Software: ResidualRandwithBootRcode.r

Program to conduct significance testing of ontogenetic-allometry trajectories using residual randomization and age category bootstrap.

Erik Otárola-Castillo August 2012

details published in

Anderson, M. J., and C. J. F. Ter Braak, 2003 Permutation tests for multi-factorial analysis of variance. J. Statist. Comput. Simul. 73(2): 85-113.

Collyer, M. L., and D. C. Adams. 2007. Analysis of two-state multivariate phenotypic change in ecological studies. Ecology. 88:683-692.

Piras, P., P. Colangelo, D.C. Adams, A. Buscalioni, J. Cubo, T. Kotsakis, C. Meloro, and P. Raia. 2010. The Gavialis-Tomistoma debate: the contribution of skull ontogenetic allometry and growth trajectories to the study of crocodylian relationships. Evolution and Development. 12:568-579"

KP McNulty, SR Frost, DS Strait (2006). Examining affinities of the Taung child by developmental simulation. Journal of Human Evolution, 51, 274-296.

```
rm(list=ls())

library(MASS)

# get data

dat<-as.data.frame(read.csv("C:/Users/EOC/Documents/Research/Manuscripts/In Progress/Human Facial Ontogeny/final-humans.csv"))


# function to use "which" multiple times using lapply, or sapply

wh<-function(a){which(pop==a)}

# population or species vector (label)

pop<-as.factor(dat[,2])

# levels

lev<-levels(pop)

# log csize

csize<-dat[,4]
```

```
# age
age<-dat[,3]
#shape data matrix
shapedat<-as.matrix(dat[,5:dim(dat)[2]])
# pairwise comparisons between all populations (n=21 comparisons)
comb<-combn(lev,2)
ncomb<-dim(comb)[2]
# pairwise combination label
lab<-character(21)
for(i in 1:21){
  lab[i]<-paste(comb[,i][1],"-",comb[,i][2],sep="")
}

# function to compute ALL species/population coefficient vectors
# using the generalized inverse
getvec<-function(a,mydata,pop,size=csize){
  data<-mydata[which(pop==a),]
  x<-size[which(pop==a)]
  Xa<-cbind(rep(1,dim(data)[1]),x)
  var<-as.matrix(ginv(t(Xa)%*%Xa)%*%t(Xa)%*%data)[2,]
  return(var)
}

# function to compute angle between two vectors
getangle<-function(v1,v2){acos(sum(v1*v2)/sqrt(sum(v1^2) * sum(v2^2)))*(180/pi)}

# function to quickly compute angles using sapply
ga<-function(a,vec){
  b<-vec[,which(colnames(vec)==a)]
  apply(vec,2,getangle,b)
}

# example
getvec(lev[1],mydata=shapedat,pop=pop)
vec<-(sapply(lev,getvec,shapedat,pop=pop));vec # compute vectors of all levels
angmat<-sapply(lev,FUN=ga,vec);angmat # compute observed angles
```

```
## Account for sample size differences:
# boot accounts for unequal sample sizes by taking an unbalanced species pair sample (where species A at age i, might have more observations than species b at age i), and returns the a random sample of the
# more abundant species equaling the smaller sample size
# if repeated multiple times, this function can be used as a bootstrap
# "Alaskan-Austrian" comparison is made.
# Alaskan N at age 1 = 4
# Austrian N at age 1 = 8
# boot constrains Austrian N at age 1 to 4, the same is done for all ages
boot<-function(pair,cat,pop){
  agecat<-unique(sort(cat))
  ncats<-length(agecat)
  groups<-lapply(pair,wh)
  g1<-lapply(pair,wh)[[1]]
  g2<-lapply(pair,wh)[[2]]
  samsiz<-apply(table(cat[unlist(lapply(pair,wh))],as.character(pop[unlist(lapply(pair,wh))])),1,min)
  agefunc<-function(agecat,g){sample(g[which(cat[g]==agecat)])}
  rem<-function(a,x){a[1:x]}
  agefunc2<-function(g){mapply(rem,lapply(agecat,agefunc,g),samsiz)}
  return(unlist(lapply(groups,agefunc2)))
}

bootfunc<-function(N){boot(combn(lev,2)[,N],age,pop)}
dat[bootfunc(1),c(1,2,3)] # test

# set up residual randomization using a model with log centroid size + species
nullmod <- lm(shapedat ~ csize + pop)
res<-resid(nullmod)

nperm<-10000 # 10 k perms takes about 10 minutes
permat<-numeric(21)
system.time(for (i in 1:length(permat)){
  pvec<-numeric(nperm)
  pvec[nperm]<-1
  for (j in 1:(nperm-1)){
    permdat<-predict(nullmod) + res[sample(nrow(res)),]
```

```
   bootsamp<-bootfunc(i)
   permdat.boot<-permdat[bootsamp,]
   csize.perm<-csize[bootsamp]
   popperm<-as.factor(as.character(pop[bootsamp]))
   vecperm<-sapply(comb[,i],getvec,permdat.boot,pop=popperm,size=csize.perm)
   an<-ifelse(getangle(vecperm[,1],vecperm[,2])>90,180-
getangle(vecperm[,1],vecperm[,2]),getangle(vecperm[,1],vecperm[,2]))
   pvec[j]<-ifelse(an >= as.vector(as.dist(angmat))[i],1,0)
  }
  permat[i]<-sum(pvec)/nperm
})
permat
re<-as.data.frame(cbind(lab,round(as.vector(as.dist(angmat)),2),permat))
re<-re[order(re[,3]),]
results<-cbind(re,.05/21:1,as.numeric(as.matrix(re[,3]))<=.05/21:1)
colnames(results)<-c("Comparison","Angle","P value","Seq. Bonferroni P value", "<Adjusted P")
results


##############################################################################
# Residual Randomization WITHOUT SAMPLE SIZE CONSTRAINT (for completeness)
##############################################################################

nullmod <- lm(shapedat ~ csize + pop)
res<-resid(nullmod)
nperm<-1000
permat<-numeric(21)
system.time(for (i in 1:21){
  pvec<-numeric(nperm)
  pvec[nperm]<-1
  for (j in 1:(nperm-1)){
   permdat<-predict(nullmod) + res[sample(nrow(res)),]
   vecperm<-sapply(comb[,i],getvec,permdat,pop=pop)
   an<-ifelse(getangle(vecperm[,1],vecperm[,2])>90,180-
getangle(vecperm[,1],vecperm[,2]),getangle(vecperm[,1],vecperm[,2]))
   pvec[j]<-ifelse(an >= as.vector(as.dist(angmat))[i],1,0)
  }
```

```
  permat[i]<-sum(pvec)/nperm
})
permat
re<-as.data.frame(cbind(lab,round(as.vector(as.dist(angmat)),2),permat))
re<-re[order(re[,3]),]
results<-cbind(re,.05/21:1,as.numeric(as.matrix(re[,3]))<=.05/21:1)
colnames(results)<-c("Comparison","Angle","P value","Seq. Bonferroni P value", "<Adjusted P")
results
```