Introduction to SIDER

Kevin Healy

2017-06-12

This package estimates Trophic Discrimination Factors (TDF) based on the imputation function within the **MCMCglmm** package (http://cran.r-project.org/web/packages/MCMCglmm/index.html) and includes functionality to include the error associated with building phylogenetic trees using the **mulTree** package (https://github.com/TGuillerme/mulTree).

Installation

To install **SIDER**, its dependency **mulTree** must first be installed directly from GitHub. If you have not sone so you can use the following:

```
# Installing devtools
if(!require(devtools)) install.packages("devtools")

# Installing mulTree
#devtools::install_github("TGuillerme/mulTree@releasev1.2.6")
library(mulTree)
```

Following this you can also install **SIDER** directly from GitHub using the following:

```
# Installing SIDER
#devtools::install_github("healyke/SIDER", ref = "release")
#library(SIDER)
```

However, if you are reading this vignette you should only need to load the package

```
library(SIDER)
```

Read in the data

SIDER has a data file which contains discrimination factors for a range of species. It also contains what tissue was used to measure the istopic value and the basic ecology of the species including its diet (herbivore, etc) and whether it is a marine or terrestrial organism. This data is used to run a model and impute an isotopic value for the new species. Lets look at the data.

```
# Read in the data
SIDER.data <- read.csv(file = system.file("extdata",
                                          "SIDER_data.csv",
                                          package = "SIDER"),
                       header = TRUE,
                       stringsAsFactors = FALSE)

# View the first 10 rows of the data frame
head(SIDER.data)
```

```
##             species     habitat taxonomic.class tissue diet.type
## 1 Rattus_norvegicus terrestrial        mammalia  liver herbivore
## 2 Rattus_norvegicus terrestrial        mammalia  liver carnivore
```

```
## 3 Rattus_norvegicus terrestrial        mammalia  liver  omnivore
## 4 Rattus_norvegicus terrestrial        mammalia  liver  omnivore
## 5 Rattus_norvegicus terrestrial        mammalia  blood herbivore
## 6 Rattus_norvegicus terrestrial        mammalia  blood carnivore
##   source.iso.13C source.iso.15N delta13C delta15N
## 1          -25.3            4.8      3.1      1.2
## 2          -16.2           12.3      3.1      0.6
## 3          -24.5            7.1      1.5      3.2
## 4          -16.7            7.4      2.2      3.0
## 5          -25.3            4.8      1.5      3.6
## 6          -16.2           12.3      0.7      3.7
```

```
# Read in the phylogenetic information
# The mammal trees
mammal_trees <- ape::read.tree(system.file("extdata",
                                           "3firstFritzTrees.tre",
                                           package = "SIDER"))
# The bird trees
bird_trees   <- ape::read.tree(system.file("extdata",
                                           "3firstJetzTrees.tre",
                                           package = "SIDER"))

# Combine them together using the tree.bind function from the mulTree package
combined_trees <- mulTree::tree.bind(x = mammal_trees,
                                     y = bird_trees,
                                     sample = 2,
                                     root.age = 250)
```

As may we want to include the error associated with building phylogenies into our analysis we take a sample of the possible trees (See Healy *et al.* 2014 (http://rspb.royalsocietypublishing.org/content/281/1784/20140298). In this case we use a small subset of a mammals and birds tree distribution and combine them at a rooted age of 250 Mya (million years ago) and sample of two of the resulting trees.

Testing the new data: `recipeSider`

In order to estimate a trophic enrichment factor for a new species we need to check that the species is already present in our phylogeny and check what data is available for the new species. `recipeSider` checks for the presence of the following data: **`tissue`** (`blood`, `claws`, `collagen`, `feather`, `hair`, `kidney`, `liver`, `milk`, `muscle`); **`habitat`** (`terrestrial`, `marine`); and **`diet.type`** (`carnivore`, `herbivore`, `omnivore`, `pellet`).

```
# Checking the data for the species we want to estimate TEF for (Meles meles)
new.data.test <- recipeSider(species = "Meles_meles",
                             habitat = "terrestrial",
                             taxonomic.class = "mammalia",
                             tissue = "blood",
                             diet.type = "omnivore",
                             tree = combined_trees)
```

If the species is not present in the phylogeny already (say the Komodo dragon - *Varanus komodoensis*), or that some values are missing (say **`tissue`**), we get an error message to indicate what is missing from our data.

N.B, the following code will throw a stop error if evaluated.

```
# Some incomplete dataset
new.data.test <- recipeSider(species = "Varanus_komodoensis",
```

```
                                habitat = "terrestrial",
                                taxonomic.class = "mammalia",
                                tissue = "NA",
                                diet.type = "omnivore",
                                tree = combined_trees)
```

The `recipeSider` function also formats the data for the new species data so that it can be combined with the data already available within the package using the `prepareSider` function.

Formatting the new data: 'prepareSider'

We now need to format the data by combining both the isotopic data already available within the package and the data from the new species. We also include what isotope we want to estimate a trophic discrimination value for (either `carbon` or `nitrogen`).

```
tdf_data_c <- prepareSider(new.data.test,
                           isotope_data,
                           combined_trees,
                           "carbon")
```

N.B. `isotope_data` is the isotopic dataset already implemented in **`SIDER`**.

We now have a `mulTree` class object, which is required by the imputation analysis. It contains the matched phylogenies, in this case two phylogenies:

```
tdf_data_c$phy
```

```
## 2 phylogenetic trees
```

and a dataset containing the TDF and related data with the new species for which you want to estimate a trophic enrichment factor at the top with a `NA` for either `delta13C` or `delta15N` depending on isotope.

```
head(tdf_data_c$data)
```

```
##                sp.col     habitat taxonomic.class tissue diet.type
## 1         Meles_meles terrestrial        mammalia  blood  omnivore
## 120 Rattus_norvegicus terrestrial        mammalia  liver herbivore
## 2   Rattus_norvegicus terrestrial        mammalia  liver carnivore
## 3   Rattus_norvegicus terrestrial        mammalia  liver  omnivore
## 4   Rattus_norvegicus terrestrial        mammalia  liver  omnivore
## 5   Rattus_norvegicus terrestrial        mammalia  blood herbivore
##     source.iso.13C delta13C           animal
## 1               NA       NA       Meles_meles
## 120          -25.3      3.1 Rattus_norvegicus
## 2            -16.2      3.1 Rattus_norvegicus
## 3            -24.5      1.5 Rattus_norvegicus
## 4            -16.7      2.2 Rattus_norvegicus
## 5            -25.3      1.5 Rattus_norvegicus
```

Running the analysis: 'prepareSider'

With the data formatted as a `mulTree` object we can decide on a model which will impute the new species estimate. In this case we will run the full model to estimate delta13C with the fixed factors of diet type and habitat type.

```
formula.c <- delta13C ~ diet.type + habitat
```

and random terms that includes the `animal` term which is required to include phylogeny into the analysis:

```
random.terms <- ( ~ animal + species + tissue)
```

As we rely on Bayesian imputation to estimate the missing value we also need to specify a prior, in this case we use a non-informative prior

```
prior <- list(R = list(V = 1, nu=0.002),
              G = list(G1=list(V = 1, nu=0.002),
                       G2=list(V = 1, nu=0.002),
                       G3=list(V = 1, nu=0.002)))
```

along with the number of iterations to run the chain (`nitt`), the burn-in (`burnin`), the sampling thinning (`thin`), the number of chains to run (`no.chains`) as recommended in the **`MCMCglmm`** guidelines (https://cran.r-project.org/web/packages/MCMCglmm/vignettes/CourseNotes.pdf)}.

```
nitt <- c(10)
burnin <- c(1)
thin <- c(1)
parameters <- c(nitt, thin, burnin)
no.chains <- c(2)
```

> N.B. These settings are only for testing and running a quick example, you can use the following parameters to produce results that pass convergence and avoid autocorrelation

```
nitt <- c(1200000)
burnin <- c(200000)
thin <- c(500)
no.chains <- c(2)
```

We need to check that our MCMC chains are converging so we use the Gelman and Rubin diagnostic to check the convergence and also check that the estimated parameters have an effective sample size >1000.

```
convergence =  c(1.1)
ESS = c(1000)
```

As the function exports the model output to avoid memory issues within `R` when running over multiple phylogenies make sure you have set the working directory to somewhere appropriate. In this example, we will use the default temporary directory determined by `tempdir()`, but ordinarily, this would be a folder of your own choosing located somewhere sensible on your machine.

> N.B. We ***strongly*** advise that you do not use this `tempdir` in your own analyses!

In this example, we store the original working directory to `origwd` so that we can return the `R` session to the original working directory after using the temporary one. Then we can finally we can run the analysis using the imputeSider function. This model will normally take approximately 5 minutes, however for brevity we will run a much shorter chain.

```
origwd <- getwd() # get the current, or original working directory.
 setwd(tempdir()) #
TDF_est.c <- imputeSider(mulTree.data = tdf_data_c,
                         formula = formula.c,
                         random.terms = random.terms,
                         prior = prior,
                         output = "test_c_run",
                         parameters = parameters,
                         chains = no.chains,
```

```
                          convergence =  convergence,
                          ESS = ESS)
```

```
## Loading required package: MCMCglmm

## Loading required package: Matrix

## Loading required package: coda

## Loading required package: ape

##
## 2017-06-12 - 15:13:02: MCMCglmm performed on tree 1
## Convergence diagnosis:
## Effective sample size is > 1000: FALSE
## 9; 9; 9; 9; 25.91214; 9; 9; 9; 9; 9; 57.69882; 9; 9; 9; 7.47497; 9; 9; 2.435133
## C1.Sol.(Intercept), C1.Sol.diet.typeherbivore, C1.Sol.diet.typeomnivore, C1.Sol.diet.typepellet, C1.S
## All levels converged < 1.1: FALSE
## 1.606104; 1.191659; 0.9246276; 3.453733; 4.774618; 2.376328; 0.9822681; 11.21836
## Individual models saved as: test_c_run-tree1_chain*.rda
## Convergence diagnosis saved as: test_c_run-tree1_conv.rda
##
## 2017-06-12 - 15:13:02: MCMCglmm performed on tree 2
## Convergence diagnosis:
## Effective sample size is > 1000: FALSE
## 9; 9; 9; 9; 9; 55.54043; 9; 9; 2.748637; 9; 9; 9; 9; 9; 9; 9; 9; 2.861955
## C1.Sol.(Intercept), C1.Sol.diet.typeherbivore, C1.Sol.diet.typeomnivore, C1.Sol.diet.typepellet, C1.S
## All levels converged < 1.1: FALSE
## 5.243942; 7.599232; 1.467542; 4.697307; 12.00127; 24.9581; 6.083791; 10.35909
## Individual models saved as: test_c_run-tree2_chain*.rda
## Convergence diagnosis saved as: test_c_run-tree2_conv.rda
##
## 2017-06-12 - 15:13:02: MCMCglmm successfully performed on 2 trees.
## Total execution time: 0.5123038 secs.
## Use read.mulTree() to read the data as 'mulTree' data.
## Use summary.mulTree() and plot.mulTree() for plotting or summarizing the 'mulTree' data.
```

```
###Now lets have a look at the files imputeSider has saved to the current working directory
list.files(pattern = "test_c_run")
```

```
## [1] "test_c_run-tree1_chain1.rda" "test_c_run-tree1_chain2.rda"
## [3] "test_c_run-tree1_conv.rda"   "test_c_run-tree2_chain1.rda"
## [5] "test_c_run-tree2_chain2.rda" "test_c_run-tree2_conv.rda"
```

`imputeSider` now runs the selected amount of chains (`no.chains`) for each of the sampled phylogeny and exports the resulting MCMC chains to the working directory. Hence in this case we have run two chains for each of the two sampled trees so there should be four files in your working directory ending with something like `run-tree1_chain2.rda`.

These are the full `MCMCglmm` model outputs for each of the chains run and can be imported back into the `R` for full inspection if required using the `read.mulTree` function. (Note this will require saving your files to a permanent working directory) Notice also that the two other files ending with something similar to `run-tree2_conv`. These give a description of the convergence diagnostics of the chains for each tree. The results of these diagnostics for each tree are also printed in `R` after running `imputeSider` returning whether the Effective sample size exceed ESS for all estimated parameters (with the number for each parameter given as a list), and whether al chains converged for each parameter.

All these models can be separately imported into `R` using `read.mulTree()`. However, since we are only

interested in the estimated TDF for our species `imputeSider` only imports the imputed posterior distribution of the estimated TDF for our species.

The contents of `TDF_est.c$tdf_global` is the posterior estimate of the imputed TDF aggregated across all chains in the model run (`TDF_est.c$tdf_estimates` is a list containing each posterior chains separately). As a `mcmc` class obect, the many functions of the `coda` package provide options for summarising and plotting the distribution.

```
# Explore the names of the list created by SIDER::imputeSider
names(TDF_est.c)
```

```
## [1] "tdf_estimates" "tdf_global"
```

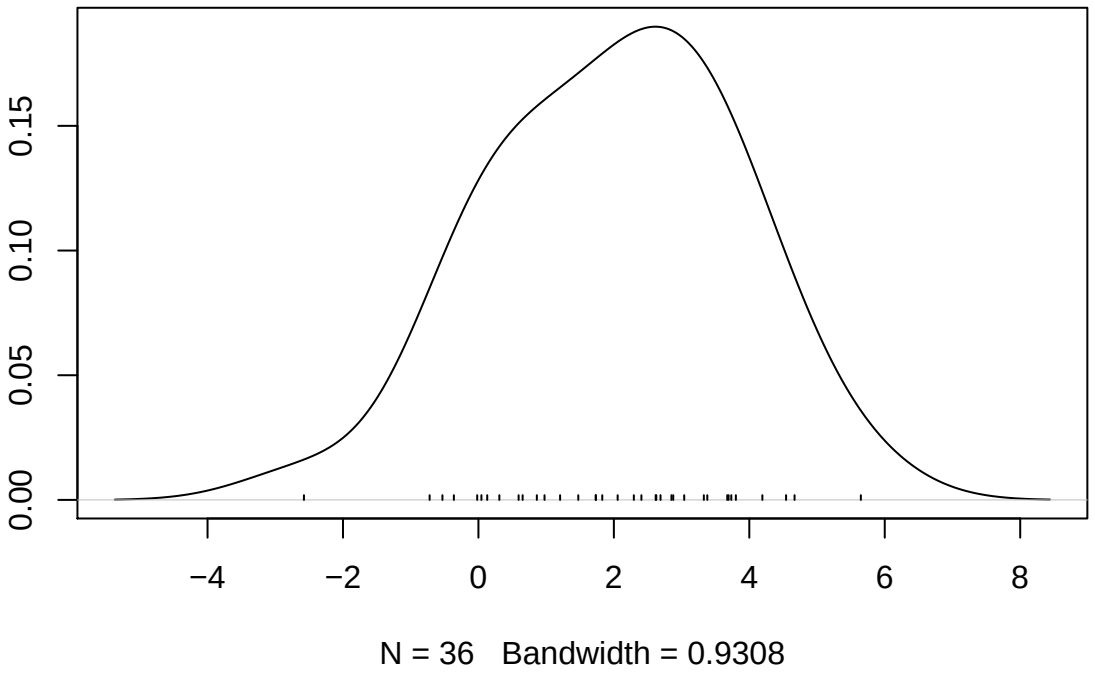
```
# Calculate summary statistics of the posterior.
# Specifically, the mean and standard deviation would be
# taken from here and used in a mixing model analysis using
# MixSIAR, MixSIR or SIAR for example.
summary(TDF_est.c$tdf_global)
```

```
##
## Iterations = 1:36
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 36
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##           Mean             SD       Naive SE Time-series SE
##         1.9848         1.7980         0.2997         0.5627
##
## 2. Quantiles for each variable:
##
##    2.5%     25%     50%     75%   97.5%
## -0.9527  0.6380  2.1752  3.3414  4.7905
```

```
# Credible intervals and the mode of the posterior are obtained
# using the hdrcde package
hdrcde::hdr(TDF_est.c$tdf_global, prob = c(50, 95, 99))
```

```
## $hdr
##           [,1]     [,2]
## 99% -2.1915099 6.138582
## 95% -0.9348443 5.020190
## 50%  0.7549744 3.536896
##
## $mode
## [1] 2.209865
##
## $falpha
##        1%        5%       50%
## 0.0350264 0.0781779 0.1431642
```

```
# You can also create density plots of the posterior
coda::densplot(TDF_est.c$tdf_global)
```




```
## logical(0)
```