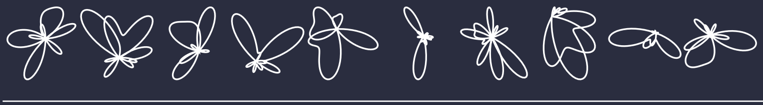


Momocs: Morphometrics using R

09 September, 2014

Vincent Bonhomme

School of Mathematics and Statistics

Sheffield, UK

Centre de Bio-Archéologie et d'Écologie

Montpellier, France

Contents

Chapter 1

Introduction to Momocs

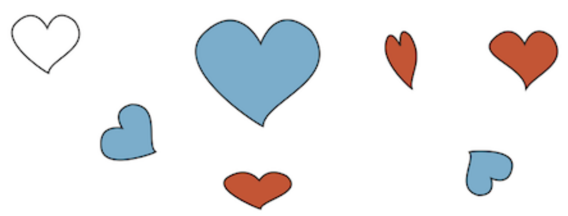
1.1 About morphometrics

The link, if there were one, between the form and the function of objects, living or inert, has been one of the most enduring questions in the realm of science. In many situations, analysing the shape variation among objects can bring deep insights into their functioning and to the underlying mechanism leading to their variation in shape.

Morphometrics aims at analyzing the variation and covariation of the *size* and *shape* of objects, defining altogether their form. Shape and form might be confusing words, used as synonyms in many languages. Hereafter, we will use the definition of shape proposed by Kendall (1989) and Small (1996) that it is:

"the total of all information invariant under translation, rotation, and isotropic rescaling".

In other words, What remains if we draw an heart, and then rotate the page, or change its size uniformly, or move the page about on the desk, is the shape of the heart (see Figure below). Shape by essence is better described in three dimensions, but here, only two dimensional shape will be considered, *e.g.*, three-dimensional objects will be viewed from one side and considered as represented by their projections on a plane.



What is the shape? Blue hearts have the same *shape* as the white one: we can obtain them by translating and/or rotating and/or rescaling the white object. Red hearts all have different shapes.

1.2 About Momocs

Momocs, which names stands for **mo**dern **mo**rphometri**cs** aims at provinding a comprehensive environment for shape analysis in R.

Momocs started with the seminal book by Julien Claude, *Morphometrics with R* that provided, for the first time, an exhaustive list of function along with a very didactical approach. Then Sandrine Picq, as part of her PhD at the UMR CBAE in Montpellier, France, wrapped some of these functions to analyse the outlines of her grapevine seeds. After my PhD, I finally had the opportunity to start morphometrics while I was doing a post-doc at the French Institute of Pondicherry in India, with Cédric Gaucherel. At that time, Momocs was only dedicated to outline analyses and has been released for the first time in May 2012. The associated paper has been published in Journal of Statistical Software but in the meantime Momocs grown, month after month, and mainly because of the great welcome it had among morphometricians *union* R-users.

More recently I started a new post-doc with Eleanor Stillman and Glynis Jones, at the University of Sheffield in March 2014, and decided to rewrite it completely to include open outlines, configuration of landmarks, global shape descriptors, etc. Primarily because we needed it to study plant remains on which we had the great chance combine all these approaches.

Our ultimate aim is to provide a very complete environment and toolkit for all morphometrics approaches, to bring them all in an open-source environment, easy to extend, improve and fine-tune. In brief, to focus more on science than on softwares.

1.3 About these vignettes

Most of the useRs will probably not need to read all of these vignettes to use Momocs but they will grab the overall logic behind Momocs and be quickly free as birds. Altogether, they are intended for form a nice and complete tutorial.

For pure R beginners, the way will be a bit longer but I hope it's feasible though. And if you come to R, thanks to morphometrics, I consider that a great news!

This tutorial assumes that you have a basic knowledge of R. If that's not the case, tons of freely available sites, books, etc. on R can be easily found and you can start there. Do not hesitate to use Graphics User Interface for R, such as the great RStudio, particularly if you are a beginner and are not yet an R masochist working on console mode or whatever. RStudio is not perfect yet, but tends to.

The aim of this manual is to provide an overview of the package, mostly graphically, neither to bring an exhaustive description of the package nor of morphometrics. For a detailed literature of shape analysis and Momocs features, have a look to:

- *Morphometrics with R* a must have book by Julien Claude, one of the Momocs' author, published by Springer, in the great UseR! collection. Julien also included an introduction to R and a nice bibliography.
- the Momocs companion paper published in JSS. So far Momocs dealt only with outline analysis.

- the R help (type `?Momocs` in R). If you need help on a specific topic, le'ts say `efourier`, you can try: `?efourier` and `example(efourier)`.

Finally, this manual itself is written using a combination of R and RMarkdown thanks to RStudio. And you can access its code source there.

1.4 How to contribute

Momocs is open-source and does not belong to someone in particular; it relies on you to signal bugs, share ideas, methods, tutorials, talks, datasets, etc.

Momocs has a GitHub repository that welcomes contributions: otherwise you can send me an email at the adress below. And naturally, any contribution you will be properly credited.

1.5 Install and load Momocs

Enough talking, let's do some morphometrics with Momocs! All the grey boxes below contain R code, most of the time with the results it produces, text and/or graphs. If you copy/paste it in your own R, it should work the same way as here.

1.5.0.1 From CRAN

Momocs is available from the CRAN and you can install it as any other packages typing :

```
install.packages("Momocs")
```

Note that you have to install it once, but to load it (via `library`) once per new R session, unless you save the environment (maybe comfortable but does not force you to have reproducible scripts):

```
library(Momocs)
```

1.5.0.2 From GitHub

But, I highly recommend to install the very last version (updated ± daily), in particular because I'm completely rewriting Momocs these days, and the code presented here is based on this last version, not on the CRAN version. My code repository is on GitHub, and we can install it directly in the console, using the package `devtools`. Just type:

```
library(devtools)
install_github("vbonhomme/Momocs")
library(Momocs)
```

1.6 A quick example

Let's start with a quick analysis on one of the datasets bundled with Momocs. The `bottles` dataset which includes 20 (randomly chosen) outlines of beer bottles, and 20 of whisky bottles. We are interested in: i) exploring the morphological diversity of these 40 bottles, and ii) in testing whether whisky and beer bottles differ in shape.

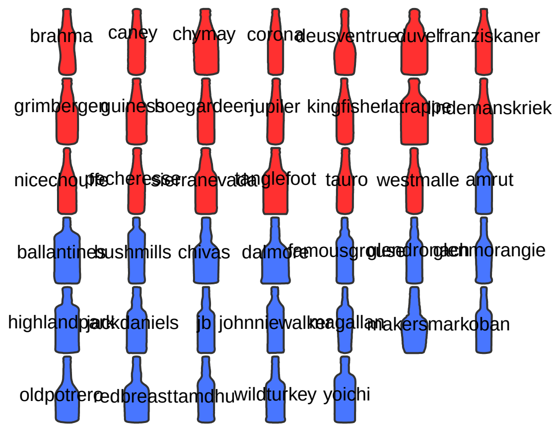
We will first load it, print it, then calculate some elliptical Fourier analysis on it. On the matrix of coefficients, we will calculate a Principal Component Analysis and plot it, then test using a Multivariate analysis of variance whether the beer bottles have different shapes than the whisly bottles.

```
library(Momocs)
data(bot)
bot
```

```
## An Out object with:
## --------------------
##  - $coo: 40 outlines (162 +/- 21 coordinates, all unclosed)
##  - $fac: 1 grouping factor:
##      'type' (2): beer, whisky.
```

This overview indicates that `bot` is a `Out` object, some outlines (x; y) coordinates, and also that we have one "classifier" in the `\$fac` slot, called "type" and that indicates whether the corresponding outline is a beer or a whisky bottle.

```
panel(bot, names=TRUE, fac="type")
```

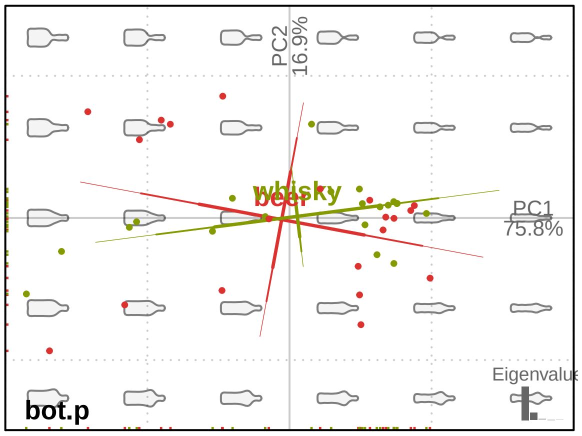


On these, bottles, let's apply an elliptical Fourier analysis with default settings. Most of the core function of Momocs have default settings but print a message when important decisions are made for you. Here, the number of harmonics chosen.

```
bot.f <- eFourier(bot, nb.h=12)
```

And then a principal component analysis, using the `type` classifier to draw groups of beer and whisky bottles.

```
bot.p <- PCA(bot.f)
plot(bot.p, "type")
```



Then, a multivariate analysis of variance on the matrix of coefficients (`bot.f`) to test if the shapes of the two groups differ.

```
Manova(bot.f, "type")
```

```
##  * 1st harmonic removed (because of normalization)
##  * 'retain' was missing. MANOVA done with 10 harmonics and the first 1 dropped

##           Df Hotelling-Lawley approx F num Df den Df Pr(>F)
## fac        1              441     36.7     36      3 0.0062 **
## Residuals 38
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So, whisky and beer bottles do not "significantly" differ in their shape.

1.6.0.3 Contact me!

If you: * struggle to "make it works" * have a doubt on something * found a bug * would like to propose an idea but do not know how to code it * would like to share some code *etc. please do not hesitate to contact me. Your remarks are precious and you will be treated accordingly ;-)

`bonhomme.vincent@gmail.com`

Also, if you would like to try morphometrics on your objects and feel that it would me more efficient/quick/pleasant to do it through a collaboration, drop me a line!

Chapter 2

Graphics

Morphometrics are, by essence, highly concerned with shape graphics and the multivariate statistics we can turn them into. Here I review the core graphical functions. They are all derived from `base`graphics yet they may be turned into a `ggplot2` unified style at some point. They are presented in a "chronological" order, from operation on single shapes, to representation of `Coo` objects that are collections of shapes.

The (various) graphics that illustrate multivariate statistics are treated in the vignette dedicated to Multivariate analyses.

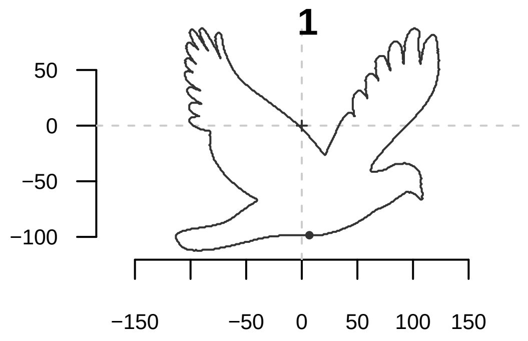
Note that graphics below may have a funny appearance. This is due to the restrictions on the graphics/vignettes sizes. Everything should be ok when code is runned on your machine.

2.1 Shape graphics

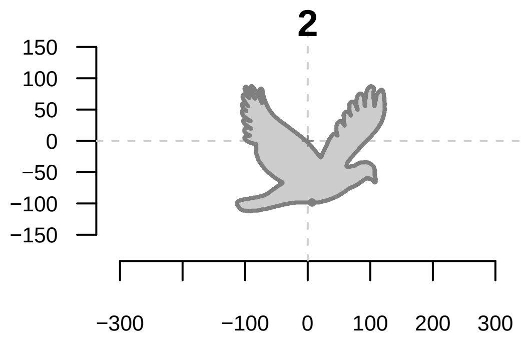
2.1.0.4 Single shape

`coo.plot` is the basic shape plotters in Momocs, used in many functions/methods. There is a bunch of options detailed exhaustively in `?coo.plot`, some are illustrated below.

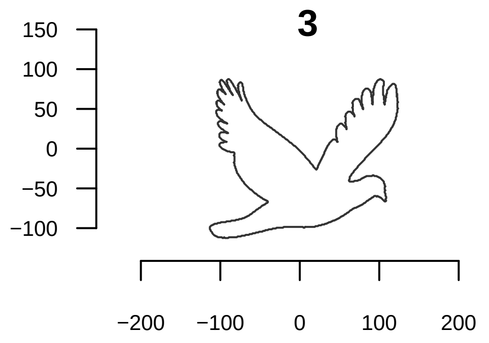
```r
library(Momocs)
data(shapes)
shp <- coo.center(shapes[22])
coo.plot(shp)
title(1)
```



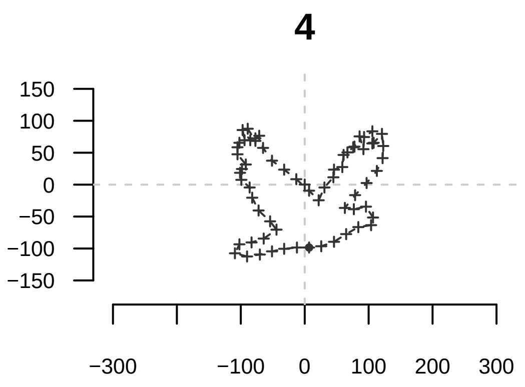
```
coo.plot(shp, col = "grey80", border = "grey50", lwd = 2,
         zoom = 1.5, )
title(2)
```



```
coo.plot(shp, ylim=c(-130, 150),
         xy.axis = FALSE, first.point = FALSE, centroid=FALSE)
title(3)
```



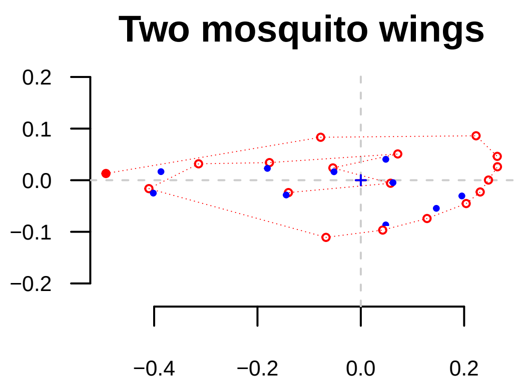
```
coo.plot(coo.sample(shp, 64), lty=2, pch=3, points = TRUE, zoom=1.5, main="4")
```



2.1.0.5 Add shape(s)

coo.draw is a shortcut for adding a shape to an existing plot. First, we shift to the wings dataset. See ?wings for details and credits.

```
data(wings)
w1 <- wings[1]
w2 <- wings[2]
coo.plot(w1, border = "#FF0000", lwd = 0.5, lty = 3, main="Two mosquito wings")
coo.draw(w2, border="#0000FF", col=.transp("000FF", 0.9), poly=FALSE, pch=20)
```

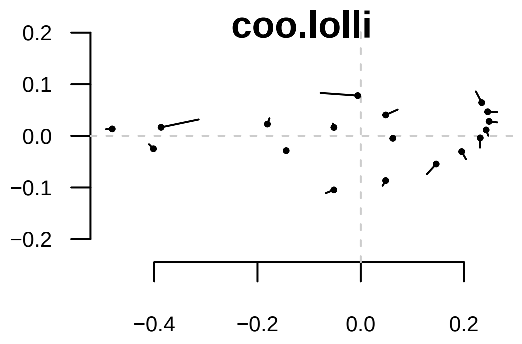


Note that the red-dotted outlines are not biologically relevant, since we want to display landmark positions, with some of them being within the wings. In other words, this should not be drawn as a polygon. That is what we have done with the second shape, with the blue dots, by passing poly=FALSE, for coo.draw which simply adds a shape in the existing graphical window.

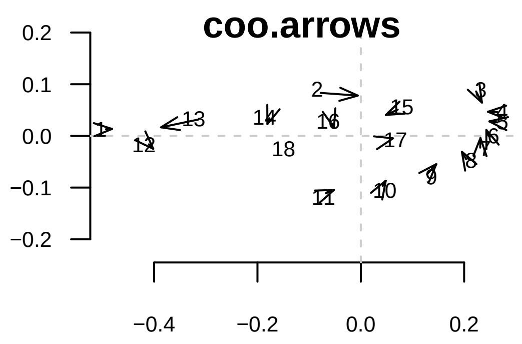
2.1.0.6 Two shapes: differences

If we want to illustrate differences in landmark positions, we can use `coo.lolli` (lollipops/lolliplots), and `coo.arrows`. In the background we add, for the first plot, landmarks that correspond to the mean shape. For the second plot, we add landmarks labels which are the row numbers for the shapes.

```
coo.lolli(w1, w2, main="coo.lolli")
```



```
coo.arrows(w1, w2, main="coo.arrows")
ldk.labels(w1)
```

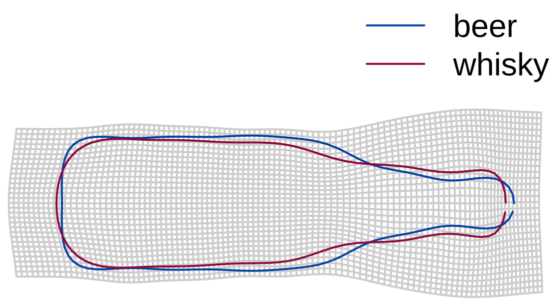


2.1.0.7 Two shapes: Thin plate splines

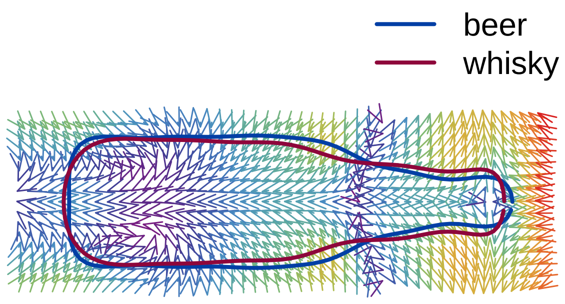
We will extract the mean shapes of whisky and beer bottles, from the bottles dataset. We have seen the first two steps before. We add some custom arguments to the `tps.` family but it “works” with only the first two.

```
data(bot)
bot.f <- eFourier(bot, 12)
ms <- mshapes(bot.f, "type") #mshapes 'type'-wise, see bot.f$fac
whisky <- ms$shp$whisky
```

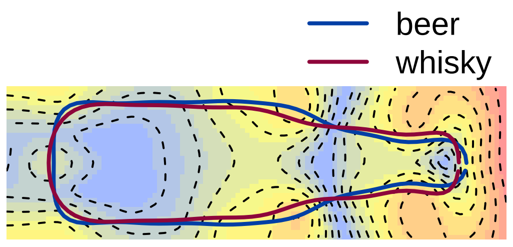
```r
beer   <- ms$shp$beer
tps.grid(beer, whisky, grid.size = 30, amp=2)
```



```r
tps.arr(beer, whisky, amp=2, palette=col.summer2)
```



```r
tps.iso(beer, whisky, amp=2, palette=col.spring)
```



2.2 Coo objects: collection of shapes

Let's work on the main objects of Momocs: `Coo` objects. Whether, we are working with `Out`, `Opn` or `Ldk` we roughly want the same thing: check singles shapes within them or plot them all, as a family picture or on the same window.

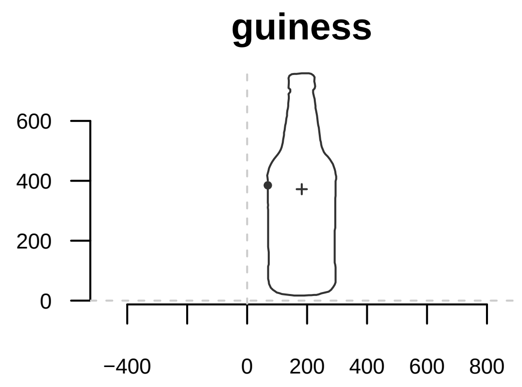
```r
data(bot)
```

2.2.0.8 Quick inspection

`Coo` objects can be directly plot, with the old-friend `plot`. If `id` is not specified, then random shapes are plotted and you will be asked to press `<Enter>` to continue your inspection.

```r
plot(bot) # random inspection (not shown)
```

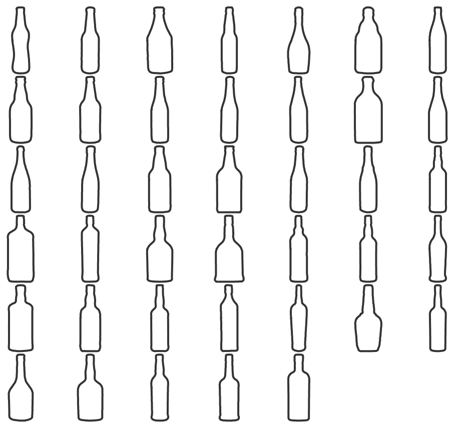
```r
plot(bot, 9) # let's go for a stout
```



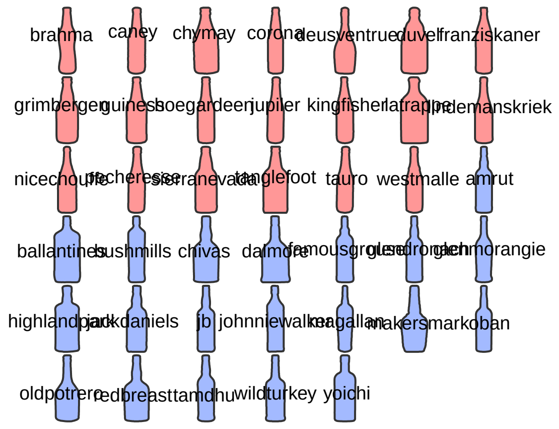
2.2.0.9 Panel

You may want to present all your data. That is a job for `panel`. Mind that shapes are templated with `coo.template` and size differences will thus be tempered (if not completely invisible). You can use a combination of `fac` and `reorder` to present graphically your groups.

```r
panel(bot)
```



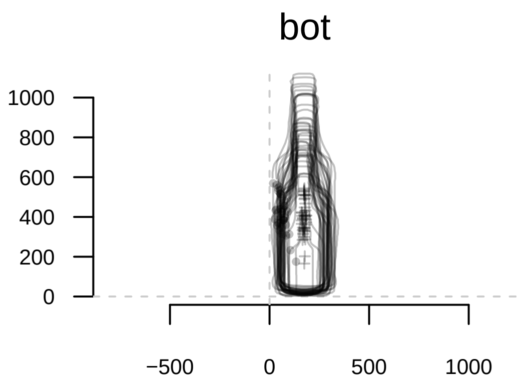
```r
panel(bot, fac = "type", palette=col.spring, names=TRUE)
```

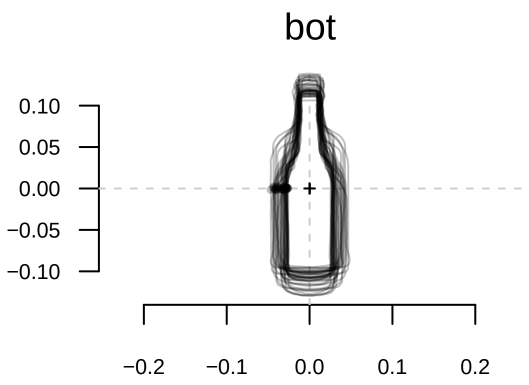


2.2.0.10 Stack

Have a look to this method: it plots all the shapes on the same window. It is obvious that shapes are neither centered nor scaled. Also, the first point of their outlines varies along their left side. This may be important to take into account, depending on the subsequent analyses (*e.g* `eFourier`).

```r
stack(bot)
bot <- coo.slidedirection(coo.scale(coo.center(bot)), "W")
stack(bot)
```





2.2.0.11 Others `Coo` objects

The functions presented above on `Out` can be applied to `Opn` and `Ldk`.

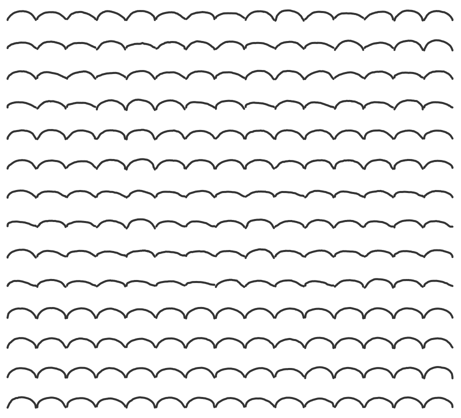
```r
data(olea) # Opn
class(olea)
```

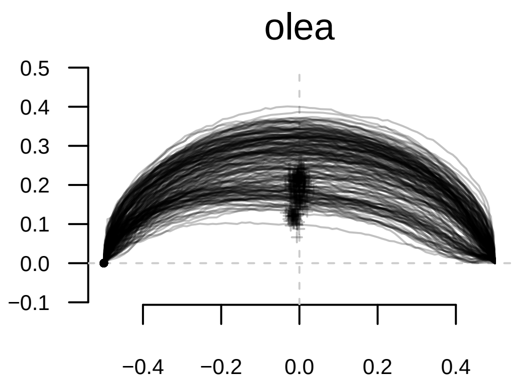
```
## [1] "Opn" "Coo"
```

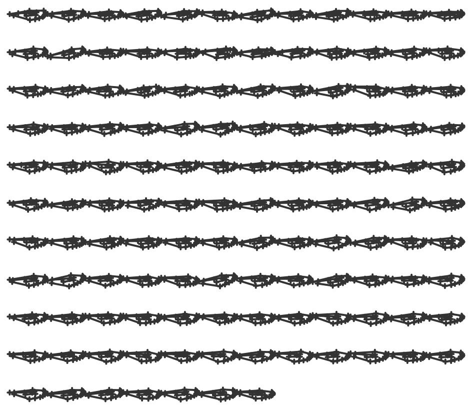
```r
panel(olea)
stack(olea)

data(wings) # Ldk
class(wings)
```

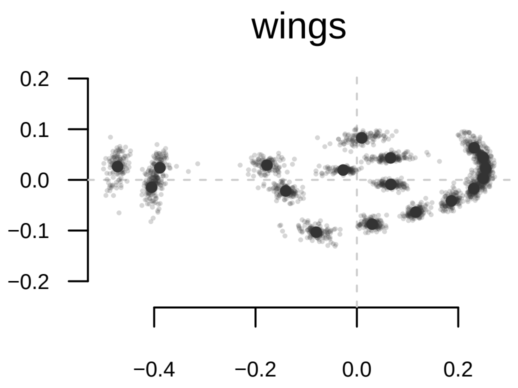
```
## [1] "Ldk" "Coo"
```

```r
panel(wings)
stack(wings)
```









Note that `Ldk` specific functions will be improved in a further version, to include, among other things, links between landmarks.

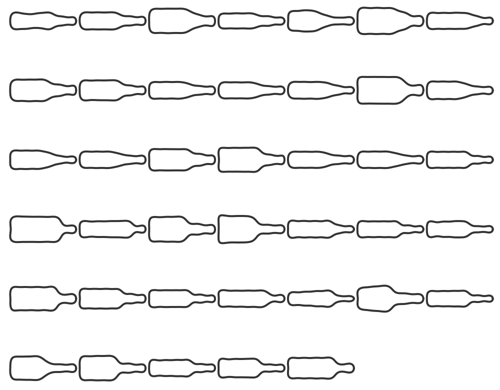
2.3 `Coe` objects: morphometric coefficients

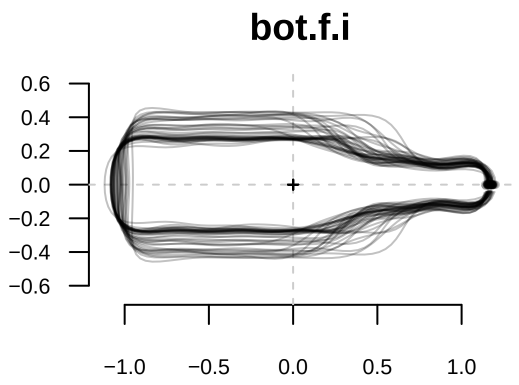
Some graphic function also apply on `Coe` objects. Try:

```
bot.f <- eFourier(bot, 12)
hist(bot.f)
boxplot(bot.f)
```

You can also pass `Coo` methods to `Coe` objects. Momocs will thus use the inverse transformation to how well (or bad) morphometric coefficients have capture shape structure.

```
panel(bot.f)
stack(bot.f)
```

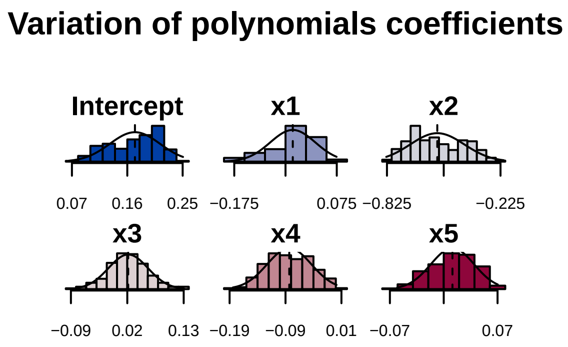


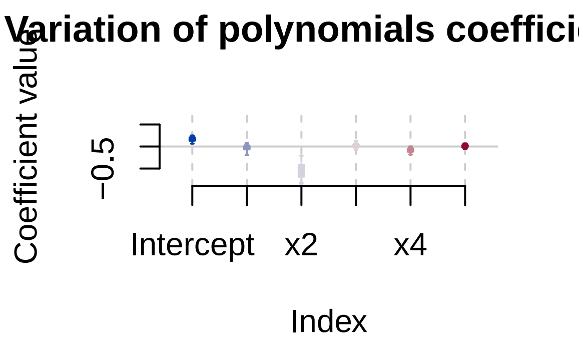


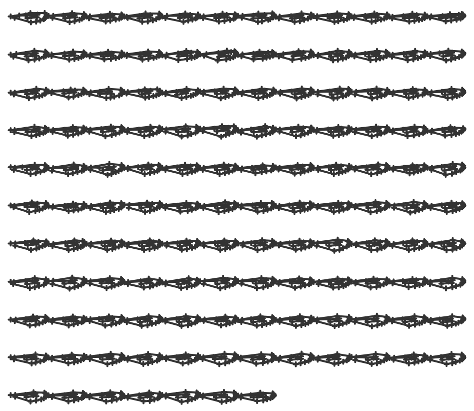
Above, it is obvious that default normalization parameters - which consumes A1, A2, A3 as seen on the graphs above - of the Elliptical Fourier Transforms are failing since some outlines have the bottleneck on the left and others on the right.

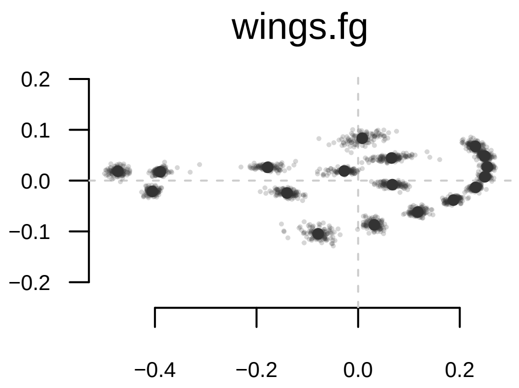
These functions also work on `OpnCoe` and on `LdkCoe`:

```r
olea.p <- orthoPolynomials(olea, degree = 5, nb.pts = 50)
hist(olea.p)
boxplot(olea.p)
#todo
#panel(olea.p)
#stack(olea.p)

wings.fg <- fgProcrustes(wings, verbose = FALSE)
#todo
#hist(wings.fg)
#boxplot(wings.fg)
panel(wings.fg)
stack(wings.fg)
```








wings.fg
0.2
0.1
0.0
−0.1
−0.2
−0.4
−0.2
0.0
0.2

Chapter 3

Outline analysis

Preliminary note.

3.1 eFourier: Elliptical Fourier Transforms

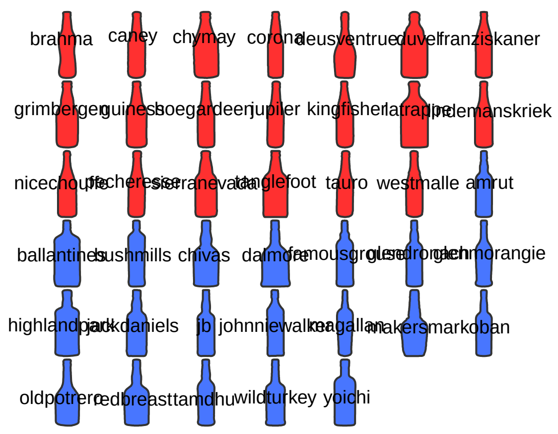
```r
library(Momocs)
data(bot)
bot
```

```
## An Out object with:
## --------------------
##  - $coo: 40 outlines (162 +/- 21 coordinates, all unclosed)
##  - $fac: 1 grouping factor:
##      'type' (2): beer, whisky.
```

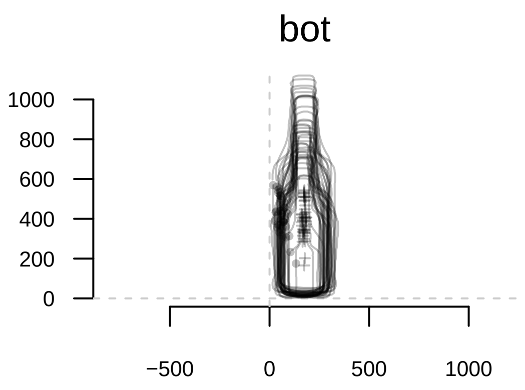
3.1.0.12 How EFT works

```r
#coo.oscillo(bot[1], method = "efourier")
```

```r
panel(bot, fac="type", names=TRUE)
```



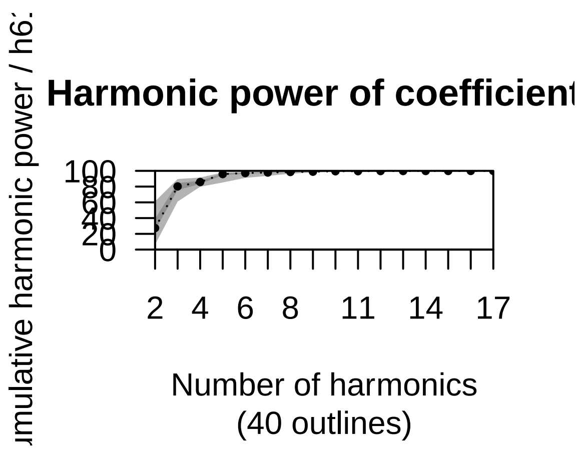
```r
stack(bot)
```



3.1.0.13 Calibration

```r
hpow(bot)
```

```
##   * Method not provided. hpow | efourier is used.
```

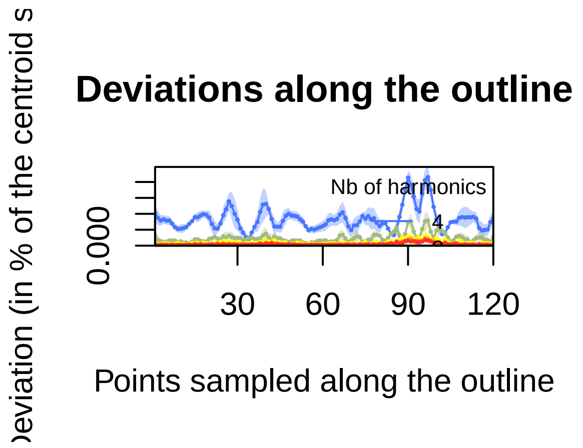


```
## $q:
```

```
##            h2    h3    h4    h5    h6    h7    h8    h9   h10   h11   h12
## q0      6.275 60.96 79.81 85.24 91.05 93.61 96.39 97.91 98.80 98.82 99.36
## q0.25 24.556 75.61 83.41 94.57 96.26 97.64 98.23 98.36 99.14 99.30 99.69
## q0.5  27.325 80.23 85.83 95.92 97.20 97.86 98.58 98.91 99.50 99.58 99.76
## q0.75 39.114 84.53 87.24 97.01 97.60 98.42 98.95 99.14 99.67 99.71 99.81
## q1    61.131 89.63 91.51 97.95 98.44 98.78 99.43 99.48 99.76 99.79 99.89
##          h13   h14   h15   h16   h17
## q0    99.56 99.72 99.76 99.80 99.80
## q0.25 99.74 99.79 99.82 99.86 99.88
## q0.5  99.80 99.84 99.86 99.89 99.91
## q0.75 99.84 99.89 99.91 99.93 99.94
## q1    99.90 99.95 99.96 99.96 99.97
##
## $minh:
##   90%   95%   99% 99.9%
##     5     5    10    17
```

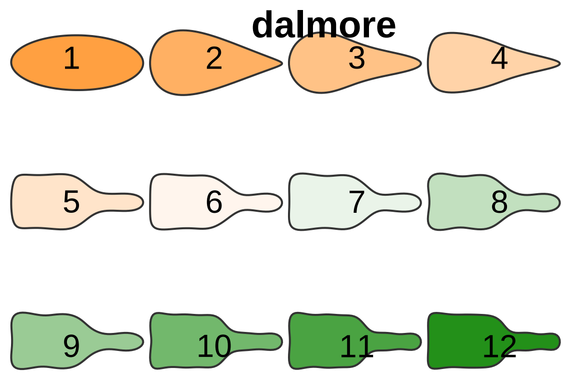
```r
hquant(bot, id=1:5)
```

```
##   * Method not provided. efourier is used.
```



```r
hqual(bot, harm.range = 1:12)
```

```
##  * Method not provided. efourier is used.
```



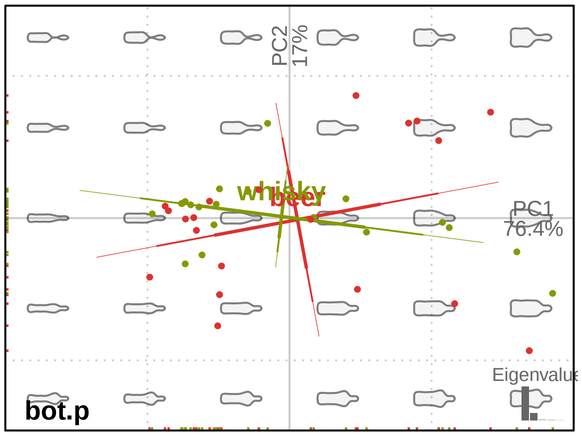
3.1.1 EFT

```
bot.f <- eFourier(bot, 9)
bot.f
```

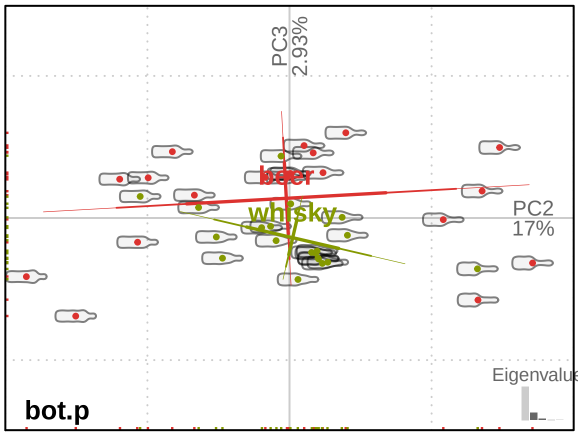
```
## An OutCoe object [ elliptical Fourier analysis ]
## --------------------
##  - $coe: 40 outlines described, 9 harmonics
##  - $coe: 1st harmonic coefficients from random individuals:
##               A1    A2    A3 B1    B2    B3 C1     C2     C3    D1     D2
## hoegardeen    1 0.010 0.092  0 0.000 0.000  0 -0.003  0.001 0.295 -0.069
## nicechouffe   1 0.015 0.091  0 0.001 0.000  0 -0.001  0.000 0.313 -0.091
## pecheresse    1 0.008 0.094  0 0.000 0.000  0  0.002  0.000 0.288 -0.071
## dalmore       1 0.039 0.073  0 0.001 0.001  0  0.001 -0.002 0.415 -0.148
## wildturkey    1 0.009 0.092  0 0.001 0.001  0  0.000 -0.001 0.319 -0.090
##                  D3
## hoegardeen  0.051
## nicechouffe 0.044
## pecheresse  0.052
## dalmore     0.044
## wildturkey  0.036
## etc.
##  - $fac: 1 grouping factor:
##      'type' (2): beer, whisky.
```

3.1.1.1 A PCA

```
bot.p <- PCA(bot.f)
plot(bot.p, "type")
```



```
# some variation around PCA
plot(bot.p, "type", xax=2, yax=3, pos="xy")
```



### MANOVA

```r
Manova(bot.f, "type")
```

```
##  * 1st harmonic removed (because of normalization)
##  * 'retain' was missing. MANOVA done with 9 harmonics and the first 1 dropped

##           Df Hotelling-Lawley approx F num Df den Df   Pr(>F)
## fac        1               58     12.7     32      7 0.00095 ***
## Residuals 38
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.1.1.2 LDA

```r
# Let's try and LDA now
bot.l <- LDA(bot.f, "type")
```

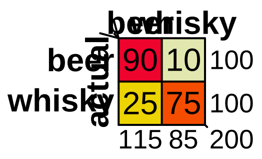
```
##  * variables A1 B1 C1 are removed since they are constant.
```

```r
bot.l
```

```
## Leave-one-out cross-validation: (82.5% - 33/40):
##         classified
## actual   beer whisky
##   beer     18      2
##   whisky    5     15
```

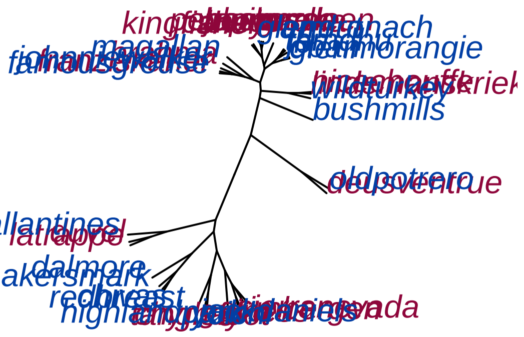
```r
#plot(bot.l)
plotCV(bot.l$CV.tab)
```

```
##           actual
## classified whisky beer
##     beer        5   18
##     whisky     15    2
```



3.1.1.3 Hierachical clustering

```r
clust(bot.f, "type", palette=col.gallus)
```



3.1.1.4 Mean shapes and Thin Plate Splines

Try:

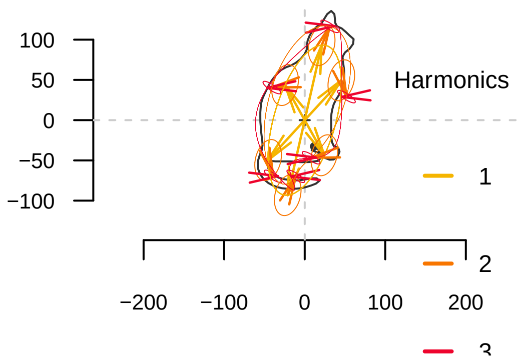
```r
ms <- mshapes(bot.f, "type")$shp
tps.iso(ms$beer, ms$whisky)
#tps.arr(ms$beer, ms$whisky)
#tps.grid(ms$beer, ms$whisky)
```

3.2 Which Fourier analysis?

```r
data(shapes)
# cats are prone to high frequency noise (well, just tfourier)
```

```
cat <- coo.sample(coo.smooth(shapes[4], 10), 120)
#coo.oscillo(cat, "all")
```

```
Ptolemy(cat)
```

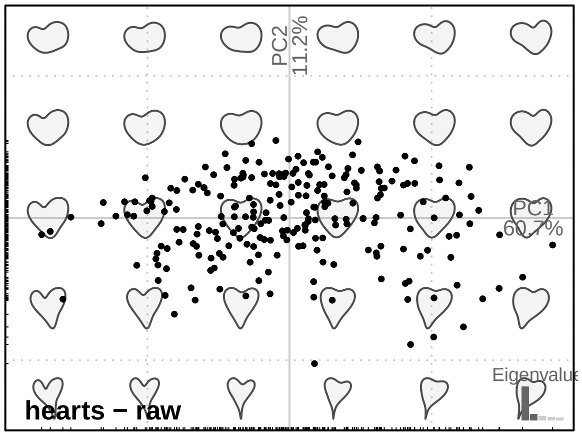


3.3 EFT

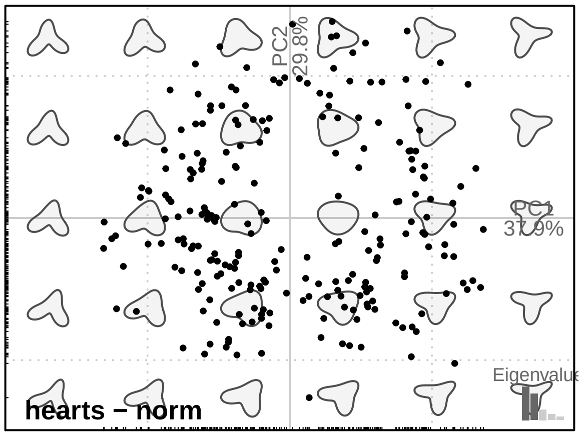
3.3.1 Normalization

3.3.1.1 numerically

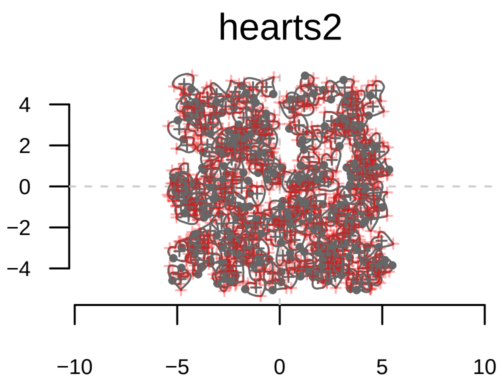
```
data(hearts)
hearts.raw  <- eFourier(hearts, nb.h=8, norm=FALSE)
#hist(hearts.raw)
hearts.norm <- eFourier(hearts, nb.h=8, norm=TRUE)
#hist(hearts.norm)
plot(PCA(hearts.raw),  border.shp="grey30", title="hearts - raw")
```



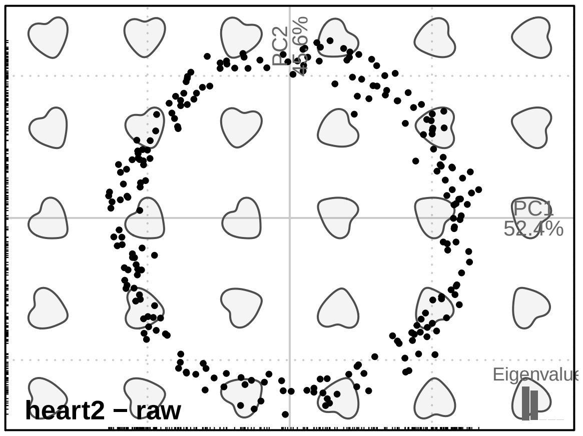
```
plot(PCA(hearts.norm), border.shp="grey30", title="hearts - norm")
```



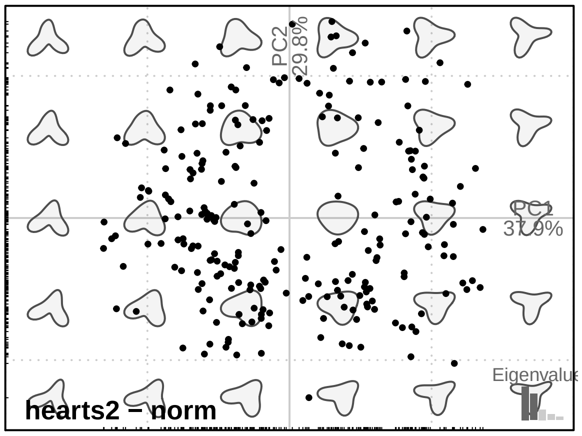
```r
hearts2 <- hearts
set.seed(123)
hearts2$coo <- lapply(hearts2$coo, function(x)
  coo.trans(coo.rotate(x, runif(1, -pi, pi)), x=runif(1, -5, 5), y=runif(1, -5, 5)))
stack(hearts2, borders="grey40")
```



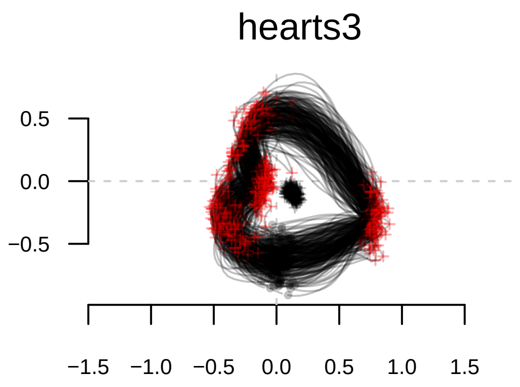
```r
hearts2.raw  <- eFourier(hearts2, nb.h=8, norm = FALSE)
hearts2.norm <- eFourier(hearts2, nb.h=8, norm = TRUE)

plot(PCA(hearts2.raw),  border.shp="grey30", title="heart2 - raw")
```



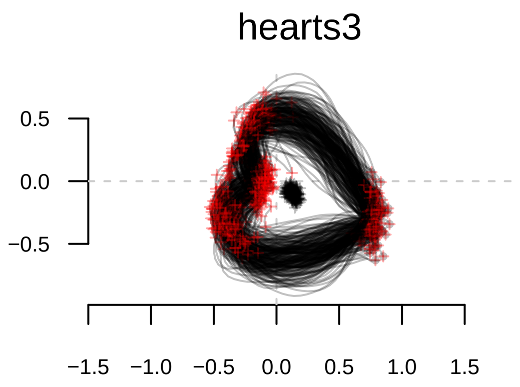
```
plot(PCA(hearts2.norm), border.shp="grey30", title="hearts2 - norm")
```



```
hearts3 <- fgProcrustes(hearts2)
stack(hearts3)
```

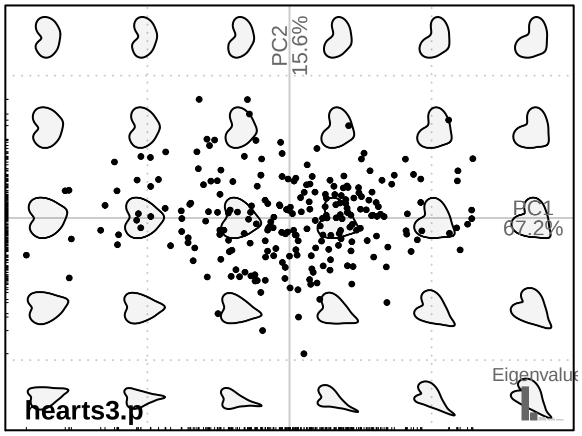


```
hearts3 <- coo.slide(coo.bookstein(hearts3, 2, 4), id1 = 4)
stack(hearts3)
```



```
hearts3.f <- eFourier(hearts3, nb.h=8, norm=FALSE)
hearts3.p <- PCA(hearts3.f)
```

```r
plot(hearts3.p, border.shp="black")
```



3.3.1.2 With landmarks

3.3.1.3 With fgP

3.3.2 Calibration

3.3.2.1 hqual

3.3.2.2 hquant

3.3.2.3 hpow

3.3.3 From outlines to coefficients

```
hist
boxplot
hcontrib
removeSymmetric
```