Categorical Trait Analysis Walkthrough

June 07, 2022

Contents

This walkthrough describes how to use the updates to RERconverge for analyzing categorical traits. This update builds on existing RERconverge objects. First time users should first read the "RERconverge Analysis Walkthrough" vignette.

Overview

The following document illustrates how to perform a categorical trait analysis after relative evolutionary rates have been calculated. To learn how to calculate relative evolutionary rates using `getAllResiduals` follow the "RERconverge Analysis Walkthrough" vignette.

Output is a list of two data objects. The first is a data frame containing a list of genomic elements with association statistics between the genomic element's evolutionary rate and the phenotype. The second object is a list of data frames for each pairwise test between the phenotype categories. For n phenotype categories, there will be $\binom{n}{2}$ data frames in this list. Each data frame is a list of genomic elements with association statistics describing the difference in relative evolutionary rates of genomic elements between the two categories.

Data Input Requirements and Formatting

The required inputs are as follows:

1. Phylogenetic trees of the same format described in the "RERconverge Analysis Walkthrough" vignette.
2. Species-labeled phenotype values

- The species labels MUST match the tree tip labels that were used in `getAllResiduals` to calculate the relative evolutionary rates (RERs)
- a named numeric vector of categorical trait values

Analysis Walkthrough

Follow the steps for installing RERconverge on the wiki, up to the "Install from Github" step. Then, load the RERconverge library.

```
if (!require("RERconverge", character.only = T, quietly = T)) {
  require(devtools)
  install_github("nclark-lab/RERconverge", ref = "New_Functions_For_Categorical_Traits")
  # ref refers to the branch of RERconverge being installed
}
library(RERconverge)
```

Follow the instructions in the "RERconverge Analysis Walkthrough" vignette in order to read in gene trees using `readTrees` and calculate evolutionary rates using `getAllResiduals`. That vignette describes how to save the RER object for later using `saveRDS`. Save both the RER object and the trees object. Make sure that you save these objects into your working directory for your project. We will read them in using `readRDS`. We will also read in the phenotype data.

It is very important that the names of the phenotype data EXACTLY match the names of the species that were used to calculate the relative evolutionary rates in `getAllResiduals`. To ensure this is the case, follow the instructions in the "RERconverge Analysis Walkthrough" vignette.

To use your own data that has already been saved in your working directory use the following code to read in the .rds files:

```
# read in the trees
toyTrees =readRDS("toyTrees.rds")
```

```
# read in the phenotype data
sleepPattern = readRDS("sleepPattern.rds")

# read in the RERs
RERmat = readRDS("sleepRERs.rds")
```

To use the same data as in this walk through, run the following code to read in the trees, load in the phenotype data, and calculate the relative evolutionary rates.

```
# find where the package is located on your machine
rerpath = find.package('RERconverge')

# read in the trees with the given file name
toytreefile = "subsetMammalGeneTrees.txt"
toyTrees=readTrees(paste(rerpath,"/extdata/",toytreefile,sep=""), max.read = 200)

# load the phenotype data into your workspace
# This will create a named vector with the name sleepPattern
data("sleepPattern")

# calculate the relative evolutionary rates with getAllResiduals
RERmat = getAllResiduals(toyTrees, useSpecies = names(sleepPattern))
```

Next, we generate the phenotype tree from the species-labeled phenotype vector using `char2TreeCategorical`.

This function uses existing function from ape or castor in order to infer the likelihoods of ancestral states for all species in the tree. The relevant citations are given below:

Paradis E. & Schliep K. 2019. ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R. Bioinformatics 35: 526-528.

Louca S, Doebeli M (2017). "Efficient comparative phylogenetics on large trees." *Bioinformatics*. doi: 10.1093/bioinformatics/btx701 (URL: https://doi.org/10.1093/bioinformatics/btx701).

This function takes the following inputs:

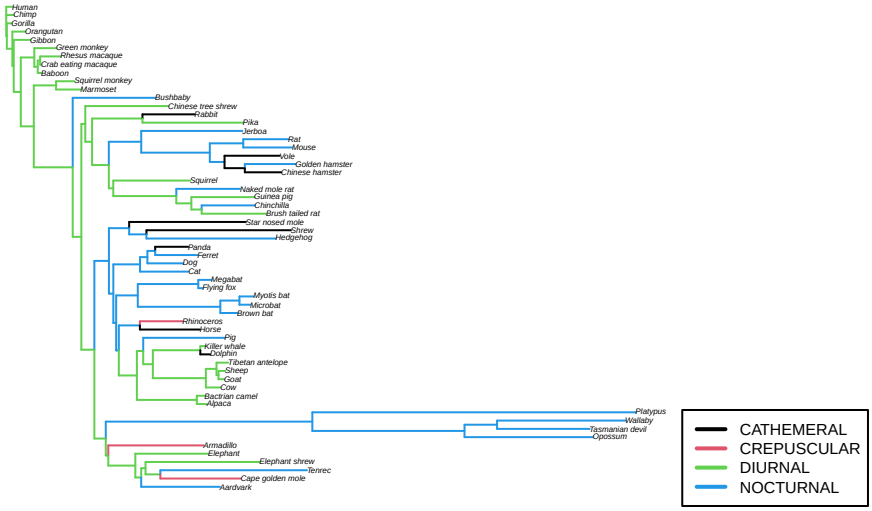
- `tipvals`: The named vector of phenotype data
- `treesObj`: The trees object containing every gene tree
- `useSpecies`: Specifies the subset of species to use in the analysis. This vector of species should match the subset used to calculate RERs.
- `use_rooted`: A boolean specifying whether to first root the tree before performing ancestral trait reconstruction. The ape function `ace` requires a rooted tree as input, therefore when use_rooted is true, the ape function is used to infer ancestral states. The castor function `asr_mk_model` does not require the tree to be rooted. The default value for `use_rooted` is `FALSE` since the input trees are unrooted.
- `outgroup`: When `use_rooted` is `TRUE`, the user must supply an outgroup to use when rooting the tree. The default value is `NULL`.
- `model`: The model used for fitting the transition rate matrix. For more information regarding input options for model when `use_rooted` is `FALSE` see the description of the `rate_model` input from castor. When `use_rooted` is `TRUE`, reference the documentation from ape for more information regarding the model. The default option is `"ER"`.
- `plot`: A boolean specifying whether to plot the phenotype tree.

By default, the castor function which does not require rooting the tree is used:

```
allspecs = names(sleepPattern)

# unrooted
```

```
phenTree = char2TreeCategorical(sleepPattern, toyTrees, useSpecies = allspecs,
                                plot = TRUE)
```

```
## [1] "The integer labels corresponding to each category are:"
##  CATHEMERAL CREPUSCULAR     DIURNAL   NOCTURNAL
##           1           2           3           4
```



Alternatively, to use the ape function and root the tree, run the following code:

```
# rooted
allspecs = names(sleepPattern)

phenTree = char2TreeCategorical(sleepPattern, toyTrees, useSpecies = allspecs,
                                use_rooted = TRUE, outgroup = "REFERENCE",
                                plot = TRUE)
```

The output of this function is a phenotype tree with the same topology as the master tree. The phenotype states are stored on the branches of the tree as edge lengths. The ancestral trait reconstruction functions use numerical (integer) tip labels. The integers do not have any biological significance and are typically assigned in alphabetical order. The integer values corresponding to the categories in the phenotype vector are printed to the console.

Additionally, to see the mapping of category names to integers, you can run the castor function that is used within `char2TreeCategorical` as shown below:

```
intlabels = map_to_state_space(sleepPattern)
print(intlabels$name2index)
```

```
##  CATHEMERAL CREPUSCULAR     DIURNAL   NOCTURNAL
##           1           2           3           4
```

Generating paths using `char2PathsCategorical`

Some of the gene trees are missing species from the analysis. To handle missing species, RERconverge generates paths. For a more detailed discussion of paths see the "RERconverge Analysis Walkthrough" vignette. For categorical traits we use the function `char2PathsCategorical`. This function has the same inputs as `char2TreeCategorical`.

```
allspecs = names(sleepPattern)
```

```
# unrooted
charP = char2PathsCategorical(sleepPattern, toyTrees, useSpecies = allspecs,
                              plot = FALSE)
```

```
## [1] "The integer labels corresponding to each category are:"
##  CATHEMERAL CREPUSCULAR     DIURNAL   NOCTURNAL
##           1           2           3           4
```

Alternatively, to use the rooted option run the code below:

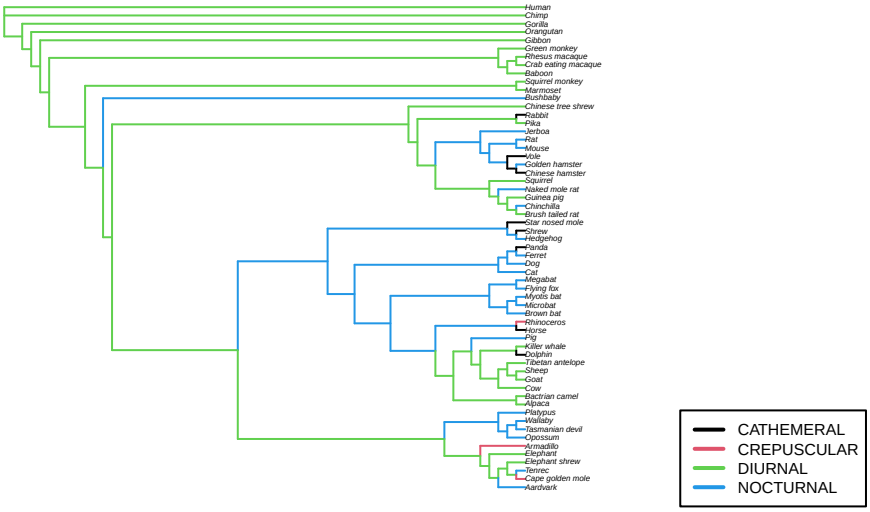
```
allspecs = names(sleepPattern)

# rooted
charP = char2PathsCategorical(sleepPattern, toyTrees, useSpecies = allspecs,
                              use_rooted = TRUE, outgroup = "REFERENCE",
                              plot = FALSE)
```

Visualization

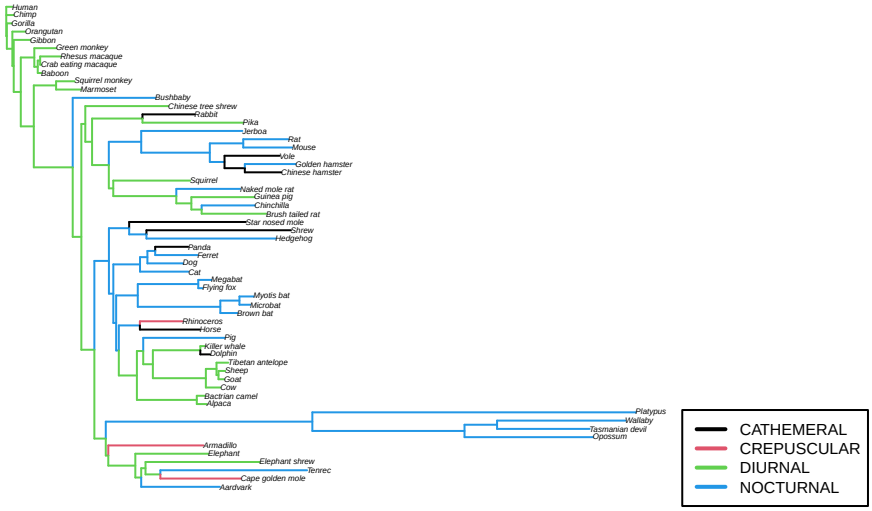
Visualizing the phenotype tree: In addition to visualizing the phenotype tree with `char2TreeCategorical` and `char2PathsCategorical` when `plot = TRUE`, the tree can be plotted with `plotTreeCategorical`. The inputs of this functiona are:

- `tree`: The phenotype tree returned by `char2TreeCategorical`
- `category_names`: If provided, the plot includes a legend with the category names and corresponding colors. The category names MUST be provided in the same numerical order as the mapping from names to integers. This can be done easily using the `intlabels` object returned by `map_to_state_space`. `category_names` can be set to `intlabels$state_names`. Otherwise the default value is `NULL` and no legend is included.
- `master`: The master tree in the trees object returned by `readTrees`. This tree will be plotted with its branches colored by the phenotypes stored in the phenotype tree.

```
# plotting phenotypes without relative evolutionary rates represented by branch length
plotTreeCategorical(tree = phenTree, category_names = intlabels$state_names)
```




```
# plotting phenotypes with branch lengths from the master tree
plotTreeCategorical(tree = phenTree, category_names = intlabels$state_names,
                    master = toyTrees$masterTree)
```



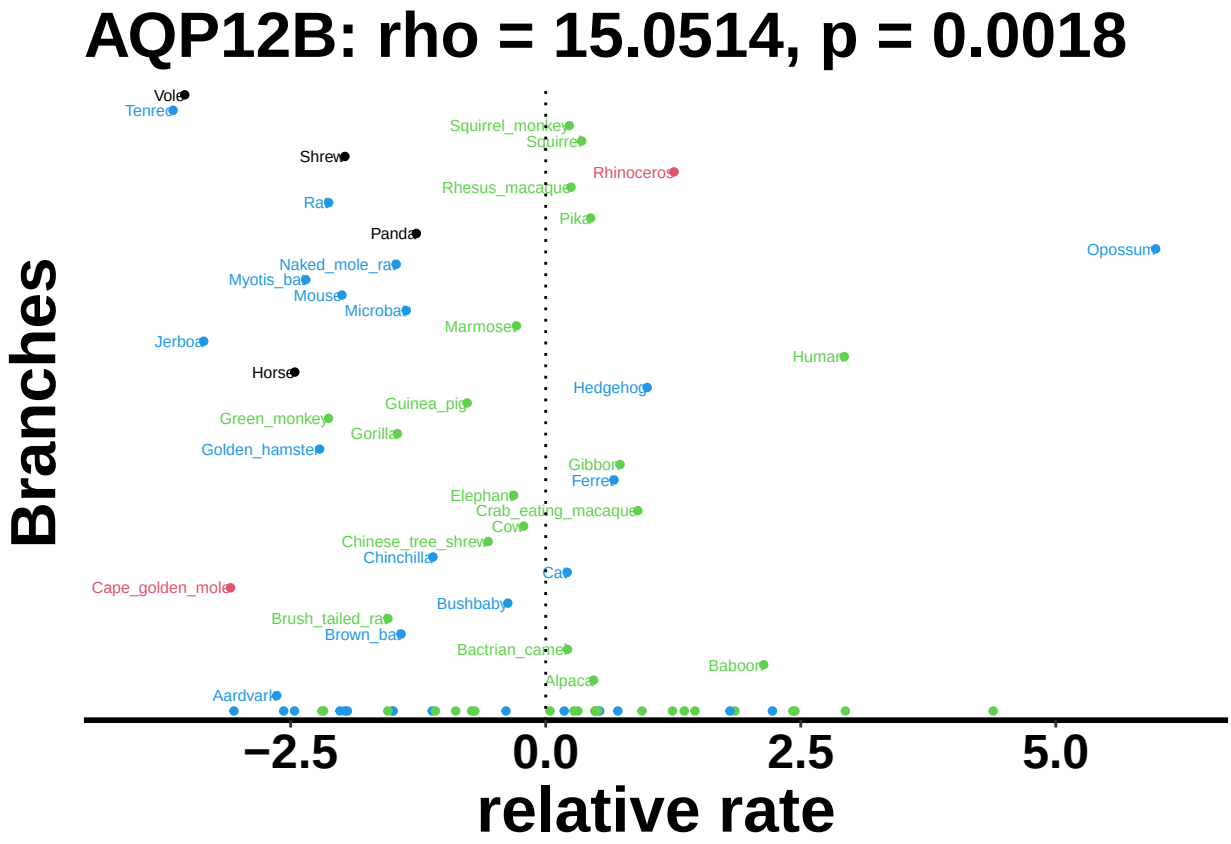
Visualizing the relative evolutionary rates: The relative evolutionary rates for a specific gene can be visualized using the function `plotRers`. For more details regarding reading this plot, see the "RERconverge Analysis Walkthrough" vignette. A negative value indicates a relative evolutionary rate that is below average while a positive value indicates a relative evolutionary rate that is above average. The colors used to distinguish categories match the colors in the phenotype tree that is plotted by `plotTreeCategorical`, `char2TreeCategorical`, or `char2PathsCategorical`.

The example below will show how to plot the RERs for one of the top genes, `"AQP12B"`. The `plotRers` function takes the following as input:

- `RERmat`: The RER matrix returned by `getAllResiduals`.
- `gene`: Either the name of the gene or the numerical index of the gene in the RER matrix.
- `phenv`: The paths generated by `char2PathsCategorical`.

The default method for calculating the correlation statistics that are displayed at the top of the plot is `"kw"` for Kruskal Wallis. To use ANOVA, use the parameter `method = "aov"`.

```
gene = "AQP12B"
plotRers(RERmat, gene, phenv = charP)
```



Correlating gene evolution with categorical trait

To correlate gene evolution with categorical trait evolution we use the function `correlateWithCategoricalPhenotype`. The function tests for association between the relative evolutionary rates of genes and the evolution of the phenotype. It takes the following as input:

- `RERmat`: The RER matrix from `getAllResiduals`.
- `charP`: The paths vector from `char2PathsCategorical`.
- `min.sp`: The minimum number of species in the gene tree in order for that tree to be included in the analysis. The default value is 10.
- `min.pos`: The minimum number of species in each category in the gene tree in order for that gene to be included in the analysis. The default value is 2.
- `method`: The input options are `"kw"` for performing a Kruskal-Wallis test (the non parametric option). The pairwise testing is done using a Dunn Test. To use an ANOVA test instead (the parametric option), use `method = "aov"`. In this case hte pairwise testing is done using a Tukey Test. When not specified, the default is `"kw"`.

The function for performing the Dunn Test comes from the package FSA and is cited below: Ogle, D.H., J.C. Doll, P. Wheeler, and A. Dinno. 2022. FSA: Fisheries Stock Analysis. R package version 0.9.2, https://github.com/droglenc/FSA.

```
# KW/Dunn (default)
cors = correlateWithCategoricalPhenotype(RERmat, charP)
```

```
# ANOVA/Tukey
cors = correlateWithCategoricalPhenotype(RERmat, charP, method = "aov")
```

The output (`cors`) is the two-element list described previously in the above section. The first element of cors

is a table with the following output for each gene: Rho, N, P, and p.adj. Descriptions of each are given in the "RERconverge Analysis Walkthrough" vignette.

Extract this table as follows:

```
allresults = cors[[1]]

# view the first few results
head(allresults[order(allresults$P),])
```

```
##             Rho   N           P     p.adj
## AQP12B 15.05141  77 0.001773239 0.3125858
## AUTS2  12.90368  96 0.004849606 0.3125858
## ABCD1  12.79350 102 0.005105144 0.3125858
## BTBD18 11.87914 102 0.007808857 0.3125858
## ATR    11.52592 111 0.009196845 0.3125858
## ASZ1   10.96691  97 0.011906131 0.3125858
```

The second element in the cors object is a list of tables from the pairwise analysis. Extract this list as follows:

```
pairwise_tables = cors[[2]]
```

Run `names(pairwise_tables)` to see the order of the pairwise comparisons in this list. They are labeled numerically so, for example, the element named `1 - 3` is the data frame with the results of the pairwise comparison between the category mapped to the number `1` and the category mapped to the number `3`.

Recall that the mapping of category names to integers was printed to the console when `char2TreeCategorical` or `char2PathCategorical` was run. Additionally, recall that you can view the mapping using functions from the castor library as shown below:

```
intlabels = map_to_state_space(sleepPattern)
print(intlabels$name2index)
```

Each data frame in the list `pairwise_tables` contains the following output for each gene:

1. Rho: Though the column is labeled Rho (in order to stay consistent), this is the test statistic returned from either the Dunn test or the Tukey test. For the Dunn test it is known as the Z statistic and for the Tukey test it is the Honest Significant Difference. It represents the relationship between the relative evolutionary rate of a gene and evolution of the phenotype.
2. P: The p-value corrected for pairwise testing, but not corrected for multiple hypothesis testing for the many genes.

```
# View the top results of the sixth pairwise comparison
table = pairwise_tables[[6]]
head(table[order(table$P),])
```

```
##             Rho           P
## ABCD1  3.218128 0.007741806
## ATR    3.189687 0.008545629
## AQP12B 3.155075 0.009627405
## AUTS2  3.117759 0.010933899
## BTBD18 3.069578 0.012861692
## ARMC9  3.043093 0.014049584
```

Enrichment Walkthrough

The enrichment analysis is performed in the same way as for binary and continuous traits.

You will need to download the gene sets and gene symbols from GSEA-MSigDB as gmtfile.gmt. Follow the instructions in the "RERconverge Analysis Walkthrough" vignette in order to properly download and save the gmt file in your current working directory. The "RERconverge Analysis Walkthrough" may say to download the file named c2.all.v6.2.symbols.gmt, however if that is not available, c2.all.v7.5.1.symbols.gmt will work. Ensure that the name of the gmt file in your working directory is "gmtfile.gmt".

Input is the output from the correlation function. This can be `allresults` (`cors[[1]]`) or any of the tables in the list, `pairwise_tables` (`cors[[2]]`). The second input is the pathways of interest with gene symbols (read in as "allannots.rds" in the code below).

Output is the enrichment statistics from each pathway including the genes in the pathway and their ranks.

```
# read in the annotations
annots = read.gmt("gmtfile.gmt")
```

```
# format in a list
annotlist=list(annots)
names(annotlist)="MSigDBpathways"

# calculate enrichment statistics for the results including all categories
allenrichments = fastwilcoxGMTall(getStat(allresults), annotlist, outputGeneVals=T)
```

```
## 23 results for annotation set MSigDBpathways
```

```
# View the stat, pval, and p.adj of the top enrichment results
head(allenrichments$MSigDBpathways[order(allenrichments$MSigDBpathways$pval),])[1:3]
```

```
##                                              stat      pval     p.adj
## REACTOME_INNATE_IMMUNE_SYSTEM             -0.19184652 0.0276999 0.6370976
## BENPORATH_CYCLING_GENES                   -0.16879433 0.0749310 0.8617065
## PEREZ_TP53_TARGETS                         0.12597547 0.1338160 0.8767631
## DODD_NASOPHARYNGEAL_CARCINOMA_UP          -0.09847328 0.1567043 0.8767631
## ZWANG_TRANSIENTLY_UP_BY_2ND_EGF_PULSE_ONLY -0.10636079 0.1906007 0.8767631
## KINSEY_TARGETS_OF_EWSR1_FLII_FUSION_UP     0.10311751 0.2366855 0.9072946
```

```
# the sixth table is for the pairwise comparison between diurnal and nocturnal species
# (with integer labels 3 and 4 respectively)
di_noc_enrichments = fastwilcoxGMTall(getStat(pairwise_tables[[6]]), annotlist, outputGeneVals=T)
```

```
## 23 results for annotation set MSigDBpathways
```

```
# View the stat, pval, and p.adj of the top enrichment results
head(di_noc_enrichments$MSigDBpathways[order(di_noc_enrichments$MSigDBpathways$pval),])[1:3]
```

```
##                                              stat      pval     p.adj
## REACTOME_TRANSPORT_OF_SMALL_MOLECULES      0.11896400 0.1106297 0.9396761
## ZWANG_TRANSIENTLY_UP_BY_2ND_EGF_PULSE_ONLY -0.12877998 0.1130391 0.9396761
## REACTOME_METABOLISM_OF_LIPIDS               0.09705882 0.2180463 0.9396761
## NABA_MATRISOME                             -0.10567376 0.2648827 0.9396761
## CHEN_METABOLIC_SYNDROM_NETWORK             -0.06183206 0.3738585 0.9396761
## MIKKELSEN_ES_ICP_WITH_H3K4ME3              -0.07494005 0.3898083 0.9396761
```

For a more in depth explanation of performing an enrichment analysis, see the "RERconverge Analysis Walkthrough" vignette.

Conclusion

This concludes the walk through of how to use the new functions in RERconverge for analyzing categorical traits. Thank you!