# Categorical Trait Analysis Walkthrough

July 05, 2023

## 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

## Installation

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 = "master")
  # ref refers to the branch 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 replacing the names of the files here with the names of your files:

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

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

# read in the RERs
RERmat = readRDS("bodyTempRERs.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 basalRate
data("basalRate")

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

## Inferring the Phenotype Tree

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

This function uses code from castor(Louca and Doebeli 2017) and internally calls the function `getAncLiks` which borrows heavily from the function `ace` in ape(Paradis and Schliep 2019) and from the functions `fitMk` and `rerootingMethod` in phytools(Revell 2012). The relevant citations are given below:

For more information on choosing a rate model (see parameter `model` below) refer to the "Ancestral State Reconstruction for Binary and Categorical Traits" vignette.

This function takes the following inputs:

- `tipvals`: The named vector of phenotype data. This may be categorical data or binary data. If the binary phenotype data is of type logical (TRUE/FALSE), `char2TreeCategorical` automatically returns the inferred ancestral states as a binary tree with branch lengths of 1 and 0.
- `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.
- `model`: A character or matrix describing the model used for fitting the transition rate matrix. The default option is `"ER"`, short for equal rates indicating that all transitions between states occur at the same rate. `"SYM"` (symmetric) and `"ARD"` (all rates different) are also options. For more information, reference the ace documentation in ape or the "Ancestral State Reconstruction for Binary and Categorical Traits" vignette, which provides an in depth tutorial for selecting a rate model.
- `plot`: A boolean specifying whether to plot the phenotype tree.
- `anctrait`: If provided, the states of ancestral species will be assigned to this trait rather than being inferred using maximum likelihood methods. If provided, it MUST be one of the traits in the phenotype vector. The default value is `NULL`.

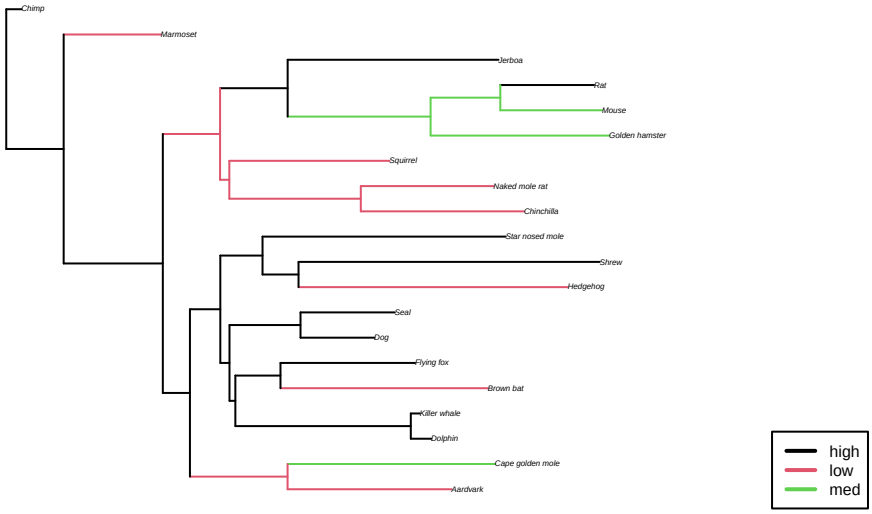
The code below shows how to obtain a phenotype tree using `char2TreeCategorical`.

```
allspecs = names(basalRate)

# unrooted
phenTree = char2TreeCategorical(basalRate, toyTrees, useSpecies = allspecs, model = "ER",
                                plot = TRUE)
```

```
## Species from master tree not present in useSpecies: Platypus,Opossum,Tasmanian_devil,Wallaby,Armadil
```

```
## [1] "The integer labels corresponding to each category are:"
## high  low  med
##    1    2    3
```




By default, `char2TreeCategorical` infers the ancestral states of species in the tree such that internal species can be assigned to any of the categories. However, there may be phenotypes in which it makes more sense for all ancestral species to belong to one category with some of the extant species belonging to the remaining categories. This is analogous to using the `clade = terminal` option in `foreground2Tree` for binary traits. (See the "RERconverge Analysis Walkthrough vignette for a more detailed discussion of the `clade` argument.)

By providing an ancestral trait via the `anctrait` argument, `char2TreeCategorical` will not infer the ancestral states of species. Instead, all ancestral species will be assigned to the ancestral trait while the extant species will be assigned to categories according to the named phenotype vector. The ancestral trait must be one of the traits in the phenotype vector, thus it is important to ensure that it is spelled the same and contains the same pattern of capitalization. When using this option with a trait that contains only two categories, the output will be a binary phenotype tree (the same output as `foreground2Tree` with `clade = terminal`), and the function will prompt you to proceed as appropriate for a binary trait analysis.

```
allspecs = names(basalRate)

phenTree = char2TreeCategorical(basalRate, toyTrees, useSpecies = allspecs, plot = TRUE,
                                anctrait = "high")
```

The output of `char2TreeCategorical` 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(basalRate)
print(intlabels$name2index)
```

```
## high  low  med
##    1    2    3
```

**Note:** `char2TreeCategorical` can automatically detect a binary phenotype and return the inferred phenotype tree as a binary tree (with branch lengths of 1 and 0). This binary tree is not necessarily equivalent to the binary tree returned by `foreground2Tree` because it uses different methods for inferring ancestral states.

#### Interactive Phenotype Tree Construction

Alternatively, RERconverge provides an interactive selection tool. To open the interactive selection tool run `click_select_foreground_branches` as shown below. Running this function should open a plot of the master tree.

```
phenTree = click_select_foreground_branches(toyTrees$masterTree)
```

Before making any selections, all the branches are automatically assigned to category 1. Begin by selecting branches corresponding to category 2. To select branches for the next category click the "New Category" button and select the corresponding branches. These will be assigned to category 3 and so on.

IMPORTANT: When using this approach, there is no method that maps category names to numbers. Please keep track of the order in which you add the categories as `map_to_state_space` will NOT give the correct mapping. When finished, click "End Selection". If you only select foreground branches for one category, this function will automatically return a binary phenotype tree in which the branch lengths are "1"s and "0"s, and it will prompt you to use `correlateWithBinaryPhenotype` instead. In this case, you should also generate paths as described in the "RERconverge Analysis Walkthrough" vignette for binary traits.

### Generating paths using `char2PathsCategorical`

The `toyTrees` object contains a separate gene tree for each gene in the analysis with branch lengths representing the evolutionary rates of that gene. All of the gene trees have the same overall topology as the master tree, but some of them are missing certain species. To handle missing species, RERconverge generates something called 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(basalRate)

# unrooted
charP = char2PathsCategorical(basalRate, toyTrees, useSpecies = allspecs, model = "ER",
                              plot = FALSE)
```

```
## Species from master tree not present in useSpecies: Platypus,Opossum,Tasmanian_devil,Wallaby,Armadil
```

```
## [1] "The integer labels corresponding to each category are:"
## high  low  med
##    1    2    3
```

Alternatively, you can generate paths directly from a phenotype tree generated by `char2TreeCategorical` or `click_select_foreground_branches` using the function `tree2Paths`.

IMPORTANT: The phenotype tree MUST have the same topology as the master tree or a subset of the master tree. This is always the case when using `click_select_foreground_branches` or `char2TreeCategorical`.

```
charP = tree2Paths(phenTree, toyTrees, categorical = TRUE)
```

(To see how to use `tree2Paths` on binary trees reference the "RERconverge Analysis Walkthrough" vignette.)

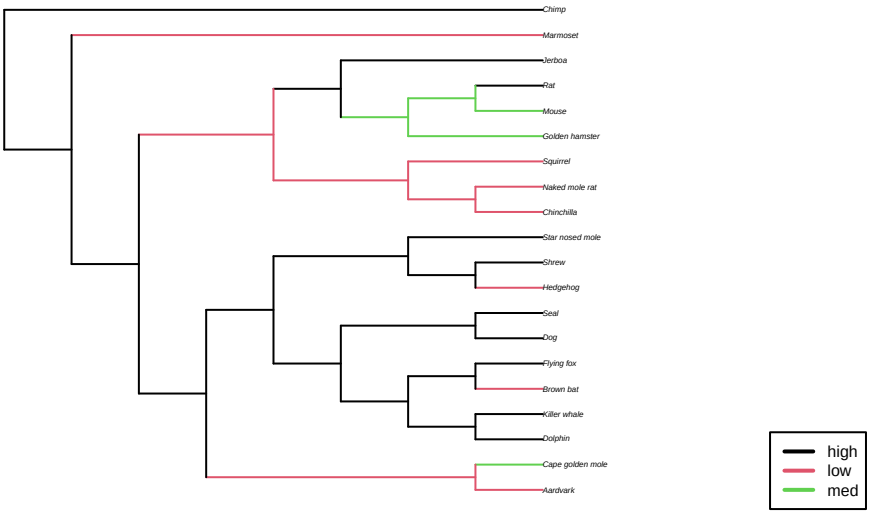
## Visualization

**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:

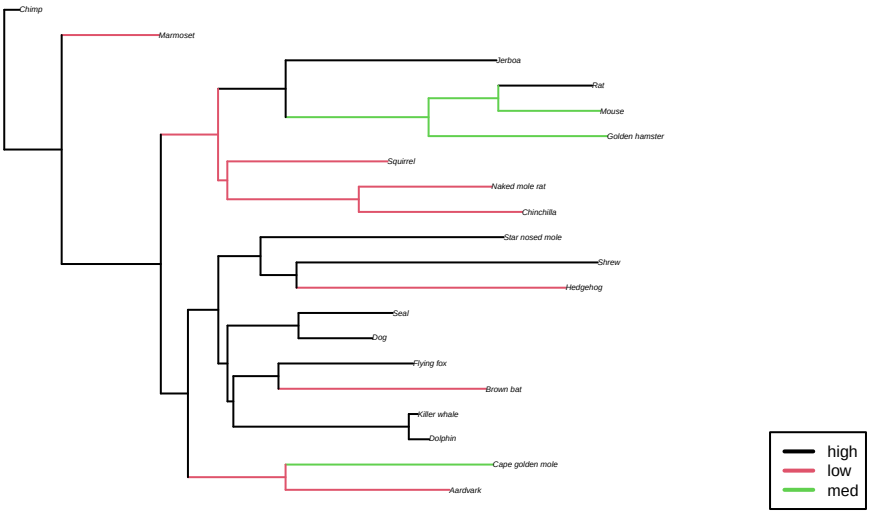
- `tree`: The phenotype tree returned by `char2TreeCategorical`
- `category_names`: If provided, the plot includes a legend with the category names and corresponding colors. The category names MUST be provided in the same numerical order as the mapping from names to integers. This can be done easily 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.
- `node_states`: The states at each node (in order of the nodes in the tree). If provided, these are used to color the vertical bars corresponding to each node in the tree. The default is `NULL`.

```
# get the states from the edges of the phenotype tree
root_state = 1 # high (1) is the state at the root
states = getStatesFromPhenTree(phenTree, root_state, intlabels$mapped_states)

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




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



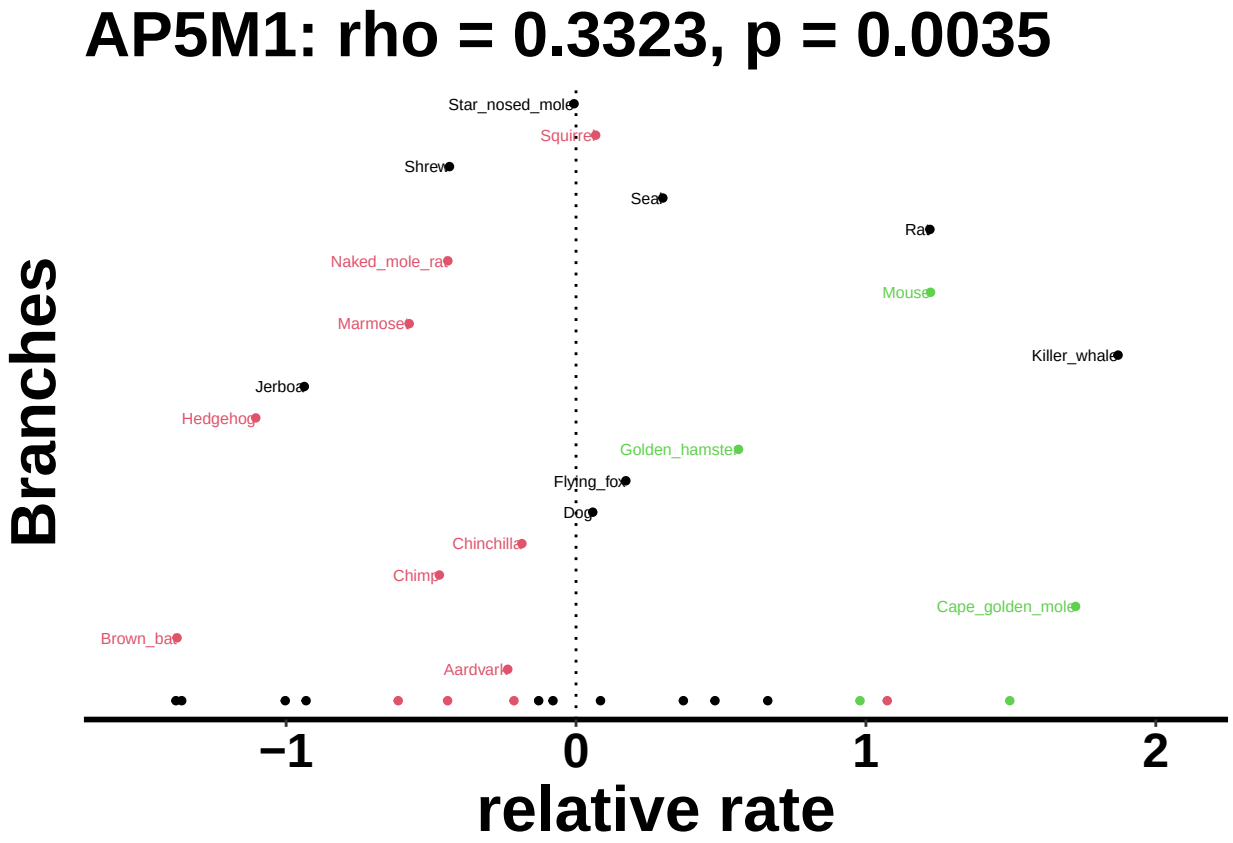

**Visualizing the relative evolutionary rates:** The relative evolutionary rates for a specific gene can be visualized using the function `plotRers`. For more details regarding reading this plot, see the "RERconverge Analysis Walkthrough" vignette. A negative value indicates a relative evolutionary rate 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, `"AP5M1"`. 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 = "AP5M1"
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 the 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(Ogle et al. 2022).

```
# 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. However, Rho is not just the test statistic, rather it is the

effect size. The Kruskal Wallis test uses epsilon squared as the measure of effect size. The epsilon squared effect size for the Kruskal Wallis test is calculated by taking the H statistic and dividing it by the number of observations minus one as described in this article. For the ANOVA test, eta squared is used as the effect size. Eta squared is calculated according to the method in this article.

Extract the first element of `cors` as follows:

```
allresults = cors[[1]]

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

```
##              Rho  N           P    p.adj
## AP5M1  0.3323218 35 0.003519378 0.363086
## BRAF   0.3700350 28 0.006768521 0.363086
## ADAD1  0.2885543 35 0.007406307 0.363086
## BRSK2  0.3040665 31 0.010451633 0.363086
## ACTL7B 0.2738857 34 0.010898678 0.363086
## ARSA   0.3112253 29 0.012814799 0.363086
```

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(basalRate)
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.
3. p.adj: The p-value corrected for multiple hypothesis testing using the Benjamini & Hochberg method.

```
# View the top results of the third pairwise comparison
table = pairwise_tables[[3]]
head(table[order(table$P),])
```

```
##              Rho           P     p.adj
## AP5M1  -3.361228 0.002327904 0.3887599
## ADAD1   3.132053 0.005207653 0.4348390
## ACTL7B  2.999228 0.008119932 0.4520095
## BRSK2   2.840025 0.013532984 0.5650021
## BIRC5  -2.682493 0.021922723 0.6003820
## ABLIM2  2.664433 0.023135501 0.6003820
```

## 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.

**Important**: The default behavior of `fastwilcoxGMTall` is to calculate p-values for a two-sided test. However, when performing an enrichment analysis on the gene correlation results from the categorical omnibus test (either ANOVA or Kruskal Wallis), the p-values should be for a one-sided test. This is because, the omnibus tests are one-sided, where only more positive test statistics indicate strength of correlation. As shown below, the user should specify this by setting the `alternative` parameter to `"greater"`. For the pairwise tests, the default two-sided behavior is correct.

**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
# specify alternative = "greater" to get p-values for a one-sided test
allenrichments = fastwilcoxGMTall(getStat(allresults), annotlist, outputGeneVals=T,
                                  alternative="greater")
```

```
## 25 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
## REACTOME_TRANSPORT_OF_SMALL_MOLECULES               0.13212670 0.03847459
## REACTOME_METABOLISM_OF_LIPIDS                       0.12070707 0.06308092
## FLECHNER_BIOPSY_KIDNEY_TRANSPLANT_REJECTED_VS_OK_DN 0.12366310 0.08660459
## DODD_NASOPHARYNGEAL_CARCINOMA_DN                    0.12408759 0.09546697
## NUYTTEN_EZH2_TARGETS_DN                             0.11970803 0.10353506
## ZWANG_TRANSIENTLY_UP_BY_2ND_EGF_PULSE_ONLY          0.09989259 0.10983787
##                                                        p.adj
## REACTOME_TRANSPORT_OF_SMALL_MOLECULES               0.4576578
## REACTOME_METABOLISM_OF_LIPIDS                       0.4576578
## FLECHNER_BIOPSY_KIDNEY_TRANSPLANT_REJECTED_VS_OK_DN 0.4576578
## DODD_NASOPHARYNGEAL_CARCINOMA_DN                    0.4576578
## NUYTTEN_EZH2_TARGETS_DN                             0.4576578
## ZWANG_TRANSIENTLY_UP_BY_2ND_EGF_PULSE_ONLY          0.4576578
```

```
# the third table is for the pairwise comparison between low and medium species
# (with integer labels 2 and 3 respectively)
low_med_enrichments = fastwilcoxGMTall(getStat(pairwise_tables[[3]]), annotlist, outputGeneVals=T)
```

```
## 21 results for annotation set MSigDBpathways
```

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

```
##                                             stat       pval     p.adj
## REACTOME_TRANSPORT_OF_SMALL_MOLECULES   0.18331053 0.01418013 0.2977827
## HOUNKPE_HOUSEKEEPING_GENES             -0.12867647 0.17515792 0.7883529
## KINSEY_TARGETS_OF_EWSR1_FLII_FUSION_UP -0.10150376 0.22785575 0.7883529
## CHEN_METABOLIC_SYNDROM_NETWORK         -0.07977621 0.26286102 0.7883529
## REACTOME_METABOLISM_OF_LIPIDS           0.08625954 0.27461457 0.7883529
## BLALOCK_ALZHEIMERS_DISEASE_UP          -0.08052885 0.29402292 0.7883529
```

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!


Louca, Stilianos, and Michael Doebeli. 2017. "Efficient Comparative Phylogenetics on Large Trees." https://doi.org/10.1093/bioinformatics/btx701.

Ogle, Derek H., Jason C. Doll, Powell Wheeler, and Alexis Dinno. 2022. "FSA: Fisheries Stock Analysis." https://github.com/fishR-Core-Team/FSA.

Paradis, E., and K. Schliep. 2019. "Ape 5.0: An Environment for Modern Phylogenetics and Evolutionary Analyses in {r}" 35: 526–28.

Revell, Liam J. 2012. "Phytools: An r Package for Phylogenetic Comparative Biology (and Other Things)." 3: 217–23.