Analyzing Outline Data

Ryan N. Felice

Loading Packages and Scripts

```
library(geomorph)
library(Momocs)
library(stringr)
library(mvMORPH)
library(MASS)
```

```
source('./utility_functions/MorphometricExtraction_Functions.r')
source('./utility_functions/MorphoFiles_Function.r')
source('./utility_functions/OutlineAnalysis_Functions.r')
```

Read data

We will just quickly load our data and run the EFA with 7 harmonics so we can move to our analyses

```
Belemnite.Full<-readland.nts("./Data/Belemnite_SmoothedOutline.nts")
Specimens<-dimnames(Belemnite.Full)[[3]]
Belemnite.R1<-Belemnite.Full[,,str_detect(Specimens, "R1")]
Belemnite.R2<-Belemnite.Full[,,str_detect(Specimens, "R2")]
#make an empty list to hold our Fourier analysis results
EFA<-list()
#create a vector of specimen names which are the
Spec.Names<-strsplit(dimnames(Belemnite.R1)[[3]], split=".", fixed=TRUE)
for (i in 1:dim(Belemnite.R1)[3]) {
  EFA[[i]]<-NEF(Belemnite.R1[,,i], Harmonics=7)
  names(EFA)[i]<-Spec.Names[[i]][1]
}
```

For this dataset, we have some additional trait and specimen data that tells us more information about each hook. Let's load that in so we can use it for downstream analyses

```
Belemnite.Covariates<-read.table("./Data/Belemnite_Data.txt",
                                 header = TRUE, sep="\t")
colnames(Belemnite.Covariates)<-c("Specimen", "Hook.name", "Hook.number",
                                  "CrossSection.R1.mm", "CrossSection.R2.mm",
                                  "Distance.to.mouth.mm")
```

Explore your data

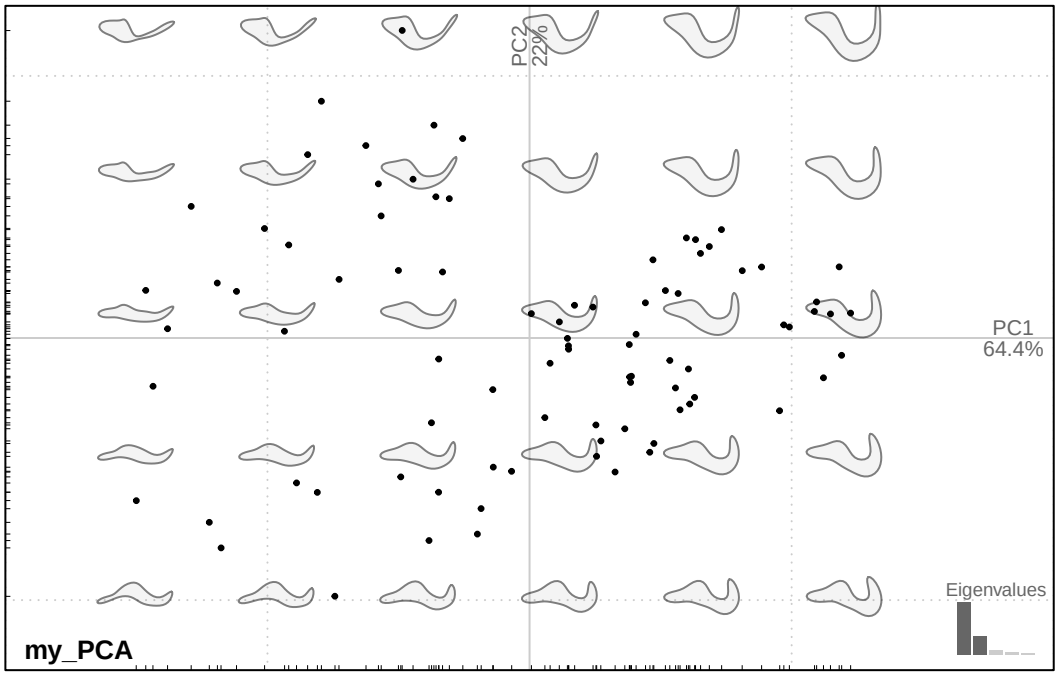
We can do a PCA with harmonic data the same way we did the PCA with landmark data. We just need to convert our data into matrix format where each row is an observation (a single hook) and each column is a harmonic coefficient.

```
#make an empty matrix with the correct number of rows and columns
EFA.Mat<-matrix(NA, length(EFA), length(EFA[[1]][[1]])*4)
#add row and column names
colnames(EFA.Mat)<-c(paste(rep("A", length(EFA[[1]][[1]])),
                           1:length(EFA[[1]][[1]]), sep=""),
                     paste(rep("B", length(EFA[[1]][[1]])),
                           1:length(EFA[[1]][[1]]), sep=""),
                     paste(rep("C", length(EFA[[1]][[1]])),
                           1:length(EFA[[1]][[1]]), sep=""),
                     paste(rep("D", length(EFA[[1]][[1]])),
                           1:length(EFA[[1]][[1]]), sep=""))
rownames(EFA.Mat)<-names(EFA)

#use a for loop to fill each specimen into the matrix
for (i in 1:length(EFA)) {
  Temp<-EFA[[i]]
  EFA.Mat[i,]<-c(Temp[[1]], Temp[[2]], Temp[[3]], Temp[[4]])
}
```

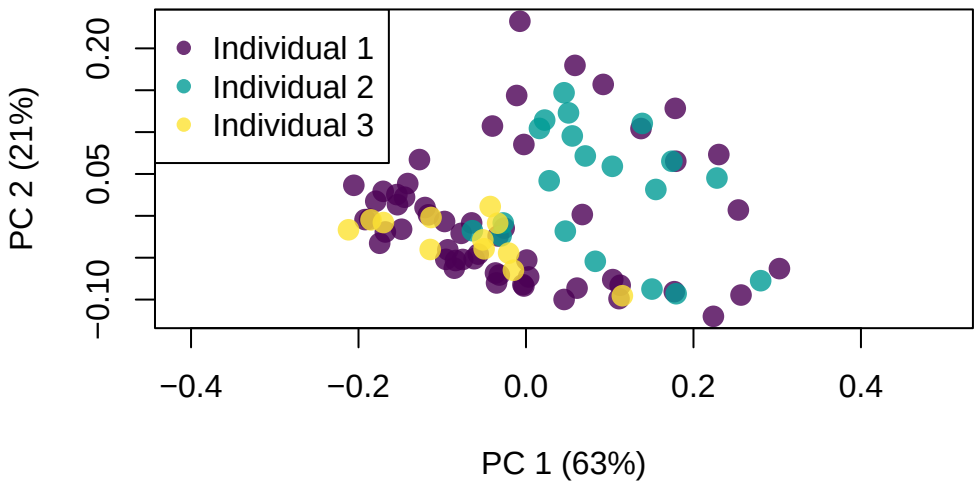
Something really cool about PCA is that you can reconstruct the shapes at any point in the "morphospace" by multiplying the mean shape by the PC scores. The easiest way to do this is with the `PCA` function in the Momocs package

```
EFA.coe <- OutCoe(EFA.Mat, method = "efourier", norm = TRUE)
my_PCA <- Momocs::PCA(EFA.coe)
plot(my_PCA, morpho=TRUE)
```



But in general you will have more power if you run and plot the PCA yourself in base R

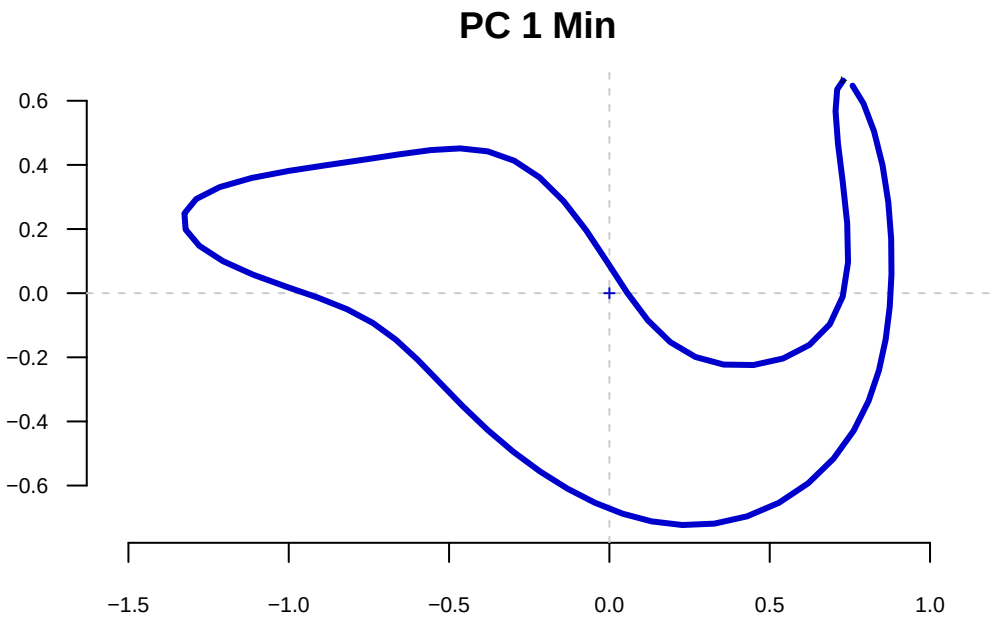
```
#remove the coefficients that were used for normalization
EFA.Mat2<-EFA.Mat[,-which(colnames(EFA.Mat) %in% c("A1", "B1", "C1"))]
#Calculate PCA
PCA2<-princomp(EFA.Mat2)
#Calculate proportion of variance explained per axis
PoV<-(PCA2$sdev^2/sum(PCA2$sdev^2))*100
#Color-code by specimen
cols<-hcl.colors(n=length(unique(Belemnite.Covariates[,"Specimen"])),
                 palette="viridis", alpha=0.8)
Col.vec<-cols[Belemnite.Covariates[,"Specimen"]]
#Plot results
plot(PCA2$scores[,1], PCA2$scores[,2], type="p",
     xlab=paste("PC 1 (", round(PoV[1], digits=0), "%)", sep=""),
     ylab=paste("PC 2 (", round(PoV[2], digits=0), "%)", sep=""), pch=16,
     cex=1.5, col=Col.vec, asp=1)
legend("topleft", pch=16, col=cols, legend=c("Individual 1",
                                             "Individual 2",
                                             "Individual 3"))
```




```
#Calculate mean shape

meanshape<-apply(EFA.Mat[,], 2, mean)
#Extract eigenvalues

ev<-PCA2$loadings
#Calculate shape at the right extreme of the PCA morphospace
Mx1<-meanshape+min(PCA2$scores[,1])*
  c(0, ev[1:6,1], 0, ev[7:12,1], 0, ev[13:18,1], ev[19:25,1])
#Note the insertion of 0 to compensate that we eliminated the 'A1', 'B1',
#and 'C1' components for the PCA calculation
#Reconstruct the shape
PC1Max<-iefourier(an=Mx1[1:7], bn=Mx1[8:14], cn=Mx1[15:21], dn=Mx1[22:28],
                  Harmonics=7, Points=70)
coo_plot(data.frame(PC1Max$x,PC1Max$y),
         border="mediumblue",
         lwd = 3,
         main = "PC 1 Min")
```



As an exercise, try to plot the minimum shapes on PC axis 1 and the min and max shapes on PC Axis 2

Hypothesis Testing

Ordination analysis such as PCA are very valuable for exploring your data and trying to understand how variation in your data is structured. However, we normally have explicit *a priori* hypotheses we want to test. This requires other approaches.

One useful tool is LDA (linear discriminant analysis, also called discriminant function analysis or canonical variate analysis). One of the oldest forms of machine learning, we first build a linear model (ordinary least squares regression in this case) and use that to predict group identity. For example, our hooks come three individual animals. We could ask whether the shape of an individual hook predicts which individual it has come from.

Let's pretend that we don't know what individual the first 3 specimens came from and try to predict them

```
Specimen <- as.factor(Belemnite.Covariates[-c(1:3),"Specimen"])
names(Specimen) <- rownames(PCA2$scores)[-c(1:3)]

#combine shape data and specimen id data into a single list
knowns<-list(spec=Specimen, shape=EFA.Mat2[-c(1:3),])
```

```r
unknowns <-  EFA.Mat2[c(1:3),]

#calculate a regression
model <- mvols(shape~spec, data=knowns)

LDA.results <- mvgls.dfa(model)
LDA.eval <- predict(LDA.results)
#check misclassification rate:

#try to predict:
LDA.predict <- predict(LDA.results, newdata = unknowns, prior = p1)
LDA.predict$class
```

How could we improve this prediction without collecting more data? Try implementing it!

Because these fossils are recovered as complete "articulated" specimens, we know which position in the tentacle each hook comes from. Thus we can ask whether hook shape differs along the length of the tentacle. To do this, we can use a regression to test whether distance to mouth predicts shape.

In this case, we will use the Procrustes linear models (RRPP) as these are more appropriate for high-dimensional data

```r
hookdist <- Belemnite.Covariates$Distance.to.mouth.mm
names(hookdist) <- rownames(PCA2$scores)

missing_dat <- which(is.na(hookdist))
#combine shape data and specimen id data into a single list
regression.Data<-geomorph.data.frame(hookdist = hookdist[-missing_dat], shape=EFA.Mat2[-missi

#calculate a regression
model <- lm.rrpp(shape~hookdist, data=regression.Data, RRPP=TRUE)

pval<-summary(model)$table[1,"Pr(>F)"]
#Plot results


# Use fitted values from the model to make prediction lines
plot1<-plot(model, type = "regression",
            predictor = regression.Data$hookdist,
            reg.type = "RegScore",
            pch = 19, col = "green")
```

Now try doing some tests of your own using a different covariate!