Supporting Information to:

"Sexual dimorphism in primate aerobic capacity: A phylogenetic test."

Computer code for phylogenetic paired *t*-test.

The following is MATLAB code for the phylogenetic paired *t*-test described in Lindenfors, Revell, and Nunn, "Sexual dimorphism in primate aerobic capacity: A phylogenetic test."

To execute the code, one must first obtain the matrix **C**, an $N \times N$ matrix (for *N* species) containing the tree length from the root to each *i*th tip in the diagonal, and the height above the root of the common ancestor of each pair of species *i* and *j* in each position *i*,*j* (e.g., Rohlf 2001; Revell and Harmon 2008). The function takes **C**, two $N \times 1$ vectors with the phenotypes ($\mathbf{x}_1$ and $\mathbf{x}_2$), two $N \times 1$ vectors with the corresponding standard errors (SEx_1 and SEx_2, set to zeroes if none are available), and two 2×1 vectors containing the limits on the likelihood space within which to optimize the parameters ε and λ (MinMaxeps and MinMaxlam). The function calls `transform_lambda()` which is also given below.

The function returns $\bar{d}$ (the phylogenetic mean difference), *t*, a P-value for the test, MLEs for ε, λ, and σ^2, and the log-likelihood. This information is also printed to screen, and the function displays a four-panel figure (similar to Figure 1 in the main text) with various visualizations of the likelihood surface for ε and λ.

Function `phyl_paired_ttest()`:

```
% function [dbar,tval,P,MLE,likelihood]=
% phyl_paired_ttest(C, x1, SEx1, x2, SEx2,MinMaxeps,MinMaxlam)
% Takes C (tree matrix), x1 (n x 1 vector), SEx1 (n x 1 vector)
% x2, SEx2, [MIN MAX] values for grid search on epsilon (eps) and lambda
% (lam).
% Returns dbar, tval, P (P-value), MLEs for epsilon, sigma^2, and lambda,
% and the log-likelihood.
% Optimization is performed on a 100 x 100 grid.

function [dbar,tval,P,MLE,likelihood]=phyl_paired_ttest(C, x1, SEx1, x2,
SEx2,MinMaxeps,MinMaxlam)

    % set grid search criterion
    STEP=0.01; % *100%

    % number of parameters
    np=3;

    % number of taxa
    n=max(size(C));

    % compute differences
    d=x1-x2;

    % compute a matrix of estimated sampling variances
    E=diag(SEx1.^2+SEx2.^2);

    % create column vector of 1.0s
    one=ones(n,1);

    % set grid search for epsilon & lambda
    lambda=MinMaxlam(1):STEP*(MinMaxlam(2)-MinMaxlam(1)):MinMaxlam(2);
    epsilon=MinMaxeps(1):STEP*(MinMaxeps(2)-MinMaxeps(1)):MinMaxeps(2);

    % determine if the user has fixed epsilon or lambda

    if (MinMaxeps(2)-MinMaxeps(1))==0
        epsilon=MinMaxeps;
        np=np-1;
    end
    if (MinMaxlam(2)-MinMaxlam(1))==0
        lamda=MinMaxlam;
        np=np-1;
    end

    % OR uncomment to fix lambda at one or zero or epsilon at zero
    % lambda=[0 0]; np=np-1;
    % lambda=[1 1]; np=np-1;
    % epsilon=[0 0]; np=np-1;

    % ok, now estimate lambda & epsilon by maximizing the likelihood
    maxL=-Inf; minL=Inf;
    logL=ones(max(size(lambda)),max(size(epsilon)))*(-Inf);
    for i=1:max(size(lambda))
```

```
        for j=1:max(size(epsilon))
            % transform by lambda
            Cl=lambda_transform(lambda(i),C);
            % add sampling error
            V=Cl+epsilon(j)*E;
            % check positive definiteness
            [R p]=chol(V);
            if(p==0)
                % compute parameter estimates and likelihood
                dbar=(one'*V^-1*one)^-1*(one'*V^-1*d);
                sigma2=(d-dbar*one)'*V^-1*(d-dbar*one)*n^-1;
                logL(i,j)=-(1/2)*(d-dbar*one)'*(sigma2*V)^-1*(d-dbar*one)...
                    -(1/2)*log(det(sigma2*V))-(n/2)*log(2*pi);
                if logL(i,j)>maxL
                    MLpos=[i j];
                    maxL=logL(i,j);
                    MLE.sigma2=sigma2;
                end
                if logL(i,j)<minL
                    minL=logL(i,j);
                    minLpos=[i j];
                end
            end
        end
    end

    % create plot of likelihood surfaces
    colormap(sort(gray,'descend'));
    eps=char(hex2dec('65')); lam=char(hex2dec('6c'));
    subplot(2,2,1); mesh(epsilon,lambda,logL);
    xlabel(eps,'fontname','symbol'); ylabel(lam,'fontname','symbol');
    zlabel('log(L)');
    title('A)                                                                    ');
    subplot(2,2,3); plot(lambda,logL(:,MLpos(2)),'k');
    xlabel(lam,'fontname','symbol'); ylabel('log(L)');
    title('C)                                                                    ');
    subplot(2,2,4); plot(epsilon,logL(MLpos(1),:),'k');
    xlabel(eps,'fontname','symbol'); ylabel('log(L)');
    title('D)                                                                    ');

    % replace -Inf with minL
    for i=1:max(size(lambda))
        for j=1:max(size(epsilon))
            if logL(i,j)==-Inf
                logL(i,j)=minL;
            end
        end
    end

    subplot(2,2,2); contour(epsilon,lambda,logL,2000);
    xlabel(eps,'fontname','symbol'); ylabel(lam,'fontname','symbol');
    title('B)                                                                    ');
    suptitle('Likelihood Surfaces');

    MLE.lambda=lambda(MLpos(1));
    MLE.epsilon=epsilon(MLpos(2));
    likelihood=maxL;
```

```
    % pause
    pause(1);

    % ok, now perform calculations using MLlambda & MLepsilon
    Cl=lambda_transform(MLE.lambda,C);

    % calculate mean difference
    V=Cl+MLE.epsilon*E;
    dbar=(one'*V^-1*one)^-1*(one'*V^-1*d);

    % calculate SE of dbar
    SEdbar=sqrt(MLE.sigma2*(n/(n-np))*(one'*V^-1*one)^-1);

    MLE.sig2e=MLE.sigma2*MLE.epsilon;

    % calculate t-statistic
    tval=dbar*SEdbar^-1;

    % calculate P-value
    P=2*(1-tcdf(abs(tval),n-np));

    % print results to screen
    fprintf(1,'dbar = %f\tt(df = %d) = %f\t',dbar,n-np,tval);
    fprintf(1,'P(t) = %f\n',P);
    fprintf(1,'MLE(lambda) = %f\tlog(L) = %f\n',MLE.lambda,maxL);
    fprintf(1,'MLE(epsilon) = %f\tlog(L) = %f\n',MLE.epsilon,maxL);
    fprintf(1,'MLE(sig2e) = %f\tlog(L) = %f\n',MLE.sig2e,maxL);

% done
```

Function `lambda_transform()`:

```
% function Ml=lambda_transform(lambda,M)
% Multiplies off-diagonals of square matrix M by lambda
% and returns Ml.
function Ml=lambda_transform(lambda,M)

    Ml=M;

    n=max(size(M));

    % transform by lambda
    for i=1:n
        for j=(i+1):n
            Ml(i,j)=M(i,j)*lambda;
            Ml(j,i)=Ml(i,j);
        end
    end

% done
```

Literature Cited in the Supporting Information

Revell, L. J. & Harmon, L. J. 2008. Testing quantitative genetic hypotheses about the evolutionary rate matrix for continuous characters. *Evol. Ecol. Res.* 10: 311-321.

Rohlf, F. J. 2001. Comparative methods for the analysis of continuous variables: Geometric interpretations. *Evolution* 55: 2143-2160.