How to handle mis-aligned shapes with 'auto3dgm'

Chris Glynn

December 12, 2013

1 Introduction

It is possible that the automated output will not always correctly align the shapes in your dataset. When this happens, there are steps you can take to visually identify the mis-aligned shapes based on the output from align_shapes. Once the shapes are identified, it is possible to split apart the minimum spanning tree where mis-alignments occur and rebuild it based on better rotations. It is left to the user to identify the "better" rotations, but 'auto3dgm' provides two functions that will help. This document walks through an example of this case and provides commented R code for steps to take.

2 R Script Example

Compute the original alignment.

```
rm(list=ls())
library(auto3dgm)

Bm_path = "/Users/christopherglynn/Dropbox/Shape_Alignment/Astragali"
Levels=c(32,64)
Ids = c('2A', '3A', '5A', '6A', '7A', '8A', '10A', '12A', '14A', '14B', '14C',
'15A', '17A', '19A', '22A', '23A', '24A', '26A', '28A', '29A', '35A', '35B',
'35C', '40A', '40B', '45A', '45B', '46A', '54A', '54B', '54C', '54D', '55A',
'55B', '55C', '55D', '55E', '55F', '56A', '56B', '56C', '56D', '56E', '57A',
'57B', '57C', '57D', '57E', '57F', '58A', '58B', '58C', '59A')

Names = c('Alouatta', 'Aotus', 'Ateles', 'Cacajao', 'Callicebus', 'Callithrix',
'Cebus', 'Cheirogaleus', 'Cynocephalus1', 'Cynocephalus2', 'Galeopterus',
'Daubentonia', 'Eulemur', 'Hapalemur', 'Lemur', 'Lepilemur', 'Microcebus',
'Nycticebus', 'Perodicticus', 'Pithecia', 'Tarsius1', 'Tarsius2', 'Tarsius3',
'Ptilocercus1', 'Ptilocercus2', 'Tupaia1', 'Tupaia2', 'Loris', 'Lepus1',
'Lepus2', 'Sylvilagus', 'Ochotona', 'Erethizon', 'Coendou', 'Marmota', 'Sciurus'
,'Aplodontia', 'Allactaga', 'Tenrec', 'Setifer', 'Hemicentetes', 'Echinops',
'Potamogale', 'Erinaceus', 'Hemiechinus', 'Suncus', 'Crocidura', 'Desmana',
'Solenodon', 'Potos', 'Arctictis', 'Nasua', 'Petrodromus')

Dataset = "astragali_dataset"
```

```
Run="astragali_branch_2"

FULL = align_shapes(Bm_path, Levels, Ids, Names, Dataset, Run,Initialize=F,
Return=c("ds","ga","pw_rotations") )
ds = FULL[[1]]
ga_full=FULL[[2]]
pa=FULL[[3]]
```

Specify a directory to store analysis in attempting to find a fix. Use the 'alignment.off', the 'map.off' and the 'MST.jpg' output files from the above alignment to visually identify improper alignments. Make note of these shapes. It is possible that an entire branch of a tree is misaligned. In this example, there are two mis-aligned branches. The first branch includes shapes with indices 33,34,35,36,37,39,40. The second branch includes shapes with indices 47 and 48.

```
fix_dir = paste(ds$ms$output_dir,"/Alignment_Fix", sep="")
unlink(fix_dir, recursive=TRUE)
dir.create(fix_dir)


A_index =c(33:37, 39, 40)
B_index = c(47,48)

Base_Tree_index = 1:ds$n
Base_Tree_index = Base_Tree_index[-c(A_index,B_index)]
```

There may be an alignment between the mis-aligned files. It might make sense to combine the two branches. It is ideal to combine the branches together if possible so that reconnecting to the base tree is easier.

Use the branch_pw_distances function to compute the procrustes distance between elements of each branch and output a pairwise alignment file for visual comparison.

```
fix_dir_AB = paste(fix_dir,"AB",sep="/")
k=1
AB = branch_pw_distances(A_index,B_index,ds,pa, k, fix_dir_AB)
write.csv(file=paste(fix_dir_AB,"proc_d",sep=""),AB)

#now consider pooling the groups.  AB.  In this case, it does not work.
#AB_index = c(A_index,B_index)
#Run ="teeth_run_AB"
```

Re-compute the MST and alignment for branch A. Since it has more than 2 elements, we should consider a new alignment and tree structure for this branch.

```
Run = "Branch_A"
ga_A = align_shapes( Bm_path, Levels, Ids[A_index], Names[A_index], Dataset,
Run,Initialize=F, Return=c("ga") )
```

```
ga_A=ga_A[[1]]
```

Again use 'branch_pw_distances' to find the best connection point between branch A and the base tree. The connection point will be the pair of best aligned shapes. This is specified by the user after examining all pairwise rotation files that are generated by 'branch_pw_distances'.

In the example below, the best connection point is row 10 of AB. AB will contain the pairs ordered from smallest procrustes distance.

Similarly, find the connection point for branch B.

```
fix_dir_A_Base = paste(fix_dir,"A_Base",sep="/")
A_Base = branch_pw_distances(A_index, Base_Tree_index,ds,pa, k, fix_dir_A_Base)
write.csv(file=paste(fix_dir_A_Base,"proc_d",sep=""),A_Base)
A_connection=as.numeric(A_Base[10,2:3])


fix_dir_B_Base = paste(fix_dir,"B_Base",sep="/")
B_Base = branch_pw_distances(B_index, Base_Tree_index,ds,pa,k, fix_dir_B_Base)
write.csv(file=paste(fix_dir_B_Base,"proc_d",sep=""),B_Base)
B_connection = as.numeric(B_Base[2,2:3])
```

Find the rotation between the connection pair for both A branches. Then compute the global alignment rotation matrices for each element of branch A.

```
#find pairwise matrix.
if(A_connection[1]<A_connection[2]){
  #this case is where shape in misaligned group is rotated to shape in base
  #group
  #Which is why I invert the matrix
A_connection_rotation = solve(pa$R[[min(A_connection)]][[max(A_connection)]])
}else{
  #this is case where shape in base group is rotated to shape in misaligned
  #group.
A_connection_rotation = pa$R[[min(A_connection)]][[max(A_connection)]]
}


ga=list(R=list())
#Now insert the pairwise alignment fix and write-alignment file

list.add = function(List,obj){
  List[[length(List)+1]]<-obj
  return(List)
}

for (j in 1:ds$n){
```

```
    ga$R = list.add(ga$R, diag(3))
}

for (j in Base_Tree_index){
    ga$R[[j]] = ga_full$R[[j]]
}

for (j in A_index){
    ga$R[[j]] = ga$R[[ A_connection[2] ]]%*%A_connection_rotation%*%
    ga_A$R[[ which(A_index==j) ]]
}



## Compute rotation matrix for graphical outputs
k = 2;
theta = pi/2
rotation_matrix = matrix(c(cos(theta), -sin(theta), 0, sin(theta), cos(theta),
0, 0, 0, 1), nrow=3, byrow=T)
rotation_matrix = rotation_matrix%*%
matrix(c(0,0,1,0,-1,0,1,0,0), nrow=3, byrow=T)%*%t(ds$shape[[1]]$U_X[[k]])
```

Now compute the pairwise rotation where the B branch will reconnect to the tree. Compute the global rotation matrices for the elements of branch B.

```
#Still need a way of connecting group B.


if(B_connection[1]<B_connection[2]){
  #this case is where shape in misaligned group is rotated to shape in base
  #group
  B_connection_rotation = solve(pa$R[[min(B_connection)]][[max(B_connection)]])
}else{
  #this is case where shape in base group is rotated to shape in misaligned
  #group which is why i take the inverse.
  B_connection_rotation = pa$R[[min(B_connection)]][[max(B_connection)]]
}

ga_B = list(R=list())

for (j in 1:length(B_index)){
  ga_B$R = list.add(ga_B$R, diag(3))
}


#Since B_index[2] is NOT the connection point, we need to find its
```

```
#rotation to the connection point.If B_index[2] were the connection point,
#then it would be ga_B$R[[1]] = solve(...)

ga_B$R[[2]]=solve(pa$R[[min(B_index)]][[max(B_index)]])

for (j in B_index){
  ga$R[[j]] = ga$R[[ B_connection[2] ]]
  %*%B_connection_rotation%*%ga_B$R[[ which(B_index==j) ]]
}
```

Write the new alignment file.

```
varargin = list(1:ds$n, 10, rotation_matrix,3.0,1)
file = paste(fix_dir,"/fixed_alignment.off", sep="" )
write_off_global_alignment(file, ds, ga, varargin)
```