

Structural EM

Introduction

This short document presents how to use the structural EM algorithm to find the maximum likelihood binary Gaussian latent tree. The code easily works for big trees (> 500 leaves) but for the presentation purposes here we compile a version with only 5 leaves.

```
devtools::install_github("pzwiernik/StructuralEM")
library(StructuralEM)
m <- 5 # number of leaves
N <- 200 # sample size
```

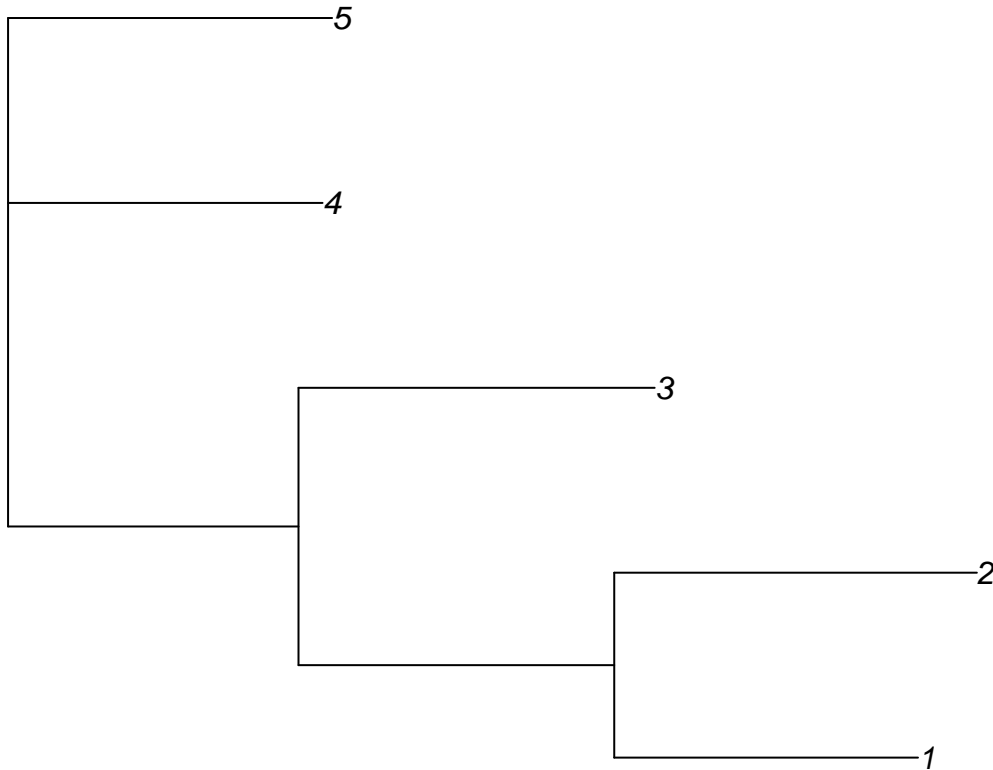
- First we work with syntetic data for some phylogenetic tree. This is done by generating a random tree with random edge correlations. We pick edge correlations uniformly from the interval $[0.5, 1]$ to obtain more stable computations.

```
# generate a random tree and some data
Ttr <- ape::rtree(m,rooted=FALSE,min=-log(1/2))
Ttr$tip.label <- 1:m
Str <- get.corr0(Ttr)
Rtr <- Str[1:m,1:m]
dat <- MASS::mvrnorm(N, rep(0,m), Rtr)
dat <- normalize.data(dat)
```

- The following is the visualization of the true tree from which the data will be generated.

```
plot(Ttr,main='The true tree')
```

The true tree



- We pick a starting point for our algorithm using distance based methods to find the best starting point for the structural EM algorithm. To this end we use the `FastME()` function in the `ape` package (see explanation there). For small trees possibly a better solution could be found.

```
# quickly compute the starting point
D <- get.dist(cor(dat))
T0 <- ape::fastme.bal(D)
#since the algorithm may produce negative lengths
T0$edge.length <- (T0$edge.length>0)*T0$edge.length
```

- Now we toggle between the E-step and the M-step. The latter is given by the Chow-Liu algorithm with an additional fix explained in papers on the structural EM algorithm.

```
ptm <- proc.time()
res <- strEM(dat,T0,tol=1e-6)
print(proc.time() - ptm)
```

```
##      user  system elapsed
##    1.791    0.039    1.838
```

```
R <- res$Sig[1:m,1:m]
# log-likelihood value at the true data generating distribution
like(Rtr,dat)
```

```
## [1] -4.935038
```

```
# log-likelihood value at the reported local maximum  
like(R,dat)
```

```
## [1] -4.882745
```

- We compare the true correlation matrix with the estimated one.

```
print("True correlation matrix")
```

```
## [1] "True correlation matrix"
```

```
Rtr
```

```
##           1           2           3           4           5  
## 1 1.00000000 0.16917420 0.07416122 0.03820449 0.03724460  
## 2 0.16917420 1.00000000 0.06338631 0.03265375 0.03183332  
## 3 0.07416122 0.06338631 1.00000000 0.07709817 0.07516106  
## 4 0.03820449 0.03265375 0.07709817 1.00000000 0.18220910  
## 5 0.03724460 0.03183332 0.07516106 0.18220910 1.00000000
```

```
print("The MLE correlation matrix")
```

```
## [1] "The MLE correlation matrix"
```

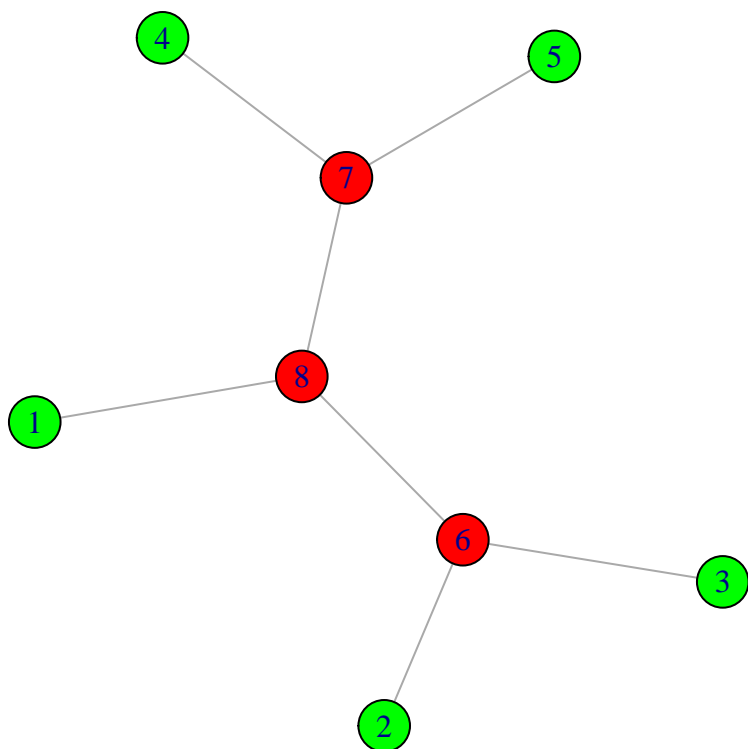
```
res$Sig[1:m,1:m]
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]  
## [1,] 1.00000000 0.12569446 0.017972659 0.003694295 0.01447923  
## [2,] 0.125694461 1.00000000 0.142986880 0.029391072 0.11519383  
## [3,] 0.017972659 0.14298688 1.000000000 0.004202538 0.01647121  
## [4,] 0.003694295 0.02939107 0.004202538 1.000000000 0.25514450  
## [5,] 0.014479226 0.11519383 0.016471206 0.255144496 1.00000000
```

```
#abs(Rtr-res$Sig[1:m,1:m])
```

- The following is the igraph plot of the estimated tree structure.

```
igraph::plot.igraph(res$tree)
```



In the paper we evaluate this method by running it many times and counting the number of times the correct tree is identified.