Principal Components Analysis in R

Mark Puttick (marknputtick@gmail.com)

24/04/2018

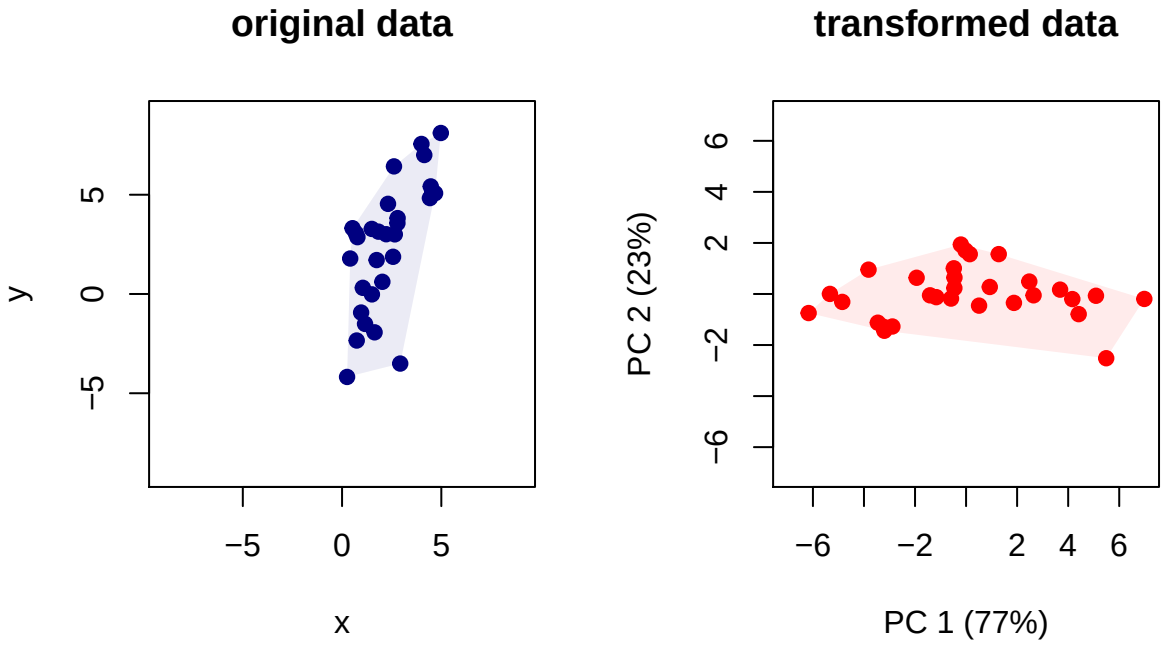
Contents

Principal Components Analysis

The backbone of Principal Components Analysis (PCA) is to identifying patterns in data with lots of dimensions. The main goal of PCA is to reduce the dimensionality of data into interpretable patterns.

Simply put PCA is the rotation of shapes until it best explains the data. All we want to do is rotate our data until we can easily visualise most of the variation

This course assumes you have some basic R knowledge, if not you may wish to consult the introduction available here



PCA: behind the scenes

Let's work through an example with few dimensions so we can do it 'by hand' to get an idea what is going on First we will simulate some data.

To make this easier to follow we will construct data with only in two dimensions

```
#some data
set.seed(101)
x <- runif(10, 0.1, 5)
y <- x + rnorm(10, sd=3)
```

Now the first thing we have to do is standardize our data by subtracting the mean of each variable x and y

```
# standardise by subtracting the mean
xad <- x - mean(x)
yad <- y - mean(y)
```

We are going to calculate the covariance of our data

$$cov(x, y) = \frac{\Sigma(\bar{y} - y)(\bar{x} - x)}{n - 1}$$

With covariance we have two measures of data and explore the sum of the product of each of their respective deviations from the mean... this is a measure of one mean with regards to another. If it is positive they increase together, if negative then as one increases the other decreases

We can calculate this in two ways with R

```
# get the covariance 'by hand'
co_var_xy <- sum((xad - mean(xad)) * (yad-mean(yad) )) /
(length(x)-1)
# to make it easier use var
co_var_xy <- var(xad, yad)
```

Ok now we can build the covariance matrix. This matrix shows the variance and convariance between each element. Here we construct it in R – the off-diagonal is identical as this the covariance between x and y.

```
cov_mat <- matrix(c(var(xad), co_var_xy , co_var_xy,
var(yad)), ncol=2, byrow=F)
rownames(cov_mat) <- colnames(cov_mat) <- c("x", "y")
cov_mat
```

```
##            x          y
## x  1.1045549 -0.1197644
## y -0.1197644  6.1407380
```

The first element in the matrix is the variance of x and the second row in the first column in the covariance between x and y.

We will now obtain the eigenvalues and eigenvectors (more on these below) using `eigen`

```
# use R eigen function to get eigenvalues and eigenvectors
ev_en <- eigen(cov_mat)
ev_en
```

```
## eigen() decomposition
## $values
## [1] 6.143585 1.101708
##
```

```
## $vectors
##             [,1]        [,2]
## [1,] -0.02376065 -0.99971768
## [2,]  0.99971768 -0.02376065
```

Here we see the eigenvalues and eigenvectors for our covariance matrix. We can use these eigenvectors to produce scores for the pc axes

```
# convert original data into scores
ev_en$vectors
```

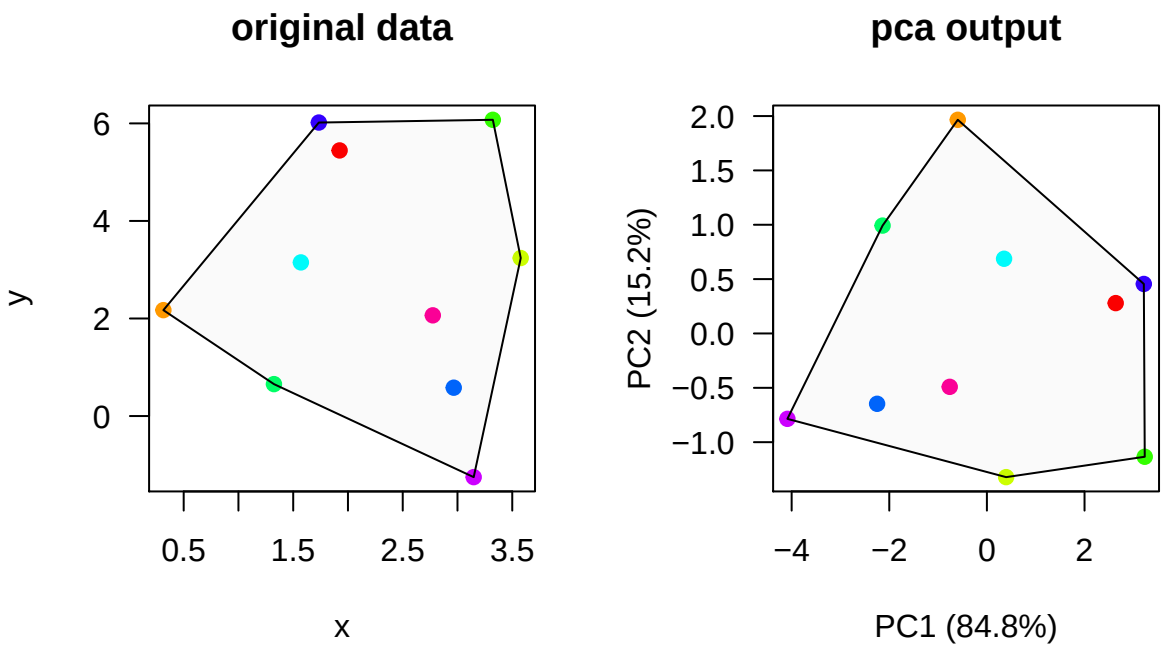
```
##             [,1]        [,2]
## [1,] -0.02376065 -0.99971768
## [2,]  0.99971768 -0.02376065
```

```
# convert original data into scores
# pc axis one
pc1 <- (xad * ev_en$vectors[1,1]) + (yad *
ev_en$vectors[2,1])
# pc axis two
pc2 <- (xad * ev_en$vectors[1,2]) + (yad *
ev_en$vectors[2,2])
# run in-built r pca analysis
pca_r <- princomp(data.frame(x, y))$scores
# check and compare our result with R's
cbind(pc1, pc2, pca_r)
```

```
##              pc1        pc2     Comp.1     Comp.2
##  [1,]  2.6386084  0.2791581  2.6386084  0.2791581
##  [2,] -0.5967953  1.9655400 -0.5967953  1.9655400
##  [3,]  0.3935166 -1.3216286  0.3935166 -1.3216286
##  [4,]  3.2332889 -1.1342818  3.2332889 -1.1342818
##  [5,] -2.1369509  0.9923089 -2.1369509  0.9923089
##  [6,]  0.3515981  0.6871175  0.3515981  0.6871175
##  [7,] -2.2491364 -0.6470418 -2.2491364 -0.6470418
##  [8,]  3.2145411  0.4553064  3.2145411  0.4553064
##  [9,] -4.0868570 -0.7854276 -4.0868570 -0.7854276
## [10,] -0.7618135 -0.4910510 -0.7618135 -0.4910510
```

We can now plot our data and compare it to our original data configuration from above (we'll learn more about these plotting options with the empirical example below)

```
par(mfrow=c(1, 2), pty="s")
par(pty="s")
plot(x, y, col=rainbow(length(x)), pch=19, las=1,
main="original data")
hpts <- chull(x, y)
hpts <- c(hpts, hpts[1])
polygon(x[hpts], y[hpts], col="#00000005")
plot(pc1, pc2, col=rainbow(length(x)), pch=19, las=1,
main="pca output", xlab="PC1 (84.8%)", ylab="PC2 (15.2%)")
hpts <- chull(pc1, pc2)
hpts <- c(hpts, hpts[1])
polygon(pc1[hpts], pc2[hpts], col="#00000005")
```



Voila! We can see that the data have simply been rotated. The distance between the data points is identical and we know get an idea of the major patterns present in the data.

Task One

- Perform and plot a PCA with the `USArrests` data built-in to R using `prcomp` (Hint – set `scaling` to TRUE)

Eigenvalues and Eigenvectors

We can use the in-built R function to discover two important aspects of our data.

The first of these is called the eigenvector. The eigenvector can be thought of as a way of transforming your data on a plot. It finds the orientation of the data that best explains the data. The eigenvectors control the way the data are transformed. An important aspect of eigenvectors in PCA is that they are all perpendicular to one another. This is also known as being orthogonal.

Eigenvalues are linked to eigenvectors. Eigenvectors are scaled by the eigenvalues and in PCA they play a key role. The PCA with the highest value shows which axis is displaying the greatest variation of the data.

Here we explore how to calculate eigenvalues and eigenvectors 'by hand' (although this is only feasible for up small matrices (3 rows x 3 columns)

The relationship between a square matrix A, an eigenvector v, and a eigenvalue λ

$$Av = \lambda v$$

So

$$Av - \lambda v = 0$$

It looks like we can factor-out v but if we want to remove v we will have to convert λ from a number into a matrix. We can do this by using an identity matrix and showing $Iv = v$

$$v(A - \lambda I) = 0$$

We can find the determinant for the co-variance matrix using our data from above

	x	y
x	1.1045549	-0.1197644
y	-0.1197644	6.1407380

To find y we can use start by substituting in our values

$$v(A - \lambda I) = 0$$

1.1045549, -0.1197644, -0.1197644, 6.1407380

$$\begin{pmatrix} 1.1045549 & -0.1197644 \\ -0.1197644 & 6.1407380 \end{pmatrix} - \lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

which is the same as

$$\begin{pmatrix} 1.1045549 & -0.1197644 \\ -0.1197644 & 6.1407380 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix}$$

substracting the matrix

$$\begin{pmatrix} 1.1045549 - \lambda & -0.1197644 - 0 \\ -0.1197644 - 0 & 6.1407380 - \lambda \end{pmatrix}$$

And calculate the determinant of the matrix

$$(1.1045549 - \lambda)(6.1407380 - \lambda) - (-0.1197644)(-0.1197644)$$

$$6.782782 - \lambda 6.1407380 - \lambda 1.1045549 - \lambda^2 - 0.01434351 = 0$$

$$6.768438 - \lambda 7.245293 - \lambda^2 = 0$$

now we just solve the quadratic equation to find eigenvalue one and two

$$\begin{aligned} x &= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \\ a &= 1 \\ b &= -7.245293 \\ c &= 6.768438 \\ eigenvalue\ one &= \frac{-b + \sqrt{b^2 - 4ac}}{2a} = 6.143585 \\ eigenvalue\ two &= \frac{-b - \sqrt{b^2 - 4ac}}{2a} = 1.101708 \end{aligned}$$

We can use these to estimate two of the possible eigenvectors given these eigenvalues

$$(A - \lambda v) = 0$$

so for the eigenvalue one

$$\begin{pmatrix} 1.1045549 - 6.143585 & -0.1197644 - 0 \\ -0.1197644 - 0 & 6.1407380 - 6.143585 \end{pmatrix}$$

which is

$$\begin{pmatrix} -5.03903 & -0.1197644 \\ -0.1197644 & -0.002847 \end{pmatrix}$$

Row one and row two are multiples, thus infinite solutions exist for the eigenvectors

If we take row one:

$$-5.03903x - 0.1197644y = 0$$

If we let one solution be *y=1*

$$\begin{aligned} -5.03903x &= 0.1197644 \\ x &= 0.1197644/-5.03903 \\ x &= -0.02376735 \end{aligned}$$

thus eigenvector one is $(-0.02376735, 1)$

For eigenvalue two:

$$\begin{pmatrix} 0.002846674 & -0.1197644 \\ -0.1197644 & 5.0390298 \end{pmatrix}$$

$$\begin{aligned} 0.002846674x - 0.1197644y &= 0 \\ y &= 1 \\ x &= 0.1197644/0.002846674 \\ x &= 42.07169 \end{aligned}$$

thus eigenvector two is $(42.07169, 1)$

Let's repeat this using R

```
#some data
set.seed(101)
x <- runif(10, 0.1, 5)
y <- x + rnorm(10, sd=3)
# standardise by subtracting the mean
xad <- x - mean(x)
yad <- y - mean(y)
# covariance
co_var_xy <- var(xad, yad)
cov_mat <- matrix(c(var(xad), co_var_xy , co_var_xy,
var(yad)), ncol=2, byrow=F)
rownames(cov_mat) <- colnames(cov_mat) <- c("x", "y")
#eigen
eigen(cov_mat)
```

```
## eigen() decomposition
## $values
## [1] 6.143585 1.101708
##
## $vectors
##             [,1]        [,2]
## [1,] -0.02376065 -0.99971768
## [2,]  0.99971768 -0.02376065
```

Easy as that. We can see the eigenvalues are identical. However, the eigenvectors are not, but this does not matter as we can show they are, in fact, multiples.

```
r.eigen <- eigen(cov_mat)
our.eigen.one <- c(-0.02376735, 1)
our.eigen.two <- c(42.07169, 1)
# comparison of eigenvecton one
r.eigen$vectors[,1] / our.eigen.one
```

```
## [1] 0.9997181 0.9997177
```

```
# comparison of eigenvecton one
r.eigen$vectors[,2] / our.eigen.two
```

```
## [1] -0.02376224 -0.02376065
```

This is fine as infinite answers exists and ours are multiples of the answer from R

Figure 1: Procrustes was not a man to accept an invitation from

Procrustes

Procrustes was a son of Zeus in Greek mythology who gave people the worst night of their lives. He would invite them to sleep in a bed. However, if they were too tall he would hacked off their legs until they fit. If too small, he stretched them out. He did this until Theseus caught him and gave him the predictable punishment... fitting him to his own bed. Different times.

Anyway, Procrustes' name has gone down as any process that manipulates data to make it fit. Procrustes analysis does three fundamental things:

- Translate data to the origin (i.e, 0,0)
- Scales them to centroid size (usually size = 1)
- Rotates until the distance between each is minimised

Ok let's get some data and see what is going on. We will be using 3-dimensional dataset of Shrew teeth (thanks to David Polly). We will go through this step-by-step so we can see what is going on

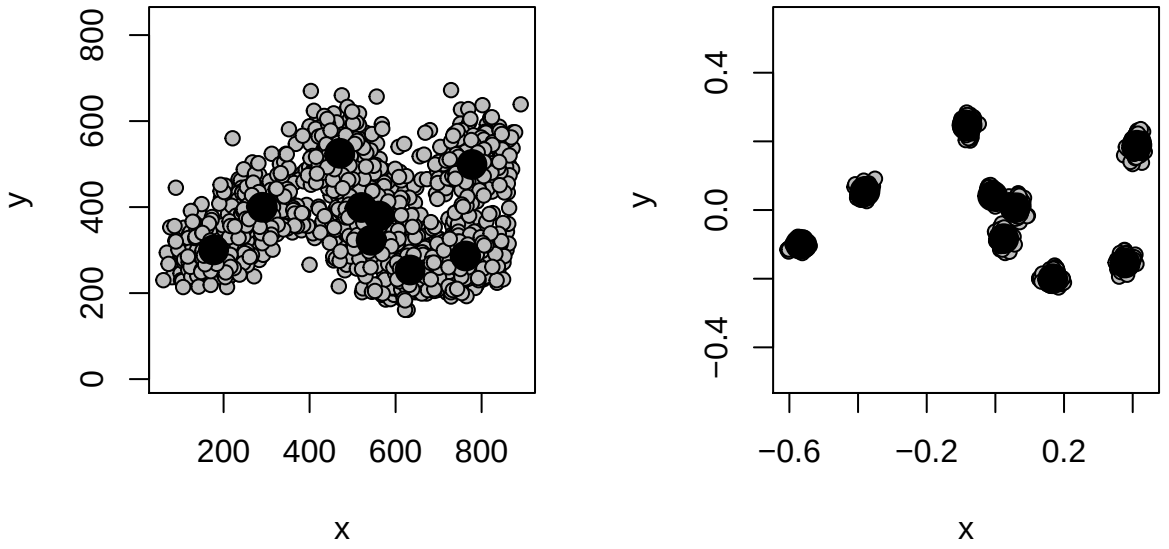
```
# we will use the package geomorph
# you may need to install in first, simply un-comment the row below
# install.packages("geomorph")
library(geomorph)
```

```
## Loading required package: rgl
```

```
## Loading required package: ape
```

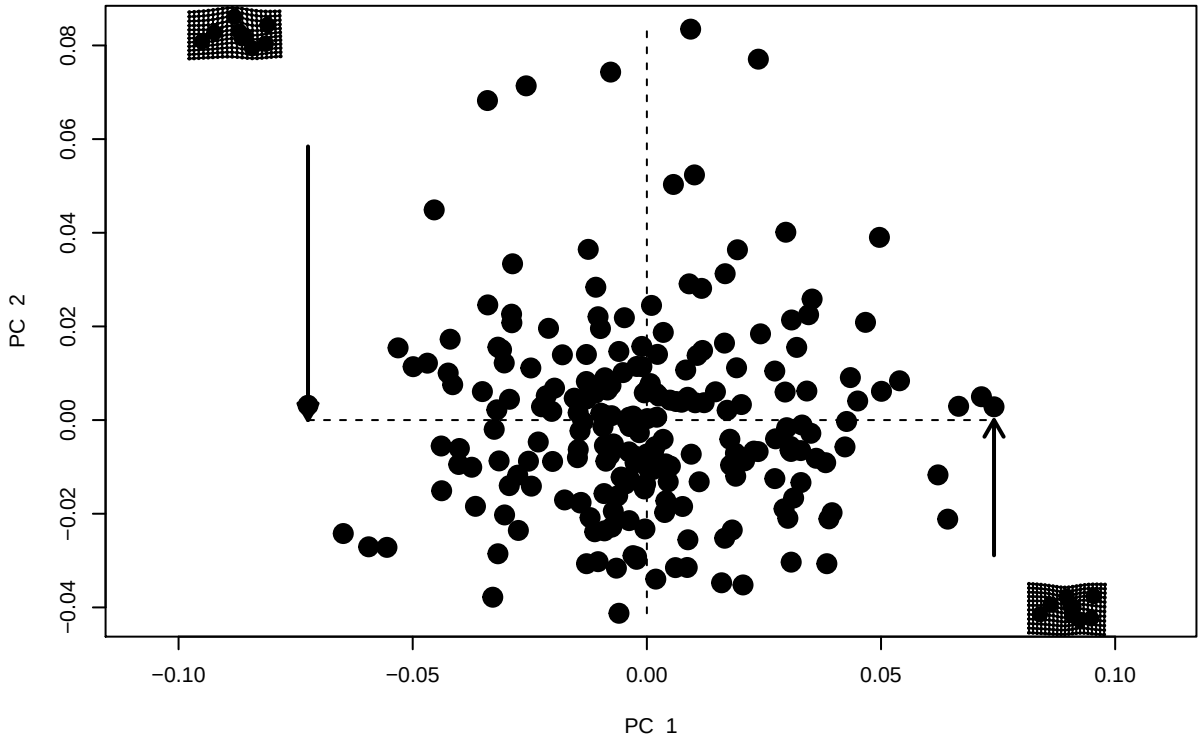
```
lands <- readland.tps("https://puttickbiology.files.wordpress.com/2018/04/polly_shrew.odt",  warnmsg=FA
# to a procrustes analysis using the gpagen function
gpa.lands <- gpagen(lands, print.progress=FALSE)
# plot the non-transformed data
par(mfrow=c(1,2), pty="s")
plotAllSpecimens(lands)
```

```
# plot the procrustes-transformed data
plotAllSpecimens(gpa.lands$coords)
```



These plots above nicely show the difference between untransformed data and data after procrustes

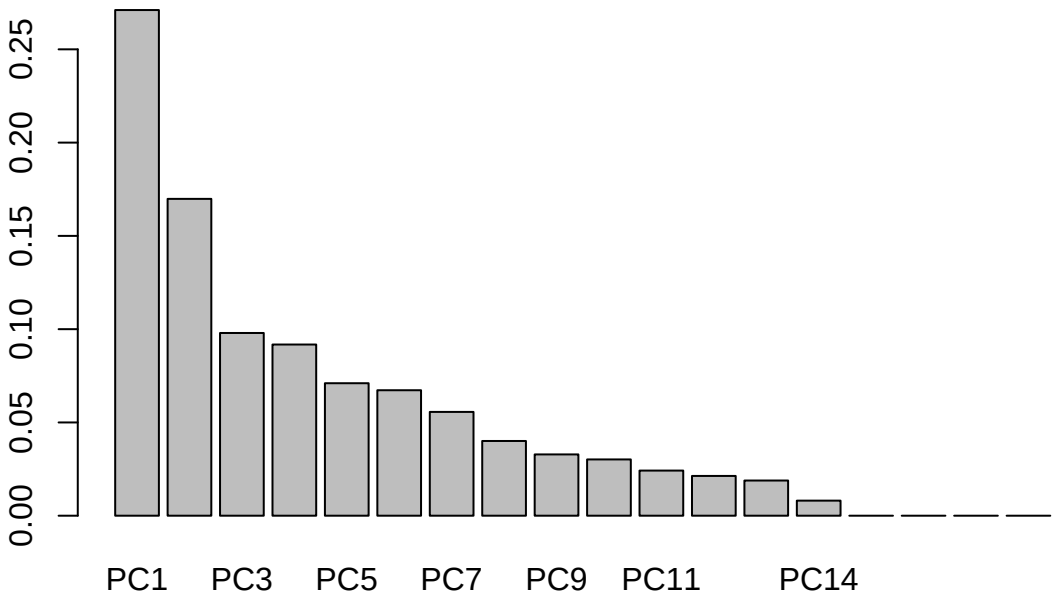
```
# perform a pca analysis on the shape data
pca.lands <- plotTangentSpace(gpa.lands$coords)
```



This shows how morphospace differs across the major axis of variation (i.e, pc1). Next we will plot a scree plot showing the amount of variation explained (as shown by the eigenvalues) by each axes. Also, we can calculate the number of axes that describe 95% of the variation.

```
# analyse output from pca analyses
cumulativeVar <- pca.lands$pc.summary$importance[3,]
```

```
propVar <- pca.lands$pc.summary$importance[2,]
# scree plot
barplot(propVar)
```




```
# how many axes explain 95% of the variation?
which(cumulativeVar >= 0.95) [1] - 1
```

```
## PC11
##   10
```

```
# Variance explained by axes one and two
propVar[1]
```

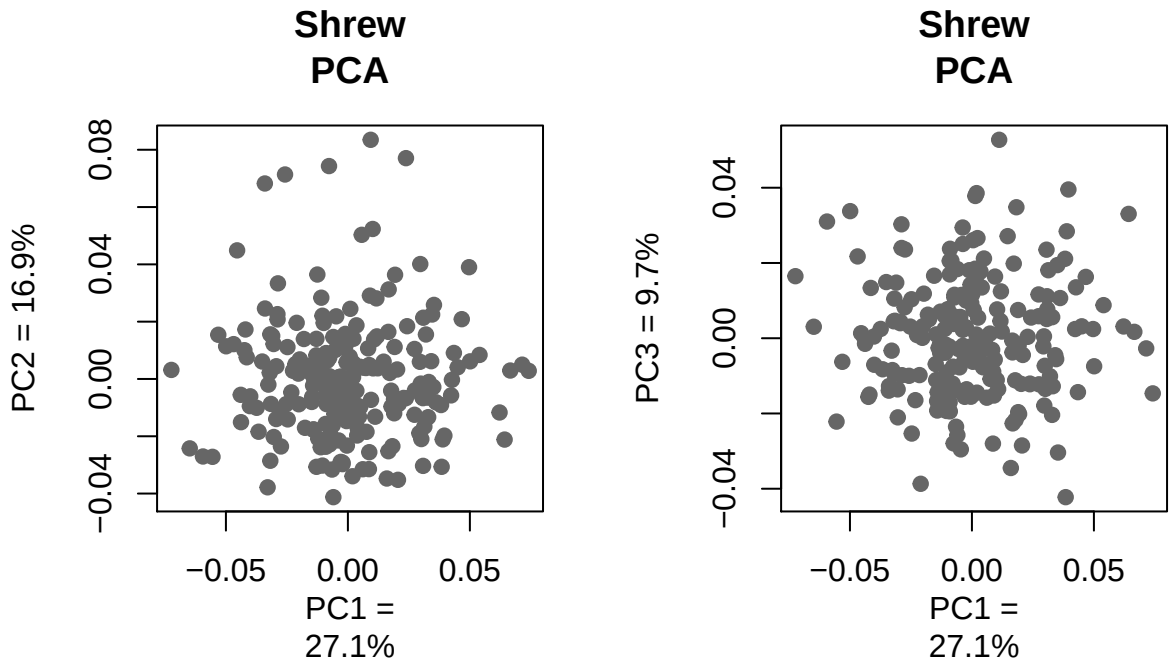
```
##     PC1
## 0.27106
```

```
propVar[2]
```

```
##     PC2
## 0.16984
```

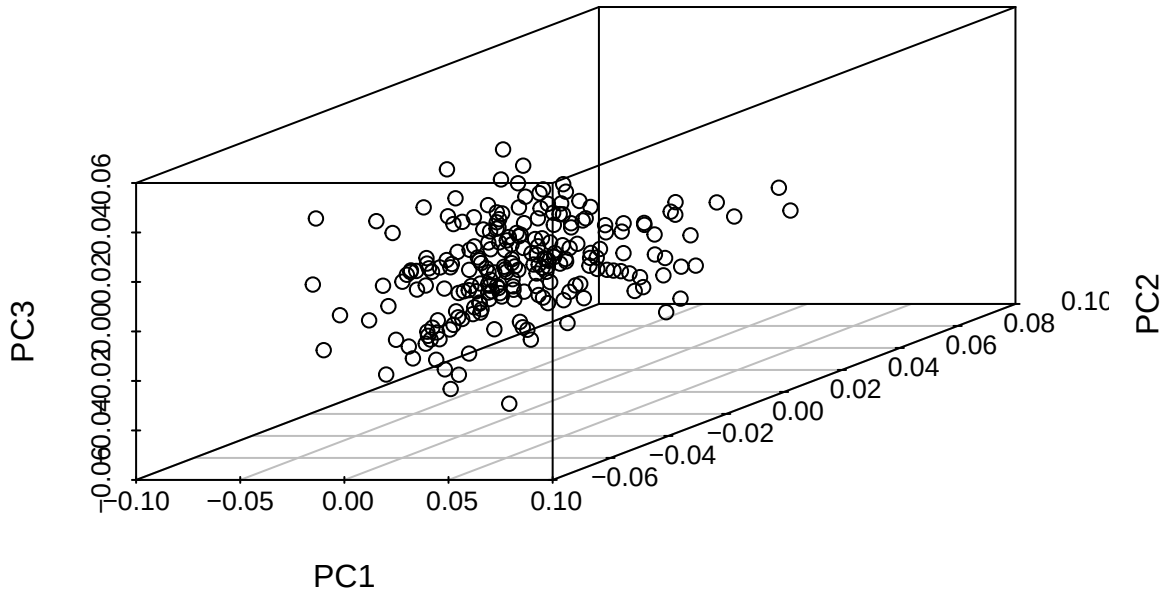
Next we can plot a PCA of the landmark data. Notice how we label the axes according the percentage of variation each of them explains

```
par(mfrow=c(1,2), pty="s")
# Plot of pca axis one and two
plot(pca.lands$pc.scores[,c(1, 2)], pch=19, xlab="PC1 =
27.1%", ylab="PC2 = 16.9%", col="grey40", main="Shrew
PCA")
# Plot of pca axis one and three
plot(pca.lands$pc.scores[,c(1, 3)], pch=19, xlab="PC1 =
27.1%", ylab="PC3 = 9.7%", col="grey40", main="Shrew
PCA")
```

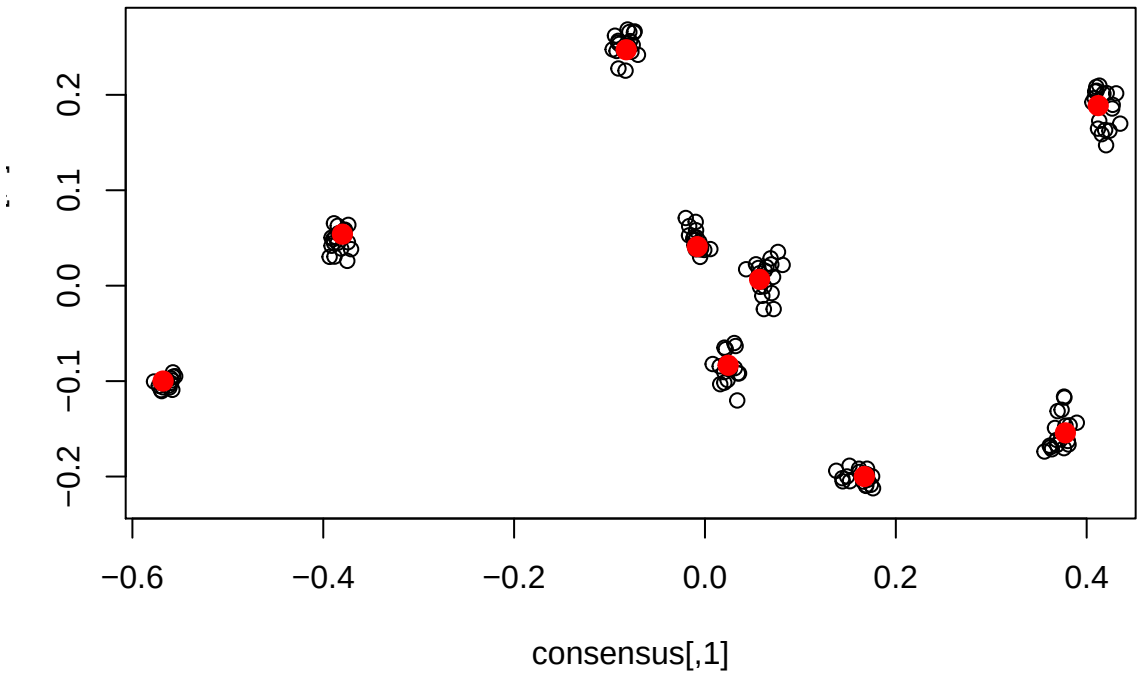


Some other cool stuff (from David Polly)

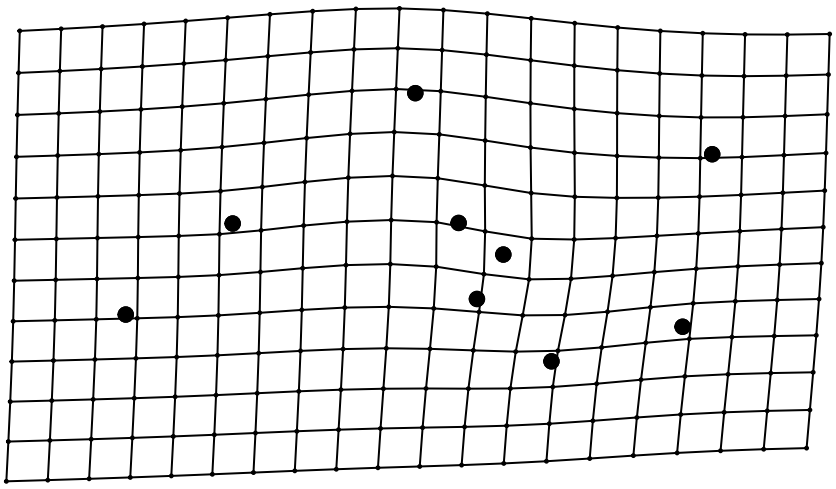
```
# we need the library scatter plot
# install.packages("scatterplot3d")
library(scatterplot3d)
# cool 3D scatterplot of data
scatterplot3d(pca.lands$pc.scores[,1:3])
```




```
# we can also analyse the consensus shape
consensus <- apply(gpa.lands$coords, c(1,2), mean)
plot(consensus,asp=1, type="n")
for(i in 1:length(gpa.lands$coords[,,3]))
points(gpa.lands$coords[,,i])
points(consensus, col="red", cex=2, pch=20)
```




```
centroid <- apply(gpa.lands$coords, 2, mean)
plotRefToTarget(consensus, gpa.lands$coords[,,2])
```

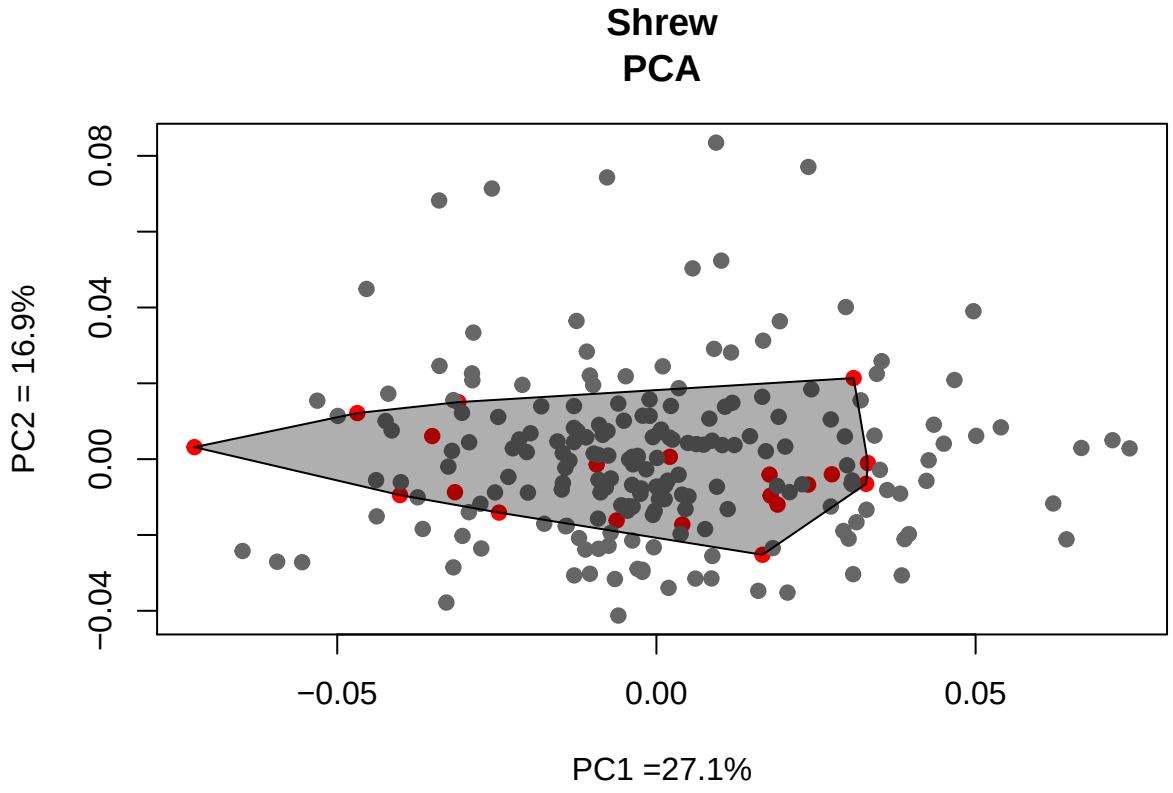


Great we have performed some PCA analyses There are plethoras of ways to do ordination analyses in R. In the above we used the geomorph package but many others are also available.

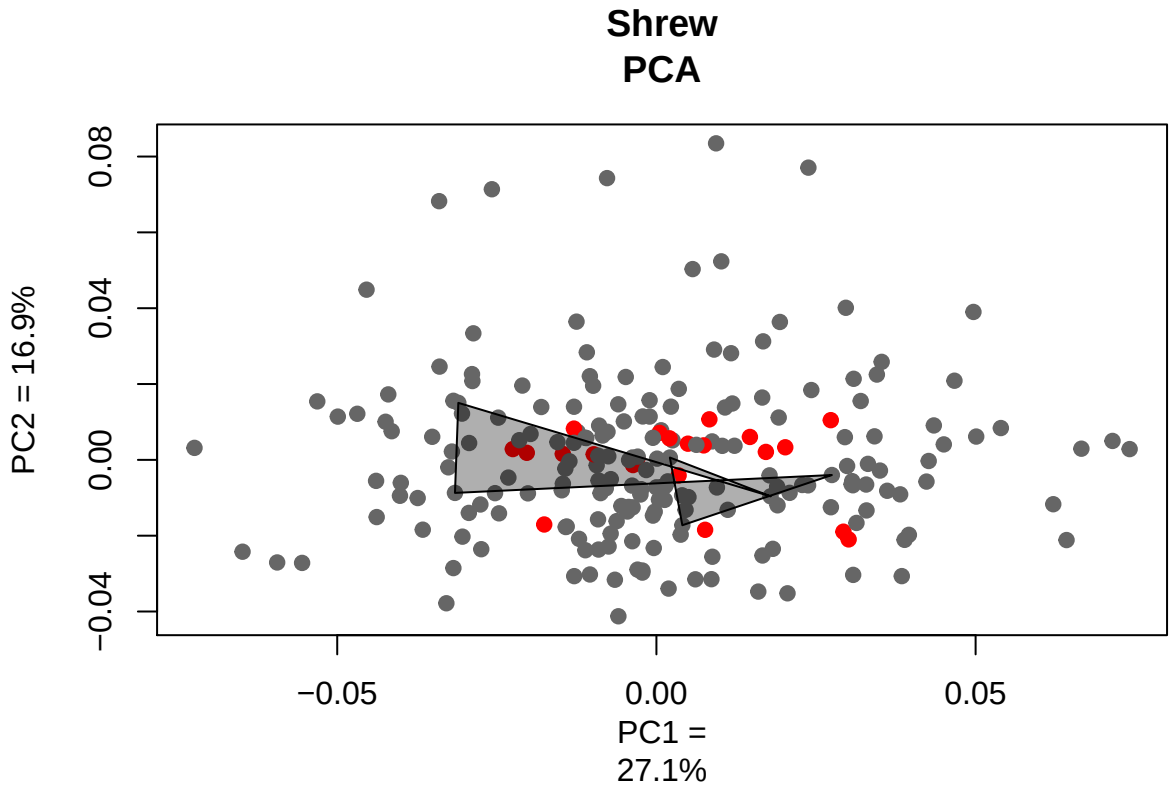
Plotting PCA and convex hulls

When we have completed a PCA (or any ordination analysis) we often want to plot our results. And more often than not we have groups, typically represented species, we wish to separate from the rest of the analysis in the plot. We can change our plotting points (i.e, change their colour) and add convex hulls to our plots. Following on from the above we can try

```
col_plot = rep("grey40", dim(pca.lands$pc.scores)[1])
col_plot[1:20] <- "red"
plot(pca.lands$pc.scores[,c(1, 2)], pch=19, xlab="PC1 =27.1%", ylab="PC2 = 16.9%", col=col_plot, main="S
PCA")
hpts <- chull(pca.lands$pc.scores[1:20,c(1, 2)])
hpts <- c(hpts, hpts[1])
polygon(pca.lands$pc.scores[hpts, 1:2], col="#00000050")
```




```
col_plot = rep("grey40", dim(pca.lands$pc.scores)[1])
col_plot[50:70] <- "red"
plot(pca.lands$pc.scores[,c(1, 2)], pch=19, xlab="PC1 =
27.1%", ylab="PC2 = 16.9%", col=col_plot, main="Shrew
PCA")
hpts <- chull(pca.lands$pc.scores[50:70,c(1, 2)])
hpts <- c(hpts, hpts[1])
polygon(pca.lands$pc.scores[hpts, 1:2], col="#00000050")
```




```
# oh dear that looks....not good
# let's try convexhull.xy in the
## install.packages("spatstat")
library(spatstat)
```

```
## Loading required package: spatstat.data
```

```
## Loading required package: nlme
```

```
## Loading required package: rpart
```

```
##
## spatstat 1.54-0       (nickname: 'Vacuous Mission Statement')
## For an introduction to spatstat, type 'beginner'
```

```
##
## Note: spatstat version 1.54-0 is out of date by more than 5 months; we recommend upgrading to the la
```

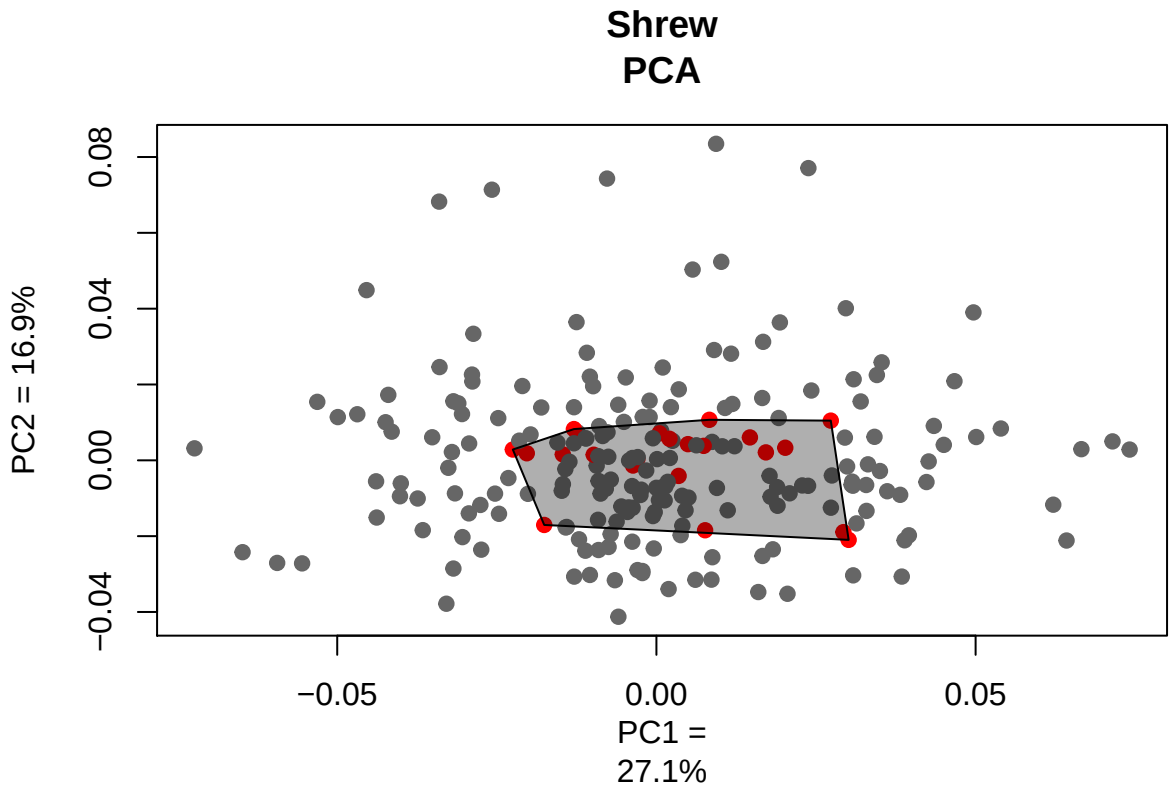
```
##
## Attaching package: 'spatstat'
```

```
## The following objects are masked from 'package:ape':
##
##     edges, rotate
```

```
col_plot = rep("grey40", dim(pca.lands$pc.scores)[1])
col_plot[50:70] <- "red"
plot(pca.lands$pc.scores[,c(1, 2)], pch=19, xlab="PC1 =
27.1%", ylab="PC2 = 16.9%", col=col_plot, main="Shrew
PCA")
hpts <- convexhull.xy(pca.lands$pc.scores[50:70,c(1, 2)])
hpts$bdry[[1]]$x
```

```
## [1]  0.027314977  0.008294586 -0.012920939 -0.022491871 -0.017574546
## [6]  0.030110961
```

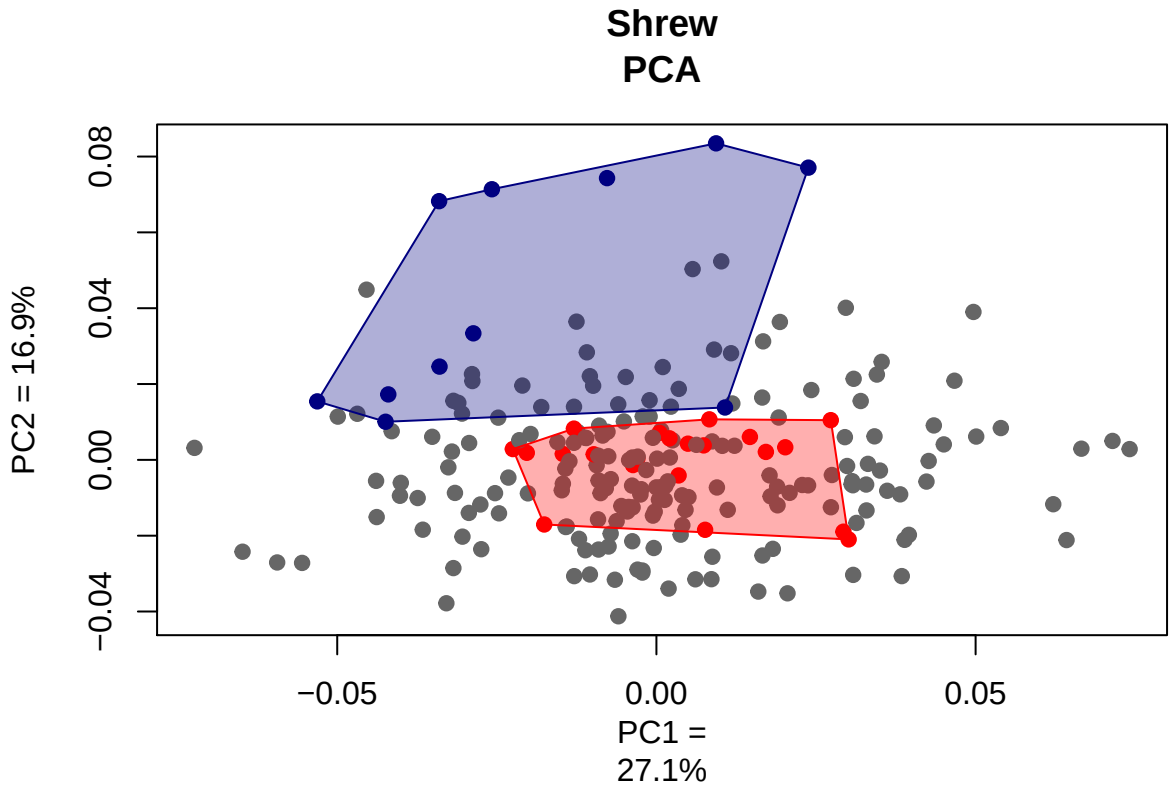
```
polygon(hpts$bdry[[1]]$x, hpts$bdry[[1]]$y,
col="#00000050")
```



Testing differences between groups

We can also investigate whether there are significant differences between groups. For example, using a manova (an anova with more than one response variables)

```
#plot two(random) groups from the PCA analysis
col_plot = rep("grey40", dim(pca.lands$pc.scores)[1])
col_plot[50:70] <- "red"
col_plot[150:160] <- "navy"
plot(pca.lands$pc.scores[,c(1, 2)], pch=19, xlab="PC1 =
27.1%", ylab="PC2 = 16.9%", col=col_plot, main="Shrew
PCA")
hpts <- convexhull.xy(pca.lands$pc.scores[50:70,c(1, 2)])
polygon(hpts$bdry[[1]]$x, hpts$bdry[[1]]$y,
col="#ff000050", border="#ff0000")
hpts <- convexhull.xy(pca.lands$pc.scores[160:150,c(1,
2)])
polygon(hpts$bdry[[1]]$x, hpts$bdry[[1]]$y,
col="#00008050", border="#000080")
```




```
groups <- rep("a", dim(pca.lands$pc.scores)[1])
groups[50:70] <- "b"
groups[150:160] <- "c"
groups <- factor(groups)
#test significance between three groups
man <- manova(pca.lands$pc.scores ~ groups)
summary(man, test="Wilks")
```

```
##            Df Wilks approx F num Df den Df    Pr(>F)
## groups      2 0.563   4.8247     28    406 2.121e-13 ***
## Residuals 216
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#test significance between the two focal groups
groupsComp <- rep("b", 32)
groupsComp[21:32] <- "c"
groupsComp <- factor(groupsComp)
man <- manova(pca.lands$pc.scores[c(50:70, 150:160),] ~
groupsComp)
summary(man, test="Wilks")
```

```
##            Df  Wilks approx F num Df den Df   Pr(>F)
## groupsComp  1 0.2176   4.3661     14     17 0.002519 **
## Residuals  30
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It appears our groups differ significantly across all PC axes

Ordination methods in R

There are lots of names thrown around for different ordination techniques. Basically there are a few directions that can help the decision as to what method is appropriate for your data

- Principal Components Analysis (PCA). Arranges data by major axes based on measured variables
- Principal Coordinates Analysis (PCO). Arranges data by major axes based on distance measures. e.g `Pcoa` in APE
- Canonical Variates Analysis (CVA) (or Discriminant Function Analysis, DFA). Finds best separation between groups. e.g, `Cca` function in vegan
- Multidimensional Scaling (MDS). Arranges data so the distances on 2D plot are as similar as possible to original multivariate distances. Data is based on ranks not Euclidean distances, so unlike PCA, it will not preserve the distance between points. e.g, `metaNDS` in vegan
- Linear discriminant analysis Separate two or more classes of data by linear combination of features. e.g., `lda` in MASS