Tim's Take on Taxonomic Hierarchy Modeling

Tim!

5/11/2022

Tim's take on the model

From Thorson et al and Thorson (2020), but basic model for the vector of life history traits corresponding to some child taxonomic group, g is:

$$g \sim MVN(x_{p(g)}, \Sigma_{l(g)}$$

where $p(g)$ is the parent taxonomy level for child group g, and $\Sigma_{l(g)}$ is the evolutionary covariance, and $l(g)$ indicates the taxonomic level of gropu g. For reference $l(g) = 1$ refers to Class.

The estimation procedure works by estimating a vector of elements of the lower triangular matrix, $\mathbf{L}_\Sigma$,so the evoluationary covariance matrix equals:

$$\Sigma_{l(g)} = \lambda_l \left(\mathbf{L}_\Sigma \mathbf{L}_\Sigma^T + \mathbf{D} \right)$$

where $\mathbf{D}$ is a diagonal matrix the additional and independent evolutionary variance in traits (σ_j^2). The parameter $\lambda_{l(g)}$ is the relative amount of of covariance explained by taxonomic level l relative to Taxonomic level Class (e.g. $\lambda_{l(g)=1} = 1$).

Now is where I get a little lost because there is an additional covariance matrix for the residuals:

$$\tilde{y} \sim MVN(x_{g(l)}, \mathbf{V})$$

where $\tilde{y}$ is the "augemented data for the ith study.

Fixed effects are the evolutionary covariance ($\mathbf{L}_\Sigma$), and the residual covariance ($\mathbf{V}$)

In the TMB code, I see something like this (with my annotations):

```
// Probability of random effects
  vector<Type> Parent_j( n_j );  # create vector of parent mean trait alues
  vector<Type> Prediction_j( n_j ); # create vector to hold predicted values of traits
  vector<Type> Deviation_j( n_j ); # MVN only works on 0 means, so create a vector of deviations
  matrix<Type> tmpCov_jj( n_j, n_j ); # not entirely sure, see below
  for( int g=0; g<n_g; g++ ){  # loop through each taxonomic grouping
    for( int j=0; j<n_j; j++ ){ # loop through trait
      if( PC_gz(g,1)==0 ) Parent_j(j) = alpha_j(j); # if group = Class, then look up values in alpha
      if( PC_gz(g,1)>=1 ) Parent_j(j) = betainput_gj(PC_gz(g,0),j); # otherwise, look up values in beta
      Prediction_j(j) = Parent_j(j); # assign prediction to equal value of parent
    }
    for( int j=0; j<n_j; j++ ){
      Deviation_j(j) = betainput_gj(g,j) - Prediction_j(j); # deviations, duh
    }
    tmpCov_jj = Cov_jj * exp(cov_logmult_z(PC_gz(g,1))); # ok, this is where I have some trouble Cov_jj
    if( Options_vec(4)==false ){
      jnll_comp(PC_gz(g,1)) += MVNORM( tmpCov_jj )( Deviation_j ); # get the NLL of the vector of j tra
```

```
    }
  }
```

I can see that we do indeed estimate two separate vectors L_v and obsL_v, which I'm guessing are the two different things. The obsCov seems to rely on this:

```
  matrix<int> RAMobs( 0, 5 );
  matrix<Type> obsCov_jj( n_j, n_j );
  obsCov_jj = cov_matrix( obsL_z, RAMobs, Options(0), n_j, Options_vec(0) );
```

The rest is pretty straitforward - using the observations ($Y_{i,j}$):

```
// Probability of data
matrix<Type> Yhat_ij( n_i, n_j ); # get matrix of predicted values for each taxa group i
for( int i=0; i<n_i; i++){ # loop through observations
  if( Options_vec(4)==false ){ # this is a toggle to turn off taxonomic hierarchy
    Yhat_ij.row( i ) = beta_gj.row( g_i(i) );
  }else{
    Yhat_ij.row( i ) = alpha_j;
  }
  jnll_comp(5) += MVNORM( obsCov_jj )( Ycomplete_ij.row(i) - Yhat_ij.row(i) ); # get NLL
}
```