Spatial Analysis and Data Exploration in History and Archaeology, Spring 2021
LDA-H313

WEEK 5 EXERCISE: Cluster and Principal Component Analysis

This week we will look at two further methods that explore patterns within data: cluster analysis and principal component analysis (PCA). The data we will apply these to originally comes from comes from David Peacock's 1981 study of Roman rotary mills.[1] Peacock developed a typological scheme for the mills, dividing them into typological groups by the characteristics, and we will see how cluster and PCA can produce similar results. The R workflows are adapted from various ones developed by Kris Lockyear at UCL Institute of Archaeology.

First, please place the **week5** folder, which contains the data we will examine, in the **sade2021** folder and set your workspace. Make sure that the path is exactly as indicated, for example without a space in the folder name:

```
Windows:
     setwd("C:\\sade2021\\week5")
Mac:
     setwd("~/Documents/sade2021/week5")
```

1. Hierarchical cluster analysis

1.1 Introduction and data overview

We will first of all see how hierarchical cluster analysis can be used to divide artefacts into groups. Such classification efforts are common in archaeology (and many other sciences and humanities disciplines) and can yield valuable information, for instance, about how technologies and tools developed or were used.

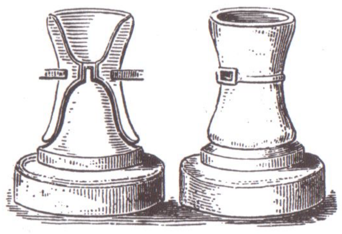
First, nstall ***cluster*** using **`install.packages()`** and then load it. Then load the Peacock data from a CSV file.

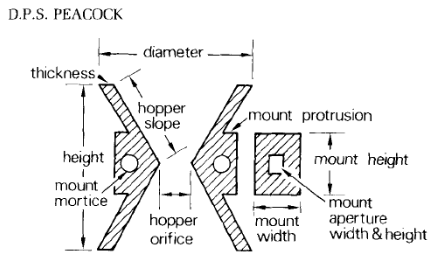
```
library(cluster)

peacock<-read.csv(file="peacock.csv", header=TRUE, sep=",")
```

[1] Peacock, D. (1989). 'The mills of Pompeii', *Antiquity*, 63 (239), 205-214. doi:10.1017/S0003598X0007592X Uploaded in the Further reading folder.

Examine it by typing the name of the objects and using the **`summary()`** function. You will see a data frame containing eleven columns of numeric ratio data, giving measurements of the various dimensions of 42 Roman mills. A twelfth column gives the type of the girth band, and a thirteenth column classifies the mills into five groups A-E by these measurements. You can create various graphs of visual the data; a boxplot could be a good one - but leave the columns ‘girth_band’ and ‘type’ out of it. Do you know why, or how you could use subsetting by square brackets in the command to easily do so?

What do you think of the resulting plot, how useful it is?





1.2 Standards deviation

A problem with the boxplot is that while all the numerical columns use the same measurement system, they measure parts of the object that are of significantly different sizes. Consequently variation among the smaller parts is nearly impossible to examine. A solution is to *standardise* the data, so that all the ratio data columns are on the same relative scale.

```
peacock.std<-scale(peacock[1:11])

peacock.std

boxplot(peacock.std)
```

Do you think this is better, and what kind of information does it give you?

1.3 Cluster analysis

Carry out the cluster analysis with the below command, and plot the results as a dendrodram (“tree diagram”). This organises the data into hierarchical relationships.

```
peacock.single<-
hclust(dist(peacock.std,method="euclid"),method="single")

plot(peacock.single,main="Single Linkage")
```

The command **`agnes()`** provides summary statistics, examine it and then use it on the standardised object.

1.4 Grouping

An inherent problem with cluster analysis is deciding how many groups the events should be divided into; there is no single good solution to it. While in more advanced workflows there are various statistical approaches and calculations that can be applied, deciding which one to pick depends on the character of the data and what kind of information one wishes to extract from it.

In humanities, it may be that the research must make an informed and argued judgement within the framework provided by the analysis but guided by their interpretation of the actual material. In this case, Peacock settled on five groups, so let us place cuts in the dendrogram accordingly.

```
rect.hclust(peacock.single,k=5)
```

Do you think this makes sense? Would you cut the plot differently?

For this exercise, we'll stick with the five-group model. A vector of group membership given as 1-5 can be obtained as follows, and combined with the original data frame as the new last column for a quick glance at which mill belongs to which cluster group.

```
peacock.single.5<-cutree(peacock.single,k=5)
peacock$single.5 <- peacock.single.5
peacock
```

How well does our cluster analysis match with Peacock's? We can summarise the overall patterns by cross-tabulating the two vectors (the original column of groups A-E and our new groups 1-5).

```
addmargins(table(peacock.single.5,peacock[,13]))
```

As this cross-table shows, Peacock divided the mills into five groups with 3, 3, 6, 10, and 20 members. Our analysis divides them into five groups with 1, 1, 2, 3 and 35 members - see also the dendrogram. This latter rather would suggest that the analysis recognises a small handful of outliers with the large majority of mills being similar at the cut-off height. While it might be interesting to examine the “outliers”, this is admittedly less interesting in terms of thinking about the bulk of the mills.

1.5 Average and Wards methods

Lets do the analysis using the “average” and “Ward’s” methods and plot all the dendrograms side by side. You can drag and resize the window to make the plots more readable, the graphics should adjust automatically.

```
peacock.ave<-
hclust(dist(peacock.std,method="euclid"),method="ave")
peacock.ward<-
hclust(dist(peacock.std,method="euclid"),method="ward")

par(mfrow=c(1,3))
plot(peacock.single,main="Single Linkage")
rect.hclust(peacock.single,k=5)
plot(peacock.ave,main="Average Linkage")
rect.hclust(peacock.ave,k=5)
plot(peacock.ward,main="Ward")
rect.hclust(peacock.ward,k=5)
```

What do you reckon matches best Peacock? We can do the cross-tables for all of the results. Study again at the distribution of the Sum columns (Peacock’s) vs Sum rows (cluster analysis results).

```
peacock.ave.5<-cutree(peacock.ave,k=5)
peacock.ward.5<-cutree(peacock.ward,k=5)

addmargins(table(peacock.single.5,peacock[,13]))
addmargins(table(peacock.ave.5,peacock[,13]))
addmargins(table(peacock.ward.5,peacock[,13]))
```

These rather different results obtained are a health-warning about naively trusting the results of a statistical analysis. It should be patently obvious that statistical analysis, in archaeology or anything else, is not an appeal to a higher authority that produces “objective” answers. Your choice of method influences the results, as do many other steps in the research process from choices made in the initial data creation phase to the decision (arbitrary? informed?) to divide the population into a certain number of groups.

Here, the results of the cluster analysis invite you to consider the data from a particular perspective. But larger question is what archaeological interpretation this advances.

2. K-Means Clustering

A second and separate cluster analysis methods is k-Means clustering. Unlike the previous hierarchical method, k-Means cluster analysis starts with a priori deciding the number of clusters and then constructing an optimal solution (from the perspective of its mathematics) to divide the data into such a number.

2.1 Determining group number

The inbuilt functions in R do not provide a neat solution to determining how many groups the data should be divided into. Refer to your weekly reading (Conolly & Lake, pp. 170-2, as well as Shennan, ch 11) for a possible solution involving an examination in decrease of the total sum of squared distances vs increase in the number of groups. R packages do not have this as an inbuilt function, but if you copy-paste the following script it will produce a chart where the "elbow" can be identified.[2] You will need to install and load the **tidyverse** package (actually, a collection of many packages) first. It is very large, so it will take a bit of time.

```
library(tidyverse)

set.seed(12)
wss <- function(k) {
  kmeans(peacock.std, k)$tot.withinss
}
k.values <- 1:10
wss_values <- map_dbl(k.values, wss)
plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```

There does indeed appear to be an "elbow" (levelling) at the five-group point. However, the method is sensitive to the **seed number** that is set at the start of the calculation. Change the number within `set.seed()` and rerun the everything to

[2] See this website for the original code, as well other material for k-Means clustering using R: https://uc-r.github.io/kmeans_clustering

see how the graph changes. But you should see that a good number of times 5 groups is a good number to go for.

Now, carry out k-Means with 5 groups and look at the data. You could add the vector of groups to the original data frame, like at section 1.4 above. And then compare group sizeswith Peacock's groups.

2.2 Carrying out k-Means partitioning

```
peacock.k5<-kmeans(peacock.std,centers=5)
peacock.k5

addmargins(table(mills.k5$cluster, mills[,14]))
```

2.3 Looking at variable pairs

Another way to think about applying k-Means partitioning (and clustering/classification in general) is to consider comparing various pairs of variables (like height and width) see of groups emerge from these. To do this, we'll create a matrix of relationships.

First of all, we need to create a vector of colours. You can pick you own colours from a list you'll see by typing **`colours()`**.[3]

```
mycols<-c("black","red","blue","green","cyan")
```

Then plot the matrix. You'll want to maximise the window: it creates scatter point diagrams between every variable in the data.

```
plot(peacock[1:11],col=mycols[peacock.k5$cluster],pch=19, cex=0.5)
```

Five groups makes for pretty confusing diagrams, in this approach this does not seem to be ideal. Lets try it with a three group solution.

```
peacock.k3<-kmeans(peacock.std,centers=3)
plot(peacock[1:11],col=mycols[peacock.k3$cluster],pch=19, cex=0.3)
```

[3] Or see http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Do you think that clusters seem be emerging more clearly now? Lets pick one particular pair of variables - height and diameter - where this is especially clear and open it up a full window.

```
dev.new(device=pdf)
plot(peacock[,1]~peacock[,2],pch=16,col=mycols[peacock.k3$clus
ter])
```

Do you think the groups are clear? What if you look at other pairs, or use a different number of groups? What do you think this shows about the set of 42 mills as a whole? Do you think this is a good approach? This maps individual pair relationships as opposed to considering the data as a whole. Consider what advantages and disadvantages these different approaches have.

3. Principal Component Analysis

Next we will carry out PCA on the mills data. As with correspondence analysis, R has a straightforward function to carry it out, and in this case it is built into the basic R functionality without needing to download a separate package. Note that the data must be, again, standardised, but this is built into the `prcomp` function so you don't need to do it separately either.

3.1 Carry out PCA

```
peacockPCA<-prcomp(peacock[,1:11],center=T,scale=T)
```

We need to determine how many components (similar to "dimensions" in correspondence analysis) to keep, so do a scree plot, get the values of the eigenvalues and look at summary statistics. As a general rule of thumb the eigenvalue of 1 can be taken as a cut-off point, so quite neatly we here we will just consider the first two components.

```
plot(peacockPCA)
peacockPCA$sdev^2
summary(peacockPCA)
```

3.2 Plot PCA

Now call up a biplot of the data pattern. It is not very neat, but similar to correspondence analysis biplot it shows the relationships between the different mills (indicated by their number), and also overlays the graph with a relationship diagram of the variables.

```
biplot(peacockPCA,main="PCA biplot of Roman mills")
```

Consider the angle between the arrows, and how this indicates whether the variables are positively or negatively correlated, or not correlated.

3.3 Compare with other analysis

You can change the numbers into letters that give Peacock's groups (note: from the 13th column of the data frame, see the subsetting) to compare how they map onto PCA. What do you think?

```
biplot(peacockPCA,xlabs=peacock[,13], main="PCA biplot of mill
data with Peacock groups")
```

It is often a good idea to explore a dataset using multiple different methods, instead of relying one a single that may have particular in-built biases.

Here, we will combine and contrast PCA with the k-Means analysis. We can easily compare the Peacock groups with K-means groups on the PCA plot by attaching labels from the respective objects onto the plot. Again, you can drag and resize the window to get a better view.

```
par(mfrow=c(1,2))
biplot(peacockPCA,xlabs=peacock[,13])
biplot(peacockPCA,xlabs=peacock.k5$cluster)
```

Looking at the Peacock groups (A-E) and the groups created with k-Means partitioning, and how the data behaves in a PCA, what do you think? Does this suggest, for instance, that Peacock's division of some of the groups is particularly valid? Would you reconsider some group memberships?

3.4 Using FactoMineR for PCA

Finally, these graphs and pretty ugly and somewhat confusing, and we can make them nicer with the package FactoMineR, which also has functions for PCA (as well as CA).

```
library("FactoMineR")
```

The carry out the PCA using the FactoMineR function. Again, the data is standardised. You get a pop up of the individual and variables plots split into two windows.

```
peacockPCA_FTMR<-PCA(peacock[1:11])
```

I am personally not fond of having them as we separate windows. Close all the windows, and you can plot them side by side (click-drag by a corner to resize).

```
par(mfrow=c(1,2))
plot(peacockPCA_FTMR,choix='ind')
plot(peacockPCA_FTMR,choix='var')
```

Among other things, the FactoMineR plot handily indicates on the axis how much of the variation of axis contributes; these are the eigenvalues we looked at earlier, and similar to the dimensions in CA.

3.5 Diagnostics

An advantage of FactoMineR is that it also gives good sets of diagnostics and summary statistics. If you just type the object name `peacockPCA_FTMR` you'll get a nice list of diagnostic data you can consider. We could, for example, look at the eigenvalues. We'll use **`round()`** function to round the figures to 2 decimals for neatness and clarity.

```
round(peacockPCA_FTMR$eig,2)
```

You should also check which variables contribute to each axis. Here the first axis is particularly strongly correlated with diameter, the mouth aperture width and hopper slope. The second axis is most closely related with the hopper orifice (the hole through the middle). From this you can start working out how the shapes of the mills relate to each other; the first axis seems to relate to fatter, perhaps heavier and sturdier mill shapes.

```
dimdesc(peacockPCA_FTMR)
```

There are other diagnostics. Look at the component loadings and square loadings. The component loading tells us the correlation between the new variable we create (i.e. compress from multiple existing variables to simply the data) and the original variables. The square loading (which is the component loading squared) how much of the old variables is accounted for by the new variable (so conversely, what is lost in the compression at the level of individual variables). Shennan pp. 288-91 discusses these more at length (see Further reading folder). The eigenvalue of a component is the sum of the squared loadings for all of the individual variables on that component.

```
round(peacockPCA_FTMR$var$cor,2)
round(peacockPCA_FTMR$var$cos,2)
```

3.6 Colour plotting

Now, you can map cluster analysis results with PCA using the FactoMineR functionalities. First, lets assign some of those colours we placed into a vector earlier to Peacock’s groups, and call up the biplot of individual mills, colouring points by their Peacock group membership. You may want to close the open plot windows first.

```
cols2<-mycols[as.numeric(peacock[,13])]
plot(peacockPCA_FTMR,choix='ind',col.ind=cols2, title="PCA and
Peacock's Mills")
```

This is a lot easier to read than the black-and-white plots from above. Lets do the same mapping for the results of the k-Means cluster analysis as well.

```
cols3<-mycols[as.numeric(peacock.k5$cluster)]
plot(peacockPCA_FTMR,choix='ind',col.ind=cols3, title="PCA and
k-Means Mills")
```

Study the commands - can you see how they are constructed? Could you plot the results of some of the other cluster analysis you carried out on the PCA biplot as well?

Finally, save your data. You have finished with this week’s exercise.