Geometric Morphometrics and Archaeological Science

Workshop Three (July 2020)

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

Introductory remarks

This guide provides a "hands-on" step-by-steph introduction into the application of geometric morphometric (GMM) methodologies in archaeological science (as cnducted through the R Environment). This guide will **provide an overview into analysing outline (open and closed) data through two case studies**.

Like the previous workshop, 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 all participants are 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.

For participants: **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 third and final workshop on GMM and Archaeological Science, organised by Lucy Timbrell and Christopher Scott and led by Dr. Christian Steven Hoggard. This markdown, along with all resources (and recordings of each workshop) can be found on the workshop GitHub repository: https://github.com/CSHoggard/-gmm_liverpool_2020.

We thank the Arts and Humanities Research Council North West Consortium Doctoral Training Partnership (AHRC NWCDTP) for supporting this workshop.

About the Code, Packages and Files

We will examine one case-study throughout this practical:

Inovaite et al. (2020). All these Fantastic Cultures? Research History and Regionalization in the Late Palaeolithic Tanged Point Cultures of Eastern Europe, European Journal of Archaeology, 23 (2): 162-185. (https://doi.org/10.1017/eaa.2019.59).

The complete repository for this dataset can be found on GitHub (https://github.com/CSHoggard/-Eastern-Europe-Tanged-Points), in addition to the Open Science Framework (https://osf.io/agrwb/).

For this final practical, three packages are required:

- **Momocs** v.1.3.0 (https://cran.r-project.org/web/packages/Momocs/index.html)
- **tidyverse** v.1.3.0 (https://cran.r-project.org/web/packages/tidyverse/index.html)
- **rio** v.0.5.16 (https://cran.r-project.org/web/packages/rio/index.html)

Similarly to the last practical we will be using the rio package to import our data into the environment. We therefore will not be required to set the working directory.

With R and RStudio 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':

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

About the Data

This data was composed to assess the robustness of cultural taxonomies in the Final Palaeolithic period of Eastern Europe, as portrayed through tanged point variants. 250 illustrations of tanged points were synthesised into one .tps file (using tpsUtil) before being digitised in tpsDig2 (https://life.bio.sunysb.edu/morph/soft-dataacq.html). Through the **outline object** function all tanged points were converted into outlines (a series of semi-landmarks, equidistant and placed using an algorithm, and of varying size). A database of all examples and their respective cultural assignment is also provided.

Creating and Importing GMM Data: Alternative Approaches

There are a number of ways with which outline data can be created. Here, all examples were digitised in tpsDig2 however functions in Momocs can also be used to extract outlines from silhouette data (black shape on white background). These are the the `Momocs::import_jpg()` and `Momocs::import_jpg1()` functions, derived from the `import_conte()` function translated into R by Claude (2008). Open outlines can be digitised through an amended `import_conte()` argument.

There are also a number of different methods with which outline daya can be imported into the R Environment. Here, for ease and replicability, the outline data (in .tps format) was stored on a GitHub repository and directly fed into the R environment, utilising the `Momocs::import_tps()` function in a .rds file. Other ways to import .tps data (if saved locally) include the above function, `geomorph::readland.tps()` and rewriting tools in Momocs e.g. `Momocs::rw_rule()`. Data from stereomorph can also be imported through the `Momocs::import_StereoMorph_ldk()` and `Momocs:import_StereoMorph_curve()` functions.

Stage 1: Importing the GMM Data into the R Environment

Let's import the tps outline data and the dataset from the GitHub workshop repository using the `rio::import()` functions

```
tpsdata <- rio::import("https://github.com/CSHoggard/-gmm_liverpool_2020/raw/master/workshop_three/tpsl

database <- rio::import("https://github.com/CSHoggard/-gmm_liverpool_2020/raw/master/workshop_three/data
```

Stage 2: Data Cleaning

With both objects now in the R Environment we can now call our tpsdata object through the `base:: View` functions (however given the size of the object it is best advised to not use this argument). 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 necessary), and 3) *scale* (the scale data if present). This echoes the structure of the first case study in the second workshop.

We can inspect our database using the `utils::head()` function, this will highlight the first six rows of the base.

```
head(database)
```

```
## # A tibble: 6 x 10
##      ID Site  Context Longitude Latitude Country Archaeological_~ File_Name
##   <dbl> <chr> <chr>       <dbl>    <dbl> <chr>   <chr>            <chr>
## 1     1 Balt~ Baltaš~      24.0     54.0 Lithua~ Baltic Magdalen~ Bal.1
## 2     2 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.1
## 3     3 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.3
## 4     4 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.4
## 5     5 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.5
## 6     6 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.6
## # ... with 2 more variables: Reference <chr>, Notes <chr>
```

We can observe that the data we wish to examine, *Archaeological_Unit* is `<chr>`, that is to say of type 'character' and not `<fctr>` ('factor'), the class which is required for our analysis. This can be corrected through the `base::as.factor()` function, as follows:

```
database$Archaeological_Unit <- as.factor(database$Archaeological_Unit)

is.factor(database$Archaeological_Unit)
```

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

Note: The factors can also be added during data importing through *readr* functions.

We can also inspect the number of different archaeological units within our dataset through the `base::summary()` function. This highlights the number of tanged points in each group. With certain taxonomic units rarely used this is reflected in the low sample sizes for certain groups e.g. Vyshegorian.

```
summary(database$Archaeological_Unit)
```

```
##      Baltic Magdalenian Bromme (Eastern Europe) Bromme (Western Europe)
##                      36                       9                      49
##                  Grensk             Krasnosillya              Perstunian
##                      55                      29                       4
##     Pitted Ware (Type A)               Podolian             Vyshegorian
##                      24                      14                       8
##              Wolkushian
##                      22
```

Stage 3: Creation of the "Out" object

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. Previous we wanted our objects to be of class LdkCoo, in this instance however our tps data is comprised of outlines, and so we wish for our data to be `OutCoo`, as to enable efourier (elliptic Fourier) analyses. Other analyses including rfourier (radii Fourier) or tfourier (tangent angle Fourier) analyses can be conducted through this process but for this workshop we're only going consider elliptic Fourier analysis (EFA).

In this instance the coordinate data (coo) is transformed into outline data through the `Momocs::Out()`. When performed, we can then enter the object (here titled 'shape') and examine its properties.

```
shape <- Out(tpsdata$coo, fac = database)
shape
```

```
## Out (outlines)
##   - 250 outlines, 1543 +/- 1370 coords (in $coo)
##   - 10 classifiers (in $fac):
## # A tibble: 250 x 10
##      ID Site  Context Longitude Latitude Country Archaeological_~ File_Name
##   <dbl> <chr> <chr>       <dbl>    <dbl> <chr>   <fct>            <chr>
## 1     1 Balt~ Baltaš~      24.0     54.0 Lithua~ Baltic Magdalen~ Bal.1
## 2     2 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.1
## 3     3 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.3
## 4     4 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.4
## 5     5 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.5
## 6     6 Baro~ Barouka      30.3     53.5 Belarus Grensk           Bor.6
## # ... with 244 more rows, and 2 more variables: Reference <chr>, Notes <chr>
##   - also: $ldk
```

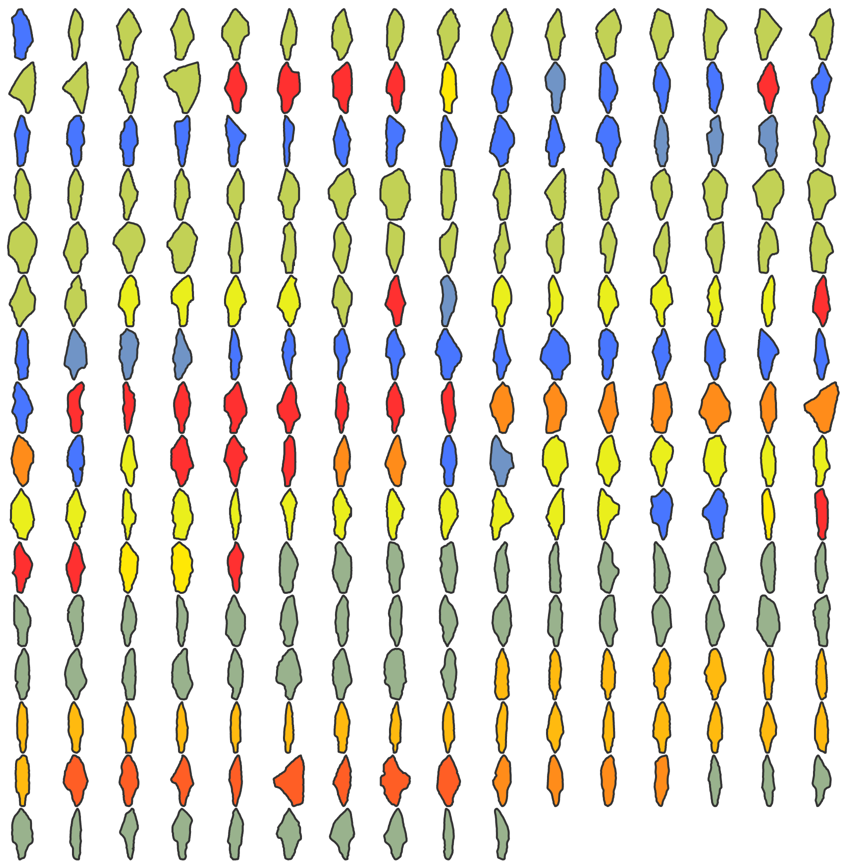
When we call our shape we see we are investigating 250 outlines, with a mean number of 1543 landmarks, and 10 different classifiers (including the factor we want to consider).

Note: It is important to note that the order of the specimens paralled the database, thus rearrangement was unnecessary. In instances where rearrangement is required, the `base::list()`, `base::match()` and `dplyr::arrange()` functions are particularly helpful.

Stage 4: Outline Visualisation

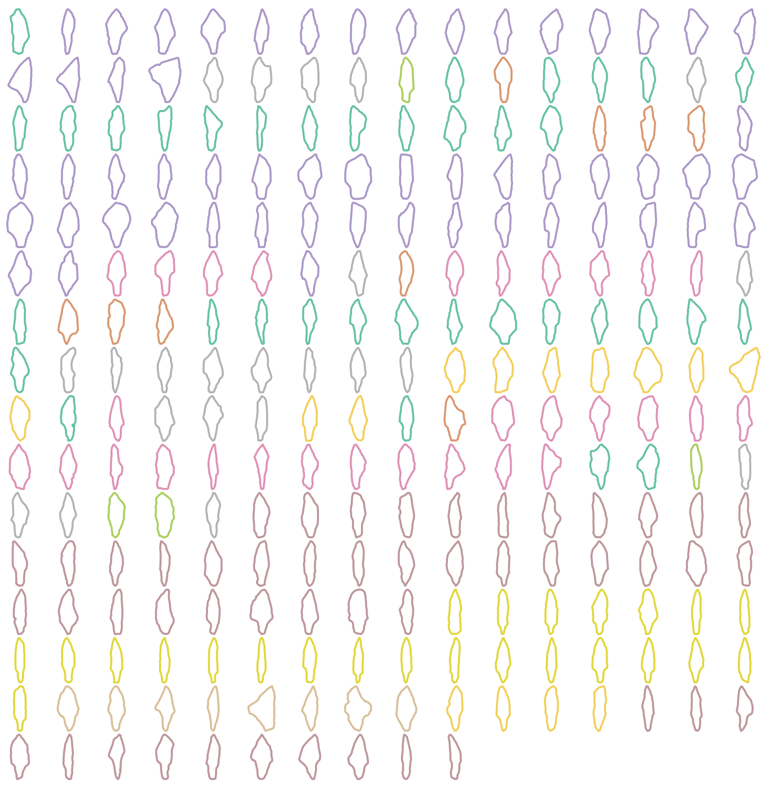
Now our data is in the R Environment and formatted appropriately for the Momocs package we can examine the outline shapes.We can first look at all outlines through the `Momocs::panel()` function. Factors can also be visualised in using the `fac` argument. For example:

```r
panel(shape, main = "", fac = "Archaeological_Unit")
```



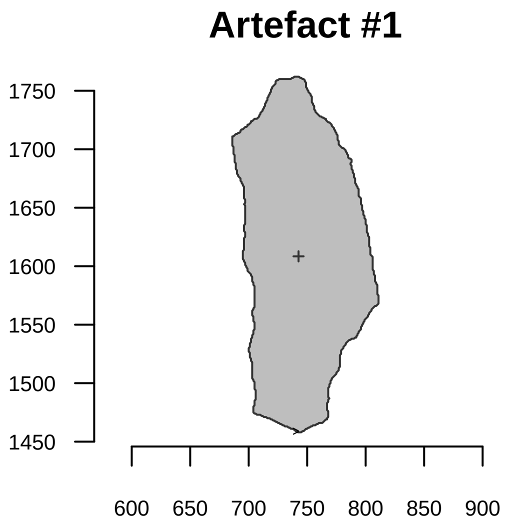
An alternative to the `Momocs::panel()` function is `Momocs::mosaic()`, an updated display function (which will soon replace panel). This does include a legend, unlike the panel function, however as the function is experimental the legend drawing options are limited.

```
mosaic(shape, ~Archaeological_Unit, asp = 1, legend = FALSE)
```



We can draw individual shapes using the `Momocs::coo_plot()` function. A number of aesthetic or stylistic changes (including line colour and fill) are also possible. For example:

```
coo_plot(shape[1], col = "grey", centroid = TRUE, main = "Artefact #1")
```



We can also extract measurements from our outline data at this point using the many `Momocs::coo_()` functions. See towards the end of this document for examples.

Stage 5: Outline Normalisation

For today's practical we will perform Elliptic Fourier Analysis. Elliptic Fourier Analysis (EFA) is one of a number of Fourier based methods of curve composition derived from the first series by Jean Baptiste Joseph Fourier (1768-1830), and developed by Giardina and Kuhl (1977) and Kuhl and Giardina (1982). In practice, a set of four parametric equations (grounded on sine and cosine transformations) are used to transform the x and y Cartesian landmarks into curves (Fourier harmonic amplitudes). The coefficients (termed A,B,C and D), when summed together, represent the approximation of artefact form, and are the framework for further analyses. These are comparable to Procrustes coordinates during the GPA procedure.

Normalisation, as stressed by Claude (2008), has long been an issue in the elliptic Fourier process. Normalisation can be performed through the actual elliptic Foruier transformation (using what is known as the "first ellipse"). As we noted in the first workshop, this process (normalisation and elliptic fitting to coefficients) is equivalent to the Procrustes Superimposition for landmark data.

It is recommended to normalise (standardise) and align your shapes before the `Momocs::efourier()` process. Rotation was considered prior outline digitisation, however rotation could also be explored in Momocs through the `Momocs::coo_aligncalliper()` function prior the `Momocs:efourier()` argument. Here we will perform three transformation processes: 1) `Momocs::coo_center()`, 2) `Momocs::coo_scale()` and 3) `Momocs::coo_close()`. These three functions perform the following actions:

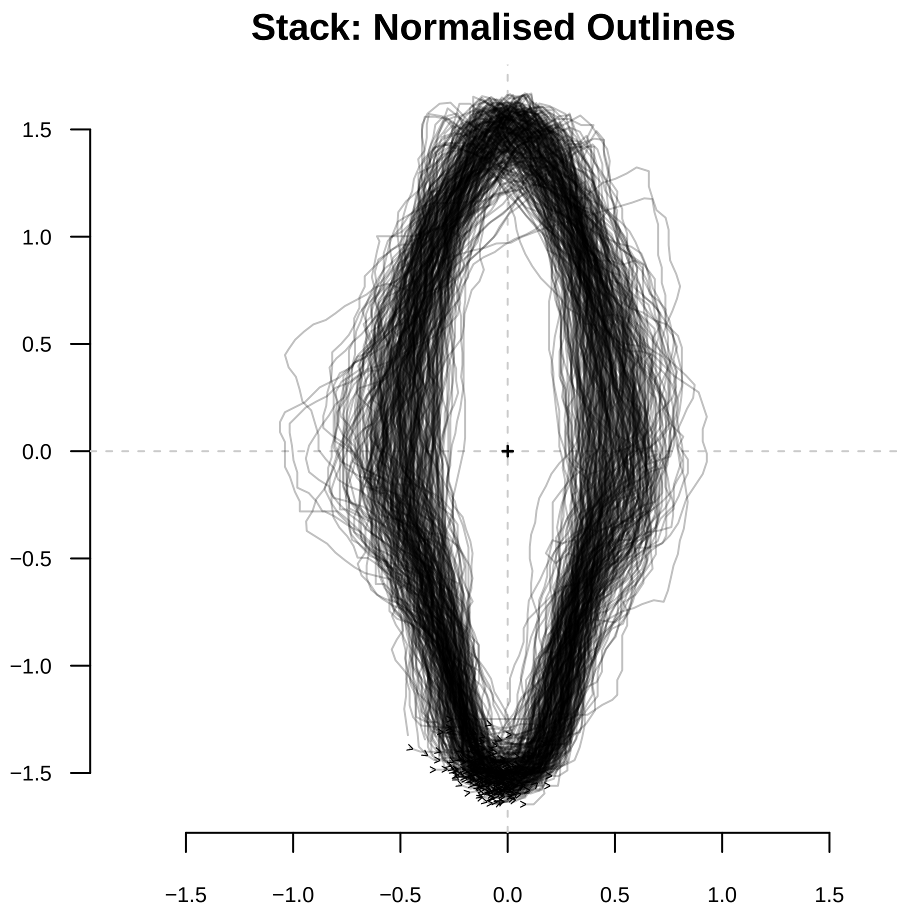
- `Momocs::coo_center()`: This action centres coordinates on a common origin/common centroid).
- `Momocs::coo_scale()`: This action scales the coordinates by their 'scale' if provided, or a given centroid size if 'scale is not provided.

- `Momocs::coo_close()`: Closes unclosed shapes (precautionary).

```r
shapenorm <- coo_center(shape)
shapenorm <- coo_scale(shapenorm)
shapenorm <- coo_close(shapenorm)
```

Following this noralisation procedure, we can then use the `Momocs::stack()` function to inspect all outlines, now according to a common centroid and of a common scale:

```r
stack(shapenorm, title = "Stack: Normalised Outlines")
```

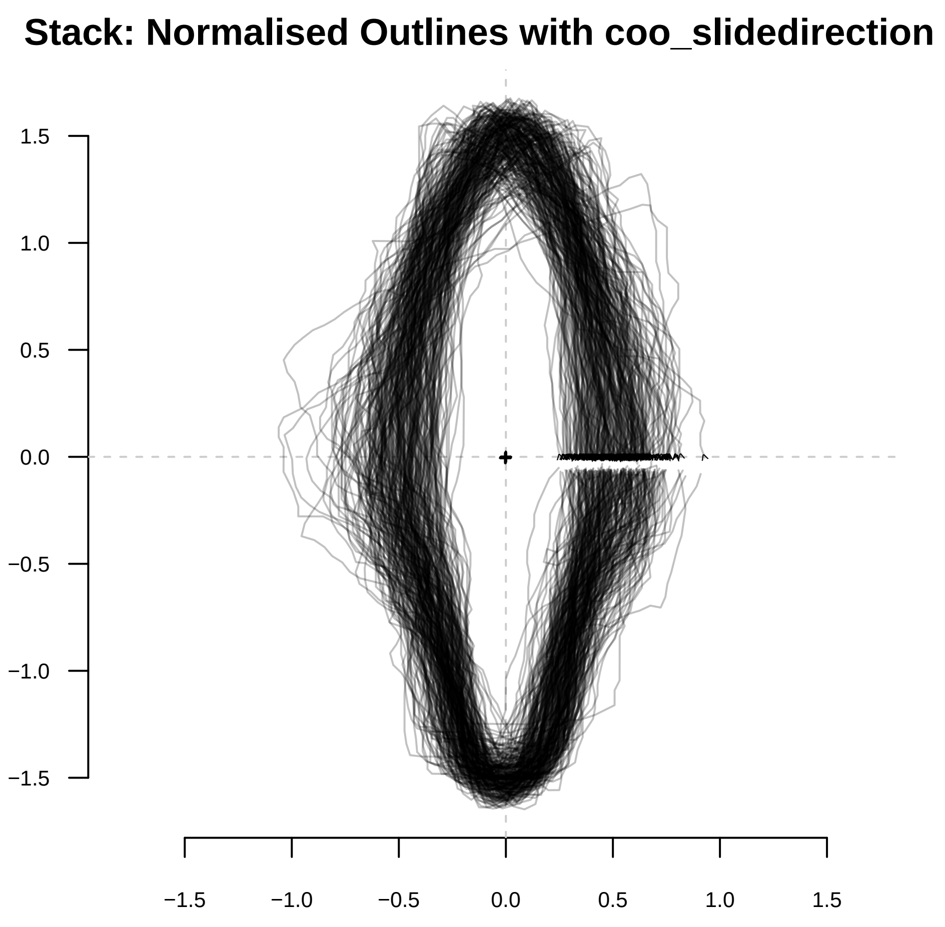


A number of other functions can be performed to normalise these outlines further including `Momocs::coo_slidedirection`, which when shapes are centered configure the first point of the outline to a particular direction. As the Momocs package (?coo_slidedirection) notes: "'right' is possibly the most sensible option (and is by default), since 0 radians points eastwards, relatively to the origin". The `Momocs::coo_untiltx()` function is also advised as to remove rotational biases. The procedure (exemplified using the dplyr piping operators) is as follows:

```
shapenorm2 <- shapenorm %>% coo_slidedirection("right") %>% coo_untiltx()
```

When we inspect this stack, the change is apparent:

```
stack(shapenorm2, title = "Stack: Normalised Outlines with coo_slidedirection()")
```



Stage 6: Elliptic Fourier Analysis

The Giardina and Kuhl parametric equations specify that an outline-specific series of elliptic shapes (of increasing spatial detail) can, when summed together, represent an approximation of the form. This reconstruction of artefact shape can vary depending on the level of accuracy that is required by the researcher; this can be changed depending on the degree of **harmonic power** which is calculated. The first harmonic (first ellipse) is responsible for rotation and defines an ellipse in the plane, with which all other harmonics fit onto. The greater the number of harmonics, the greater the level of detail, and the closer the curves resemble the shape. However, a considerable level of statistical noise is produced if there is too much detail (and thus too many harmonics), and so an appropriate level of harmonics are necessary.

When a level of harmonic power (shape complexity) is determined by the researcher (95%, 99%, 99.9%, 99.99% shape approximation), a series of procedures can be implemented to test how many harmonics are necessary:

- `Momocs::calibrate_harmonicpower_efourier()`: This function estimates the number of harmonics required for the elliptic Fourier process (and all other Fourier processes).

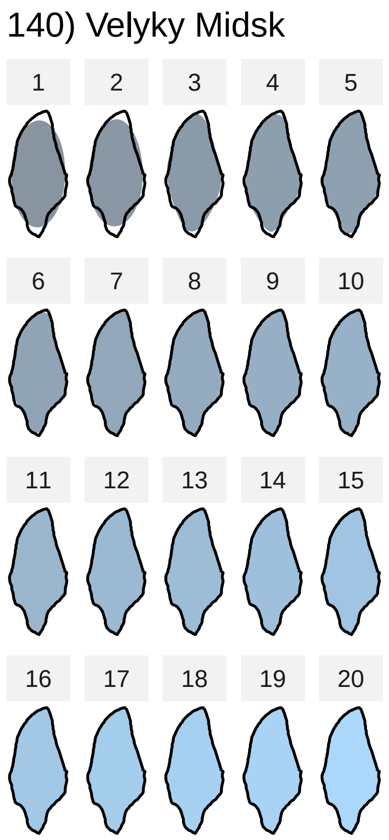
- `Momocs::calibrate_reconstructions_efourier()`: This procedure calculates reconstructed shapes for a series of harmonic numbers. This process best demonstrates the harmonic process.
- `Momocs::calibrate_deviations_efourier`(): This procedure calculates deviations from the original and reconstructed shapes for a series of harmonic numbers.

Let's perform these functions on our normalised dataset:

```
calibrate_harmonicpower_efourier(shapenorm2, nb.h = 20, plot = FALSE)
```

This first function provides us with the necessary number of harmonics required for the varying degree of shape accuracy. It details (visually and in tabular form) that seventeen harmonics are necessary to achieve 99.9% harmonic power, ten for 99%, six for 95% and five for 90% harmonic powers. This function can require some time to perform, participants can trial the function for an individual object through the addition of "id = x", where x is an artefact number.

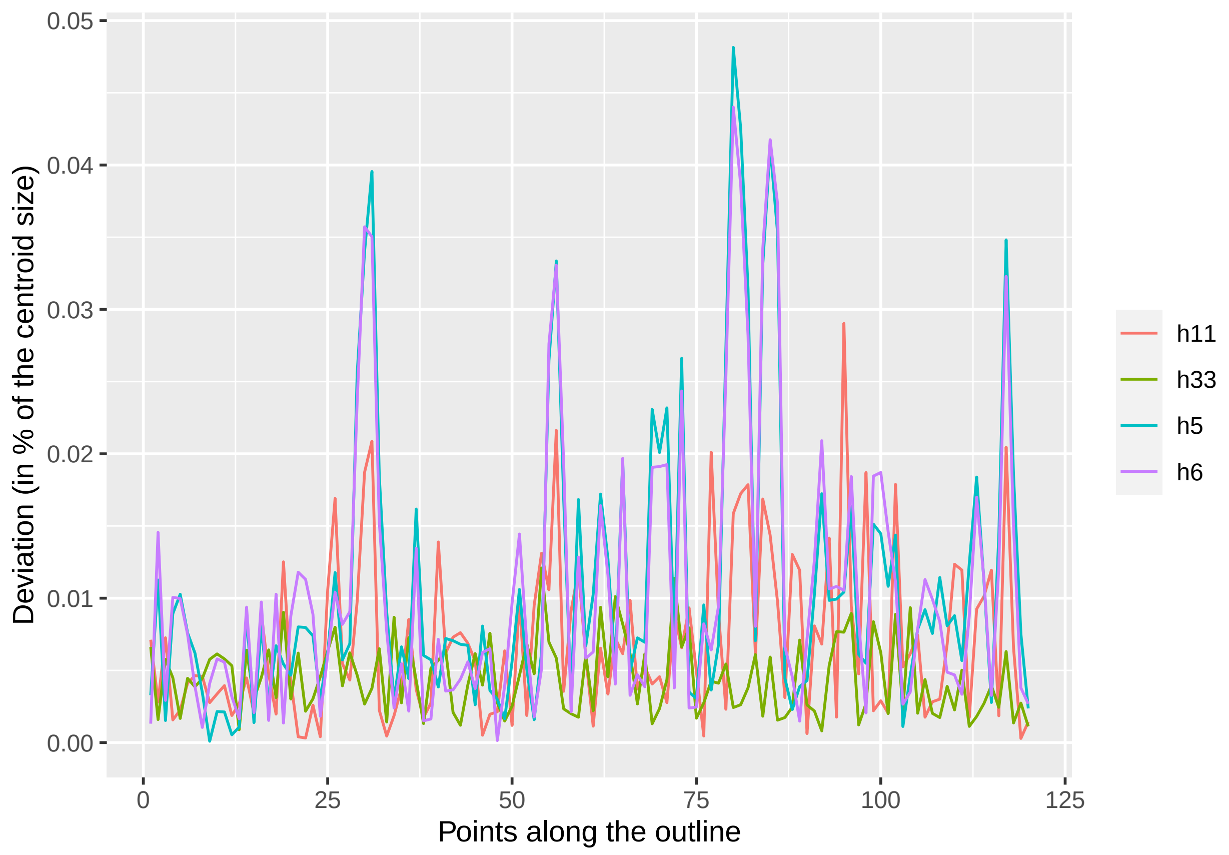
```
calibrate_reconstructions_efourier(shapenorm2, range = 1:20)
```



This second function calculates and displays reconstructed shapes using a range of harmonic numbers (here coded as the first 20). This best illustrates the reconstruction of shape through varying harmonic powers. Again, we see that by the seventeenth harmonic captures (almost) the entire shape. It is important to note that your results may vary as this function considers a random artefact within the Out object.

Note: this may vary from artefact to artefact (given their varying complexity) and so the number of harmonics is a guide for the collection of artefacts.

```
calibrate_deviations_efourier(shapenorm2, id = 4)
```



This last function, for a select shape or the range of shapes, calculates deviations from original and reconstructed shapes, along the shape outline, for a range of harmonic numbers.

Once we are satisfied with how many harmonics will be required, we can input this number into the `Momocs::efourier()` function to generated out OutCoe (the outline coefficients). Here we will use 99.9% harmonic power and thus the first seventeen harmonics.

```
efashape <- efourier(shapenorm2, nb.h = 17, smooth.it = 0, norm = TRUE)
```

Technical note: certain artefact shapes may be prone to bad alignment among the first ellipses and result in not-as-ideal homologous coefficients, and in certain instances upside-down (or 180 degrees rotated) shapes on the morphospace (e.g. PCA plots) may occur. It is considered good practice to normalise outlines (as we have done here) and performing the efourier function with `norm = FALSE`.

Other normalisation procedures not performed here include the addition of landmarks, in order to anchor your artefacts, or through `Momocs::fgProcrustes` through your calliper length.

`norm = TRUE`, and the use of a numerical alignment ("through the first ellipse"), is also a suitable option following prior normalisation. The degree of prior normalisation is dependent on the complexity of artefact shape and is thus the choice of the researcher.

Stage 7: Principal Component Analysis

With our new elliptic fourier coefficients we can begin the exploratory and analytical procedure. We will start by exploring the main theoretical differences in shape through a **Principal Component Analysis (PCA)**. Please refer to the first workshop for a detailed explanation of PCA, and the second workshop for further examples. In the second workshop we needed to turn our LdkCoe() into a PCA class object through the `Momocs::PCA()` function. Here we need to repeat the process (or as demonstrated previously we can utilise the dplyr piping operators).

We can then explore the main sources of shape variation through the `Momocs::scree()`, `Momocs::PCcontrib()`, and `Momocs::plot_PCA()` functions.

```
pcashape <- PCA(efashape)
```

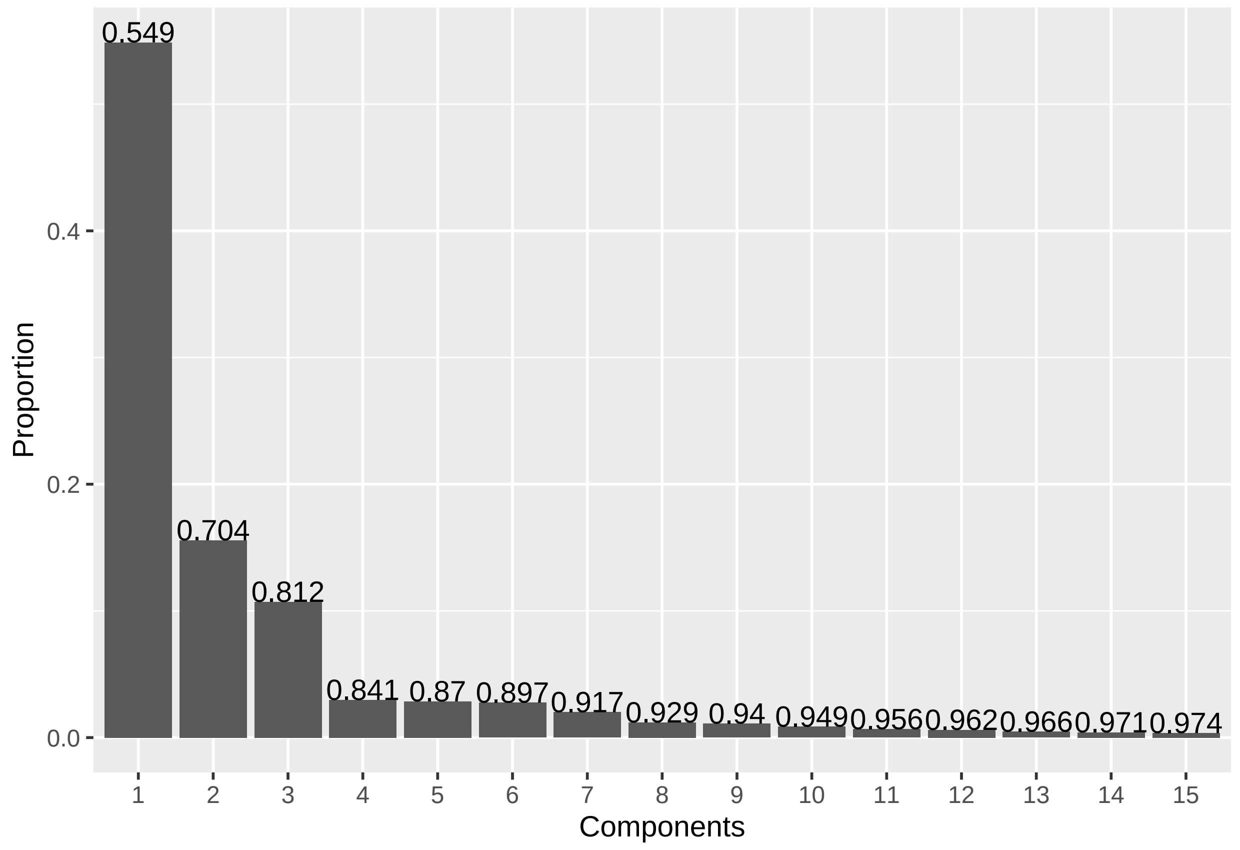
With our new PCA class object we can assess the the proportion and cumulated proportion of the principal components (sources of shape variation) through `Momocs::scree()`:

```
scree(pcashape)
```

```
## # A tibble: 68 x 3
##     axis proportion cumsum
##    <int>      <dbl>  <dbl>
##  1     1    0.549    0.549
##  2     2    0.156    0.704
##  3     3    0.107    0.812
##  4     4    0.0297   0.841
##  5     5    0.0286   0.870
##  6     6    0.0276   0.897
##  7     7    0.0200   0.917
##  8     8    0.0119   0.929
##  9     9    0.0110   0.940
## 10    10    0.00863  0.949
## # ... with 58 more rows
```

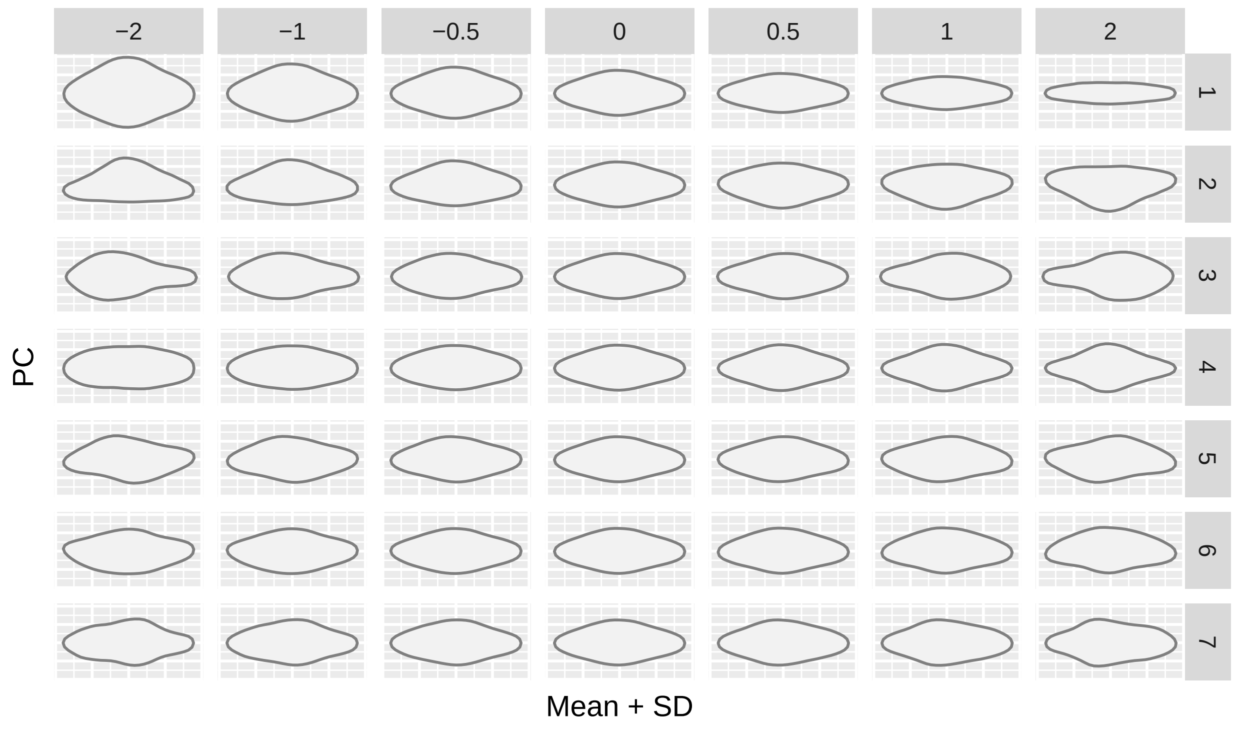
Through the scree function we observe that the first seven axes account for 90% cumulative shape variance, eleven axes account for 95% cumulative shape variance, and twenty-four axes account for 99% cumulative shape variance (this will be important to note for subsequent analyses). This can also be visualised as a scree plot:

```
scree_plot(pcashape, nax = 1:15)
```



Here we have visualised the first fifteen components, however this can be easily changed through amending the nax argument. We can now investigate what the principal components represent through the `Momocs::PCcontrib()` function:

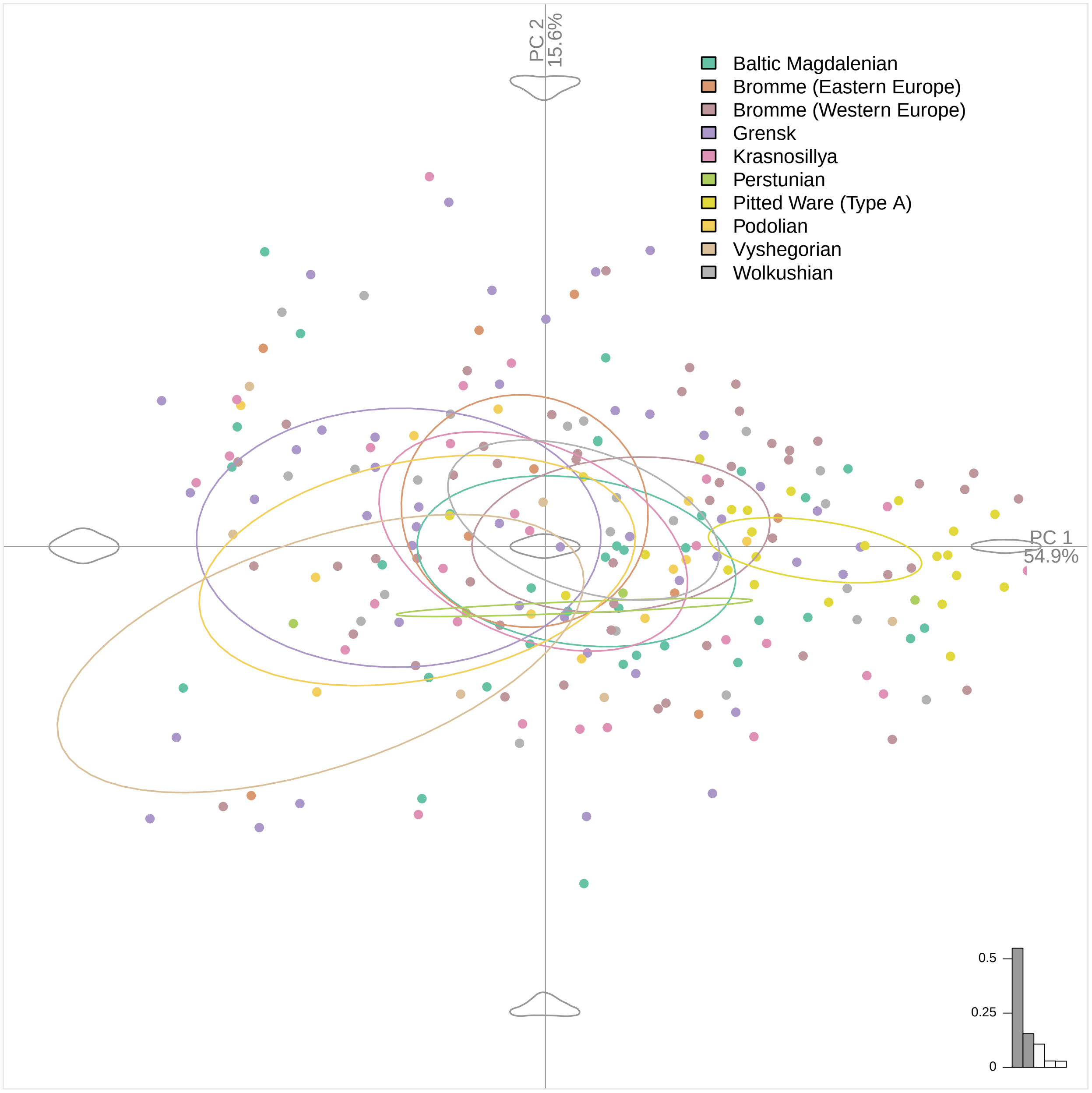
```
PCcontrib(pcashape, nax = 1:7)
```



We can see through this function that Principal Component 1 (PC1), i.e. the main source of shape variation among the tanged points, range from thin tanged points to wider-tanged examples, and that Principal Component 2 (PC2), i.e. the second main source of shape variation, extends from left-exaggerated tangs to right-exaggerated tangs (this shape variance can be removed through flipping all examples on the horizontal axis, however technological profiles were preserved). This function can be set to display as many sources of shape variation as required by the researcher.

Now we know what each principal component represents, and their values to overall shape variation within our analysis, we can plot our artefacts within a morphospace representative of these components. Here we will use the highly-customisable `Momocs::plot_PCA()` function:

```
plot_PCA(pcashape, axes = c(1, 2), ~Archaeological_Unit, morphospace_position = "full_axes",
    zoom = 2, chull = FALSE) %>% layer_points(cex = 1) %>% layer_ellipses()
```

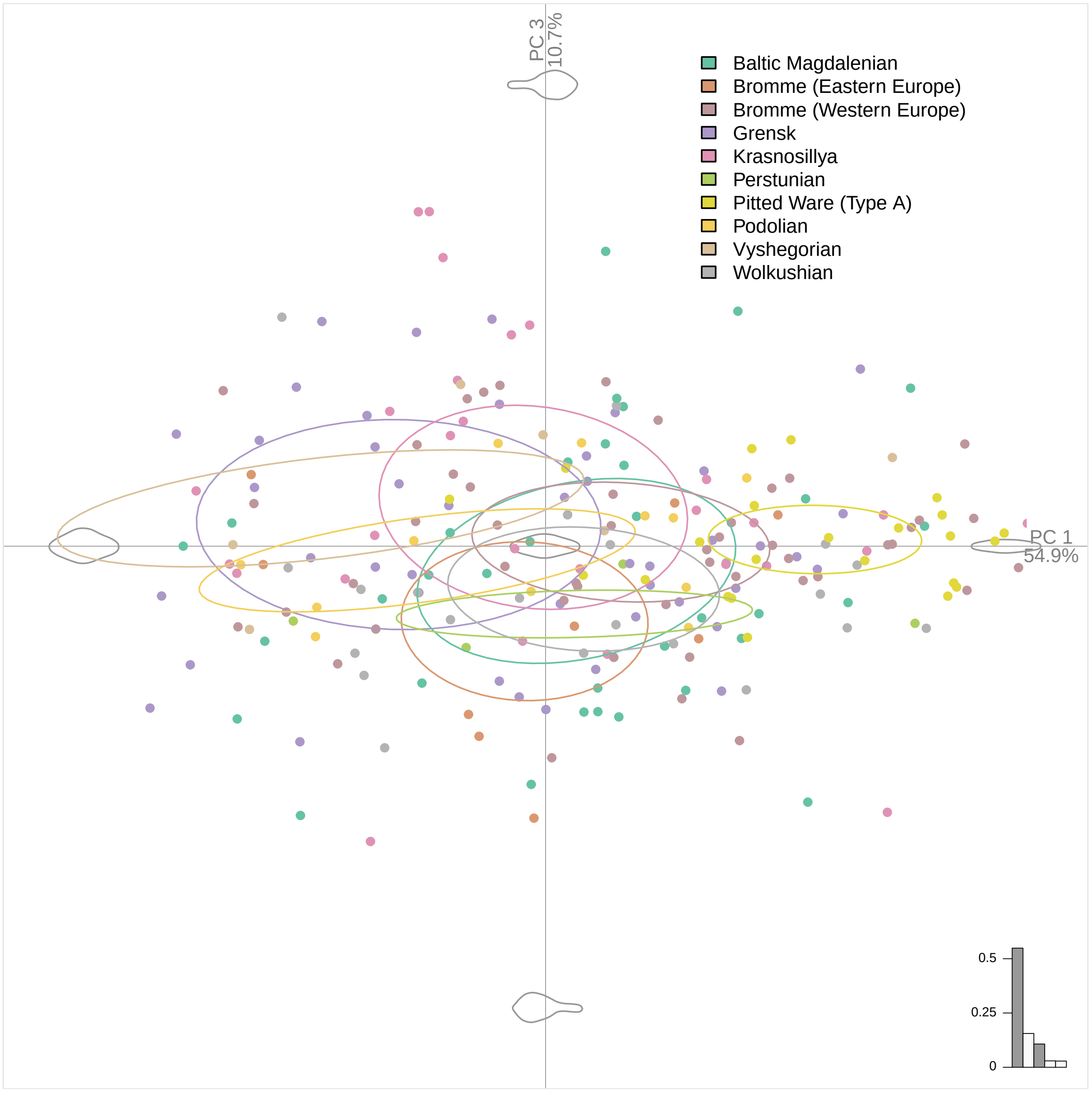


In this diagram we can observe the different distributions of each archaeological unit within the morphospace, and the relative clustering of each unit within this graph. Some clusters appear incredibly tight, while others are broad, meaning that the archaeological unit is represented by varying shapes, some seen in other units. It's important to remember that this graph only represents the first two principal components, and we may wish to examine other sources of shape variation (some which may be of importance to the archaeological relevance and discriminantory powerful of shapes).

Note: pipes (%>%) are used here to processes multiple arguments at the same time. Momocs supports piping with the whole process able to be 'piped'. For teaching purposes we are doing the 'long way' of GMM (as previous).

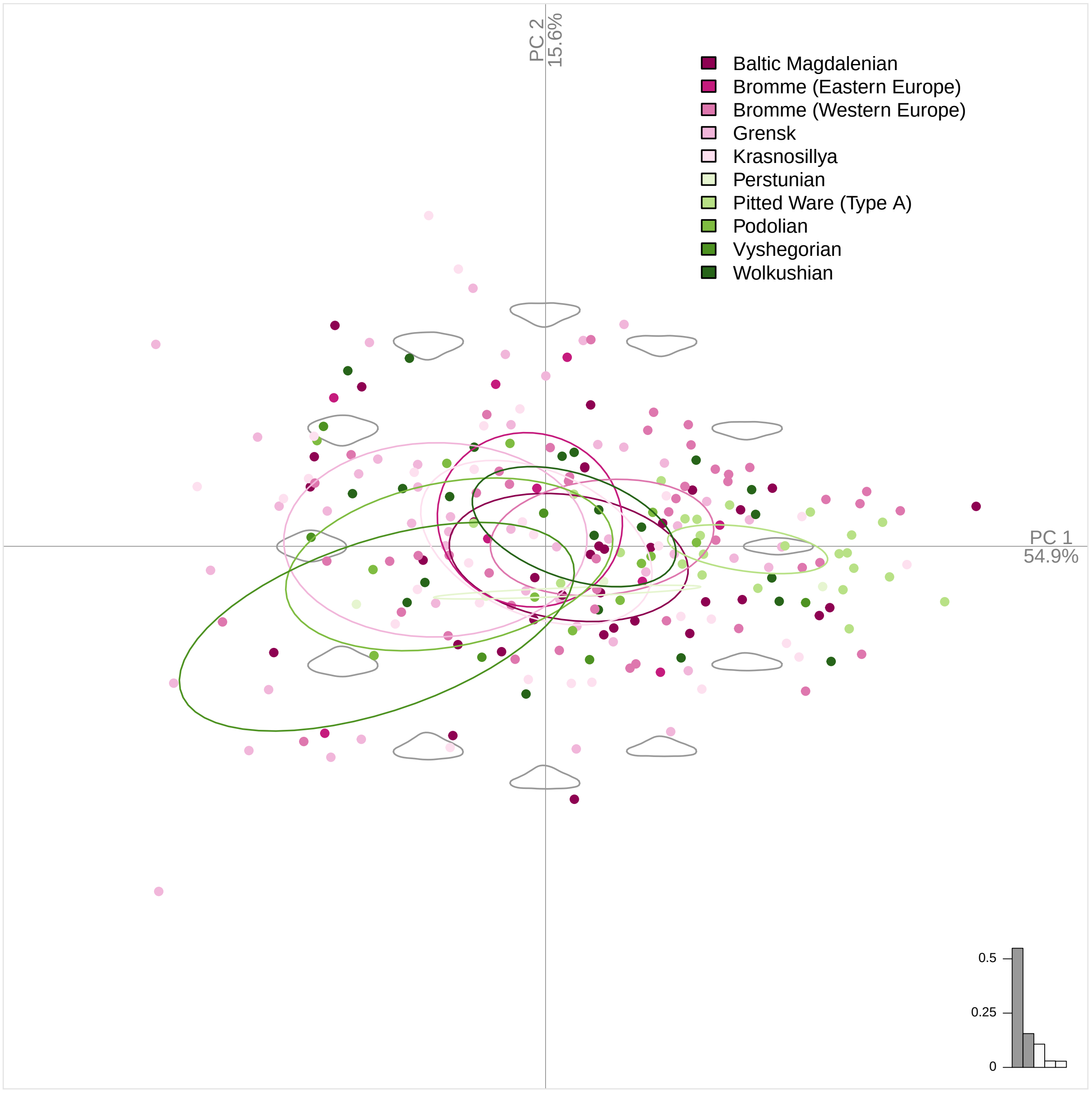
If we wish to examine the relationship between different principal components we can use the `axes` argument to change our graph configuration. For example, if we wish to examine differences in shape between PC1 and PC3 we can specify the `axes` argument in the following way:

```r
plot_PCA(pcashape, axes = c(1, 3), ~Archaeological_Unit, morphospace_position = "full_axes",
    zoom = 2, chull = FALSE) %>% layer_points(cex = 1) %>% layer_ellipses()
```



The 'morphospace_position' and 'palette' arguments are incredible useful wonderful visualisation tool here, for example:

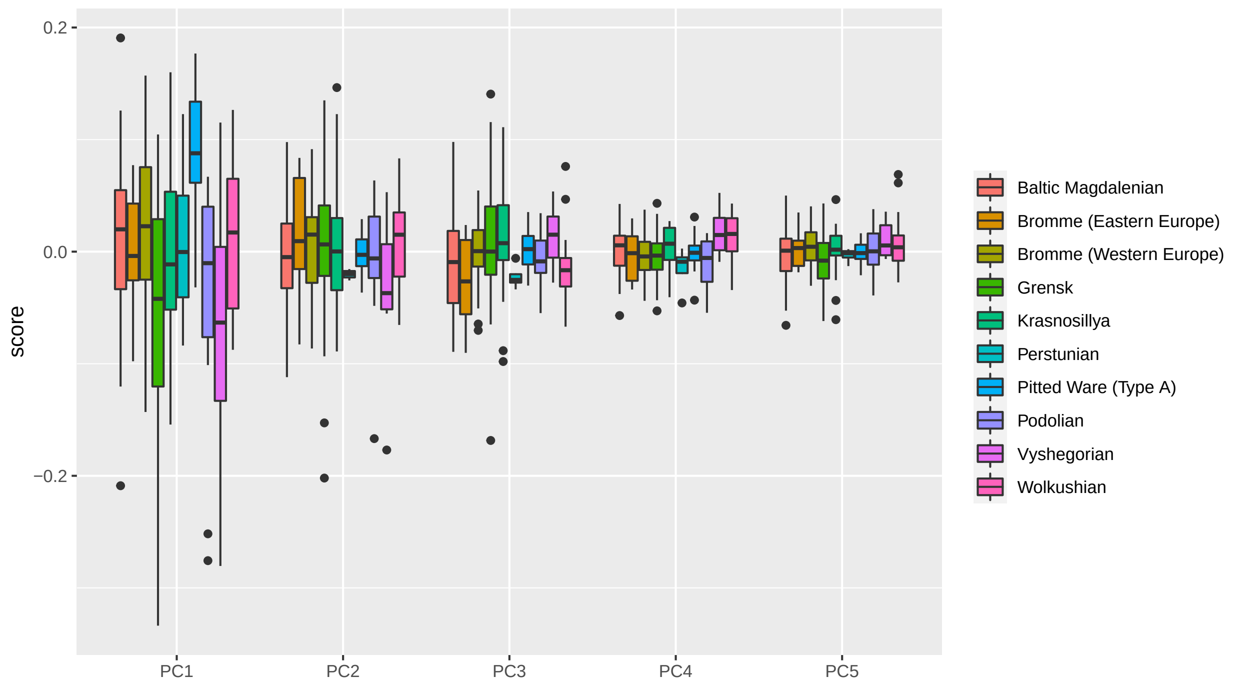
```r
plot_PCA(pcashape, axes = c(1, 2), ~Archaeological_Unit, palette = pal_div_PiYG,
    morphospace_position = "circle", zoom = 1.5, chull = FALSE) %>% layer_points(cex = 1) %>%
    layer_ellipses()
```



Note: the `pal_manual()` argument can be added to allow custom palettes.

Alternatively, we can visualise these principal components, and the variance within different archaeological units through the `Momocs::boxplot()` function. This is particularly useful for capturing variance on many different principal components in one image.

```r
boxplot(pcashape, ~Archaeological_Unit, nax = 1:5)
```



Note: for all of these images, tidy versions can be provided through minor data wrangling (utilising 'pcashape$x' and a factor).

Stage 8: Discriminant Analysis (LDA/DA/CVA)

As we highlighted in the first workshop, PCA explores differences in shape variation irrespective of group composition (*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 elliptic Fourier coefficients 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 e.g. the degree of dimension reductionality. 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 higher degree of unintentional statistical importance, with smaller unimportant variables taking precedence, and so an optimal amount of data is necessary.

For this practical, let's perform three discriminant analyses on different data: 1) the Fourier coefficients, 2) 95% cumulative shape variance as expressed in PC scores, and 3) 99% cumulative shape variance expressed in PC scores.

```
dashapefc <- LDA(efashape, ~Archaeological_Unit)
dashape95 <- LDA(pcashape, ~Archaeological_Unit, retain = 0.95)
dashape99 <- LDA(pcashape, ~Archaeological_Unit, retain = 0.99)

dashapefc$CV.correct
```

```
## [1] 0.24
```

```
dashapefc$CV.ce
```

```
##      Baltic Magdalenian Bromme (Eastern Europe) Bromme (Western Europe)
##               0.1944444               0.0000000               0.2653061
##                  Grensk           Krasnosillya             Perstunian
##               0.3090909              0.2068966              0.2500000
##    Pitted Ware (Type A)               Podolian            Vyshegorian
##               0.3750000              0.0000000              0.2500000
##              Wolkushian
##               0.2272727
```

```
dashape95$CV.correct
```

```
## [1] 0.3
```

```
dashape95$CV.ce
```

```
##      Baltic Magdalenian Bromme (Eastern Europe) Bromme (Western Europe)
##              0.27777778              0.00000000              0.46938776
##                  Grensk            Krasnosillya              Perstunian
##              0.36363636              0.03448276              0.00000000
##    Pitted Ware (Type A)                Podolian             Vyshegorian
##              0.54166667              0.00000000              0.37500000
##              Wolkushian
##              0.22727273
```

```
dashape99$CV.correct
```

```
## [1] 0.308
```

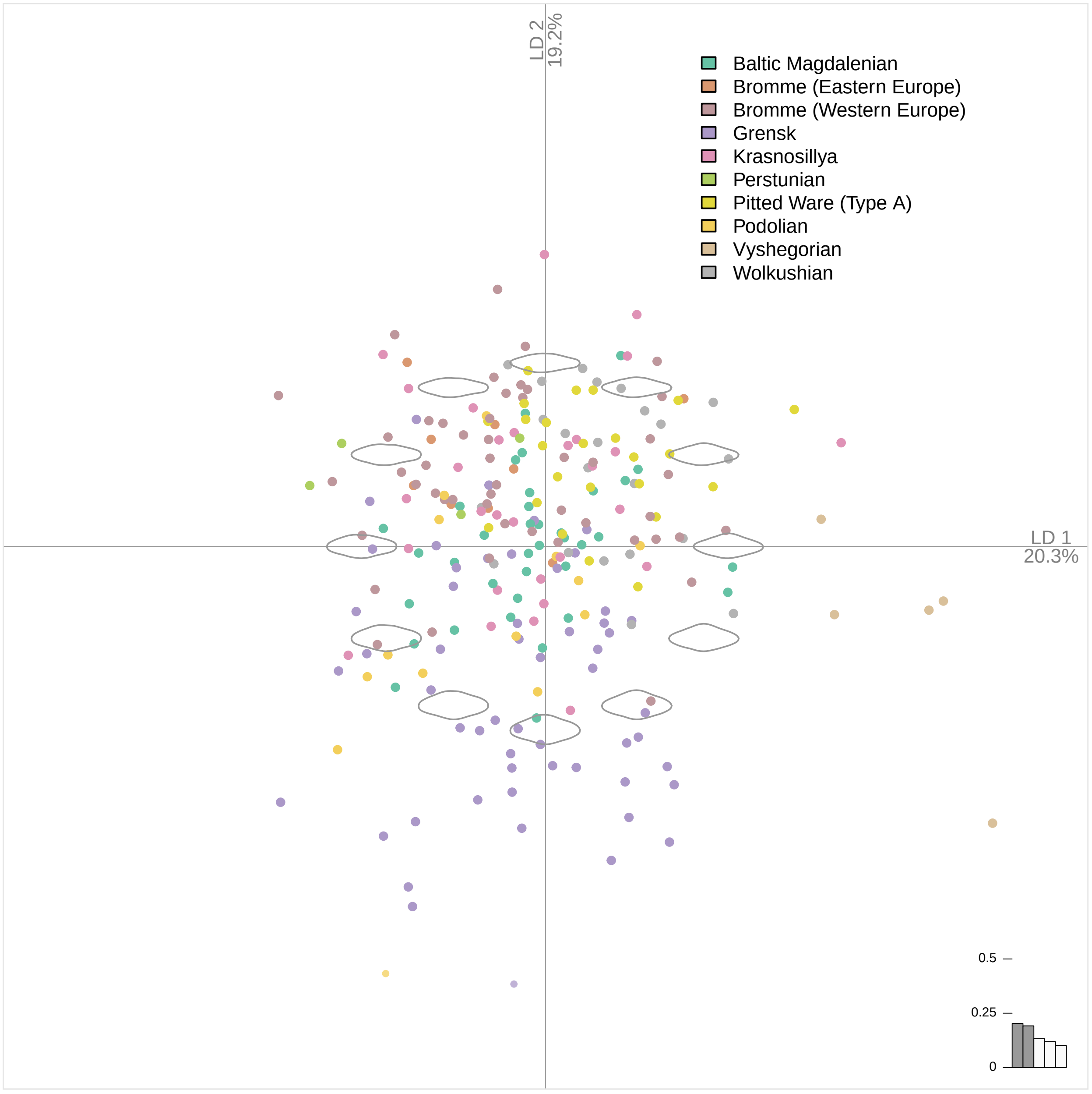
```
dashape99$CV.ce
```

```
##      Baltic Magdalenian Bromme (Eastern Europe) Bromme (Western Europe)
##               0.2777778               0.0000000               0.4081633
##                  Grensk            Krasnosillya              Perstunian
##               0.4909091               0.1034483               0.0000000
##    Pitted Ware (Type A)                Podolian             Vyshegorian
##               0.5000000               0.0000000               0.1250000
##              Wolkushian
##               0.1818182
```

When we examine the Fourier coefficients, a Leave-one-out cross-validation score of 24% (60/250) is obtained, with the Pitted Ware (Type A) and the Grensk forms with the highest class accuracy (37.50% and 30.90% success). With 95% cumulative shape variance a 30% (75/250) classification accuracy is obtained, with higher class accuracies for Pitted Ware and Grensk (54.16% and 36.36%) in addition to considerably higher class accuracy for Bromme (Western Europe) and the Vyshegorian (46.94% and 37.50% respectively), while 99% cumulative shape variance provides a 30.8% (77/250) classification score, with high values for the Pitted Ware, Grensk and Bromme (Western Europe) groups (50.00%, 49.09% and 40.81%). These three discriminant analyses highlight the relative robustness of certain groups, and the weakness of many others, irrespective of how much data is provided. More detailed metrics are included in the `Momocs::classification_metrics()` function (not covered here). Alternatively, these data can be transformed and assessed through further supervised and unsupervised classificatory techniques through tidy machine learning techniques (see the parsnip package for example).

If we wish to visualise our plot, as is common in exploratory procedures we can use the `Momocs::plot_LDA()` function, using similar arguments to `Momocs::plot_PCA()`. For example, to visualise the discriminant analysis for the Fourier coefficients:

```
plot_LDA(dashapefc, axes = c(1, 2), zoom = 1.5, chull = FALSE) %>% layer_points(cex = 1) %>%
    layer_morphospace_LDA(position = "circle")
```



Note: it is now relatively straight forward to impose the shapes onto the graph using the `layer_morphospace_LDA()` argument.

Stage 9: Multivariate Analysis of Variance (MANOVA)

So far we have explored the differences in shape within the whole group of artefacts and explored how well they can be seperated through their units and their group variance. Now we can test, within an Null Hypothesis Significance Testing (NHST) framework, whether there is difference between the different archaeological units. Again, this can be conducted on the Outline data (Fourier Coefficients) or the PCA scores.

Once we have chosen a desired alpha level as of marker of difference (that is to say the boundary with which we are able to reject the null hypothesis of same populations) e.g. 0.05 we can use the `Momocs::MANOVA()` function, noting "Archaeological_Unit" to be our factor which we want to consider. For example, through piping, and for the three different methods above:

```
efashape %>% MANOVA(~Archaeological_Unit)
```

```
##              Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
## fac           9           4.8301   1.4693    576   1577 4.334e-09 ***
## Residuals 240
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pcashape %>% MANOVA(~Archaeological_Unit, retain = 0.95)
```

```
##              Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
## fac           9          0.94475   2.1779     99   2054 5.943e-10 ***
## Residuals 240
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pcashape %>% MANOVA(~Archaeological_Unit, retain = 0.99)
```

```
##              Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
## fac           9           1.6797   1.6737    216   1937 2.357e-08 ***
## Residuals 240
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In each instance, we can reject the null hypothesis of same populations and infer that there is a statistical difference in the shape of different tanged point archaeological units. This, however, doesn't tell us where the differences lie. For the Principal Component scores we can use the `Momocs::MANOVA_PW()` function:

```
pcashape %>% MANOVA_PW(~Archaeological_Unit, retain = 0.95)

pcashape %>% MANOVA_PW(~Archaeological_Unit, retain = 0.99)
```

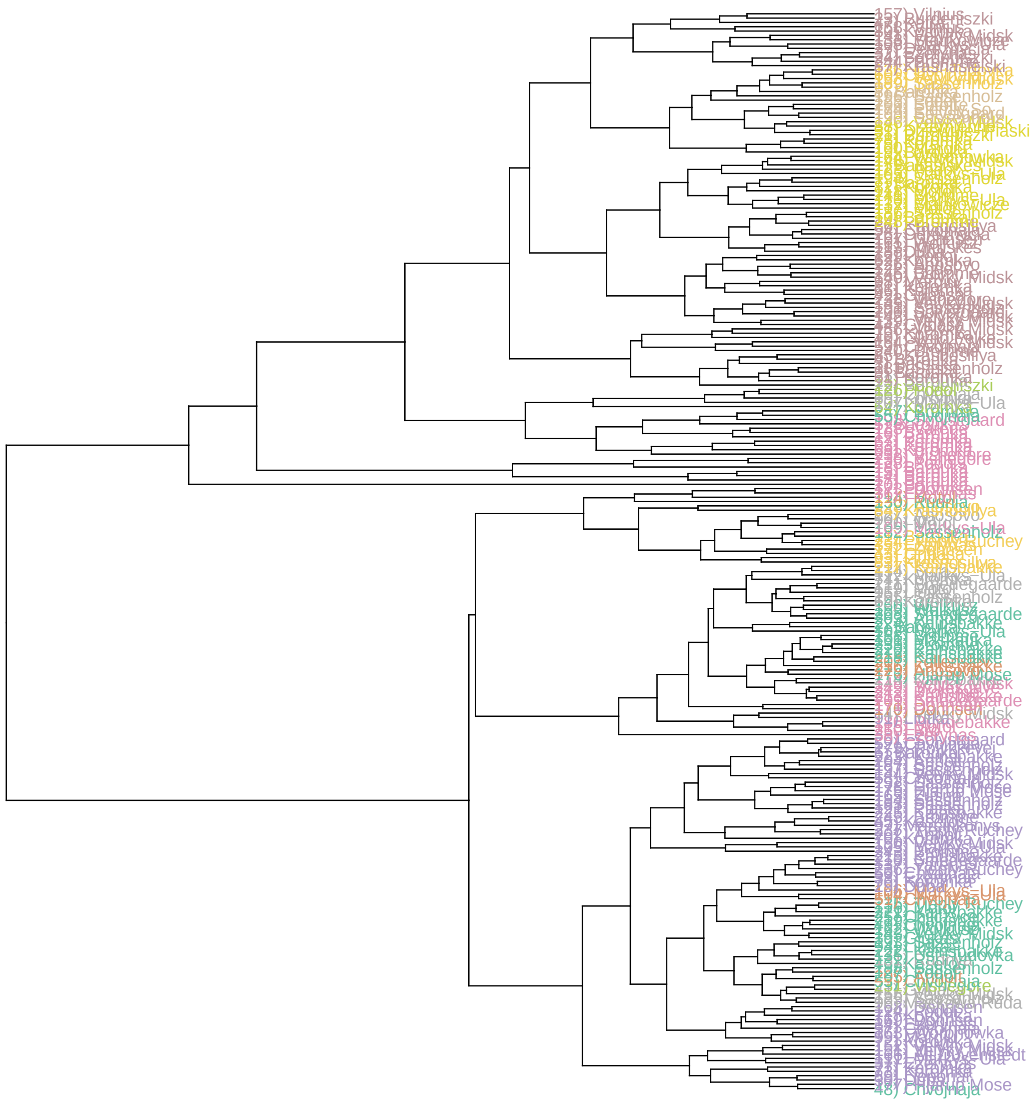
This rather large amount of information provides the p values for each combination of archaeological units and depicts level of significance in star form. In terms of analysis this data highlights, as previously the degree to which specific archaeological units can be distinguished from others in terms of their two-dimensional outline shape.

Stage 10: Hierarchical and K-Means Cluster Analysis

We can now use the elliptic Fourier coefficients and/or the PCA data to examine, irrespective of previous groupings, how similar objects relate to one another within the overall set of examples. The end-point here will be the construction a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar in two-dimensional outline shape. This can be done through two different methods in Momocs: Hierarchical Cluster Analysis, where the structure is provided, or through a K-Means analysis which partitions the shapes into k groups.

To perform a Hierarchical Cluster Analysis we can use the `Momocs::CLUST()` function, a wrapper of `stats::dist()` and `stats::hclust()`. We can specify what type of shape we wish for our tree to be using the `type` argument (horizontal as default), and the specific `hclust` (complete as default) and `dist_method` (euclidean as default). Again, we can use the number of PCA scores as we find applicable or use the elliptic Fourier coefficients.

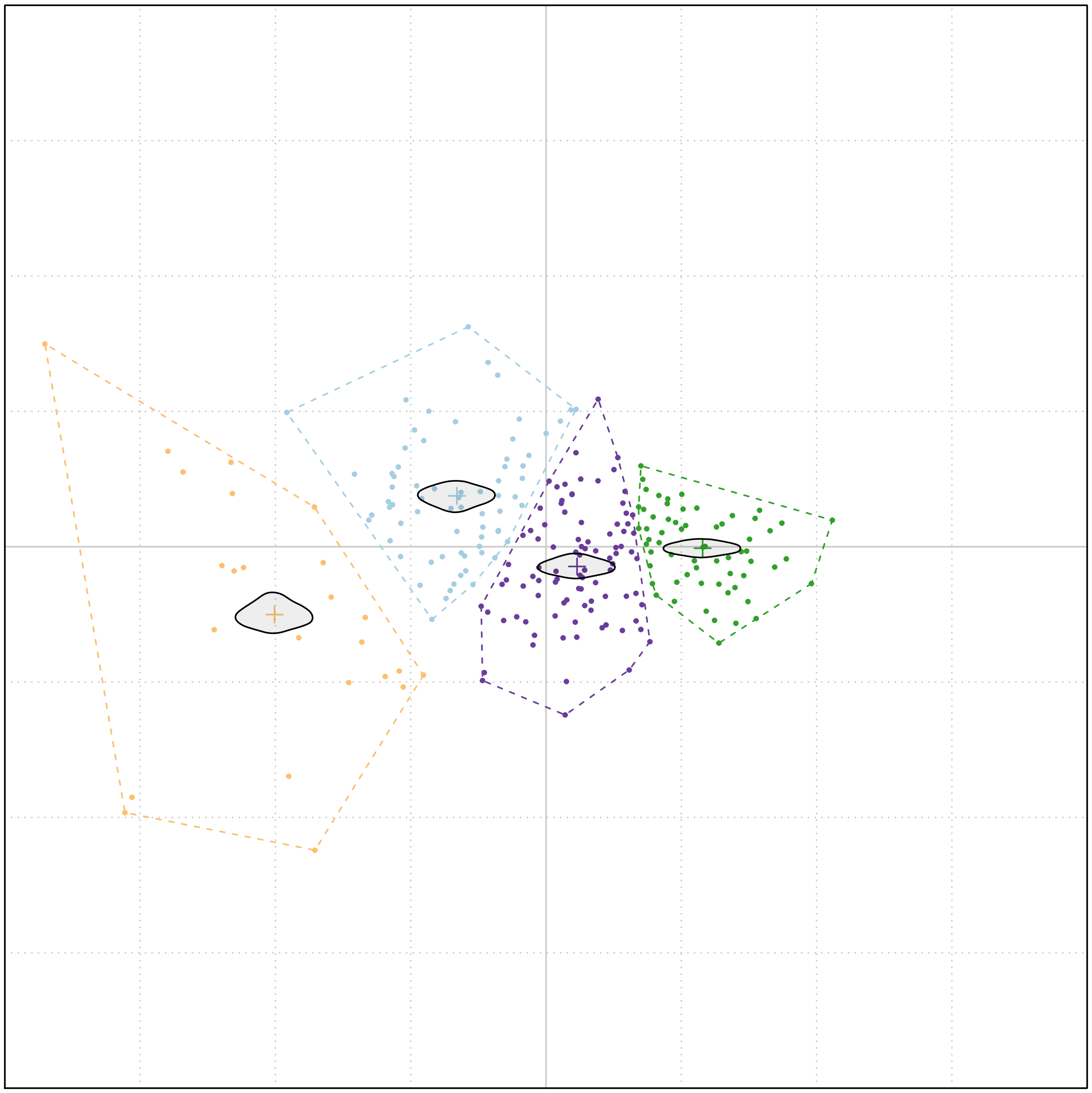
```
CLUST(pcashape, ~Archaeological_Unit, dist_method = "euclidean", hclust_method = "complete",
    palette = pal_qual)
```



This tree can be further examined in the `ape` package and customised further through tree-specific packages e.g. `ggtree`.

Alternatively we can use the `Momocs::KMEANS()` function to derive four x number of groups from the data. For example, if we wish for four groups:

```
KMEANS(pcashape, centers = 4)
```



```
## K-means clustering with 4 clusters of sizes 70, 63, 24, 93
##
## Cluster means:
##           PC1           PC2
## 1 -0.05934544  0.0338508602
## 2  0.10431100 -0.0009912443
## 3 -0.18080392 -0.0451604247
## 4  0.02066539 -0.0131533509
##
## Clustering vector:
##     1) Baltasiskes         2) Barouka         3) Barouka         4) Barouka
##                  4                  2                  1                  1
##         5) Barouka         6) Barouka         7) Barouka         8) Barouka
##                  3                  2                  1                  4
##          9) Barouka         10) Barouka         11) Barouka         12) Barouka
##                   1                   1                   4                   3
##         13) Barouka         14) Barouka         15) Barouka         16) Barouka
##                   1                   3                   3                   3
##         17) Barouka         18) Barouka         19) Barouka         20) Barouka
##                   3                   3                   4                   3
##     21) Burdeniszki     22) Burdeniszki     23) Burdeniszki     24) Burdeniszki
##                   1                   1                   1                   4
##       25) Chilczyce      26) Dereznycia      27) Dereznycia            28) Duba
##                   4                   1                   4                   4
##            29) Duba            30) Duba 31) Dziewule-Piaski        32) Ezerynas
##                   4                   4                   1                   4
##        33) Ezerynas        34) Ezerynas        35) Ezerynas        36) Ezerynas
##                   2                   4                   4                   4
##        37) Ezerynas        38) Ezerynas          39) Glukas     40) Glyno Pelke
##                   4                   2                   4                   4
##         41) Gribasa         42) Gribasa         43) Gribasa         44) Gribasa
##                   4                   1                   4                   1
##         45) Kasetos         46) Kasetos           47) Katra       48) Chvojnaja
##                   2                   4                   1                   4
##       49) Chvojnaja       50) Chvojnaja       51) Chvojnaja       52) Chvojnaja
##                   4                   4                   4                   4
##       53) Chvojnaja       54) Chvojnaja       55) Chvojnaja       56) Chvojnaja
##                   4                   1                   3                   3
##       57) Chvojnaja       58) Chvojnaja       59) Chvojnaja       60) Chvojnaja
##                   4                   4                   1                   1
##         61) Koromka         62) Koromka         63) Koromka         64) Koromka
##                   1                   1                   3                   1
##         65) Koromka         66) Koromka         67) Koromka         68) Koromka
##                   3                   1                   3                   3
##         69) Koromka         70) Koromka         71) Koromka         72) Koromka
##                   2                   2                   4                   4
##         73) Koromka         74) Koromka         75) Koromka         76) Koromka
##                   4                   2                   1                   4
##         77) Koromka         78) Koromka         79) Koromka         80) Koromka
##                   4                   1                   1                   1
##         81) Koromka         82) Koromka    83) Krasnosillya    84) Krasnosillya
##                   1                   1                   4                   4
##    85) Krasnosillya    86) Krasnosillya   87) Krasnasielski      88) Krzemienne
##                   1                   1                   4                   4
##       89) Lieporiai            90) Lipa          91) Liutka          92) Liutka
##                   4                   4                   4                   1
##          93) Liutka          94) Liutka          95) Liutka    96) Mackowa Ruda
##                   1                   2                   2                   4
##     97) Marcinkonys          98) Margiu          99) Margiu         100) Margiu
##                   2                   1                   1                   1
##       101) Maskauka     102) Markys-Ula     103) Markys-Ula     104) Markys-Ula
##                   2                   2                   2                   4
##     105) Markys-Ula     106) Markys-Ula     107) Markys-Ula     108) Markys-Ula
##                   3                   2                   3                   4
##     109) Markys-Ula     110) Markys-Ula     111) Markys-Ula     112) Markys-Ula
##                   4                   1                   4                   2
##      113) Mitriskes          114) Motol          115) Motol          116) Motol
##                   4                   4                   2                   4
##          117) Motol          118) Motol          119) Motol          120) Motol
##                   1                   1                   2                   4
##         121) Plaska          122) Podol          123) Podol          124) Podol
##                   2                   1                   1                   4
##          125) Podol          126) Podol          127) Podol          128) Podol
##                   1                   3                   4                   3
##          129) Podol         130) Rudnia         131) Rudnya    132) Stankowicze
##                   1                   4                   4                   1
##    133) Stankowicze          134) Suraz    135) Ust-Tudovka    136) Ust-Tudovka
##                   1                   2                   4                   1
##         137) Varena         138) Varene   139) Velyky Midsk   140) Velyky Midsk
##                   4                   3                   3                   1
##   141) Velyky Midsk   142) Velyky Midsk   143) Velyky Midsk   144) Velyky Midsk
##                   1                   1                   4                   4
##   145) Velyky Midsk   146) Velyky Midsk   147) Velyky Midsk   148) Velyky Midsk
##                   1                   4                   2                   1
##   149) Velyky Midsk   150) Velyky Midsk   151) Velyky Midsk   152) Velyky Midsk
##                   2                   2                   4                   4
##   153) Velyky Midsk   154) Velyky Midsk   155) Velyky Midsk   156) Velyky Midsk
##                   1                   1                   4                   1
##        157) Vilnius        158) Vilnius        159) Wolkusz        160) Wolkusz
##                   1                   1                   2                   2
##        161) Wolkusz        162) Wolkusz      163) Woronowka      164) Woronowka
##                   4                   4                   4                   1
##          165) Zusno 166) Alt Duvenstedt        167) Dohnsen        168) Dohnsen
##                   2                   4                   4                   4
##        169) Dohnsen        170) Dohnsen        171) Dohnsen        172) Dohnsen
##                   4                   2                   2                   4
##        173) Dohnsen      174) Elemly So    175) Hjarup Mose    176) Hjarup Mose
##                   4                   1                   2                   2
##    177) Hjarup Mose    178) Hjarup Mose     179) Rolykkevej     180) Rundebakke
##                   4                   4                   2                   2
##     181) Sassenholz     182) Sassenholz     183) Sassenholz     184) Sassenholz
##                   1                   4                   2                   2
##     185) Sassenholz     186) Sassenholz     187) Sassenholz     188) Sassenholz
##                   4                   4                   4                   4
##     189) Sassenholz     190) Sassenholz     191) Sassenholz     192) Sassenholz
##                   1                   4                   1                   4
##     193) Sassenholz     194) Sassenholz     195) Sassenholz     196) Sassenholz
##                   4                   1                   2                   1
##     197) Sassenholz    198) Solystgaard    199) Solystgaard    200) Solystgaard
##                   4                   3                   1                   1
##    201) Solystgaard         202) Anholt         203) Anholt         204) Anholt
##                   2                   2                   2                   2
##         205) Anholt        206) Sotofte     207) Follenslev    208) Smedegaarde
##                   4                   1                   2                   2
##    209) Smedegaarde    210) Smedegaarde    211) Smedegaarde     212) Kainsbakke
##                   2                   4                   2                   2
##     213) Kainsbakke     214) Kainsbakke     215) Kainsbakke     216) Kainsbakke
##                   2                   2                   2                   2
##     217) Kainsbakke     218) Kainsbakke     219) Kainsbakke     220) Kainsbakke
##                   2                   2                   4                   2
##     221) Kainsbakke     222) Kainsbakke     223) Kainsbakke     224) Kainsbakke
##                   4                   2                   4                   2
##     225) Kainsbakke        226) Anosovo        227) Anosovo        228) Anosovo
##                   2                   1                   4                   4
##        229) Anosovo      230) Vishegore      231) Vishegore      232) Vishegore
##                   2                   3                   4                   3
##      233) Vishegore  234) Tieply Ruchey  235) Tieply Ruchey  236) Tieply Ruchey
##                   1                   4                   4                   4
##  237) Tieply Ruchey         238) Bromme         239) Bromme         240) Bromme
##                   2                   2                   2                   1
##         241) Bromme         242) Bromme         243) Bromme         244) Bromme
##                   1                   2                   2                   4
##         245) Bromme         246) Bromme         247) Bromme         248) Bromme
##                   2                   1                   3                   1
##    249) Trollesgave            250) Bro
##                   2                   2
##
## Within cluster sum of squares by cluster:
## [1] 0.2052452 0.1017226 0.2574383 0.2054117
##  (between_SS / total_SS =  71.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

More computationally-intensive tree-building exercises can be explored through **maximum likelihood**, using the the `RPhylip` package (this requires the Phylip software to be installed on a computer already).

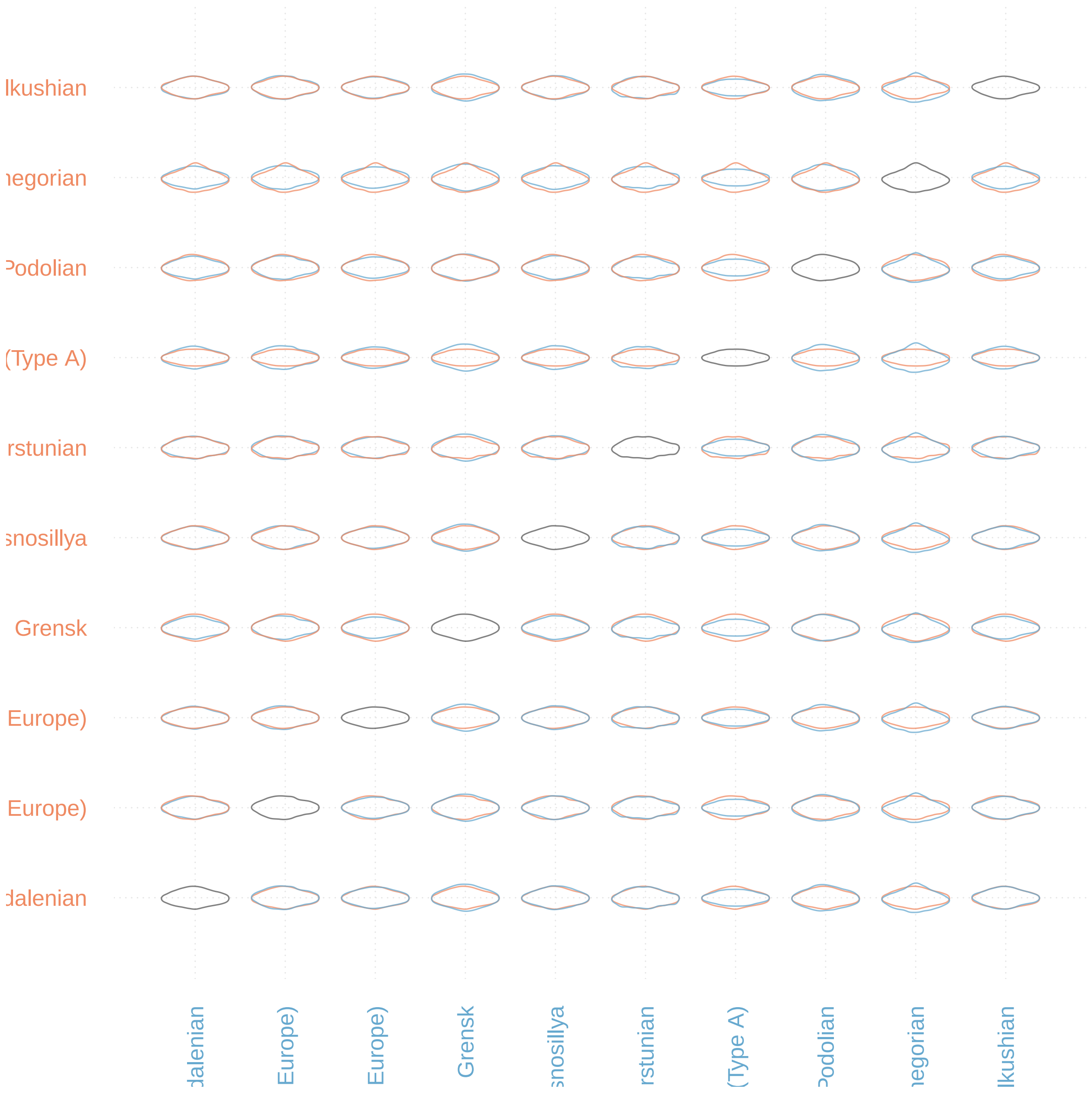
Stage 11: Constructing Mean Shapes

If we wish, we can retrieve mean shapes for a provided factor (e.g. "Archaeological_Unit"), using the elliptic Fourier coefficients or PCA scores. This is done through the `Momocs::MSHAPES()` function with the object first being made.

```
meanshape <- MSHAPES(efashape, ~Archaeological_Unit)
```

The `Momocs::plot_MSHAPES()` function is particularly useful for displaying the mean shapes for all the archaeological units and the visualisation of different configurations of mean shapes.

```
plot_MSHAPES(meanshape, size = 0.75)
```



Stage 12: Further Exploratory Work

Throughout this exercise we have only examined shape, however we still have size data (as all images have a scale). From this we can extract various different measures including length, width, circularity, rectangularity, elongation and symmetry through the `Momocs::coo_()` functions (noting that the converted value for measurements is in pixels and trasnformation is necessary!). We could take any of these measures and examine this new factor against our data.

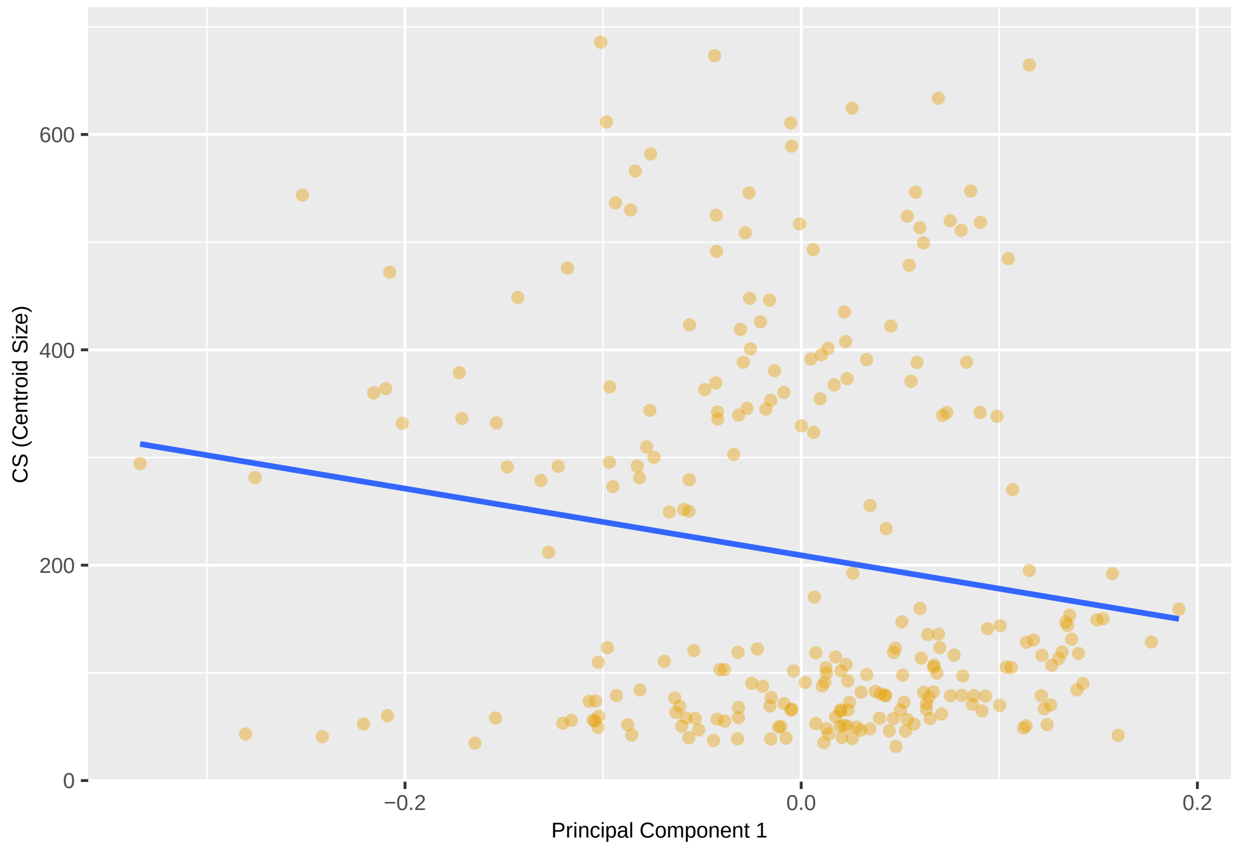
Centroid size is perhaps the best measure of size, incorporating the distance from all points of interest in relation to the shape. This can be extracted from the original shape data, using the `Momocs::coo_centsize()` function. We can then take this data and the principal component scores, and merge them into one database. There are a variety of ways this can be done, this is just one example.

```
centroidsize <- as_tibble(coo_centsize(shape))
centroidsize <- rename(centroidsize, CS = "value")
```

```
pcascores <- as_tibble(pcashape$x)
databasedata <- cbind(database, centroidsize, pcascores)
```

We can now explore these through regression and correlation-based analyses. For example, using the ggplot functions we can create a scatter plot and add a regression line. As a demonstration, let's compare the first source of variation (PC1) with centroid size:

```
ggplot(databasedata, aes(PC1, CS)) + geom_point(size = 2, pch = 16, alpha = 0.4,
    colour = "#E69F00", fill = "#ffd475") + geom_smooth(method = lm, se = FALSE) +
    theme(text = element_text(size = 8), axis.text = element_text(size = 8)) + xlab("Principal Componen
    ylab("CS (Centroid Size)")
```



We can then perform a correlation (and test) using the `cor` and `cor.test` functions:

```
cor(databasedata$PC1, databasedata$CS)
```

```
## [1] -0.1612085
```

```
cor.test(databasedata$PC1, databasedata$CS)
```

```
##
##  Pearson's product-moment correlation
##
## data:  databasedata$PC1 and databasedata$CS
## t = -2.5724, df = 248, p-value = 0.01068
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.27968167 -0.03789949
## sample estimates:
##        cor
## -0.1612085
```

Concluding Remarks

This workshop was designed to highlight how geometric morphometrics (outline analysis) can be examined for archaeological material in the R Environment, from data importing to visualisation and analysis. It is worth stressing that Momocs is only one of a number of packages in the R Environment, and the methods showcased here are only one way (and one style) of conducting GMM. Only through exploring the packages and their functions will you be able understand what workflow works best for your research question and process.

If there are any questions please feel free to contact me: C.S.Hoggard@soton.ac.uk