Geometric Morphometrics and Archaeological Science

Workshop Two (July 2020)

Dr. Christian Steven Hoggard (University of Southampton, United Kingdom)

Introductory remarks

This guide provides a "hands-on" step-by-step introduction into the application of geometric morphometric (GMM) methodologies in archaeological science (as conducted through the R Environment). This guide will **provide an overview of some different methods of creating landmark data, before detailing three case studies (using 2D and 3D landmark data)**.

We will spend roughly 3-5 minutes per 'chunk'. Chunks can be used as a means of rendering R output into documents, or to simply display code for illustration. The chunks presented here will minimise error resulting from manual input and ensure everyone is at the same stage. To run a 'chunk' (displayed as a shaded area and representing a function or suite of actions), we can press the "Run Selected Chunk" button, represented by a play button, or alternatively use the shortcut `ctrl + enter` on the highlighted code. **When you complete a function please use a "thumbs up" emoji on Slack. If there is an issue please raise your query in the Zoom Chat**. We are allowing time between functions to ensure that all participants (of varying R knowledge) can keep up; if you finish a particular process early please explore the functions in the packages used throughout this workshop, or individual functions through the 'Help' tab in the 'Packages' window.

This practical constitutes the second (of three) workshops on GMM and Archaeological Science, organised by Lucy Timbrell and Christopher Scott and led by Dr. Christian Steven Hoggard.

About the Code, Packages and Data

One published dataset and two unpublished datasets are used in this workshop. The data from the first practical originates from: Vestergaard, C. and Hoggard, C.S. (2019). A Novel Geometric Morphometric (GMM) Application to the Study of Bronze Age Tutuli. *Danish Journal of Archaeology*, 8: 5-28.

Data for this publication is stored on the Open Science Framework (https://osf.io/fcp43/) and is stored (for ease) on the workshop repository (https://github.com/CSHoggard/-gmm_liverpool_2020/tree/master/workshop_two). Data for the second and third case studies is unpublished and stored on the workshop repository. This data is copyright protected under ownership law; please ask the repository owner (Dr. Christian Hoggard) for use beyond the remit of this workshop.

For this workshop we will be focusing on the analysis of two- and three-dimensional landmark data. The following packages are required:
* **geomorph** v.3.3.1 (analysis of landmark data)
* **Momocs** v.1.3.0 (analysis of landmark data)
* **tidyverse** v.1.3.0 (visualisation of data)
* **rio** v.0.5.16 (import files from GitHub)

As we are using the rio package we will not be required to download the data to a working directory, and setting our RStudio accordingly (as is standard practice). Through the execution of all chunks in this markdown document all data will be imported, analysed and visualised.

Once R and RStudio have been installed, and this markdown file opened within RStudio (through `File -> Open file`), we need to install the aforementioned packages. For this workshop we will install these packages

through the below 'chunk'.

```
install.packages("geomorph", repos = "http://cran.us.r-project.org")
install.packages("Momocs", repos = "http://cran.us.r-project.org")
install.packages("tidyverse", repos = "http://cran.us.r-project.org")
install.packages("rio", repos = "http://cran.us.r-project.org")
```

As the tidyverse and Momocs packages may take time to install given the size of the files *please ensure that these are downloaded prior the workshop.* Once installed we can now activate and use these packages through the `library()` function.

```
library(geomorph)
library(Momocs)
library(tidyverse)
library(rio)
```

Case Study 1: Tutuli in the Nordic Bronze Age (2D)

In this first case study we will examine 376 tutuli from the Nordic Bronze Age. Tutuli are small circular plates and were originally thought to have been designed for practical purposes e.g. shield-buckles (Rafn 1856). More recently, archaeologists have argued that they function as clothing accessories e.g. beltware and cape buttons (Bergerbrant 1999). Here we will focus on the strength of pre-existing classificatory schemes, specifically:

- How successful can the four groups (types A/C/D/E) be differentiated?
- How successful can different shapes be attributed to different periods within the NBA?

28 two-dimensional landmarks were digitised (in tpsDig2) from professional illustrations of tutuli cross-sections. As these shapes are typically symmetric, and given their abundance in catalogues (Aner et al. 1973, 1976-1978, 1981, 1986, 1991, 1995, 2001, 2005, 2008, 2011, 2014, Aner and Kersten 1979;Aner, Kersten and Neumann 1984; Aner, Kersten and Koch 1990; Aner et al. 1993), these cross-section diagrams represent a source of great interpretive potential.

To do this we will first import the data, perform the necessary data registration method (**Generalised Procrustes Analysis**), explore the main sources of shape variation through a **Principal Component Analysis (PCA)**, before testing the robustness of these groups through a **MANOVA (Multivariate Analysis of Variance)** and a **Discriminant Function Analysis (DFA)**.

We will first import the data into the R Environment. As this data is .tps in format we could use either `Geomorph::readland.tps` or `Momocs::import_tps()`. As we will use Momocs throughout this first case study the `Momocs::import_tps()` is your best choice. However, as we are downloading this data from the GitHub repository we will use `rio::import` as follows:

```
tutuli_lm <- rio::import("https://github.com/CSHoggard/-gmm_liverpool_2020/raw/master/workshop_two/tutul

tutuli_data <- rio::import("https://github.com/CSHoggard/-gmm_liverpool_2020/raw/master/workshop_two/tu
```

Note: for the purpose of this workshop I will detail in-text the function and its constitutent package e.g. `geomorph::readland.tps()`, however only the function is what will be 'used' so-to-speak e.g. `readland.tps()`. This helps you to understand what packages the functions originate from. With our data now in the R Environment we can now call our tpsdata object through the `base:: View` functions. The `base::View()` function will highlight the three constituent parts of the tps file: the 1) *Coo* (coordinate data), 2) *cur* (the curve data if applicable), and 3) *scale* (the scale data if present). It is the Coo data which we will take forward (size is not considered here), with the database, to examine shape variation.

We can examine the database using the `utils::head()`.

```
head(tutuli_data)
```

```
## # A tibble: 6 x 4
##   artefact_id site           date  classification
##   <chr>       <chr>          <chr> <chr>
## 1 1141III-1   "Valbyg\xe5rd" II    C
## 2 1141III-2   "Valbyg\xe5rd" II    A
## 3 1148-2      "Landsgrav"    II    C
## 4 1148-5      "Landsgrav"    II    D
## 5 1148-6      "Landsgrav"    II    C
## 6 1148-7      "Landsgrav"    II    C
```

In order to use these columns later we require them to be of class *factor*; at present they are `<chr>`, that is to say of type 'character', an object that contains character strings. We can do this quite easily using the `base::as.factor()` argument on the specific column (which can be called using the dollar sign). We can then double-check to ensure they are of class factor with the `base::is.factor()` as follows:

```
tutuli_data$date <- as.factor(tutuli_data$date)
tutuli_data$classification <- as.factor(tutuli_data$classification)

is.factor(tutuli_data$date)
```

```
## [1] TRUE
```

```
is.factor(tutuli_data$classification)
```

```
## [1] TRUE
```

Finally, we can convert our artefact id column into the row names using the `tibble::column_to_rownames` function.

```
tutuli_data <- column_to_rownames(tutuli_data, var = "artefact_id")
```

Our database is now formatted appropriately.

Central to Momocs are a specific suite of shape classes for: 1) *outlines* (OutCoo), *open outlines* (OpnCoo) and *landmarks* (LdkCoo), with often one class specific to your own dataset. While some operations in Momocs are generic and do not depend on one of these classes, many functions require your data to be one of these specific 'S3 objects'. In this instance our tps data is comprised of landmarks, and so we wish for our data to be `LdkCoo`, as to enable a Generalised Procrustes Analysis and subsequent exploratory and analytical procedures.

The coordinate data (coo) must therefore be turned into outline data through the `Momocs::Ldk()` function for the workflow to work. Once performed, we can then enter the object (here titled 'tutuli_shape') and examine its properties. In addition, for landmark visualisation we can input the links between the landmarks. This can be imported from a .csv file or notepad, however today we will create the links in-house, so to speak. This must be imported prior the `Momocs::Ldk()` function. the `base::rbind` function takes a series of vector data or a matrix and creates x number of columns. Each number corresponds to a landmark, with each paid being defined within one concatenated set of parentheses:

```
outline <- rbind(c(1, 2), c(2, 3), c(3, 4), c(4, 5), c(5, 6), c(6, 7), c(7, 8), c(8,
    9), c(9, 10), c(10, 11), c(11, 12), c(12, 13), c(13, 14), c(14, 15), c(15, 16),
    c(16, 17), c(17, 18), c(18, 19), c(19, 20), c(20, 21), c(21, 22), c(22, 23),
    c(23, 24), c(24, 25), c(25, 26), c(26, 27), c(27, 28), c(28, 1))
```

We can now perform the `Momocs::Ldk()` argument:

```
ldk_tutuli <- Ldk(tutuli_lm$coo, links = outline, fac = tutuli_data)
```

Note: we need to call the coordinate component of the object. If we wish we can now call the object and observe that there are 375 landmark configurations, each with 28 landmarks and 3 classifiers (including site name). We can also use square parentheses to call up individual objects. For example, if we wish to see the coordinate data for item #3 we can use `[3]`:

```
ldk_tutuli[3]
```

```
##       [,1] [,2]
##  [1,]  108   77
##  [2,]  108   61
##  [3,]  108   45
##  [4,]   99   39
##  [5,]   92   31
##  [6,]   86   24
##  [7,]   93   21
##  [8,]  101   21
##  [9,]  108   22
## [10,]  108   17
## [11,]  108   13
## [12,]   96   13
## [13,]   85   15
## [14,]   74   17
## [15,]   62   15
## [16,]   51   14
## [17,]   39   13
## [18,]   28   13
## [19,]   16   13
## [20,]    6   15
## [21,]   20   17
## [22,]   34   18
## [23,]   49   19
## [24,]   62   25
## [25,]   74   32
## [26,]   86   41
## [27,]   94   52
## [28,]  100   65
```

Now our data is in the R environment and in the appropriate class required for Momocs, we can examine the landmark configurations We can first look at all landmarks through the `Momocs::panel()` function (Note: this will take some time and it is advised not to use this function during the workshop). We can also use the `Momocs::inspect()` and `Momocs::coo_plot()` functions for additional visualisations (we will focus on these next week with our outline data). But for example:

```
coo_plot(ldk_tutuli[9], cex = 1.5, pch = 20, col = "grey", poly = TRUE, centroid = FALSE,
    border = TRUE)
```

We can now remove all factors external to shape (scaling, translation and rotation) through a *Generalised Procruses Analysis*. In Momocs there are a number of Procrustes procedures we could implement:

- Full Generalised Procrustes (`fgProcrustes`: default)
- Full Generalised Procrustes with sliding landmarks (`Momocs::fgsProcrustes`)
- Full Generalised Procrustes between two shapes (`Momocs::fProcrustes`)
- Partial Procrustes between two shapes (`Momocs::pProcrustes`)

Which procedure you use is depending on the types of shapes you have (sliders vs. non-sliders) and the nature of the superimposition. We require the `Momocs::fgProcrustes` function. This iterative process will return

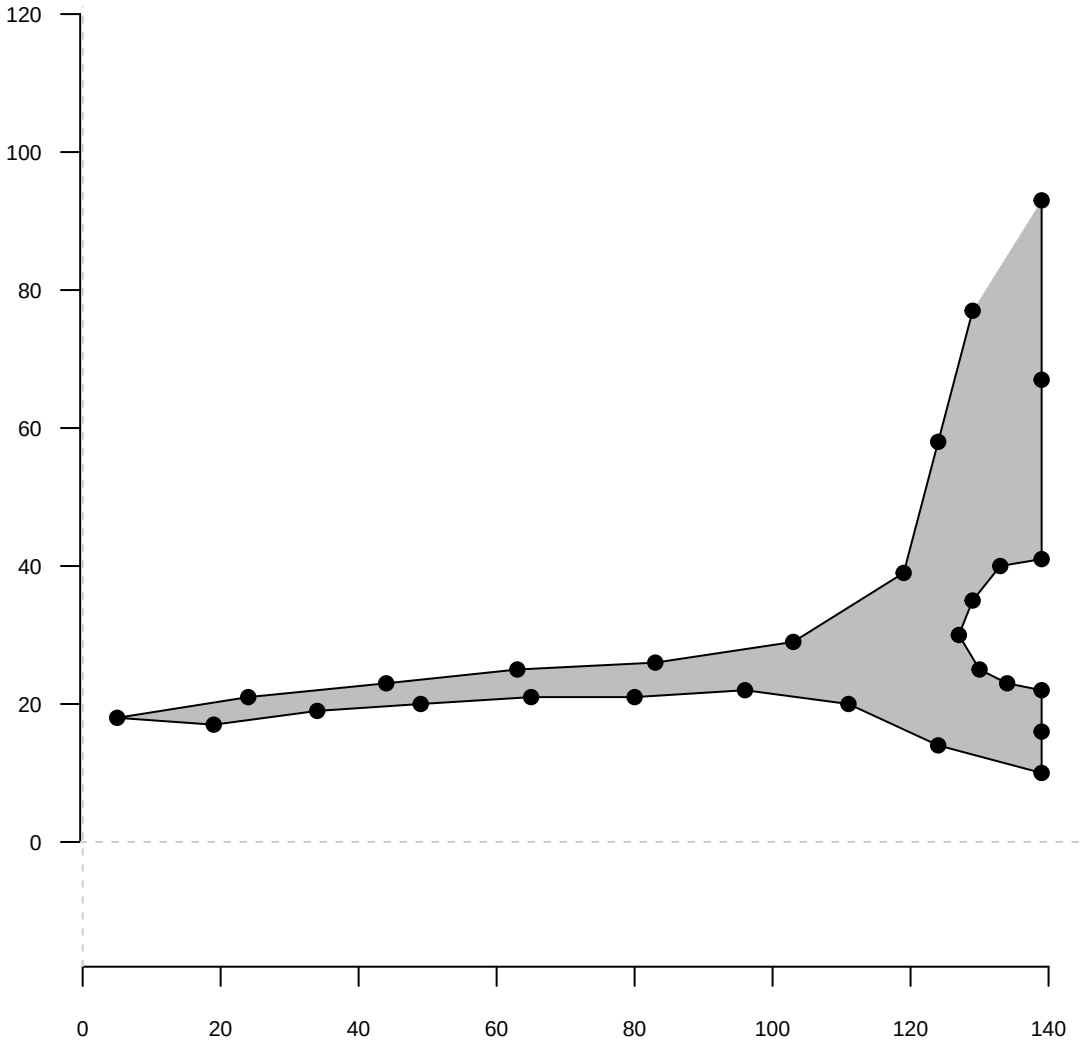


Figure 1: An example plot of the tutuli using the `Momocs::coo_plot()` function. The size and the shape of landmarks can be customised using the `cex` and `pch` arguments.

a number of useful components including the number of iterations, the pairwise distance measures, the mean shape configuration and, if scale has been included, the centroid sizes. This object is now of class *LdkCoe* (landmark coefficients), this is important when considering what functions can be used in Momocs. Note: we could alternatively import and analyse our data through geomorph, using the `geomorph::readland.tps()` and `geomorph::gpagen()` functions.

```
gpa_tutuli <- fgProcrustes(ldk_tutuli, tol = 0.1)
```

```
## iteration:  1    gain: 3544700
## iteration:  2    gain: 4.8593
## iteration:  3    gain: 0.48471
## iteration:  4    gain: 0.06586
```

Following this process we can visualise our shapes through the `Momocs::stack()` function. This image may appear variable, with landmarks failing to align, however there is great variability in the shapes of tutuli.

```
stack(gpa_tutuli, title = "")
```

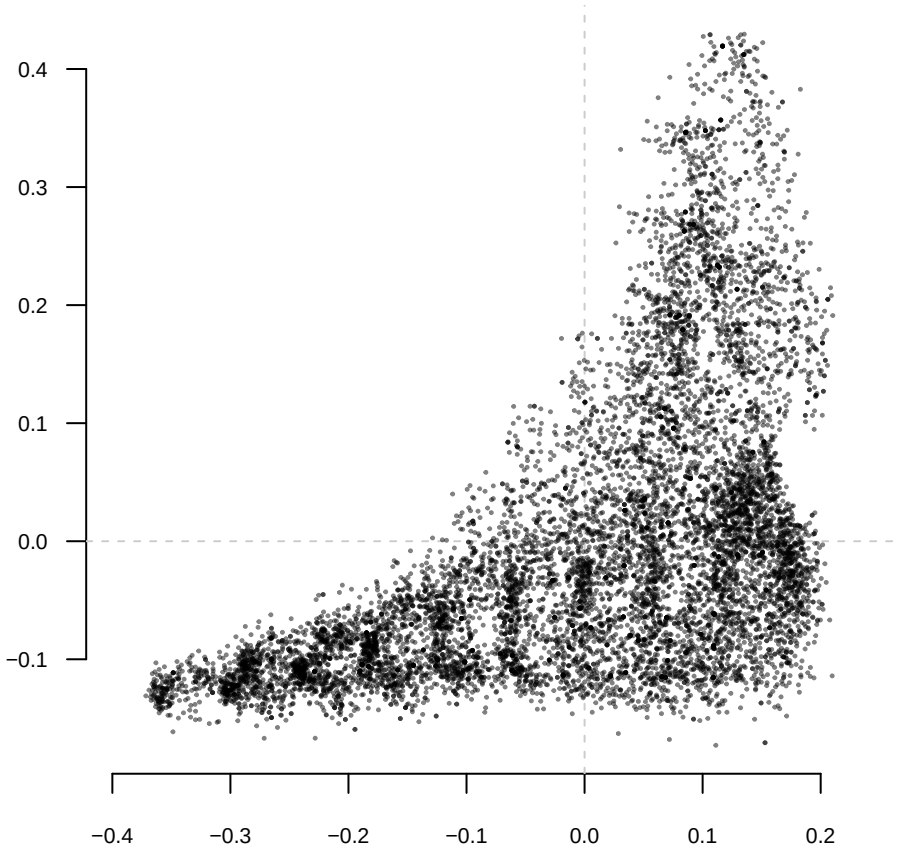


Figure 2: Generalised Procrustes Analysis (GPA) of all 375 tutuli (using the `Momocs::fgProcrustes` function). Note: the image may appear variable, with landmarks failing to align, however this is the result of great variability in the shapes of tutuli - see `Momocs::panel()`

With our Procrustes-alligned coordinates we can now examine shape variance among all examples, and between-group shape variance.

To examine the main sources of shape variation we will first perform a **Principal Component Analysis** (please refer to the first workshop for a detailed explanation of PCA). This analysis will allow us to understand the main sources of variation within our data, and how our groups (classification and date) map onto this morphospace. We can also begin to understand how much each source of variation contributes to the overall variation within a dataset.

Our landmark coefficients first need to be of class 'PCA'. We will therefore use the `Momocs::PCA()` function (using `base::prcomp()`). As our factors are embedded within the procrustes coordinates (as we linked them at the beginning) we do not need to add them here. The code goes as follows:

```
pca_tutuli <- PCA(gpa_tutuli)
```

Technical note: scaling and centering are `TRUE` by default (however this can be changed through the scale and center arguments).

With the PCA object we can assess the contribution of each source of shape variation through a **scree table** and **scree plot** using the `Momocs::scree()` and `Momocs::scree_plot()` arguments.

```
scree(pca_tutuli)
```

```
## # A tibble: 56 x 3
##     axis proportion cumsum
##    <int>      <dbl>  <dbl>
##  1     1    0.688    0.688
##  2     2    0.155    0.843
##  3     3    0.0528   0.896
##  4     4    0.0341   0.930
##  5     5    0.0191   0.949
##  6     6    0.0153   0.964
##  7     7    0.0107   0.975
##  8     8    0.00721  0.982
##  9     9    0.00430  0.986
## 10    10    0.00256  0.989
## # ... with 46 more rows
```

Here we can see that the first source of shape variance (the first principal component) accounts for 68.76% of all shape variation within our dataset, and the first six sources accounting for greater than 95% cumulative shape variance. We can also visualise this in bar graph form:

```
scree_plot(pca_tutuli, 1:10)
```

We can now plot of PCA through the `Momocs::plot_PCA()` function. This plot highlights that the four different classifications can be teased apart quite successfully in the first two principal components. We can also see that first principal component extends from narrow thin-lipped tutuli to high-peaking tutuli, with the second principal component accounting for the cross-section cavity. Convex hulls are visualised to show the distribution of the group in its entirety, however these are prone to outliers and confidence ellipses would be a preferred visual tool (both can be achieved through the `Momocs::plot_PCA()` tools). Many of these arguments (from initialisation to visualisation) can be handled through Dplyr's forward-pipe operator (%>%). You can use this operator to pass the left-hand side input through the right-hand side operator. This will be demonstrated during the practical (for teaching purposes we're taking the long way around).

```
plot_PCA(pca_tutuli, axes = c(1, 2), ~classification, chull = FALSE, chullfilled = TRUE,
    morphospace_position = "range_axes", zoom = 1)
```

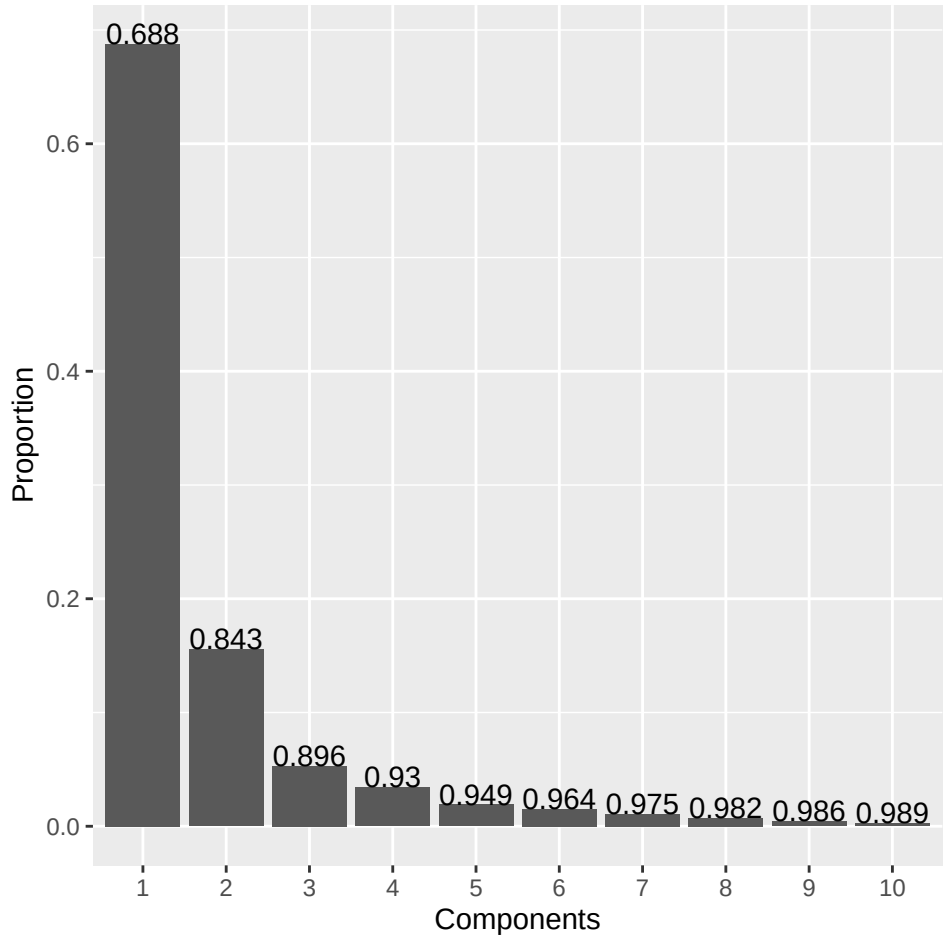
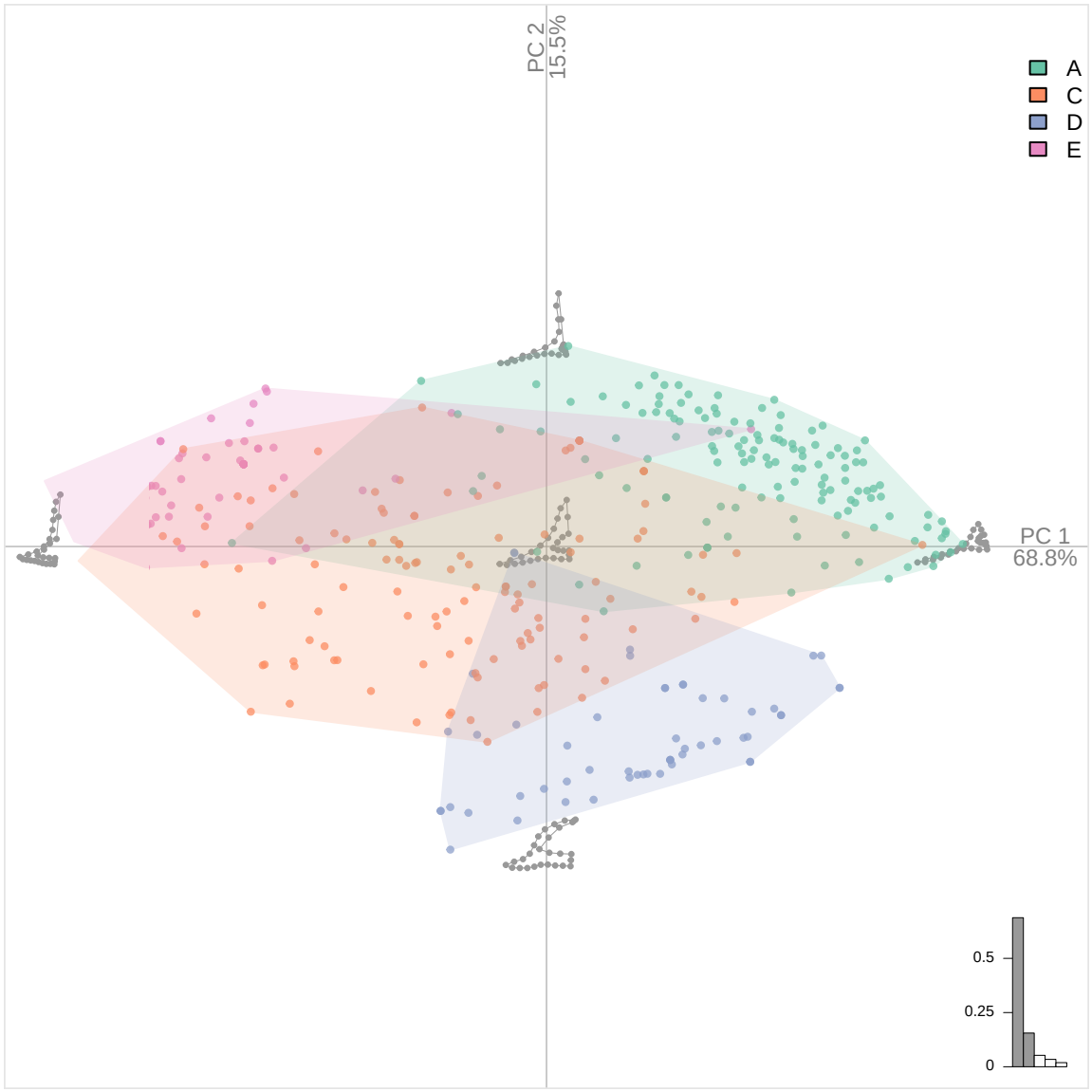


Figure 3: A scree plot of the first ten principal components (using the `Momocs::scree_plot()` function



It is important to note that these sources are shape variation pertain to the entire group, and not specific classifications. It is therefore important to examine other principal components in order to identify principal components which are good discriminators (i.e. those that are archaeologically relevant). The axes on this plot can be changes through amending the `axes = c(1,2)` argument to whichever principal components are best.

We can repeat this process for periodisation, as follows:

```
plot_PCA(pca_tutuli, axes = c(1, 2), ~date, chull = FALSE, chullfilled = TRUE, morphospace_position = "
    zoom = 1)
```

Again, clustering can be seen with each period of the Nordic Bronze Age with stage three featuring more negative PC1 vaues and stage two corresponding with more positive values.
We can also visualise these principal components, and the variance within different archaeological units, in an alternative way, through the `Momocs::boxplot()` function. Boxplots have the advantage of condensing a multi-dimensional space into two axes (subject to graphical parameters). The code is as follows:

```
boxplot(pca_tutuli, ~date, nax = 1:5)
```

If we wish, we can use `Momocs::plot_MSHAPES()` function for displaying the mean shapes for both factors.

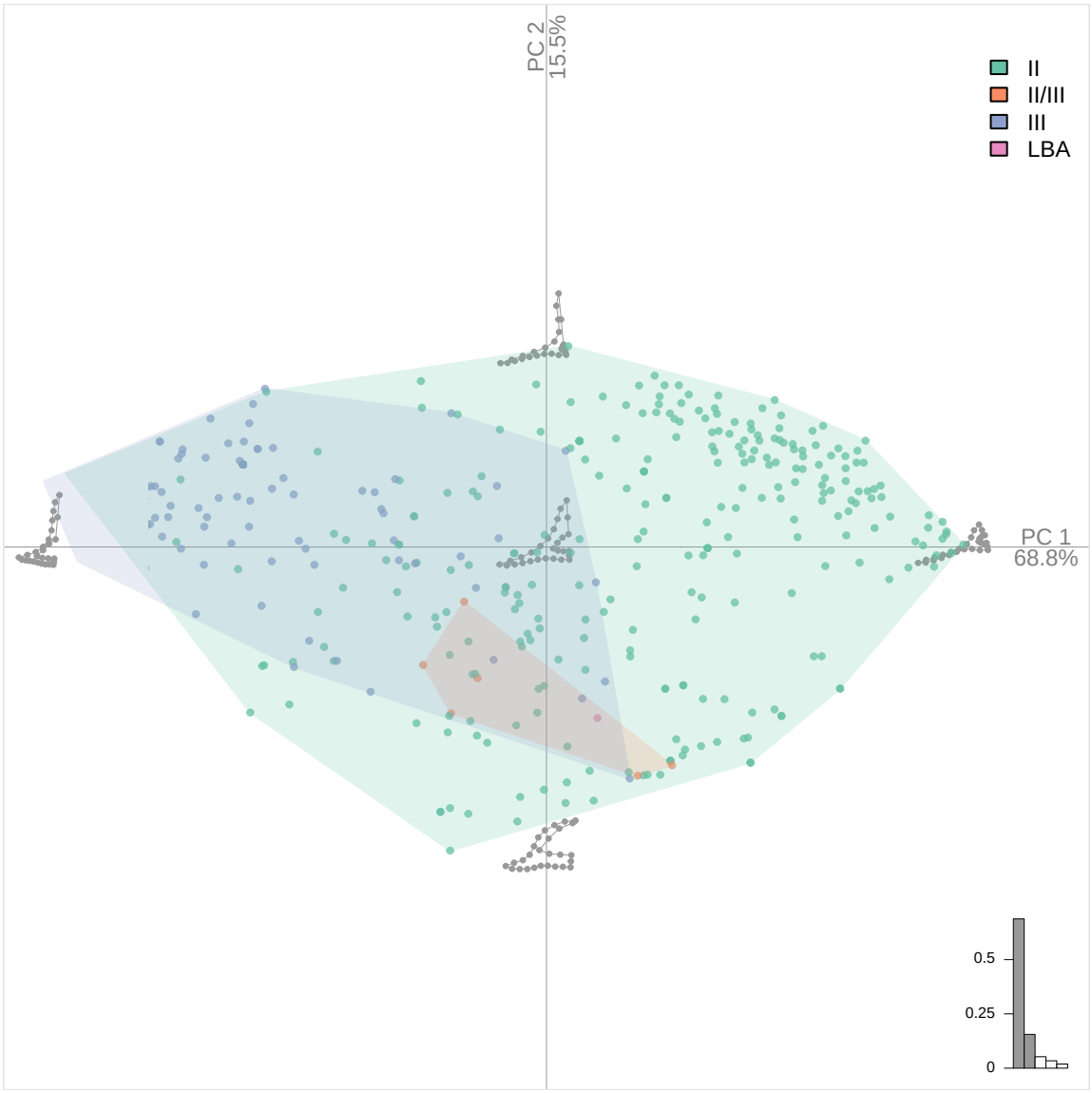


Figure 4: A Principal Component Analysis (PC1 vs. PC2) of NBA tutuli. Colours correspond to period in the Nordic Bronze Age (NBA).

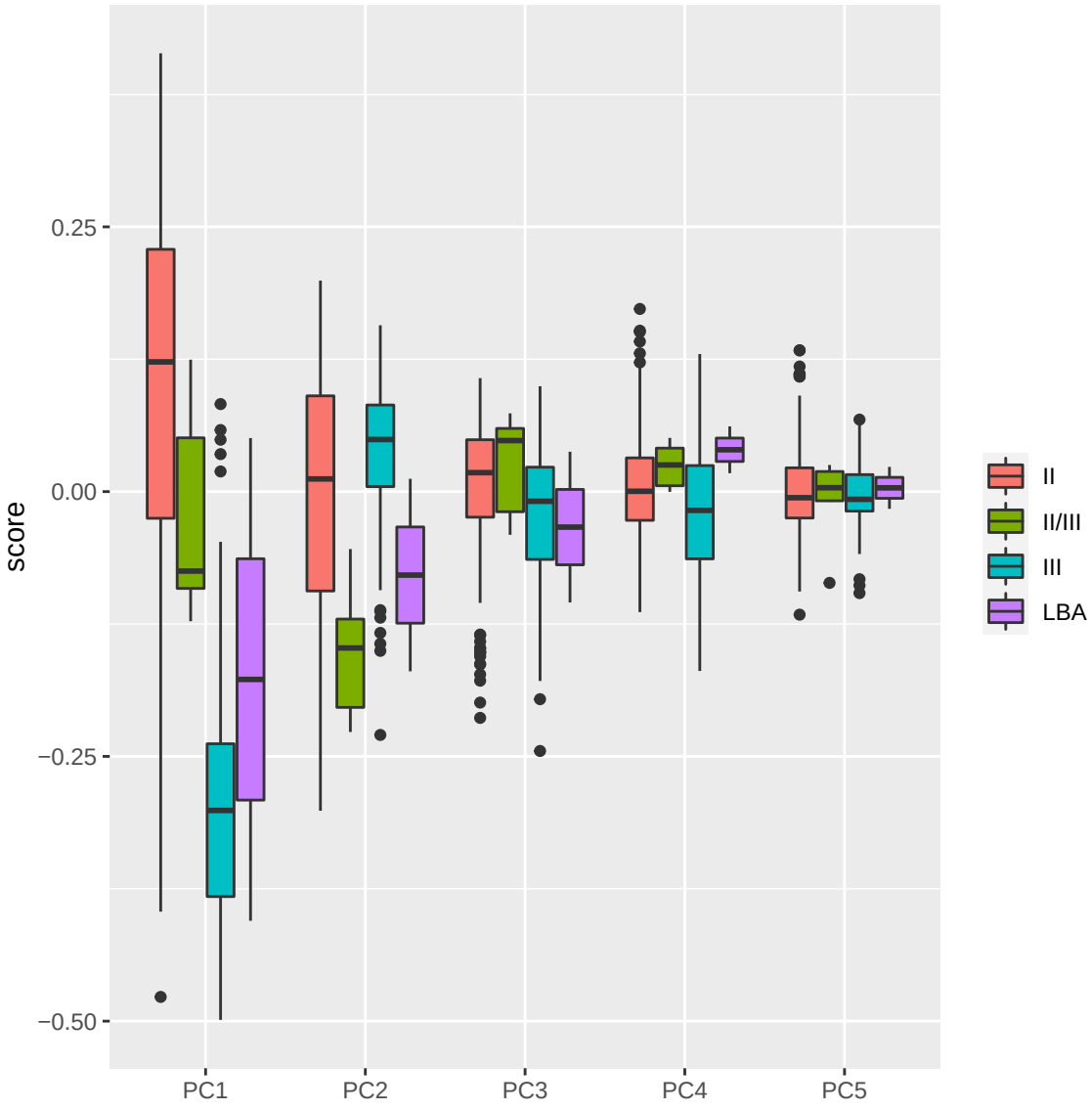


Figure 5: A Principal Component Analysis (PC1 vs. PC2) of NBA tutuli represented as a box-plot. Colours correspond to period in the Nordic Bronze Age (NBA).

A variety of methods are available however `Momocs::plot_MSHAPES()` is the most convenient method. Here I exemplify piping of mean shapes for classification:

```
gpa_tutuli %>% MSHAPES(~classification) %>% plot_MSHAPES()
```

Figure 6: Mean shapes of different tutuli classifications (as plotted through `Momocs::plot_MSHAPES()`

As we highlighted in the first workshop, PCA explores differences in shape variation irrespective of group composition (i.e. *a priori* groupings). Through a discriminant analysis we can examine differences in shape as based on their maximum group seperation (between-group variation in contrast to within-group variation). In Momocs, we use the `Momocs::LDA()` function on either the landmark coefficients (Procrustes Coordinates) or the PCA scores to produce our class accuracy, plots and correction scores. There is no correct answer as to which to use, it depends on the data you wish to examine. In using the PCA scores it is possible to retain a number of components that are deemed important, this can be either: 1) the first nth components, 2) the number of components representing a certain level of shape variance (e.g. 95%, 99%, 99.9%), or 3) all principal components. The coefficients, in contrast would encapsulate all shape data.

With greater levels of data you may include a degree of statistical noise, with smaller unimportant variables

taking precedence, and so an optimal level of data is necessary if you persue PCA scores (see Kovarovic et al. 2011 for more information).

First we must create a LDA object (or alternatively perform piping):

```
lda_tutuli_class <- LDA(gpa_tutuli, ~classification)
lda_tutuli_date <- LDA(gpa_tutuli, ~date)
```

We can now examine different aspects of our discriminant analysis data, including the cross-validation table (actual vs. predicted categories for artefacts) and the proportion of correctly classified individuals.

```
lda_tutuli_class$CV.correct
```

```
## [1] 0.8853333
```

```
lda_tutuli_class$CV.ce
```

```
##         A         C         D         E
## 0.9264706 0.8130081 0.9636364 0.8688525
```

```
lda_tutuli_date$CV.correct
```

```
## [1] 0.88
```

```
lda_tutuli_date$CV.ce
```

```
##        II    II/III       III       LBA
## 0.9236364 0.1666667 0.8152174 0.0000000
```

When we use the `CV.correct` argument we see that 88.53% of tutuli can be appropriately differentiated by the Montelius system. We can examine this in further detail through the `CV.ce` argument. Higher percentages of success can be seen for groups D (96.37%) and A (92.65%), and admirable success rates for groups E (86.89%) and C (81.30%). For designated periodisation, these can be successfully discriminated 88% of the time with 92.36^ and 81.52% for the second and third parts of the NBA, and little success with the transitional and LBA examples (this is the result of an incredibly low sample size). Pursuing this further through Machine Learning methodologies for undoubtedly increase the success of these groups, and highly possible anomalies in their classification.

Finally, for this case study let's test our interpretations through a MANOVA (through PC scores). A Procrustes ANOVA could be used through geomorph, however as we have used Momocs throughout this first case study let's finish off here with a MANOVA (the next examples will perform a Procrustes ANOVA.

For this we require the `Momocs::MANOVA()` function. Again, this can be piped using Dplyr's operator or done manually. A manual version would look like this:

```
manova_tutuli_class <- MANOVA(pca_tutuli, ~classification)
manova_tutuli_date <- MANOVA(pca_tutuli, ~date)
manova_tutuli_class_pw <- MANOVA_PW(pca_tutuli, ~classification)
```

```
## $stars.tab
##   A C   D   E
## A   *** *** ***
## C       *** ***
## D           ***
##
## $summary (see also $manovas)
##       Df Pillai approx F num Df den Df    Pr(>F)
## A - C  1 0.7468    66.22     11    247 3.209e-67
## A - D  1 0.9253   201.60     11    179 1.350e-94
## A - E  1 0.8877   132.93     11    185 1.315e-81
```

```
## C - D  1 0.8413    79.99     11    166 1.984e-60
## C - E  1 0.6568    29.92     11    172 1.985e-34
## D - E  1 0.9755   377.16     11    104 1.782e-78
```

```
manova_tutuli_date_pw <- MANOVA_PW(pca_tutuli, ~date)
```

```
## $stars.tab
##        II II/III III LBA
## II        **     *** .
## II/III           *** -
## III                 .
##
## $summary (see also $manovas)
##              Df  Pillai approx F num Df den Df    Pr(>F)
## II - II/III   1 0.03455    4.974      2    278 7.545e-03
## II - III      1 0.53275  207.512      2    364 7.214e-61
## II - LBA      1 0.01795    2.504      2    274 8.362e-02
## II/III - III  1 0.31699   22.046      2     95 1.365e-08
## II/III - LBA  1 0.18539    0.569      2      5 5.989e-01
## III - LBA     1 0.05095    2.443      2     91 9.260e-02
```

We have used the `Momocs::MANOVA_PW()` function to include pairwise values, highlighting the statistical relationships between factor sub-groups

```
manova_tutuli_class
```

```
##            Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
## fac         3           9.5626   104.22     33   1079 < 2.2e-16 ***
## Residuals 371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
manova_tutuli_date
```

```
##            Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
## fac         3           1.6485   17.967     33   1079 < 2.2e-16 ***
## Residuals 371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
manova_tutuli_class_pw
```

```
## $manovas
## $manovas$`A - C`
##            Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i       1 0.74677   66.216     11    247 < 2.2e-16 ***
## Residuals 257
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`A - D`
##            Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i       1 0.92531    201.6     11    179 < 2.2e-16 ***
## Residuals 189
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`A - E`
```

```
##            Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i       1 0.88769   132.93     11    185 < 2.2e-16 ***
## Residuals 195
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`C - D`
##            Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i       1 0.84129   79.993     11    166 < 2.2e-16 ***
## Residuals 176
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`C - E`
##            Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i       1 0.65677   29.921     11    172 < 2.2e-16 ***
## Residuals 182
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`D - E`
##            Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i       1 0.97555   377.16     11    104 < 2.2e-16 ***
## Residuals 114
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## $summary
##       Df    Pillai  approx F num Df den Df       Pr(>F)
## A - C  1 0.7467651  66.21626     11    247 3.209365e-67
## A - D  1 0.9253099 201.59734     11    179 1.350094e-94
## A - E  1 0.8876913 132.93138     11    185 1.315039e-81
## C - D  1 0.8412888  79.99318     11    166 1.984097e-60
## C - E  1 0.6567737  29.92065     11    172 1.985287e-34
## D - E  1 0.9755450 377.15568     11    104 1.782326e-78
##
## $stars.tab
##   A C   D   E
## A   *** *** ***
## C       *** ***
## D           ***
##
## $pvalue.tab
##    A            C            D            E
## A NA 3.209365e-67 1.350094e-94 1.315039e-81
## C NA           NA 1.984097e-60 1.985287e-34
## D NA           NA           NA 1.782326e-78
```

```
manova_tutuli_date_pw
```

```
## $manovas
## $manovas$`II - II/III`
##            Df   Pillai approx F num Df den Df   Pr(>F)
## fac.i       1 0.034547   4.9738      2    278 0.007545 **
```

```
## Residuals 279
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`II - III`
##           Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i      1 0.53275   207.51      2    364 < 2.2e-16 ***
## Residuals 365
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`II - LBA`
##           Df  Pillai approx F num Df den Df  Pr(>F)
## fac.i      1 0.01795   2.5041      2    274 0.08362 .
## Residuals 275
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`II/III - III`
##           Df  Pillai approx F num Df den Df    Pr(>F)
## fac.i      1 0.31699   22.046      2     95 1.365e-08 ***
## Residuals 96
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $manovas$`II/III - LBA`
##           Df  Pillai approx F num Df den Df Pr(>F)
## fac.i      1 0.18539  0.56897      2      5 0.5989
## Residuals  6
##
## $manovas$`III - LBA`
##           Df   Pillai approx F num Df den Df Pr(>F)
## fac.i      1 0.050951   2.4428      2     91 0.0926 .
## Residuals 92
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## $summary
##               Df     Pillai    approx F num Df den Df       Pr(>F)
## II - II/III    1 0.03454687   4.9738463      2    278 7.544593e-03
## II - III       1 0.53274886 207.5121620      2    364 7.213840e-61
## II - LBA       1 0.01795008   2.5041101      2    274 8.361771e-02
## II/III - III   1 0.31699442  22.0455516      2     95 1.365061e-08
## II/III - LBA   1 0.18539396   0.5689681      2      5 5.989203e-01
## III - LBA      1 0.05095139   2.4427497      2     91 9.260295e-02
##
## $stars.tab
##         II II/III III LBA
## II         **     *** .
## II/III            *** -
## III                   .
##
## $pvalue.tab
```

```
##         II      II/III          III        LBA
## II      NA 0.007544593 7.213840e-61 0.08361771
## II/III  NA          NA 1.365061e-08 0.59892035
## III     NA          NA           NA 0.09260295
```

These again highlight the degree with which different categories can be partitioned on the basis of cross-sectional shape. We can therefore conclude that the date and classification categories which archaeologists perscribe tutuli to work (that is not to say that they are chronologically correct just that the groupings work!).

Case Study 2: Cranial morphology vs. sex (3D)

In this next case study we will investigate three-dimensional shape changes in cranial morphology, and differences in cranium according to sex. As a simple demonstration we will use six crania (2 female and 4 male).

In the previous case study we used Momocs to investigate our two-dimensional landmark data (in .tps format). It provided the most immediate form of analysis for our data, and allowed a number of interesting visualisations. In this example we will use geomorph to examine our three-dimensional data (in .ply ASCII format). Geomorph relies more on base R (in comparison to Momocs which is more 'tidy'), and so our grammar may change in areas.

Files of .ply format can be fed into R through `geomorph::read.ply` through the sample code (providing that the data is saved to a working:

```
SK1 <- read.ply("skull_1.ply", ShowSpecimen = FALSE, addNormals = FALSE)
```

With the SK1 object we can now begin digitising landmarks onto the cranium. In this case study we will digitise 23 landmarks (refer to the landmark guides and documentation in the GitHub repository). For increased resolution we will include 200 surface sliding semilandmarks. Please see Gunz et al. (2005) and Mitteroecker and Gunz (2009) for more information.

In order to digitise our landmarks through this procedure we will first need to build a template for the digitisation of landmarks across specimens. This will be performed through the `geomorph::buildtemplate` function. This function requires an object of class mesh3d/shape3d (our .ply file), the number of fixed landmarks (23) and the number of surface.sliders (200). A graphical user interface (GUI) then appears allowing you to digitise your 23 landmarks; the 200 surface sliding landmarks will be automatically placed following this digitisation process (and for every subsequent specimen). Please refer to the R documentation (`?geomorph::buildtemplate`) for more information. This will be demonstrated throughout the workshop. I have included all six .ply files to allow digitisation following the workshop.

The resulting object is stored in the R Environment, with an .nts file saved into the working directory. Once the first skull has been digitised you continue the process with the `geomorph::digitsurface()` function.

When all the landmarks are digitised we can feed them all in together using the `geomorph::readmulti.nts`. For the purpose of the practical this object has been created in an .rds file and will be imported from the workshop GitHub repository (like in the previous case study.

```
skull <- rio::import("https://github.com/CSHoggard/-gmm_liverpool_2020/raw/master/workshop_two/skull.rds
```

As you will observe in the environment the data is in n x d x s, with n being the number of landmarks in total, d being the dimensions explored, and s being the sample size. This is an array, a series of rows, columns and layers. In this example, if we wish to view a particular specimen we would use our square parentheses but this time include two commas (as we're interested in the third aspect of this array) e.g. for the first specimen we would code…

```
skull[, , 1]

# OR
```

```
rgl.open()
rgl.points(skull[, , 1], r = 0.2)
```

If a sufficient number of examples were included we could also use the `geomorph::plotOutliers` argument to examine which specimens have a Procrustes Distance from the mean (incase of a landmarking issue, for example).

A further two objects are necessary: the metadata (the .csv table with the skull data) and the slider file (denoting which landmarks slide). Again, with the the `rio::import()` function we can include them into this workspace:

```
surface_lm_skull <- rio::import("https://github.com/CSHoggard/-gmm_liverpool_2020/raw/master/workshop_t

skull_data <- rio::import("https://github.com/CSHoggard/-gmm_liverpool_2020/raw/master/workshop_two/skul
```

We need to inspect the skull_data to verify that are columns are correct. Using the `base::head()` function we realise we again need to convert our sex column into `factor` class (we will not worry about location as we are only investigating sex here). Like previous, we can use the `base::is.factor()` argument to verify that our change is correct.

```
head(skull_data)
```

```
##      Code Location Sex Age
## 1 skull_1        A   F  30
## 2 skull_2        B   M  40
## 3 skull_3        B   F  60
## 4 skull_4        B   M  50
## 5 skull_5        C   M  50
## 6 skull_6        C   M  40
```

```
skull_data$Sex <- as.factor(skull_data$Sex)
is.factor(skull_data$Sex)
```

```
## [1] TRUE
```

With the landmarks not imported into the R Environment, and our metadata in the correct format we can now begin registering our coordinate data through a Generalised Procrustes Analysis. Previously we used `Momocs::fgProcrustes()` with our two-dimensional coordinate data, for this dataset we will use the `geomorph::gpagen()` function. We input our landmark data and include our surface sliding semilandmark data.

```
gpa_skull <- gpagen(skull, Proj = TRUE, ProcD = TRUE, curves = NULL, surfaces = surface_lm_skull)
```

```
##   |                                                                              |
```

A class of gpagen data is returned. this includes the Procrustes coordinates, the centroid size, aspects of the dataset including the variance-covariance matrix and its format. We can inspect the result through the `graphics::plot()` function and customise the output using the rgl package.

```
plot(gpa_skull)
```

Using the `geomorph::mshape()` we can extract the mean shape (PCA graph centroid) from our skulls as follows:

```
mean_skull <- mshape(gpa_skull$coords)
```

One can then use the plotting code to produce a number of landmark positions (from mean shape to a specific specimen, for example), or to add links and review landmark positions.

We can now inspect variation in our skull shape through a PCA. In geomorph, a PCA is formed through the `geomorph::gm.prcomp()` argument, with the input being our Procrustes coordinates. If phylogenetic data is provided then a Phyogenetically-Aligned Principal Component Analysis (PaCA) can be performed, however we are using a traditional PCA based on OLS-centering and projection.

```
pca_skull <- gm.prcomp(gpa_skull$coords)
summary(pca_skull)
```

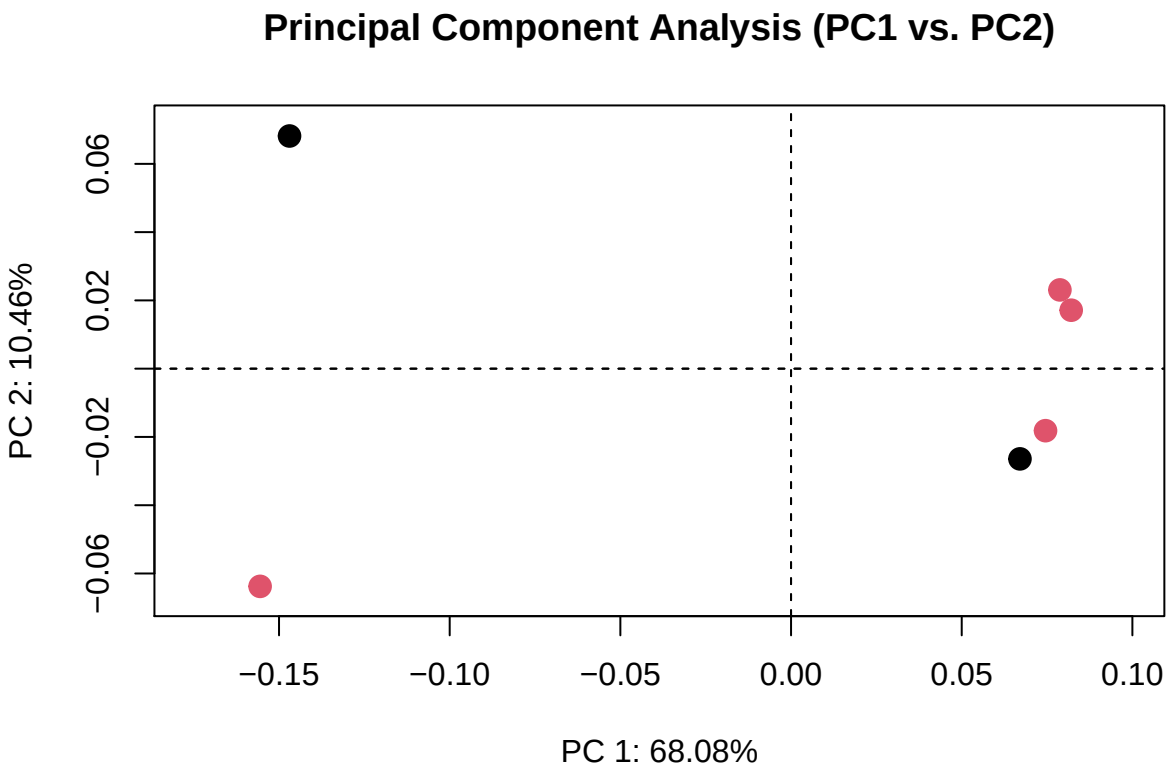
```
##
## Ordination type: Principal Component Analysis
## Centering and projection: OLS
## Number of observations 6
## Number of vectors 6
##
## Importance of Components:
##                             Comp1       Comp2       Comp3      Comp4
## Eigenvalues            0.01375499 0.002113578 0.001887654 0.00141315
## Proportion of Variance 0.68083677 0.104616670 0.093434000 0.06994731
## Cumulative Proportion  0.68083677 0.785453437 0.878887437 0.94883475
##                              Comp5        Comp6
## Eigenvalues            0.001033695 2.937654e-33
## Proportion of Variance 0.051165252 1.454063e-31
## Cumulative Proportion  1.000000000 1.000000e+00
```

We can observe that in this example the first principal component accounts for 68.08% of all cumulative shape variation, with the first five components accounting for all variation (this is surprising as the number of components is limited by your degrees of freedom).

We can also look at the three-dimensional configurations of different aspects of the principal components through an investigation of the `$shapes` component of the object.

With this we can now plot our data in base R graphics as below (or use `rgl::plot.3d()` if we're being fancy! As our .nts files are in the correct order as the dataset we can attribute colour as attached. Match functions will be necessary if these are not alligned (with the id label mirrored in the .nts and dataset files).

```
plot(pca_skull, axis1 = 1, axis2 = 2, main = "Principal Component Analysis (PC1 vs. PC2)",
    pch = 19, cex = 1.5, col = skull_data$Sex, legend = TRUE)
```



Changes in shape can be inferred through `geomorph::plotRefToTarget`, for example:

```
plotRefToTarget(mean_skull, pca_skull$shapes.comp1$max)
```