ERC Analysis Walkthrough

June 20, 2025

Contents

Installing and loading ERC

Make sure to visit the **installation page** to get the code and prerequisite packages before following this tutorial.

ERC Pipeline Walkthrough

You can follow along in R or RStudio, or you can read along in **runERC.R**. First, if you are using RStudio, you need to set your directory. Change the string in `setwd` to the directory your runERC.R file is in, generally the repository you cloned/downloaded.

```
setwd("~/Documents/GitHub/erc")
```

Once you set your directory, you can source the relevant packages and ERC files.

```
require(devtools)
remotes::install_github("ms609/TreeTools")
source("ERC_functions.R")
source("ERC.R")
Rcpp::sourceCpp("cppFuncs.cpp")
```

File setup

To run the workflow, you need to delineate two things: the tree file to read in, and your output file. Find the path for the tree file, and choose a name (and optionally a path) for your output file. Here we set `treefile` and `outputfile` accordingly.

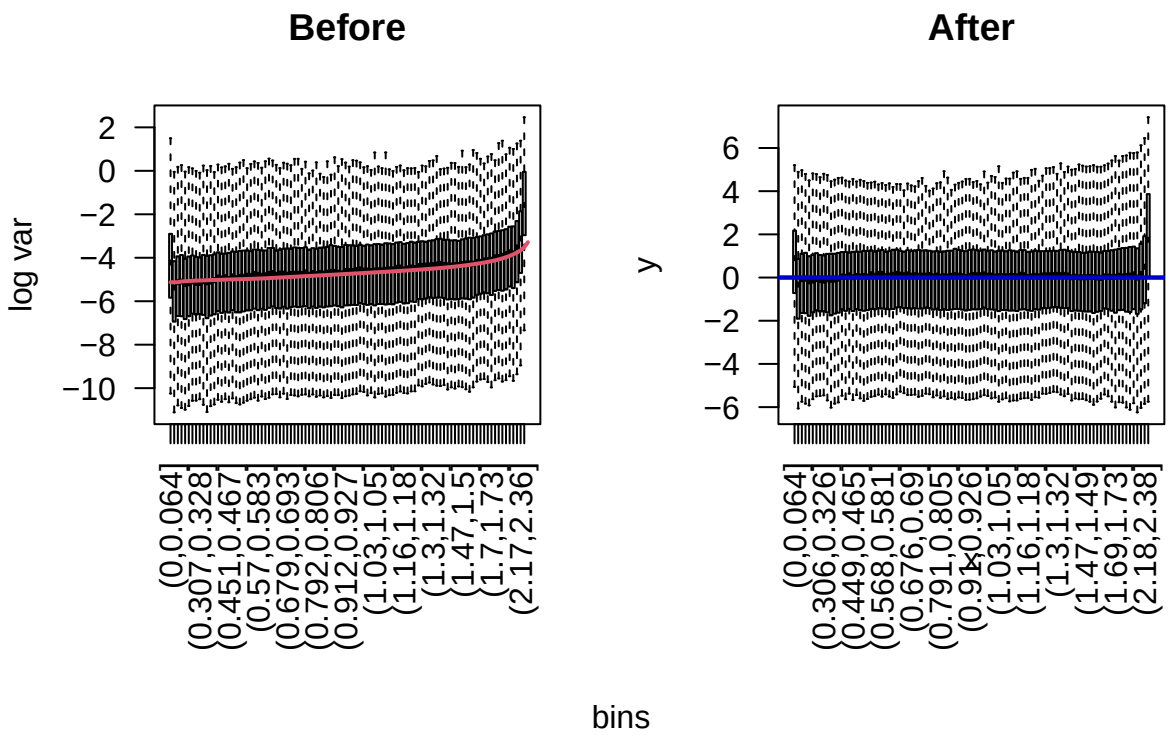
```
treefile = "physical_interaction_paper/domains_trees.tre"
outputfile = "out.RDS"
```

Workflow

Now that you've selected your file names, you can begin running the main functions. For a detailed description of what they do and their parameters, visit the functions page. First, your tree file is read in using `readTrees`. The trees are then transformed (via a square root transform) with `transformPaths`.

```
trees=readTrees(treefile)

compTrees = transformPaths(trees, transform = "sqrt",impute = F)
```



Above is a plot of the tree paths before and after a square root transform.

Next, Relative Evolutionary Rates (RERs) are calculated by `coreGetResiduals`, and finally those rates are formatted into a matrix with `getRMat`. `getAllResiduals` (commented out below) has the same output as running these three together, but here we do not use it because we will later need the compTrees object.

```
compResid = coreGetResiduals(compTrees, n.pcs=0)
residuals = getRMat(compResid, all = T, rmatweights = compTrees$weights)

# Wrapper function: (you still need to separately get compTrees
#                    for the later clusterList function)
# residuals = getAllResiduals(trees, impute=F, n.pcs=0, all=T)
```

Finally, we get a list of gene clusters for the ERC function.

```
clusterList = getClusterList(compTrees)
```

ERC function

Finally, you can compute the ERC values for your trees. You can tune many parameters (also visible on the **functions page**), but the main ones you want to worry about are below: - Here is where you will edit the threshold you want, `minSp` is the number of species two genes have to share - If you only want to run a few genes you can set the parameter `doOnly = c("genea","geneb")` - If you want the plot of the RERs set `plot = T` (I would only recommend doing this for a few genes because it uses up a lot of space)

```
corres=computeERC(residuals, compTrees, clusterListOutput = clusterList,
                  minSp = 15, saveFile = outputfile)
```

```
## Done!
```

Fisher transformation

After you create your ERC matrices, we recommend Fisher transforming them. This creates a single matrix taking into account the two matrices of the `corres` object: the correlation matrix and the matrix of observation/branch counts for each correlation. We also make it symmetrical here, but if your matrix is too large you may just want to make subsets symmetrical as you need them.

```
ft_data = fisherTransform(corres)

#makes the matrix symmetrical
sym_ft = make_symmetric(ft_data)
```

Congratulations for making it to the end! Now with `ft_data` you have the data we usually operate on. Below we have some sample analysis you can do with it.

Next Steps:

Example: examine 10 genes' relations to each other

In this example, we show how to visualize ERC data. We take a sample ten genes, and create a symmetrical ERC matrix of their values (we round the values at the end to make display clearer).

```
genes = c("NSE5_1", "NSE6_3",  "CSE1_3",  "CSE1_1",  "EXO70_1",
          "MCM2_4",  "MDY2_1",  "ATP1_2",  "MCM5_1",  "SEC8_2")
# You could also generate a random sample:
# genes = colnames(ft_data)[sample(1:length(ft_data), 10, replace=FALSE)]


# makes a matrix of the 10 genes against themselves
# (it can be against different genes too)
ft_filtered = betweencomplex(genes,genes,sym_ft)

ft_filtered = round(ft_filtered,3)
#output
ft_filtered
```

```
##         NSE5_1 NSE6_3 CSE1_3 CSE1_1 EXO70_1 MCM2_4 MDY2_1 ATP1_2 MCM5_1 SEC8_2
## NSE5_1   0.000     NA     NA     NA   1.048 -1.169 -0.225  1.713 -0.888 -0.451
## NSE6_3      NA  0.000  2.551 -0.188   1.651 -0.467  0.086  0.464  0.150  1.812
## CSE1_3      NA  2.551  0.000  8.725   2.891  5.159  4.058  2.522  2.134 -0.663
## CSE1_1      NA -0.188  8.725  0.000   4.683  2.385  2.836  2.911  1.212  1.744
## EXO70_1  1.048  1.651  2.891  4.683   0.000  5.282  1.694  1.585  1.511  1.388
## MCM2_4  -1.169 -0.467  5.159  2.385   5.282  0.000  2.161  2.227  3.794  0.655
## MDY2_1  -0.225  0.086  4.058  2.836   1.694  2.161  0.000  0.353 -0.093 -0.597
## ATP1_2   1.713  0.464  2.522  2.911   1.585  2.227  0.353  0.000  2.838  0.350
## MCM5_1  -0.888  0.150  2.134  1.212   1.511  3.794 -0.093  2.838  0.000  1.539
## SEC8_2  -0.451  1.812 -0.663  1.744   1.388  0.655 -0.597  0.350  1.539  0.000
```

Another example: non-Fisher transformed data

We can also use our raw ERC correlation data from before the Fisher transformation. Again, we round to simplify the display.

```
filtered = betweencomplex(genes,genes,corres[["cor"]])

sym = round(make_symmetric(filtered),3)
```

```
#output
sym
```

```
##          NSE5_1 NSE6_3 CSE1_3 CSE1_1 EXO70_1 MCM2_4 MDY2_1 ATP1_2 MCM5_1 SEC8_2
## NSE5_1    1.000     NA     NA     NA    0.091 -0.099 -0.019  0.148 -0.075 -0.045
## NSE6_3       NA  1.000  0.217 -0.016    0.144 -0.040  0.007  0.041  0.013  0.177
## CSE1_3       NA  0.217  1.000  0.328    0.115  0.199  0.274  0.100  0.084 -0.028
## CSE1_1       NA -0.016  0.328  1.000    0.185  0.093  0.194  0.115  0.048  0.073
## EXO70_1   0.091  0.144  0.115  0.185    1.000  0.206  0.119  0.063  0.060  0.059
## MCM2_4   -0.099 -0.040  0.199  0.093    0.206  1.000  0.146  0.087  0.146  0.027
## MDY2_1   -0.019  0.007  0.274  0.194    0.119  0.146  1.000  0.024 -0.006 -0.046
## ATP1_2    0.148  0.041  0.100  0.115    0.063  0.087  0.024  1.000  0.111  0.015
## MCM5_1   -0.075  0.013  0.084  0.048    0.060  0.146 -0.006  0.111  1.000  0.064
## SEC8_2   -0.045  0.177 -0.028  0.073    0.059  0.027 -0.046  0.015  0.064  1.000
```