ShiVa

Wensha Zhang, Lam Si Tung Ho, Toby Kenney

November 2, 2022

The main useage of this package is to detect the evolutionary shifts in both variance and optimal value based on minimizing the loss function with L1 penalty.

To demonstrate the usage of the package, we will simulate a dataset and apply the main functions on the dataset. We first generate a dataset with shifts in variance. The main input of our functions are a phylogenetic tree for analysis and a vector of univariate trait values. In the codes below, we generate a phylogenetic tree (tree) with 80 tip nodes. And we generate the trait values using the OU model and put a shift in its variance parameter (σ^2) on branch 2 ($\Delta\sigma^2 = 10$). We will show in the following functions, how we can detect the shift.

```
library(ape)
library(MASS)
library(PIGShift)
library(ShiVa)
alpha = 1
sigma2_0 = alpha*2
true_shifts_var = c(2) #shift configuration in variance
size_var = 10 #shift size in variance
true_shifts_mean = 0 #shift configuration in mean
size_mean = 0 #shift size in mean
set.seed(123)
# generate a tree
tree = rcoal(80)
tree$edge.length = tree$edge.length/max(node.depth.edgelength(tree))
# generate design matrix
X = generate_design_matrix(tree,type='simpX')
V = OU.vcv(tree,alpha)
tb = node.depth.edgelength(tree)[tree$edge[,1]]
q = exp(-2*alpha*max(node.depth.edgelength(tree)))/(2*alpha)*(1-exp(2*alpha*tb))
# simulate shifts in variance
gamma = rep(0,ncol(X))
gamma[true_shifts_var]  = size_var
Sigma = V*sigma2_0 +  X%*%diag(gamma)%*%t(X)*V+ X%*%diag(gamma*q)%*%t(X)
```

```
# simulate shifts in mean
beta = rep(0,ncol(X))
beta[true_shifts_mean] = size_mean
# generate simulated data
eps = mvrnorm(n = 1, mu = rep(0,nrow(X)), Sigma = Sigma)
Y = X%*%beta+eps
```

`get_mean_var_shifts(Y, tree, alpha, lambda1, lambda2)`

Performs the estimation of shifts in variance and optimal value given fixed lambda1 and lambda2 (the degree of penalty for shifts in optimal value and for shifts in variance). alpha is the selection strength parameter. Now we apply the function on the generated dataset.

Example:

```
result = get_mean_var_shifts(Y, tree, 1, 1, 0.02)
sv_mean = (1:ncol(X))[result$beta!=0]
sv_var = (1:ncol(X))[result$gamma!=0]
sv_mean # shifts in optimal values

## integer(0)

sv_var # shifts in variance

## [1] 2
```

`fit_OU_mean_var(tree, Y ,alpha, sv_mean, sv_var)`

Provides the estimation of shift sizes for shifts in optimal value and shifts in variance given the shift positions. The shift sizes of the previous function are biased with L1 penalty. Therefore we need to refit the model with no penalty to get the unbiased estimation of shift sizes.

Example:

```
OModel = fit_OU_mean_var(tree, Y ,1, sv_mean, sv_var)
OModel$gamma[sv_var]

## [1] 4.50441
```

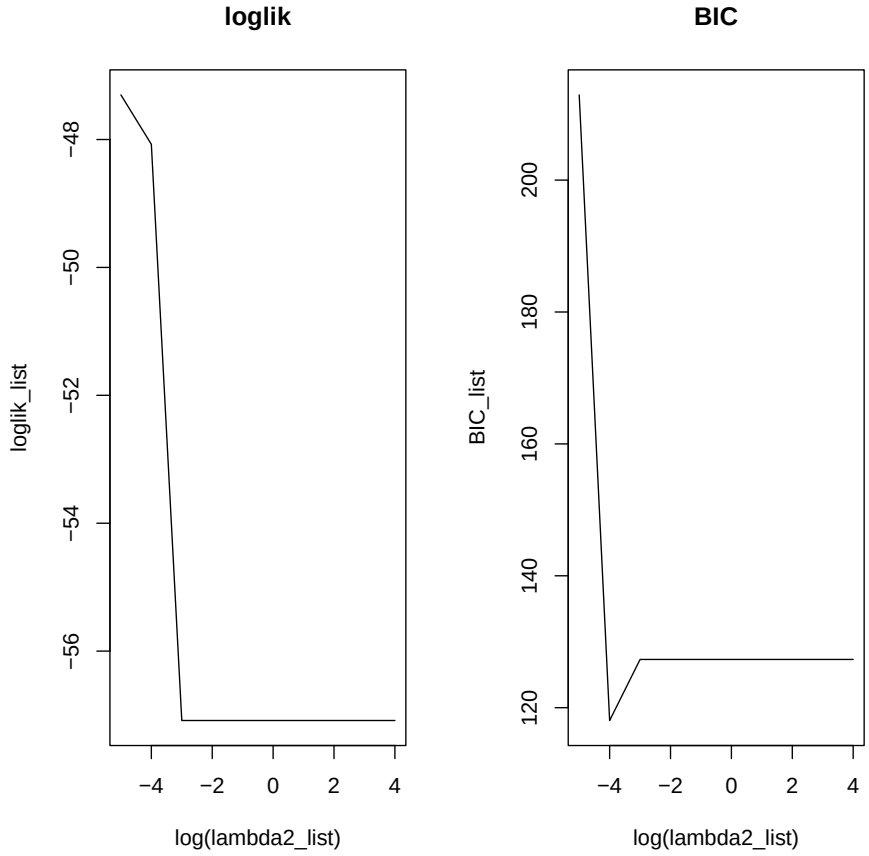
To tune the two hyperparameters, the function

`get_mean_var_shifts_model_selection(Y,tree,alpha,lambda1_list,lambda2_list)`

Provides the estimation of shifts in variance and optimal value given two lists of regularization penalty parameters. `get_mean_var_shifts` only gives estimate when lambda1 and lambda 2 doesn't need to be tuned. And this function will perform the model selection over different regularization penalty values. lambda1_list is a list of lambda1 values. It can be NULL, and it will be generated by "glmnet" if it is NULL. lambda2_list is a list of lambda2 values and it cannot be NULL.

Example:

```
result = get_mean_var_shifts_model_selection(Y,tree,1,NULL,exp(1:10-6))
```




```
sv_mean = (1:ncol(X))[result$beta!=0]
sv_var = (1:ncol(X))[result$gamma!=0]
sv_mean # shifts in optimal values

## integer(0)

sv_var # shifts in variance

## [1] 2

result$gamma[sv_var] # shift sizes

## [1] 4.50441

result$lambda1
```

```
## [1] 0.7973339
```

```
result$lambda2
```

```
## [1] 0.01831564
```