RERconverge Analysis Walkthrough

August 23, 2018

Contents

This walkthrough provides instructions for implementing the RERconverge package to identify genes whose evolutionary rates shift in association with change in a trait. For information on how to download and install RERconverge, see the wiki. Source code and a quick start guide are available on github.

Overview

The following document describes the steps necessary to perform a standard RERconverge analysis to identify genomic elements with convergent **rates of evolution** in phenotypically convergent species using a **binary** trait or a **continuous** trait.

Output is a the list of genomic elements with statistics that represent the strength and directon of the relationship between the genomic element's evolutionary rate and the phenotype. These statistics can be used to make inferences about the genomic elements' importances to the phenotype. Genomic elements that evolve more *slowly* for a given phenotype may be under *increased evolutionary constraint* because their function is important for the development of the convergent phenotype, for example. On the other hand, genomic elements that evolve more *quickly* for a given phenotype may either be under *decreased evolutionary constraint* due to loss of function or relatively decreased functional importance or, conversely, undergoing *directional selection* to increase or alter functionality. The ranked gene list can be further used as input for functional enrichment methodologies to find pathways or other functional groups under convergent evolutionary pressures.

Data Input Requirements and Formatting

The analysis requires two sources of data as input:

1. Phylogenetic trees for every genomic element with branch lengths that represent element-specific evolutionary rates
   - Trees should be in Newick format with tip labels and no node labels
   - Tree topologies must all be subsets of the same master tree topology
2. Species-labelled phenotype values
   - Species labels must match tree tip labels

- For continuous traits: a named numeric vector
- For binary traits: foreground species names or a tree with 0 and 1 branch lengths

When choosing a dataset to work with, consider the availability and accuracy of both genomic and phenotypic data, and be sure to select a valid convergent phenotype that is observed at high and low levels in multiple independent clades in your phylogeny.

For a more detailed description of data formatting requirements and examples, please see the relevant sections of the walkthrough.

Detailed Walkthrough

Installing and loading RERconverge

Note: Prior to running the vignette, be sure to follow all the steps for installation on the wiki, up to the "Install from Github" step.

In R, load the RERConverge library.

```
if (!require("RERconverge", character.only=T, quietly=T)) {
    require(devtools)
    install_github("nclark-lab/RERconverge") #change ref once branch is merged
}
library(RERconverge)
```

This should also download all the files we will be working with to your computer, in the directory where your R library lives. If you'd like to visualize or work with any of these files separately, this is where you can find them:

```
#Old version of finding package through .libPaths()
#wpath = NA
#for (i in 1:length(.libPaths())) {
#  if (file.exists(paste(.libPaths()[i],"/RERconverge/R/RERconverge",sep=""))) {
#      wpath = i
#      break
#  }
#}
#if (is.na(wpath)) {
#    print("RERconverge not found in R paths; check installation.")
#}
#print(paste(.libPaths()[wpath],"/RERconverge",sep="")) #This is the path to the files
#rerpath = paste(.libPaths()[wpath],"/RERconverge",sep="")
```

```
rerpath = find.package('RERconverge') #If this errors, there is an issue with installation
print(rerpath)
```

```
## [1] "/Users/wynnmeyer/Library/R/3.3/library/RERconverge"
```

Reading in gene trees with `readTrees`

To run RERconverge, you will first need to supply a file containing **gene trees** for all genes to be included in your analysis. This is a tab delimited file with the following information on each line:

Gene_name Newick_tree

An example file is provided in *extdata/subsetMammalGeneTrees.txt*, which you can view in any text editor.

Now in R, read in the gene trees. The `readTrees` function takes quite a while to read in trees for all genes, so we will limit ourselves to the first 200 using `max.read` (this will still take a minute or so, so be patient):

```
toytreefile = "subsetMammalGeneTrees.txt"
toyTrees=readTrees(paste(rerpath,"/extdata/",toytreefile,sep=""), max.read = 200)
```

```
## max is  62
```

```
## estimating master tree branch lengths from 32 genes
```

First, the code tells us that there are 1000 items, or gene trees, in the file. Since we have set `max.read = 200`, it will only read the first 200 of these. Then it says that the maximum number of tips in the gene trees is 62 and, later, it reports that it will use the 32 genes in this set that have data for all 62 species to estimate a **master tree**. The master tree will be used for subsequent analyses.

Estimating relative evolutionary rates (RER) with `getAllResiduals`

The next step is to estimate **relative evolutionary rates**, or RERs, for all branches in the tree for each gene. Intuitively, a gene's RER for a given branch represents how quickly or slowly the gene is evolving on that branch relative to its overall rate of evolution throughout the tree.

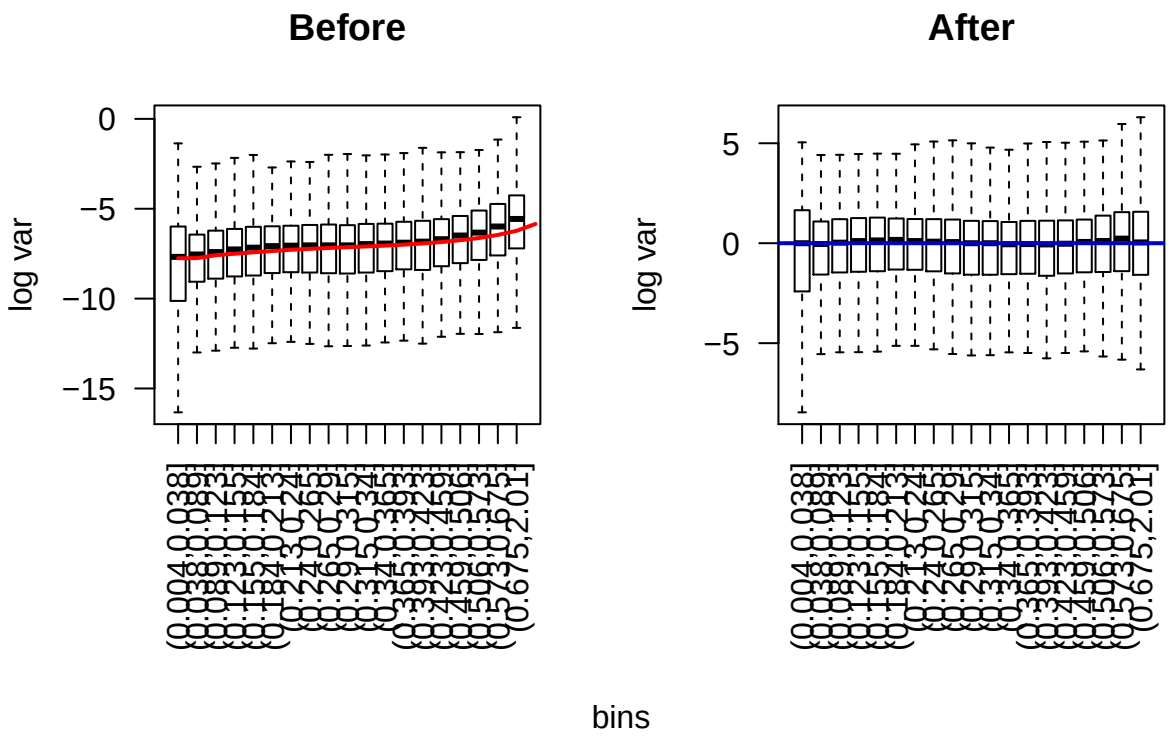
Briefly, RERs are calculated by normalizing branch lengths across all trees by the master branch lengths. Branch lengths are then corrected for the heteroscedastic relationship between average branch length and variance using weighted regression. For a more detailed description of how RERs are computed, see (Chikina, Robinson, and Clark 2016) and (Partha et al. 2017).

We will use the `getAllResiduals` function to calculate RERs. This uses the following input variables (all the options set here are also the defaults):

- `useSpecies`: a vector that can be used to specify a subset of species to use in the analysis. Here we will use the species in our AdultWeightLog vector that will be used for continuous trait analysis. Note that these are also the same species used for binary trait analysis.
- `transform`: the method used to transform the raw data. By transforming the raw data, we reduce the heteroscedasticity (relationship between mean and variance) and the influence of outliers. Here we will use a square-root transform.
- `weighted`: whether to use a weighted regression to estimate RER. Weighting allows further correction for the relationship between mean and variance, which can be directly estimated from the data.
- `scale`: whether to scale the individual branches of the gene trees to account for variance across trees. This scales the variance, though not the mean, of each branch length, using the R function `scale`.

Here is the basic method, with the recommended settings:

```
data("logAdultWeightcm")
mamRERw=RERconverge::getAllResiduals(toyTrees,useSpecies=names(logAdultWeightcm),
                            transform = "sqrt", weighted = T, scale = T)
```



The first output of this function tells you that the cutoff is set to 3.2e-05. Any branches shorter than this will be excluded from the analysis. It then prints out i= 1 ... 200, showing the progress as it estimates correlations for sets of gene trees.

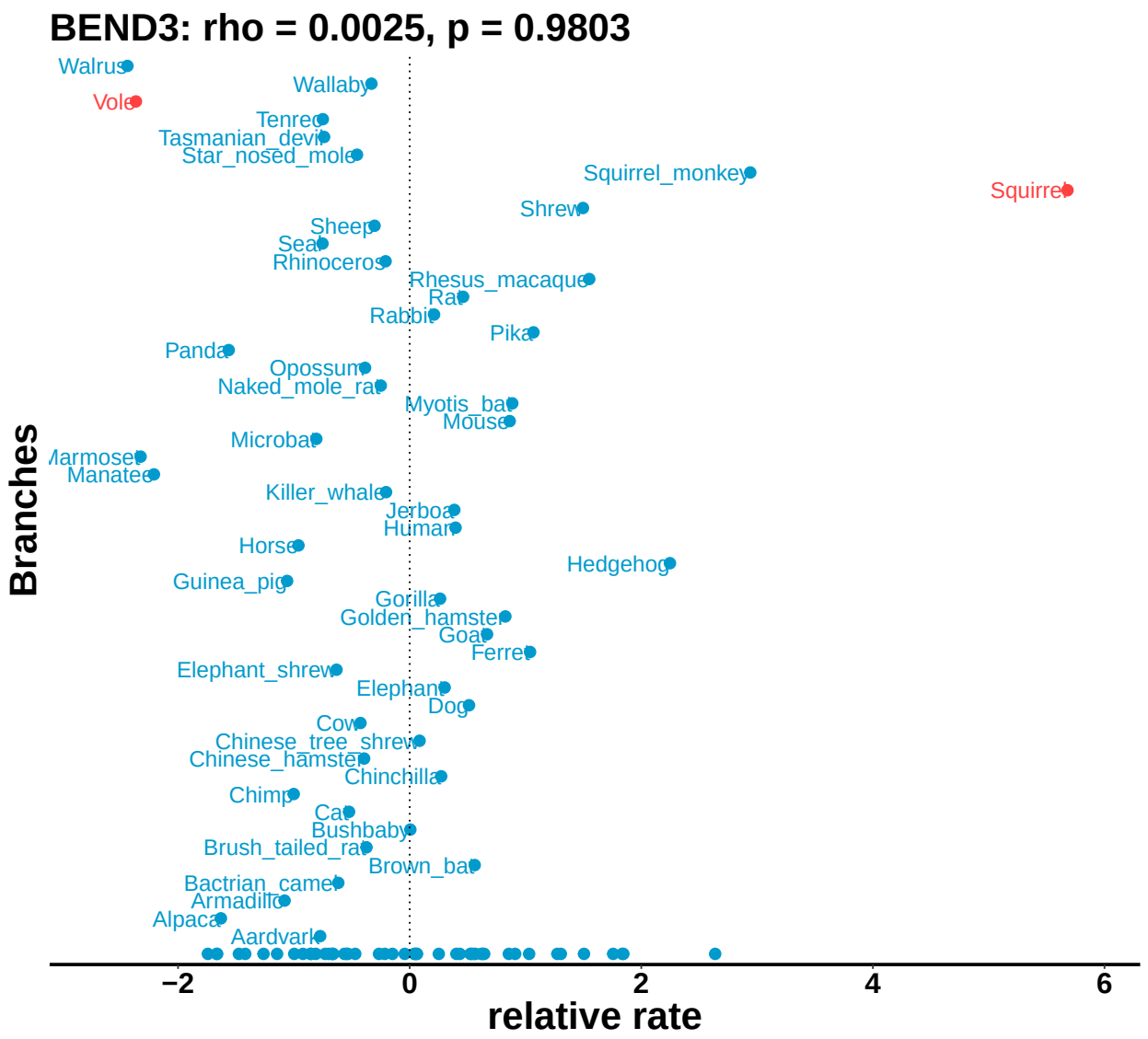
The plots generated by this function show the heteroscedasticity in the original data (on the left) and the data after transformation and weighted regression (on the right). The x-axis displays bins of branch lengths on the tree, and the y-axis is the (log-scaled) variance in these branch lengths across trees. As you can see by comparing the right plot to the left plot, transforming and performing a weighted regression reduces the relationship between the mean branch length (x-axis) and the variance in branch length (y-axis).

Now that we have RERs, we can visualize these for any given gene using the `plotRers` function. Here is an example.

```
#make average and gene tree plots
noneutherians <- c("Platypus","Wallaby","Tasmanian_devil","Opossum")
par(mfrow=c(1,2))
avgtree=plotTreeHighlightBranches(toyTrees$masterTree, outgroup=noneutherians,
                                  hlspecies=c("Vole","Squirrel"), hlcols=c("blue","red"),
                                  main="Average tree") #plot average tree
bend3tree=plotTreeHighlightBranches(toyTrees$trees$BEND3, outgroup=noneutherians,
                                    hlspecies=c("Vole","Squirrel"), hlcols=c("blue","red"),
                                    main="BEND3 tree") #plot individual gene tree
```

The left plot is a tree with branch lengths representing the average rates across all genes. The right plot is the same tree, but with branch lengths representing rates specifically for the BEND3 gene.

```
#plot RERs
par(mfrow=c(1,1))
phenvExample <- foreground2Paths(c("Vole","Squirrel"),toyTrees)
plotRers(mamRERw,"BEND3",phenv=phenvExample) #plot RERs
```



This plot represents the estimated RERs for terminal branches. The foreground branches (set here using `foreground2Paths`) are highlighted in red. Notice how the RER for vole is negative; this is because the branch leading to vole in the BEND3 tree is shorter than average. On the other hand, the RER for squirrel is positive because the branch leading to squirrel in the BEND3 tree is longer than average.

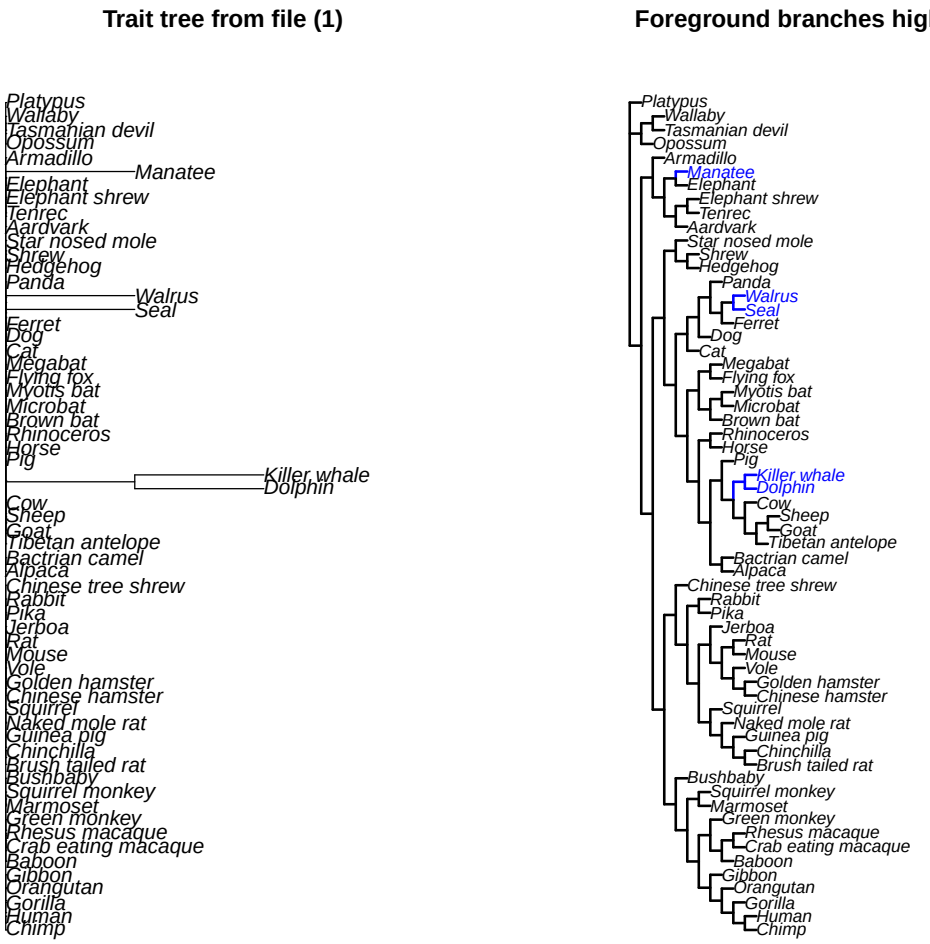
Binary Trait Analysis

Now we will associate variation in these RERs with variation in a **binary trait** across the tree. To do so, we first need to provide information about which branches of the tree have the trait of interest (**foreground branches**). There are several possible ways to do this:

1) Provide a binary trait tree file. This should be a file in Newick format with branch lengths zero for background branches and one for foreground branches. An example is provided in *extdata/MarineTreeBinCommonNames.txt*. This tree must have the same topology as the master tree or a subset of the master tree.

```
marineb=read.tree(paste(rerpath,"/extdata/MarineTreeBinCommonNames_noCGM.txt",sep=""))
marinebrooted = root(marineb,outgroup=noneutherians)
```

```
mb1 = marineb
mb1$edge.length = c(rep(1,length(mb1$edge.length)))
par(mfrow=c(1,2))
plot(marinebrooted, main="Trait tree from file (1)")
#alternative way of representing the tree
binplot1=plotTreeHighlightBranches(mb1, outgroup=noneutherians,
                                   hlspecies=which(marineb$edge.length==1), hlcols="blue",
                                   main="Foreground branches highlighted (1)")
```



The plot on the left shows the tree you provided, with branch lengths 0 for all background lineages and branch lengths 1 for foreground lineages. The plot on the right displays the tree with all branch lengths 1 and the foreground lineages highlighted in blue. This binary tree represents the following as foreground lineages: all terminal branches leading to extant marine species, plus the branch leading to the common ancestor of the killer whale and the dolphin.

2) Generate a binary tree from a vector of foreground species using `foreground2Tree`. By modifying the `clade` argument, you can choose one of several options for how to handle the branches ancestral to the foreground species, as follows:

2a) `clade = "ancestral"`: Use maximum parsimony to infer where transitions from background to foreground occurred in the tree, and set those transition lineages to foreground.

```
marineextantforeground = c("Walrus","Seal","Killer_whale","Dolphin","Manatee")
marineb2a = foreground2Tree(marineextantforeground, toyTrees, clade="ancestral")
```

Here the branch leading to the common ancestor of killer whale and dolphin, as well as the branch leading to the common ancestor of walrus and seal, are foreground, along with the terminal branch leading to the manatee.

2b) `clade = "terminal"`: Set only terminal lineages leading to foreground species as foreground.

```
marineb2b = foreground2Tree(marineextantforeground, toyTrees, clade="terminal")
```

Here the each terminal branch leading to a marine species is foreground, but no internal branches are foreground.

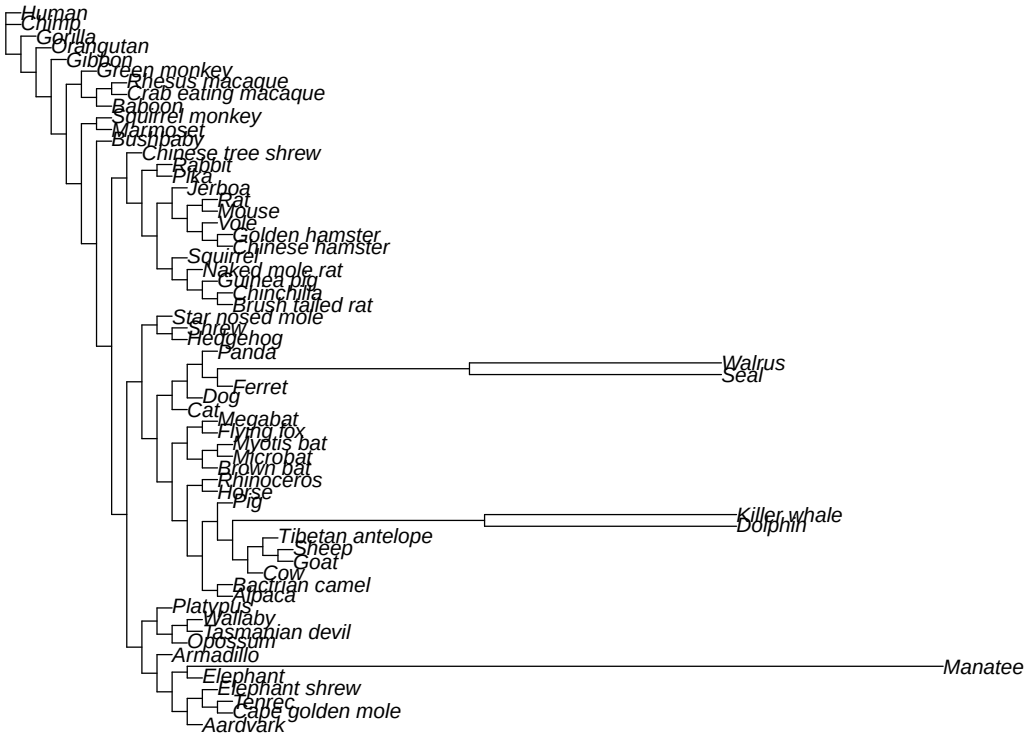
2c) `clade = "all"`: Use maximum parsimony to infer where transitions from background to foreground occurred in the tree, and set those transition lineages, along with all daughter lineages, to foreground.

```
marineb2c = foreground2Tree(marineextantforeground, toyTrees, clade="all")
```

Here the foreground branches are all those inferred to be transitional from 2a, as well as the terminal branches. If we had a case in which some branches daughter to the transition branches were not terminal, those would be included in the foreground as well.

2d) `clade = "weighted"`: Infer transition and daughter branches as in 2c, but spread a weight of 1 evenly across all branches within each independent foreground clade.

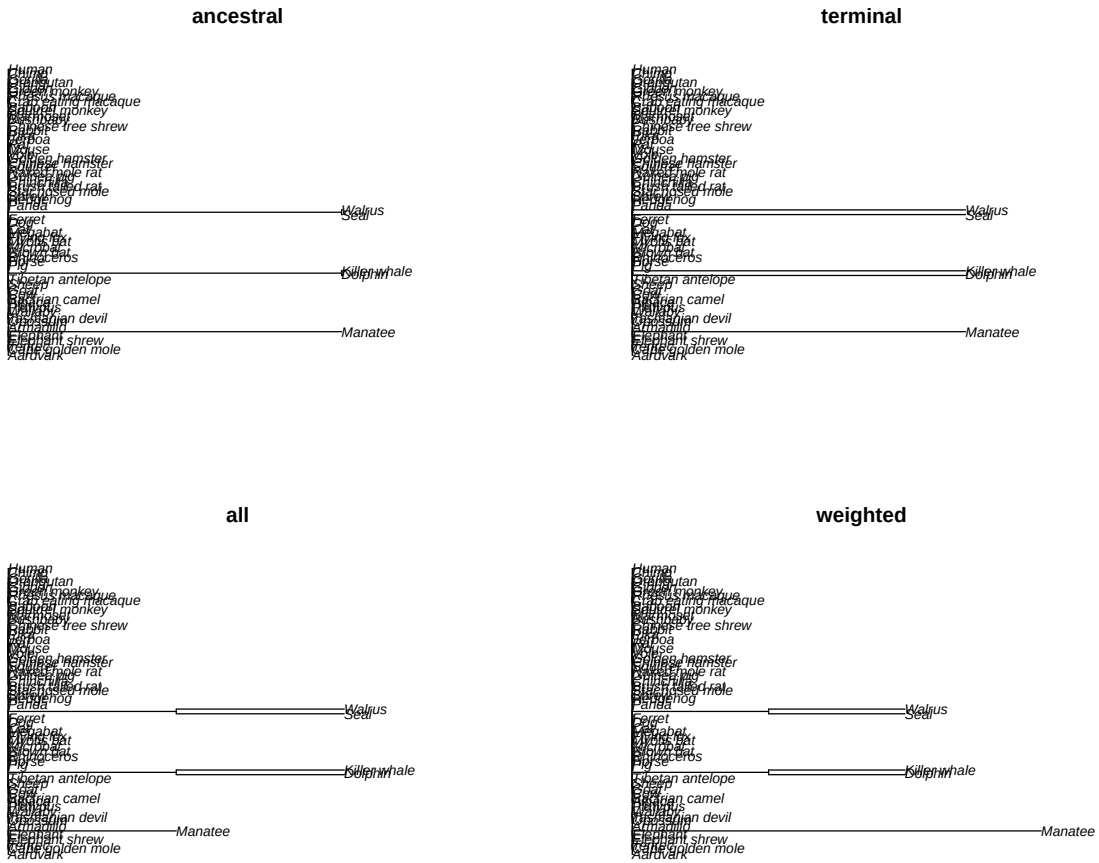
```
marineb2d = foreground2Tree(marineextantforeground, toyTrees, clade="weighted")
```



Here all branches in the cetacean and pinniped clades have length 1/3, whereas the terminal branch leading to the manatee has length 1. This is a way of distributing the weight given to each independent convergence event evenly across clades, rather than across lineages.

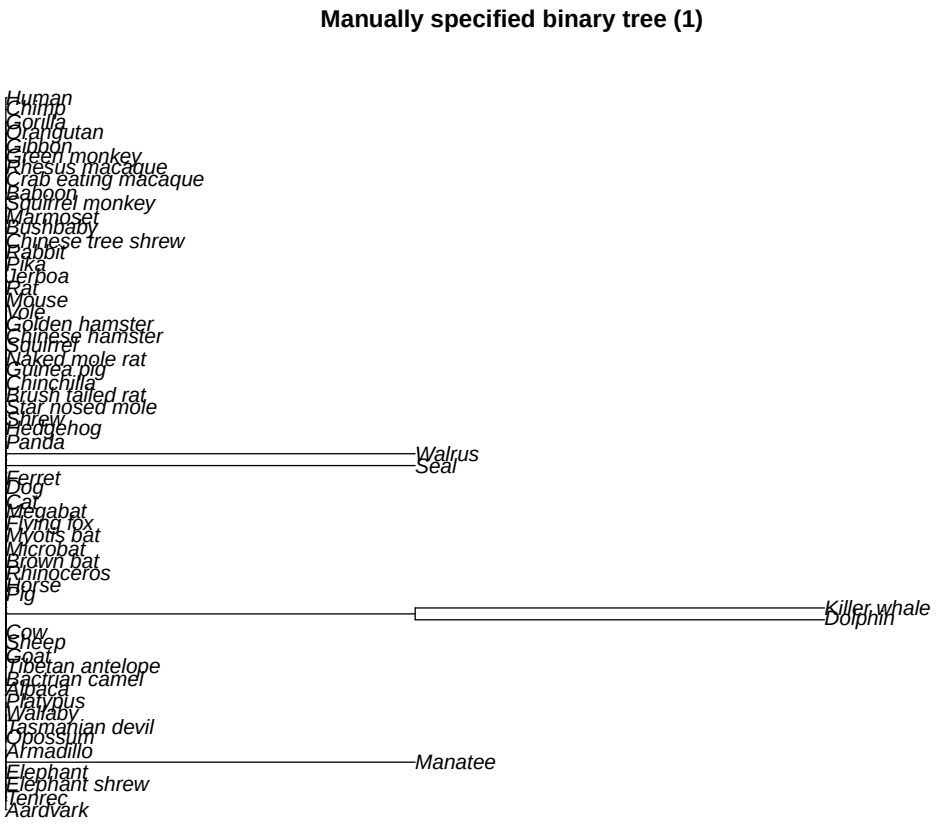
If you plot all the resulting trees, you can see how the different choices for `clade` influence the resulting branch lengths.

```
par(mfrow=c(2,2))
plot(marineb2a,main="ancestral")
plot(marineb2b,main="terminal")
plot(marineb2c,main="all")
plot(marineb2d,main="weighted")
```



None of these are the same as the binary tree that you specified in (1)!

```
plot(marineb,main="Manually specified binary tree (1)")
```



Note that, in this tree, the branch representing the ancestor of the killer whale and the dolphin has a branch length of 1, whereas the branch representing the ancestor of seal and walrus has a branch length of 0. This shows that providing a binary trait tree file (1) allows you to specify which branches should be foreground with more flexibility than providing a set of foreground species to `foreground2Tree`.

3) Use the interactive branch selection tool (not currently a part of the main RERconverge package; email the developers if you wish to use it). *If you have it installed*, the following should open a plot of the master tree. When the GUI opens, select the marine foreground branches (Walrus, Seal, Killer whale, Dolphin, Killer whale-Dolphin ancestor, and Manatee), and click 'End selection.'

```
marineb3=selectforegroundbranches(toyTrees$masterTree)
```

You can double check that the tree you created by manual selection (3) matches the binary tree you provided in (1) (it should) using the following code:

```
marineb3=selectforegroundbranches(toyTrees$masterTree)
marineb3rooted=root(marineb3,outgroup=c("Platypus", "Wallaby","Tasmanian_devil","Opossum"))
plot(marineb3rooted, main = "Trait tree from manual selection (3)")
```

Generating paths using `tree2Paths` or `foreground2Paths`

Some of the genes (like BEND3 above) may not have data for all species, meaning that their phylogeny will be a subset of the full phylogeny. To plot RERs for these genes and to correlate them with trait evolution,

we run one of two functions that determine how the trait would evolve along all/many possible subsets of the full phylogeny, generating a set of **paths**. The function `tree2Paths` takes a binary tree as input, and the function `foreground2Paths` takes a set of foreground species as input. `foreground2Paths` has the same arguments as `foreground2Tree` described above (see option 2 in *Reading in or generating trait trees*).

Important note: If you are using a binary tree to create paths, it **must** have the same topology as the **master tree** or a subset of the **master tree** (see previous section for how this is generated).

```
phenvMarine=tree2Paths(marineb, toyTrees)
#phenvMarine2=foreground2Paths(marineextantforeground, toyTrees, clade="all")
phenvMarine2=tree2Paths(marineb2b, toyTrees)
```

Correlating gene evolution with binary trait evolution using `correlateWithBinaryPhenotype`

Now that we have estimates for the RERs for all genes of interest, as well as a representation of how the trait of interest evolves across the tree, we can use `correlateWithBinaryPhenotype` to test for an association between relative evolutionary rate and trait across all branches of the tree.

This uses the following input variables (all the options set here are also the defaults):

- `min.sp`: the minimum number of species in the gene tree for that gene to be included in the analysis. The default is 10, but you may wish to modify it depending upon the number of species in your master tree.
- `min.pos`: the minimum number of independent foreground (non-zero) lineages represented in the gene tree for that gene to be included in the analysis. The default is 2, requiring at least two foreground lineages to be present in the gene tree.
- `weighted`: whether to perform a weighted correlation where branch lengths represent weights. This can be used with the `clade=weighted` option in `foreground2Tree` or `foreground2Paths` to distribute phenotype weights across multiple lineages in a clade. The default is "auto", which will use weighted correlation if any branch lengths are between 0 and 1, and will use standard correlation otherwise.

```
corMarine=correlateWithBinaryPhenotype(mamRERw, phenvMarine, min.sp=10, min.pos=2, weighted="auto")
```

The text displayed shows which correlation method is used to test for association. Here it uses the default for binary traits: the Kendall rank correlation coefficient, or Tau.

The `correlateWithBinaryPhenotype` function generates a table with the following output for each gene:

1) Rho: the correlation between relative evolutionary rate and trait across all branches
2) N: the number of branches in the gene tree
3) P: an estimate of the P-value for association between relative evolutionary rate and trait.
4) p.adj: an estimate of the P-value adjusted for multiple comparisons using the Benjamini-Hochberg procedure (i.e., an estimate of the FDR).

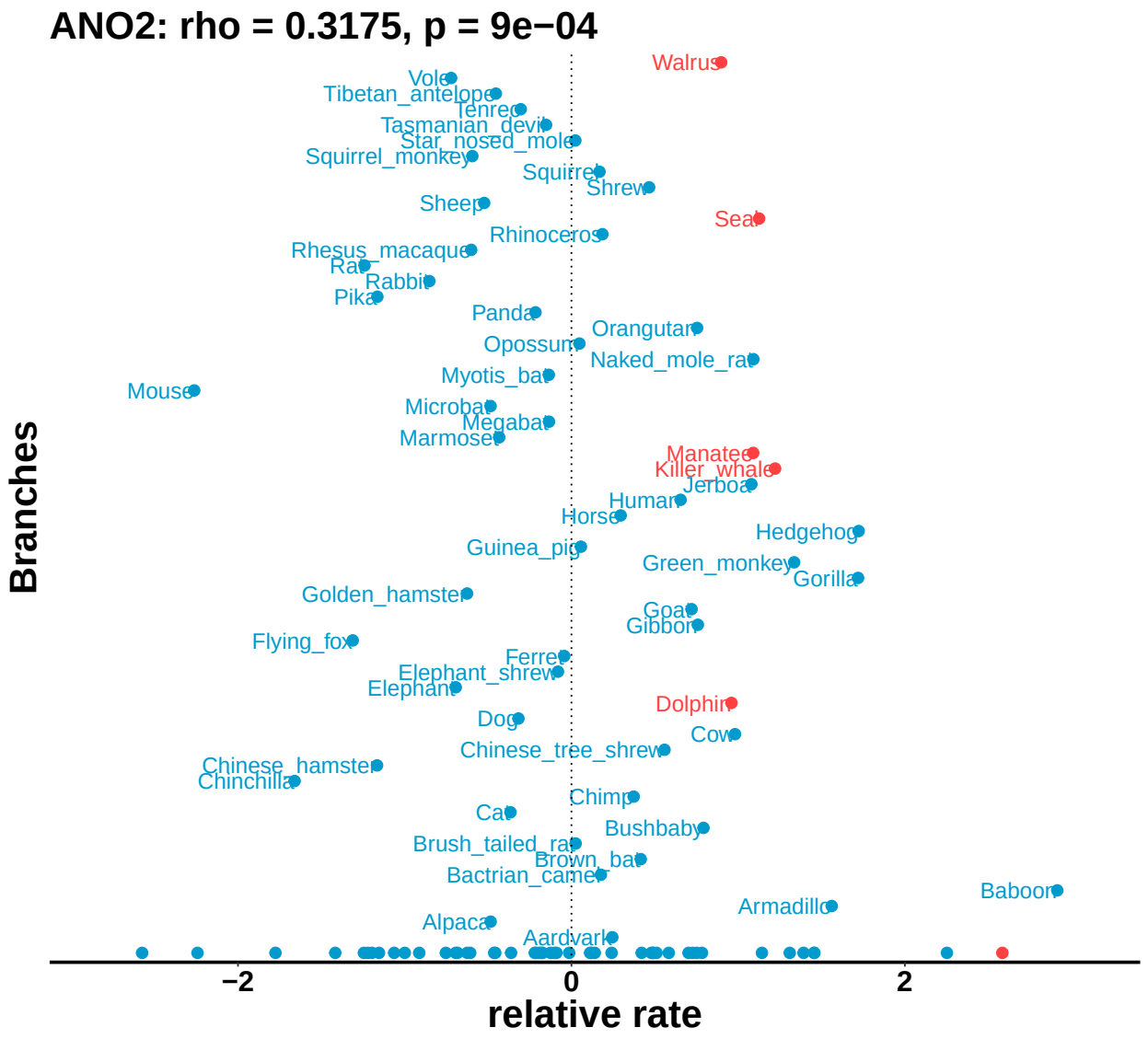
Let's take a look at some of the top genes within this set.

```
head(corMarine[order(corMarine$P),])
```

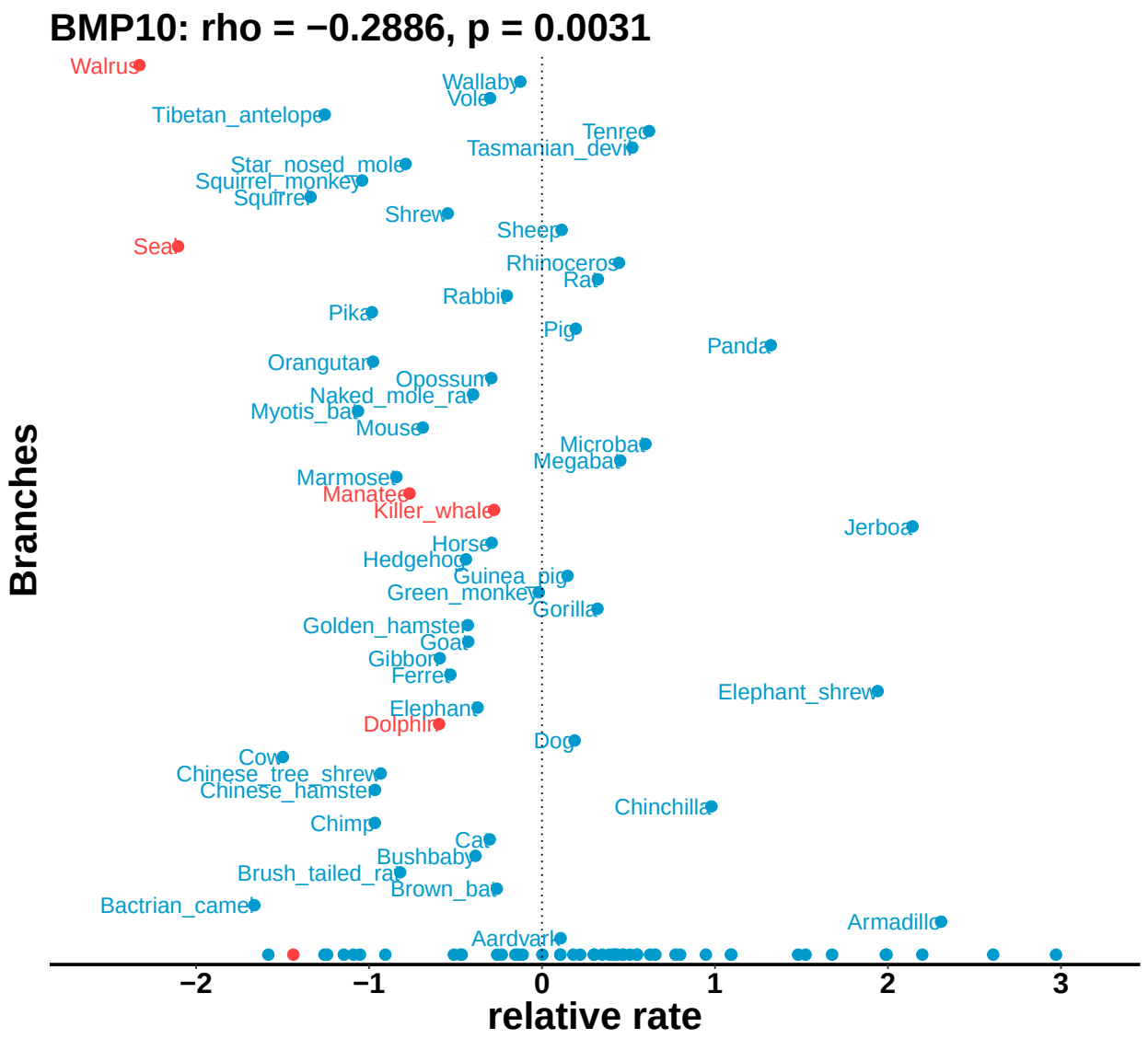
```
##                 Rho   N           P     p.adj
## ANO2      0.2604946 106 0.001138448 0.1575810
## AK124326 -0.3064213  68 0.002292048 0.1575810
## BDH1      0.2423160 102 0.002997042 0.1575810
## BMP10    -0.2367746 103 0.003561152 0.1575810
## ATP2A1    0.2312257 101 0.004832241 0.1710613
## ASB15     0.2136137 107 0.007338788 0.1977725
```

ANO2 and BMP10 are two of the top genes.

```
plotRers(mamRERw,"ANO2",phenv=phenvMarine)
```




```
plotRers(mamRERw,"BMP10",phenv=phenvMarine)
```

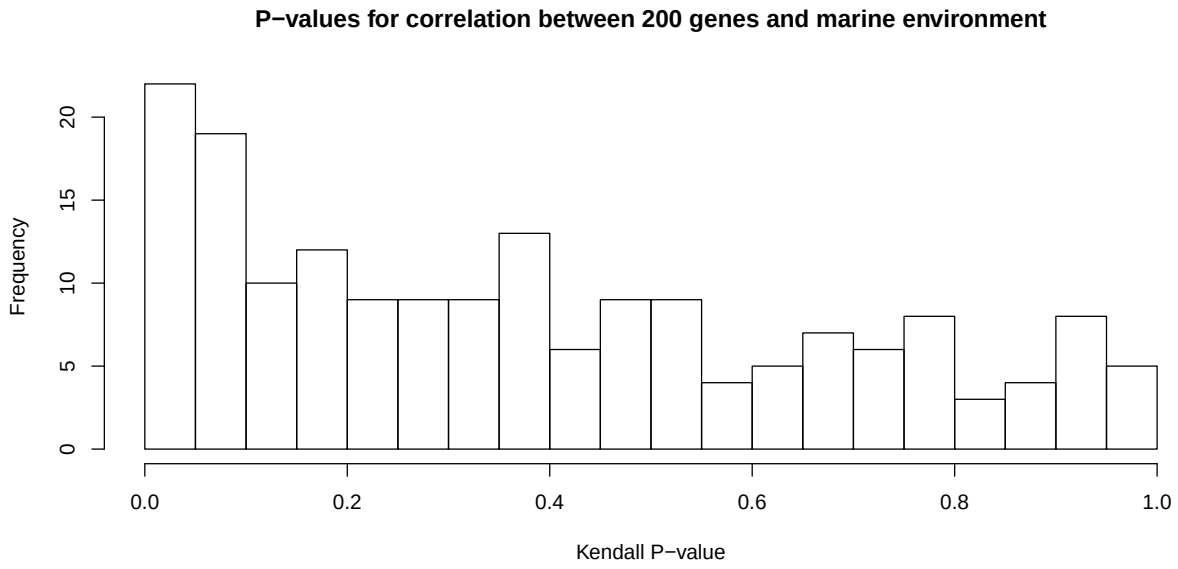


In these RER plots, the marine lineages specified in the foreground are highlighted in red. The terminal branches are listed in alphabetical order, and internal branches are displayed at the bottom; note the red point at the bottom of each plot indicating the RER for killer whale-dolphin ancestral branch.

In the ANO2 tree, marine lineages have high RER, leading to a positive Rho and a low p-value. In contrast, in the BMP10 tree, marine lineages have low RER. This also yields a low p-value, but with a negative Rho.

To see what the overall pattern of association is across all genes in the set, we can plot a p-value histogram.

```
hist(corMarine$P, breaks=15, xlab="Kendall P-value",
     main="P-values for correlation between 200 genes and marine environment")
```



There appears to be a slight enrichment of low p-values, but since we have only evaluated the first 200 genes from our ~19,000 gene genome-wide set, we should hold off on drawing conclusions from this.

Continuous Trait Analysis

In addition to supporting binary trait analyses, RERconverge can also calculate correlations between rates of evolution of genes and the change in a *continuous* trait. To perform a continuous trait analysis, start with a named vector in R. Vector names must match the names in the trees read in previously. Here are the first few entries in the vector we will use for continuous trait analysis:

```
head(logAdultWeightcm)
```

```
##             Alpaca            Dolphin Chinese_tree_shrew
##          15.919981          17.609640           7.643856
##         Tree_shrew            Manatee                Pig
##           7.643856          18.296701          16.988152
```

We must convert the trait vector to paths comparable to the paths in the RER matrix. To do that, we can use the function 'char2Paths' as shown here:

```
charpaths=char2Paths(logAdultWeightcm, toyTrees)
```

```
## using metric diff, with filtering constant -1
```

```
## Species not present: Cape_golden_mole
```

We are using `metric diff`, which means that branch lengths are assigned to the trait tree based on the difference in trait values on the nodes connected to that branch.

The function tells us that there is one species in the master tree that is not present in the trait tree: the cape golden mole.

The `char2Paths` function creates a paths vector with length equal to the number of columns in the RER matrix. The phylogenetic relationships represented in the "char2Paths" output are the same as those represented in the RER matrix.

Finally, we can perform our ultimate analysis to find correlations between the rate of evolution of a genomic element (encoded in the RER matrix) and the rate of change of a phenotype (encoded in charpaths) using `correlateWithContinuousPhenotype`. The final output is the list of input genes with relevant statistics. As input, we provide the RER matrix and trait path.

This function uses the following input variables (all the options set here are also the defaults):

- `min.sp`: the minimum number of species in the gene tree for that gene to be included in the analysis. The default is 10, but you may wish to modify it depending upon the number of species in your master tree.
- `winsorize`: pulls the outermost three (or whatever number you set) points in all four directions to the center of the data before calculating correlations to mitigate the effects of outlier points.

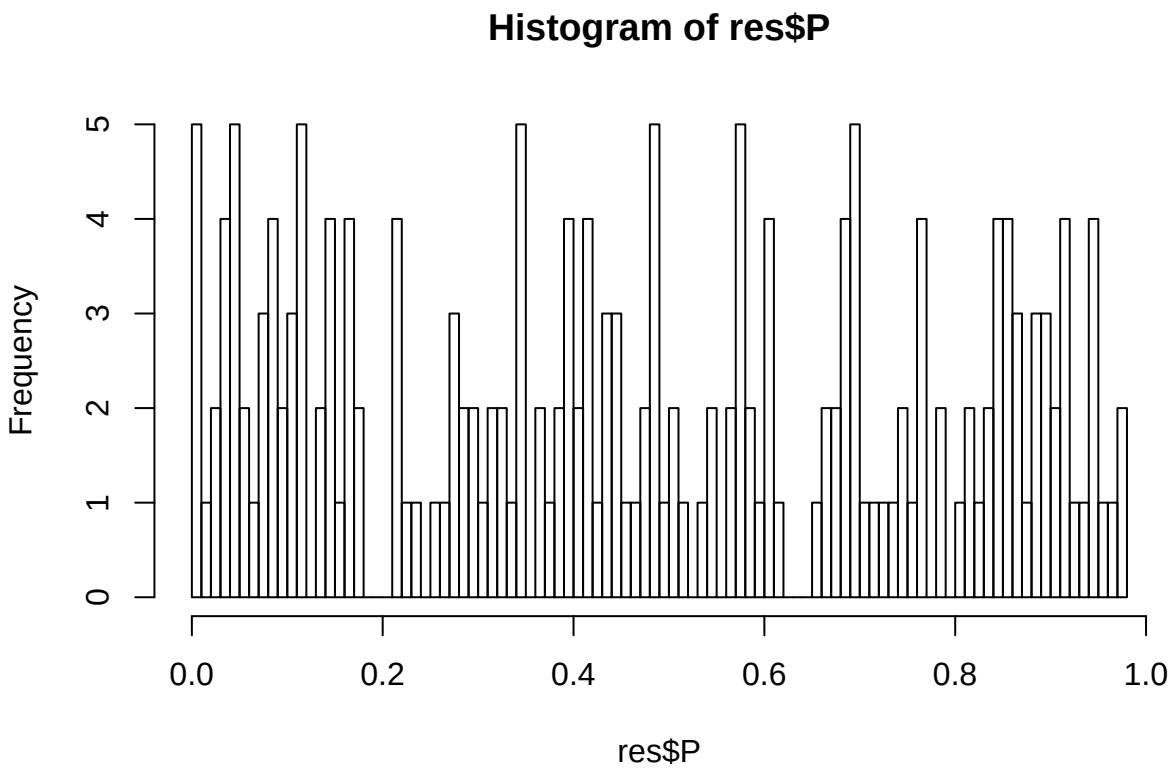
```
res=correlateWithContinuousPhenotype(mamRERw, charpaths, min.sp = 10, winsorize = 3)
head(res[order(res$P),])
```

```
##                 Rho   N           P     p.adj
## ANKRD18B -0.3391882  80 0.002085102 0.2630160
## TTN       0.2728150 119 0.002683837 0.2630160
## ABCB4     0.2529241 107 0.008578280 0.3696541
## ATP10A   -0.2528102 105 0.009269941 0.3696541
## ADH7      0.2903854  79 0.009429953 0.3696541
## B3GALT1   0.4074747  34 0.016759579 0.5404546
```

In these results, Rho is the standard statistic for a Pearson correlation, N is the length of the genomic element, and P is the uncorrected correlation p-value. Since huge numbers of statistical tests are being performed in these analyses, it is essential to correct p-values using a method such as the Benjamini-Hochberg correction.

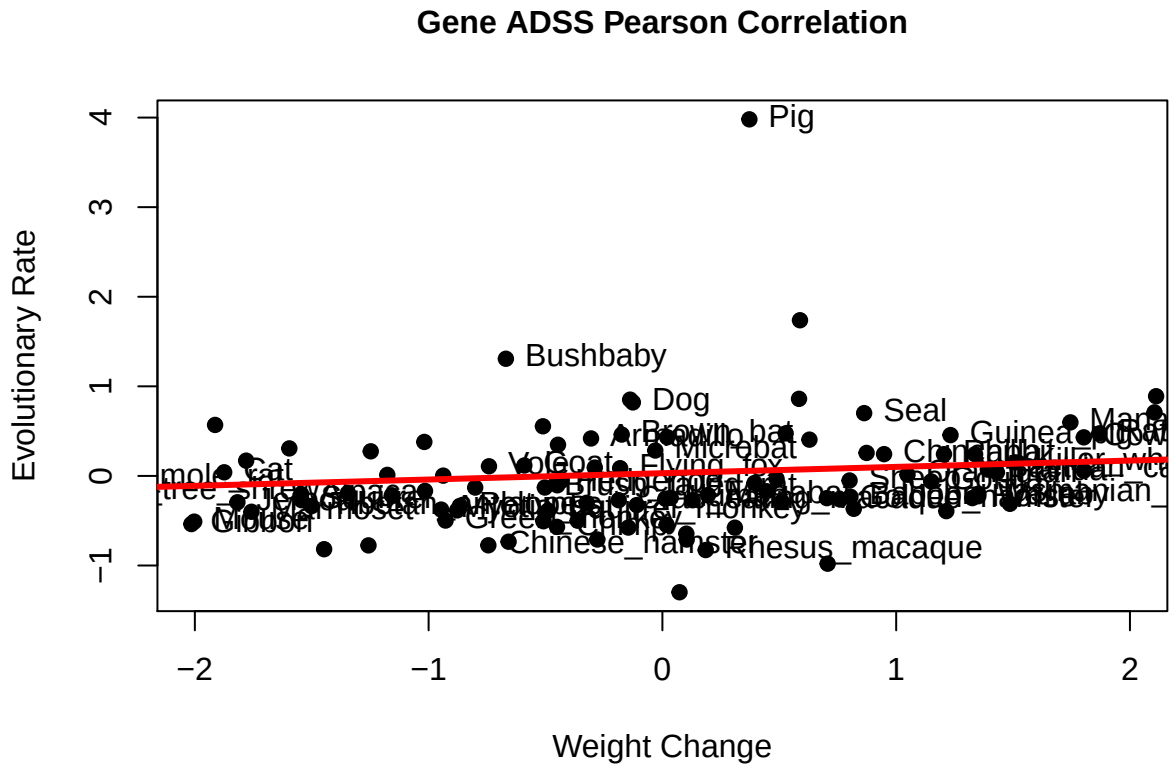
We can also plot a distribution of our uncorrected p-values to allow us to speculate if they will remain significant after correction. A mostly uniform distribution with an elevated frequency of low p-values indicates the presence of genomic elements whose rates of evolution are correlated with the phenotype (note that this trend will also be increasingly distinct when larger numbers of genomic elements are considered).

```
hist(res$P, breaks=100)
```



One important consideration of the results is the impact of one or a few species on the overall correlation. To assess this risk, we can examine individual correlation plots as follows:

```
x=charpaths
y=mamRERw['TTN',]
pathnames=namePathsWSpecies(toyTrees$masterTree)
names(y)=pathnames
plot(x,y, cex.axis=1, cex.lab=1, cex.main=1, xlab="Weight Change", ylab="Evolutionary Rate", main="Gene
text(x,y, labels=names(y), pos=4)
abline(lm(y~x), col='red',lwd=3)
```



In this case, we see that the positive correlation is driven by all species and not just a single clade. Note that labelled points are terminal nodes in the phylogeny and unlabelled points are internal nodes.

Further analyses could include using functional enrichment detection methods to find functionally-related genomic elements that are experiencing convergent evolutionary rates as a group and using branch-site models to determine if fast-evolving genes are experiencing relaxation of constraint or undergoing directional selection.

Conclusion

We've now walked through the basic workflow for RERConverge. For more information about these methods and some results relevant to marine and subterranean adaptation, see (Chikina, Robinson, and Clark 2016) and (Partha et al. 2017).

References

Chikina, Maria, Joseph D Robinson, and Nathan L Clark. 2016. "Hundreds of Genes Experienced Convergent Shifts in Selective Pressure in Marine Mammals." *Molecular Biology and Evolution* 33 (9): 2182–92. doi:10.1093/molbev/msw112.

Partha, Raghavendran, Bharesh K Chauhan, Zelia Ferreira, Joseph D Robinson, Kira Lathrop, Ken K Nischal, Maria Chikina, and Nathan L Clark. 2017. "Subterranean mammals show convergent regression in ocular genes and enhancers, along with adaptation to tunneling." *ELife* 6 (October). eLife Sciences Publications Limited: e25884. doi:10.7554/eLife.25884.