62"
## [31] "19"   "23"   "86"   "55"   "63"   "67"   "85"   "78"   "35"   "48"
## [41] "2"    "92"   "65"   "97"   "60"   "52"   "61"   "45"   "41"   "65"
## [51] "57"   "51"   "42"   "23"   "43"   "43"   "52"   "52"   "61"   "42"
## [61] "47"   "59"   "24"   "29"   "30"   "51"   "48"   "47"   "52"   "51"
## [71] "53"   "39"   "52"   "35"
```

```
if (is.null(fire.tree$node.label) || any(fire.tree$node.label == "")) {
  fire.tree$node.label <- paste0("Node", 1:fire.tree$Nnode)
}
fire.tree$node.label <- NULL

if (is.null(basics.tree$node.label) || any(basics.tree$node.label == "")) {
  basics.tree$node.label <- paste0("Node", 1:basics.tree$Nnode)
}
basics.tree$node.label <- NULL
```

```r
# print("fire.tree$node.label:")
# str(fire.tree$node.label)
# print("fire.tree$tip.label:")
# str(fire.tree$tip.label)
# print("fire.tree$edge:")
# str(fire.tree$edge)
# print("fire_hab_values:")
# str(fire_hab_values)

fire_habitat_ancestral =
  ace (
    fire_hab_values,
    fire.tree,
    type="discrete",
    method="ML",
    model="ER"
  )

basics_habitat_ancestral =
  ace (
    basics_hab_values,
    basics.tree,
    type="discrete",
    method="ML",
    model="ER"
  )

str(fire_habitat_ancestral)
```

```
## List of 6
##  $ loglik     : num -6.57
##  $ rates      : num 0.809
##  $ se         : num 0.397
##  $ index.matrix: num [1:4, 1:4] NA 1 1 1 1 NA 1 1 1 1 ...
##  $ lik.anc    : num [1:8, 1:4] 1.18e-01 6.85e-09 4.38e-03 1.38e-06 9.85e-01 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:8] "10" "11" "12" "13" ...
##   .. ..$ : chr [1:4] "2" "3" "4" "7"
##  $ call       : language ace(x = fire_hab_values, phy = fire.tree, type = "discrete", method = "ML"
##  - attr(*, "class")= chr "ace"
```

```r
names(fire_habitat_ancestral)
```

```
## [1] "loglik"       "rates"        "se"           "index.matrix" "lik.anc"
## [6] "call"
```

```r
str(basics_habitat_ancestral)
```

```
## List of 6
##  $ loglik     : num -103
##  $ rates      : num 1.17
##  $ se         : num 0.174
##  $ index.matrix: num [1:9, 1:9] NA 1 1 1 1 1 1 1 1 1 ...
##  $ lik.anc    : num [1:74, 1:9] 9.72e-10 1.92e-04 1.92e-04 3.15e-10 1.51e-09 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:74] "76" "77" "78" "79" ...
##   .. ..$ : chr [1:9] "0" "1" "2" "3" ...
##  $ call        : language ace(x = basics_hab_values, phy = basics.tree, type = "discrete", method =
##  - attr(*, "class")= chr "ace"
```

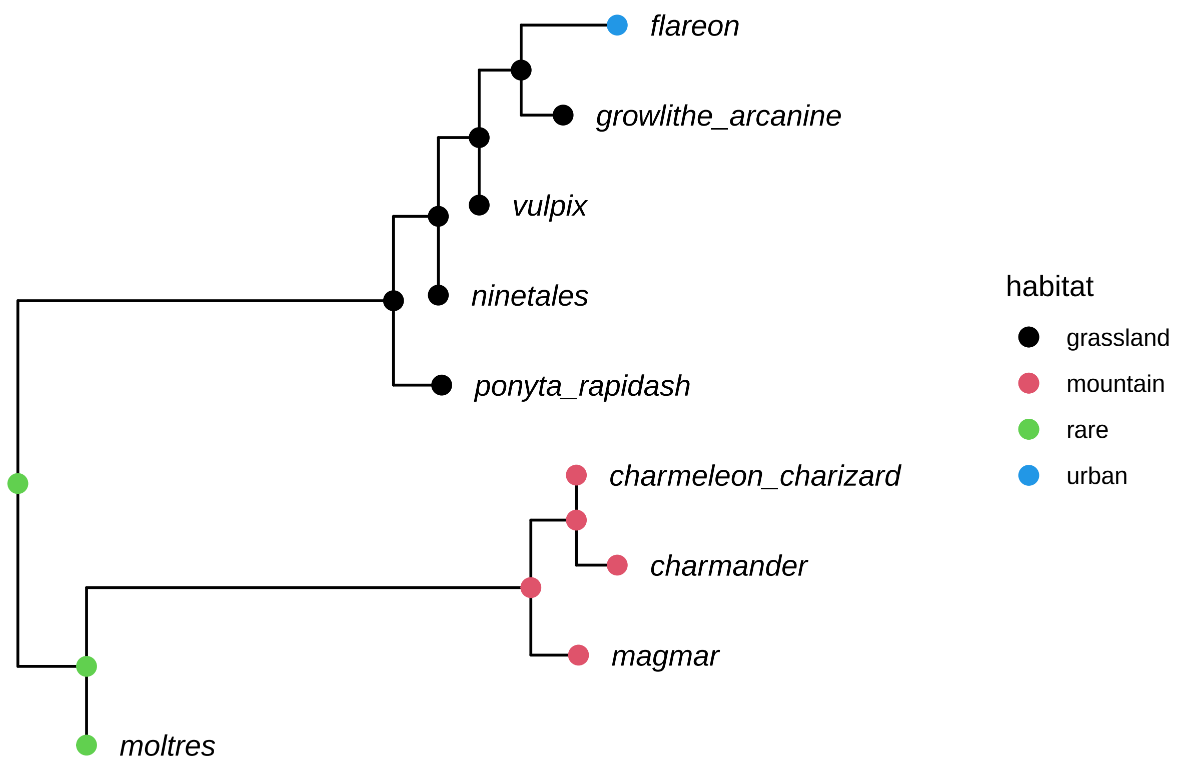
```
names(basics_habitat_ancestral)
```

```
## [1] "loglik"       "rates"        "se"           "index.matrix" "lik.anc"
## [6] "call"
```

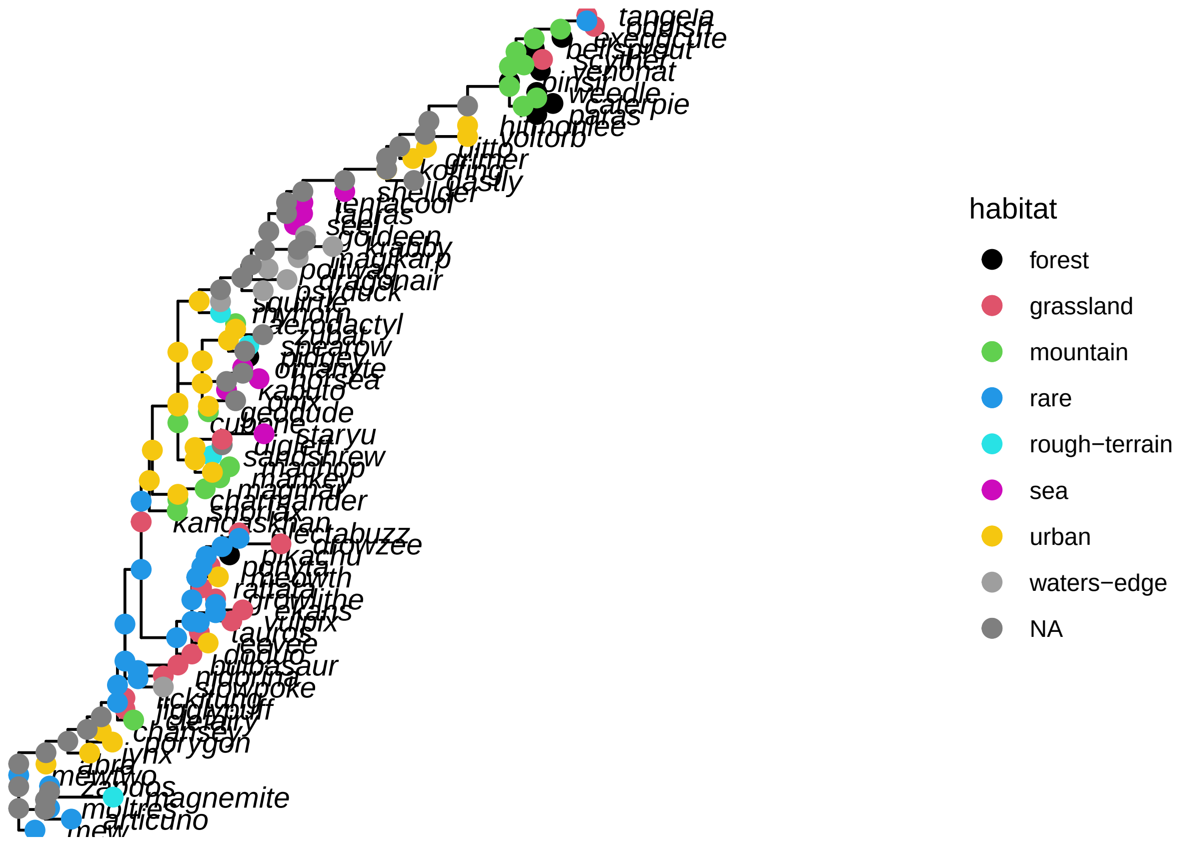
```
## why isn't it showing me these... :sob:
# '$ace' contains the reconstructed ancestral character states
# '$sigma2' is the estimate of the Brownian motion parameter
# '$CI95' are the confidence intervals on the ASR
```

Map ancestral states onto phylogeny

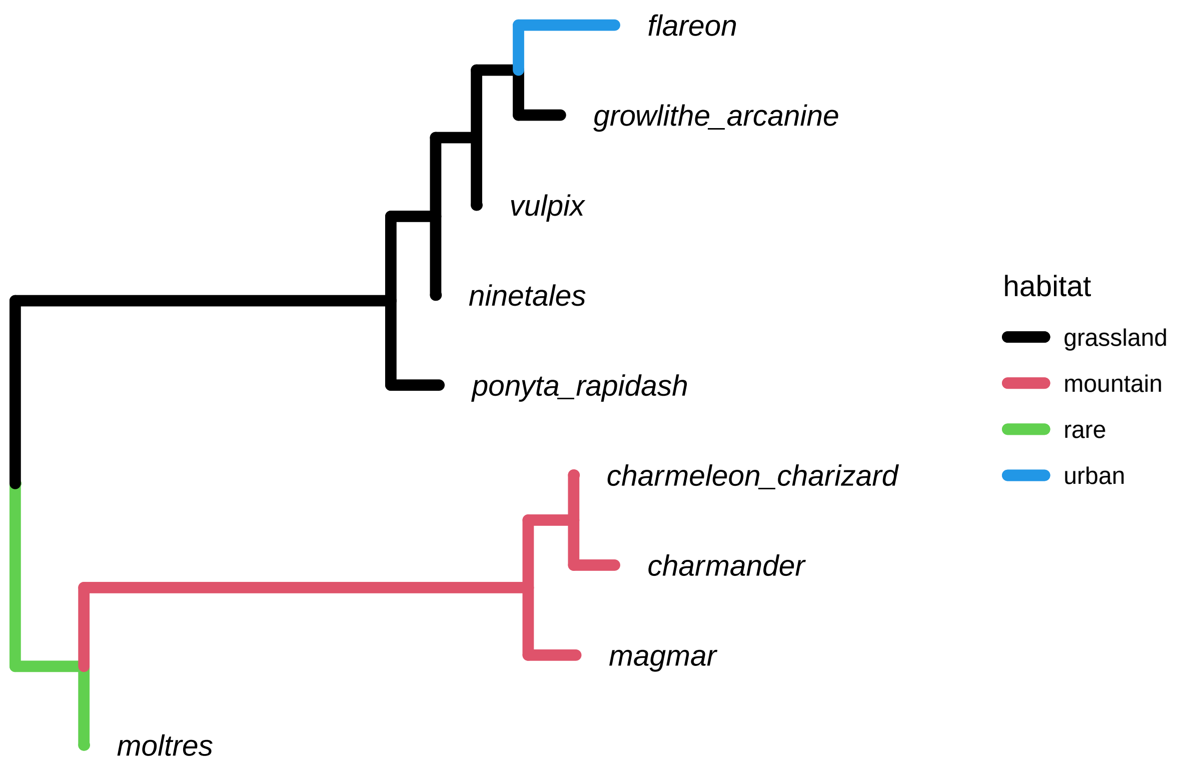
```
# get habitat data ready to map onto the phylogeny
ancestral_states_index <- apply(fire_habitat_ancestral$lik.anc, 1, which.max)
state_values <- c(2, 3, 4, 7)
ancestral_states <- state_values[ancestral_states_index]
fire_hab_values <- factor(fire_hab_values, levels = state_values)
ancestral_states <- factor(ancestral_states, levels = state_values)

basics_ancestral_states_index <- apply(basics_habitat_ancestral$lik.anc, 1, which.max)
basics_state_values <- c(1, 2, 3, 4, 5, 6, 7, 8)
basics_ancestral_states <- state_values[basics_ancestral_states_index]
basics_hab_values <- factor(basics_hab_values, levels = basics_state_values)
basics_ancestral_states <- factor(basics_ancestral_states, levels = basics_state_values)

# create and call plot with circles at each node
node_vals = c(fire_hab_values, ancestral_states)
basics_node_vals = c(basics_hab_values, basics_ancestral_states)

this_palette <- palette()
colors <- setNames(c(this_palette[1:length(state_values)]), state_values)
basics_colors <- setNames(c(this_palette[1:length(basics_state_values)]), basics_state_values)

legend_labels <- c("grassland", "mountain", "rare", "urban")
basics_legend_labels <- c("forest", "grassland", "mountain", "rare", "rough-terrain", "sea", "urban", "

fire_rect = ggtree(fire.tree) + geom_tiplab(fontface = "italic",
  offset = label_offset_fire) +
  xlim(0, tree_width_fire) +
  geom_tippoint(aes(color=node_vals), size=3, alpha=1) +
  geom_nodepoint(aes(color=node_vals), size=3, alpha=1) +
  scale_color_manual(
    values = colors,
    labels = legend_labels,
    name = "habitat"
  )
fire_rect
```



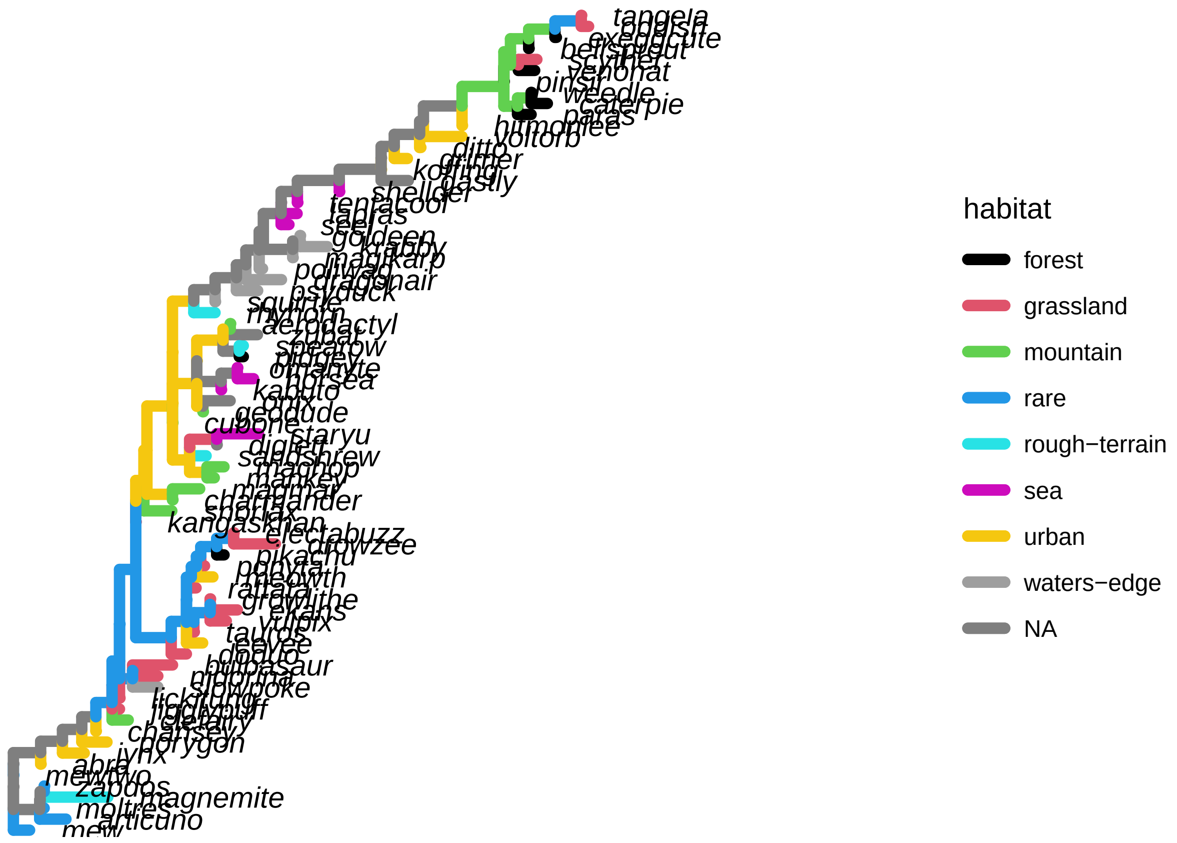
```r
basics_rect = ggtree(basics.tree) + geom_tiplab(fontface = "italic",
  offset = label_offset_basics) +
  xlim(0, tree_width_basics) +
  geom_tippoint(aes(color=basics_node_vals), size=3, alpha=1) +
  geom_nodepoint(aes(color=basics_node_vals), size=3, alpha=1) +
  scale_color_manual(
    values = basics_colors,
    labels = basics_legend_labels,
    name = "habitat"
  )
basics_rect
```



```
?ggtree
# creating a plot where branches are colored by state
fire.tree$state <- as.factor(node_vals)
fire_branch_colors = ggtree(fire.tree) +
  geom_tree(aes(color = fire.tree$state), size = 2) +
  geom_tiplab(fontface = "italic", offset = 0.05) +
  xlim(0, tree_width_fire) +
  scale_color_manual(
    values = colors,
    labels = legend_labels,
    name = "habitat"
  )
fire_branch_colors
```



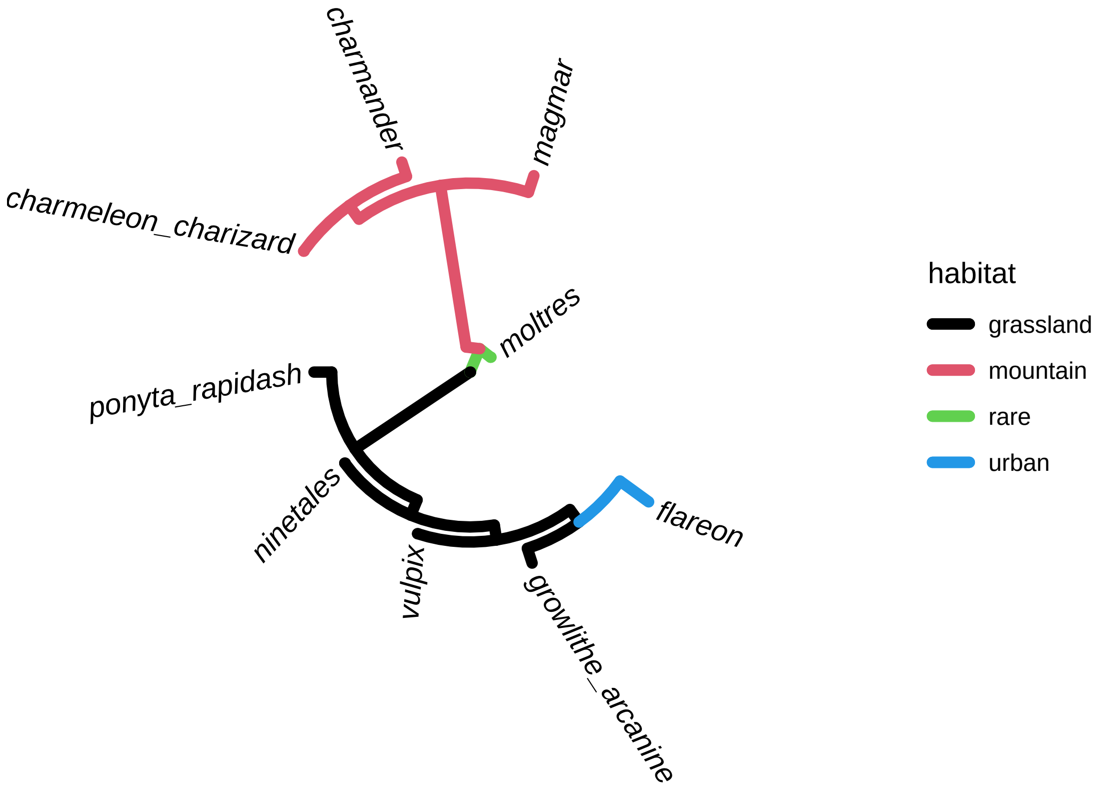
```
basics.tree$state <- as.factor(basics_node_vals)
basics_branch_colors = ggtree(basics.tree) +
  geom_tree(aes(color = basics.tree$state), size = 2) +
  geom_tiplab(fontface = "italic", offset = 0.05) +
  xlim(0, tree_width_basics) +
  scale_color_manual(
    values = basics_colors,
    labels = basics_legend_labels,
    name = "habitat"
  )
basics_branch_colors
```



```r
# ^ but fan
fire_fan = ggtree(fire.tree, layout = "fan") +
  geom_tree(aes(color = fire.tree$state), size = 2) +
  geom_tiplab(fontface = "italic", offset = 0.05) +
  xlim(0, tree_width_fire) +
  scale_color_manual(
    values = colors,
    labels = legend_labels,
    name = "habitat"
  )
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

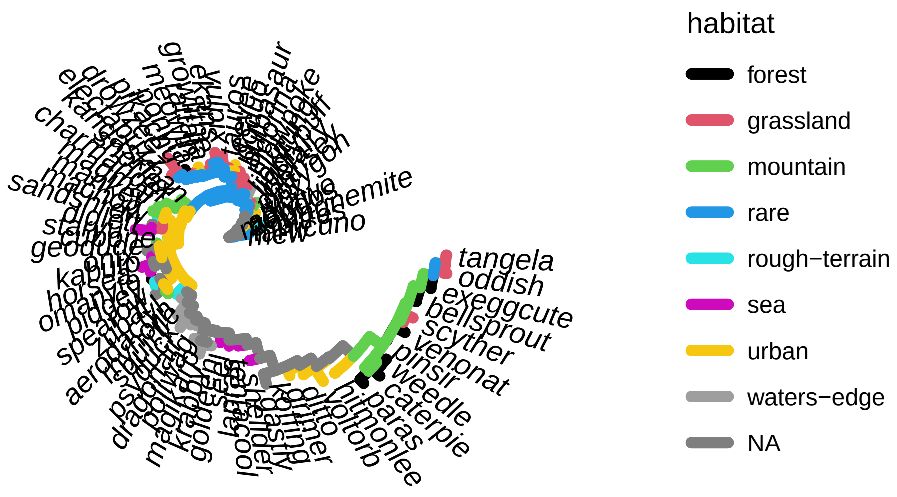
```r
fire_fan
```



```r
basics_fan = ggtree(basics.tree, layout = "fan") +
  geom_tree(aes(color = basics.tree$state), size = 2) +
  geom_tiplab(fontface = "italic", offset = 0.05) +
  xlim(0, tree_width_basics) +
  scale_color_manual(
    values = basics_colors,
    labels = basics_legend_labels,
    name = "habitat"
  )
```

```
## Scale for y is already present.
## Adding another scale for y, which will replace the existing scale.
```

```r
basics_fan
```



```r
## posterior probabilities as pie charts on internal nodes
?make.simmap

# create hab_states factor for tip states
hab_states <- state_values[as.integer(fire_hab_values)]
hab_states <- factor(hab_states, levels = state_values)
names(hab_states) <- fire.tree$tip.label

basics_hab_states <- basics_state_values[as.integer(basics_hab_values)]
basics_hab_states <- factor(basics_hab_states, levels = basics_state_values)
names(basics_hab_states) <- basics.tree$tip.label

print("smth")
```

```
## [1] "smth"
```

```r
str(basics_hab_states)
```

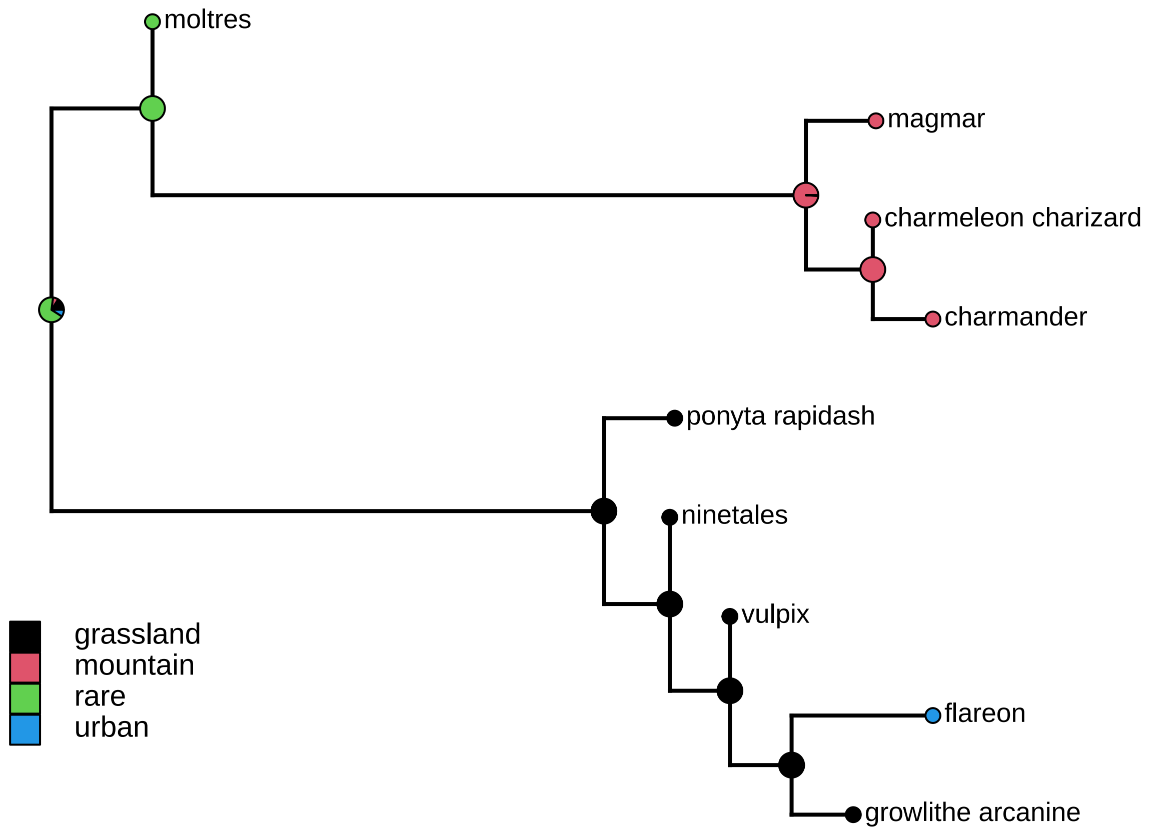
```
##  Factor w/ 8 levels "1","2","3","4",..: 2 3 3 8 1 1 1 2 2 1 ...
##  - attr(*, "names")= chr [1:75] "bulbasaur" "charmander" "magmar" "squirtle" ...
```

```r
trees <- make.simmap(fire.tree, hab_states, model="ER", nsim=100)
```

```
## make.simmap is sampling character histories conditioned on
## the transition matrix
##
## Q =
##             2          3          4          7
## 2 -2.4265393  0.8088464  0.8088464  0.8088464
## 3  0.8088464 -2.4265393  0.8088464  0.8088464
## 4  0.8088464  0.8088464 -2.4265393  0.8088464
## 7  0.8088464  0.8088464  0.8088464 -2.4265393
## (estimated using likelihood);
## and (mean) root node prior probabilities
## pi =
##    2    3    4    7
## 0.25 0.25 0.25 0.25
```

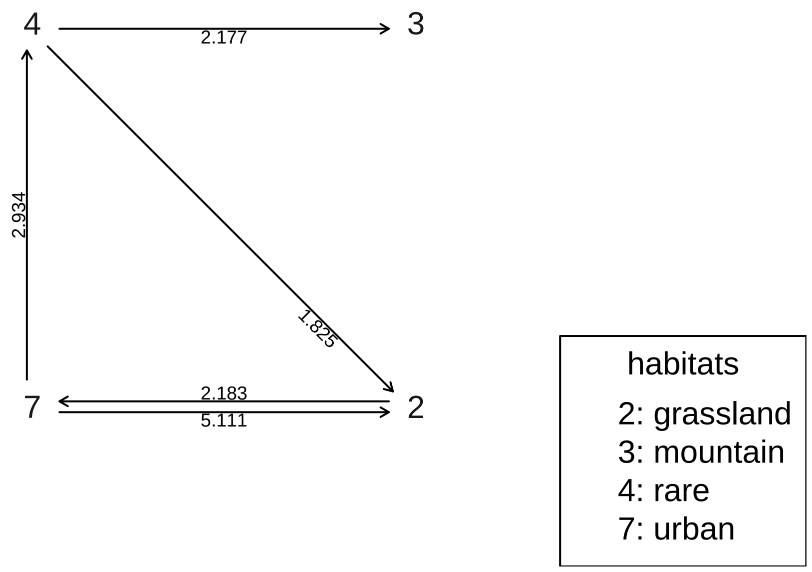
```
## Done.
```

```r
# basics_trees <- make.simmap(basics.tree, basics_hab_states, model="ER", nsim=100)

obj <- summary(trees, plot=FALSE)
plot(obj, colors=colors, fsize=0.8, cex=c(0.5, 0.3))
?add.simmap.legend
add.simmap.legend(
  colors=colors,
  x=0.9*par()$usr[1],
  y=0.3*par()$usr[4],
  prompt=FALSE,
  fsize=0.9,
  leg = legend_labels
)
```



```r
# basics_obj <- summary(basics_trees, plot=FALSE)
# plot(basics_obj, colors=basics_colors, fsize=0.8, cex=c(0.5, 0.3))
# add.simmap.legend(
#   colors=basics_colors,
#   x=0.9*par()$usr[1],
#   y=0.3*par()$usr[4],
#   prompt=FALSE,
#   fsize=0.9,
#   leg = basics_legend_labels
# )

# ARD model
legend_labels <- c("2: grassland", "3: mountain", "4: rare", "7: urban")
basics_legend_labels <- c("1: forest", "2: grassland", "3: mountain", "4: rare",
                          "5: rough-terrain", "6: sea", "7: urban", "8: waters-edge")


fit.ARD<-fitMk(fire.tree, hab_states, model="ARD")
plot(fit.ARD, show.zeros=FALSE, main="fitted ARD model for habitat evolution")
legend("bottomright", legend = legend_labels, title = "habitats")
```

fitted ARD model for habitat evolution



```
# basics_fit.ARD<-fitMk(basics.tree, basics_hab_states, model="ARD")
# plot(basics_fit.ARD, show.zeros=FALSE, main="fitted ARD model for basics habitat evolution")
# legend("bottomright", legend = basics_legend_labels, title = "basics habitats")
```

Things I Would Love To Do But I Tried For a While And Got Nowhere

```
# # cool but only when already know asr states :eyeroll:
# # ## for discrete states (ANOLE EXAMPLE)
# data(anoletree)
# cols<-setNames(palette()[1:6],mapped.states(anoletree))
# # cols <- setNames(palette()[1:length(unique(X))], sort(unique(X)))
#
# plot(anoletree, cols, type="fan", fsize=0.8, lwd=3, ftype="i")
# add.simmap.legend(colors=cols, x = 0.9*par()$usr[1],
#     y = 0.9*par()$usr[4], prompt=FALSE, fsize=0.9)
#
# ## figure out what mapped.states() is doing
# ?mapped.states
# str(anoletree)
# result <- mapped.states(anoletree)
# print("result:")
# print(result)
#
# # plot discrete tip node states as colors
# plotTree(fire.tree, type="fan", fsize=0.8, ftype="i")
# cols <- setNames(palette()[1:length(unique(X))], sort(unique(X)))
# tiplabels(pie=to.matrix(X, sort(unique(X))), piecol = cols, cex = 0.3)
# add.simmap.legend(colors=cols,prompt=FALSE,x=0.9*par()$usr[1],
#     y=-max(nodeHeights(tree)),fsize=0.5)
```

Interesting Results

Legendary pokémon Moltres *failed* the chi2 test when compared with all the other fire gen1 pokémon! Gap/Ambiguity Composition p-value 1 charmander 0.00% passed 61.69% 2 charmeleon_charizard 0.00% passed 69.95% 3 vulpix 0.00% passed 88.29% 4 ninetales 0.00% passed 98.01% 5 growlithe_arcanine 0.00% passed 88.29% 6 ponyta_rapidash 0.00% passed 89.76% 7 magmar 0.00% passed 24.30% 8 flareon 0.00% passed 77.79% 9 moltres 0.00% failed 3.34% **** TOTAL 0.00% 1 sequences failed composition chi2 test (p-value<5%; df=11)