João Coelho

20 de Julho de 2015

## Getting started

# First we make sure to load the required R packages and its dependencies:  
  
if (!require('knitr')){  
 install.packages('knitr', dependencies = TRUE)  
 library(knitr)  
}

## Loading required package: knitr

if (!require('rmarkdown')){  
 install.packages('knitr', dependencies = TRUE)  
 library(knitr)  
}

## Loading required package: rmarkdown

options(knitr.table.format = 'markdown')  
  
if (!require('geomorph')){  
 install.packages('geomorph', dependencies = TRUE)  
 library(geomorph)  
}

## Loading required package: geomorph  
## Loading required package: rgl  
##   
## Attaching package: 'rgl'  
##   
## The following objects are masked from 'package:knitr':  
##   
## hook\_rgl, hook\_webgl

if (!require('pastecs')){  
 install.packages('pastecs', dependencies = TRUE)  
 library(pastecs)  
}

## Loading required package: pastecs  
## Loading required package: boot

if (!require('car')){  
 install.packages('car', dependencies = TRUE)  
 library(pastecs)  
}

## Loading required package: car  
##   
## Attaching package: 'car'  
##   
## The following object is masked from 'package:boot':  
##   
## logit

if (!require('RWeka')){  
 install.packages('RWeka', dependencies = TRUE)  
 library(RWeka)  
}

## Loading required package: RWeka

source('./R/pts2array.R') # gets a function written in collaboration with David Navega. It reads all .pts files inside a directory into R and saves them as an array.

All the results presented within this manuscript have been automatically generated as a .docx file through the knitr package for R. Even this paragraph itself was not written in Word, but instead in R. Contrary to the standard, this has the advantage of creating a text accompanied by the exact code that generated the graphics, tables and other results. Hence, increasing the reproducibility criteria, allowing other researchers to easily run the data analysis into their personal computers as long as the data is shared. This adheres to the tenants of the Open Data movement and Science 2.0 philosophy by promoting data analysis transparency among scientists.

If you desire to execute in your computer the whole data analysis present in this thesis you should first make sure to:

* Have installed R and the latest version of RStudio
* Install the latest version of the knitr package, by writing install.packages("knitr") in your RStudio console.

To run the code that produced my thesis output:

* Open RStudio, and go to File > New > R Markdown
* Paste in the contents of the resultsgenerator.Rmd available in the thesis' CD-ROM or by contacting me: [joao@osteomics.com](mailto:joao@osteomics.com)
* Click Knit Word

Don't forget to setwd("YOUR/FOLDER/LOCATION") to where the landmark files and 3D meshes are located. Or even easier, just open the HOTProject-GM.Rproj file, since it automatically loads everything.

Everything should run fine if you follow what is said above. Just to clarify, from here on, every time you see chunks of code, you can easily identify them through the box with light background and different font type that get’s colored by association of the type of programming object. There you also see some commands that usually have a *#* symbol followed by some sentence, which are explanatory commentaries upon what the code is doing to help non-experts understanding what is being programmed. If it's a double *##* it is an output generated by the code.

## Data Input

Our landmarks were obtained from the free software Landmark Editor. While we tried to use auto3Dgm for the same purpose, in order to compare the two approaches (i.e. manual versus automatic), as it can be seen in Fig. 4 that was generated from the output of the auto3Dgm algorithm, 5 burnt bones were completely inadequately aligned. However, this would effectively reduce the geometric *n* by 10, since their counterparts (i.e. the unburnt) would not be used for any particular purpose without their pair. Considering that no way was found within R to solve this problem, this approach was discontinued mid-way through the project because it would considerable reduce our *n*.

SW <- read.csv(file = './data/SingularWarps.csv', header = TRUE) # loads all the variables that aren't shape-data.  
  
rawdata <- pts2array(pts.dir = './data/pts-files') # loads raw landmark data into R, similary to readland.nts(), check ?readland.nts  
  
dimnames(rawdata)[[3]] # reads each geometric configuation's name. Useful to confirm if the order and naming are correct and consistent with singular warps order.

## [1] "CEIXXI05F" "CEIXXI05F900" "CEIXXI08F" "CEIXXI08F700"   
## [5] "CEIXXI17M" "CEIXXI17M900" "CEIXXI24F" "CEIXXI24F800"   
## [9] "CEIXXI26F" "CEIXXI26F900" "CEIXXI29M" "CEIXXI29M800"   
## [13] "CEIXXI32F" "CEIXXI32F800" "CEIXXI35M" "CEIXXI35M900"   
## [17] "CEIXXI43M" "CEIXXI43M800" "CEIXXI49F" "CEIXXI49F850"   
## [21] "CEIXXI50F" "CEIXXI50F900" "CEIXXI51M" "CEIXXI51M900"   
## [25] "CEIXXI53F" "CEIXXI53F800" "CEIXXI57M" "CEIXXI57M900"   
## [29] "CEIXXI64M" "CEIXXI64M800" "CEIXXI65F" "CEIXXI65F900"   
## [33] "CEIXXI79M" "CEIXXI79M900" "CEIXXI86M" "CEIXXI86M1000"  
## [37] "CEIXXI97F" "CEIXXI97F1050"

## Data pre-processing and GPA

Some *humerii* from our sample don’t have anatomical parts where specific landmarks should be located. This was either due to taphonomic processes or because of the heating experiment itself. However robust methods have been devised within the Geometric Morphometrics approach in order to handle missing data. For solving this, the very convenient estimate.missing() function from the geomorph package is used to generate estimates of the missing landmarks.

EM <- estimate.missing(rawdata, method = 'TPS') # This command estimates missing landmarks using either Thin Plate Spline or Regression. Here TPS was choosen.

Perfoming GPA over your data is the quintessential step to start a Geometric Morphometrics analysis. In geomorph this is achieved with the gpagen() command. Our aligned Procrustes coordinates, and specimens' centroid sizes are recorded as procrustes$coords and procrustes$Csize, respectively.

procrustes <- gpagen(EM, ShowPlot = FALSE) # Generalized Procrustes Analysis, creates a viable dataset for applying geometric morphometrics methods.

Right now we have everything we need to start a multivariate statistical analysis of shape and its covariation with other variables. However it is better to do some graphical exploration, in order to understand our data and correctly obtain fruitful inferences from it.

## Exploratory Data Analysis

An essential step of every data analysis that rarely takes the spotlight is the visual data exploration that antecedes inference or modeling. It is extremely important in the sense that allows us to allocate our precious time in more fruitful avenues, rather than trying everything for all the variables without a rigorous aim in mind. The exploratory data analysis was broken into two main steps. First we intend to summarize, describe and visualize general aspects of our data. This is achieved in 3.3.1, but also complemented with the Table 4 in Appendix 7.1. Second, we use a method based on the Geometric Morphometrics toolkit to look for possible errors in the landmark placing protocol.

### Descriptive Statistics

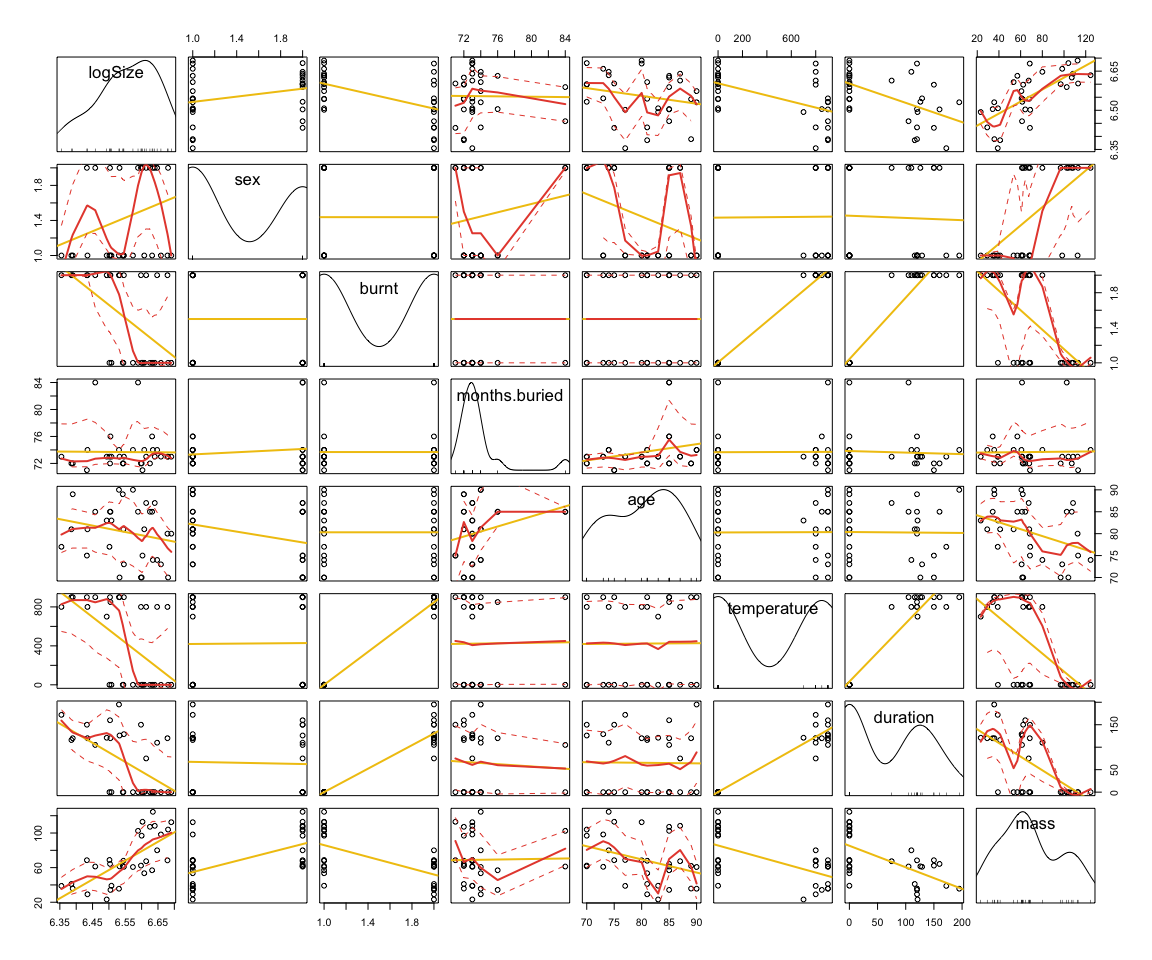
Now, we perform quick and very simple data manipulation to allow the construction of graphical plots and tables. We provide counts of values, and many classic measures in the theory of probability distributions.

logSize <- log(as.numeric(procrustes$Csize)) # The log of Centroid Size is useful for creating models.  
  
Shapes <- procrustes$coords # Gives a new name easier to remember.  
  
kable(stat.desc(cbind(logSize,SW[,-c(1:4)]), norm = TRUE), digits = 2) # some descriptive statistics of our non-shape variables. Categorical variables were removed because most of these stats only make sense for numeric variable.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | logSize | months.buried | age | temperature | duration | mass |
| nbr.val | 38.00 | 32.00 | 38.00 | 38.00 | 38.00 | 38.00 |
| nbr.null | 0.00 | 0.00 | 0.00 | 19.00 | 19.00 | 0.00 |
| nbr.na | 0.00 | 6.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| min | 6.35 | 71.00 | 70.00 | 0.00 | 0.00 | 23.20 |
| max | 6.69 | 84.00 | 90.00 | 1050.00 | 195.00 | 124.49 |
| range | 0.34 | 13.00 | 20.00 | 1050.00 | 195.00 | 101.29 |
| sum | 249.39 | 2358.00 | 3050.00 | 16500.00 | 2569.00 | 2588.03 |
| median | 6.59 | 73.00 | 81.00 | 350.00 | 37.50 | 62.16 |
| mean | 6.56 | 73.69 | 80.26 | 434.21 | 67.61 | 68.11 |
| SE.mean | 0.01 | 0.52 | 1.02 | 71.96 | 11.64 | 4.44 |
| CI.mean.0.95 | 0.03 | 1.06 | 2.08 | 145.80 | 23.59 | 9.00 |
| var | 0.01 | 8.61 | 39.87 | 196770.98 | 5151.98 | 750.38 |
| std.dev | 0.09 | 2.93 | 6.31 | 443.59 | 71.78 | 27.39 |
| coef.var | 0.01 | 0.04 | 0.08 | 1.02 | 1.06 | 0.40 |
| skewness | -0.54 | 2.64 | -0.11 | 0.05 | 0.26 | 0.33 |
| skew.2SE | -0.70 | 3.19 | -0.14 | 0.06 | 0.34 | 0.43 |
| kurtosis | -0.65 | 6.63 | -1.31 | -1.99 | -1.66 | -1.01 |
| kurt.2SE | -0.43 | 4.10 | -0.88 | -1.33 | -1.11 | -0.67 |
| normtest.W | 0.95 | 0.60 | 0.94 | 0.71 | 0.78 | 0.94 |
| normtest.p | 0.08 | 0.00 | 0.04 | 0.00 | 0.00 | 0.06 |

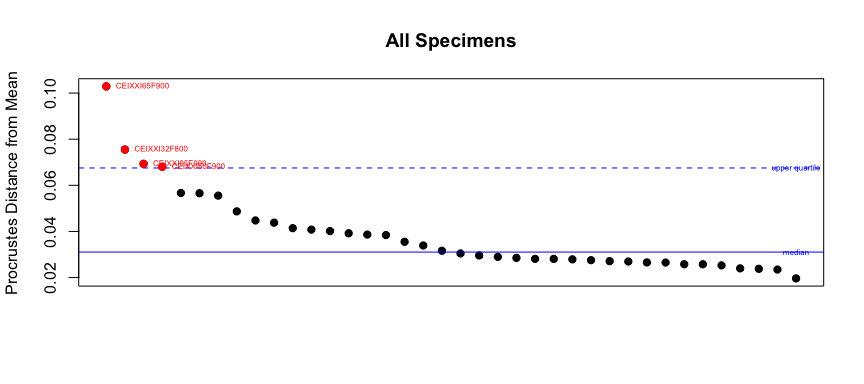
Next, we check potential correlations and possibility for regression models in our non-shape related data (i.e. Singular Warps) and the logarithm of Centroid Size. This last measure is also included because it is a vector of size and can be easily represented. Procrustes landmarks are a *p*  k \* x\* data matrix, in our case, a shape configuration of 35 landmarks, 3 dimensions and 38 individuals. Therefore, it would be too complicated to visualize through the graphical device of Fig. 6 or Table 1 and so shape data wasn’t included.

scatterplotMatrix(cbind(logSize, SW[,-c(1,2)]), col = c('#f1c40f', '#e74c3c', 'black'), lwd = 2)



Now looking at our shape data we try to see if there are any outliers within our dataset. This function is very useful because it allow us to verify if any specimen was incorrectly digitized (e.g. landmarks out of order or misplaced).

outliers <- plotOutliers(Shapes) # Plots potential outliers through calculation of Procrustes Distance.

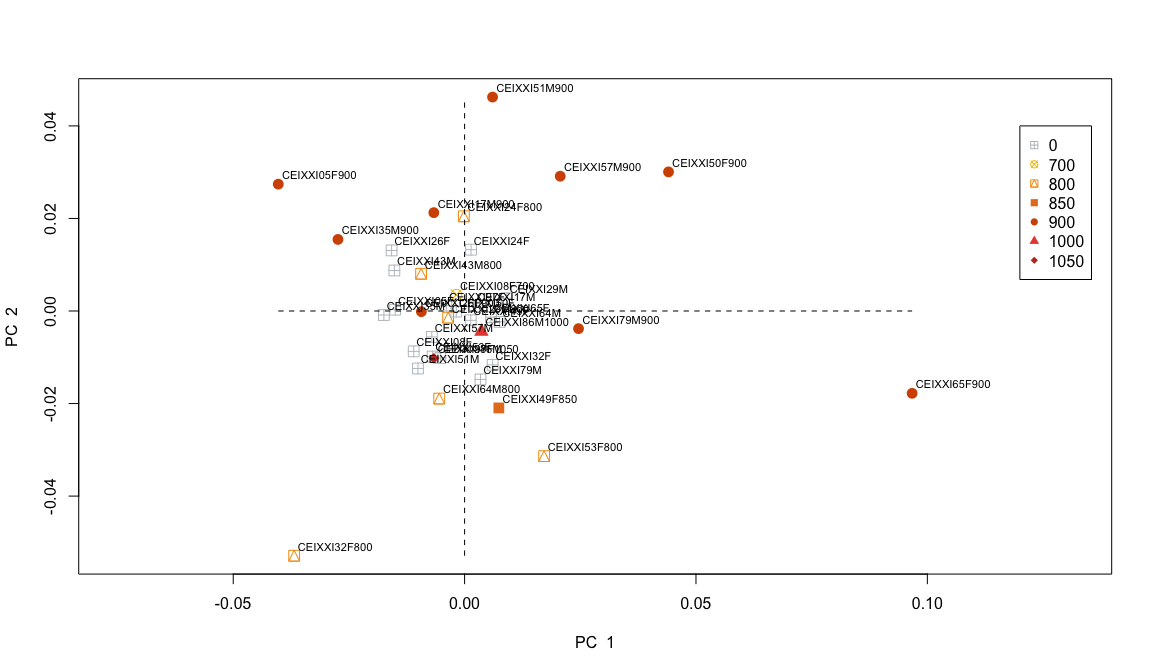


Interestingly, after a visual check in Landmark Editor, none of “outliers” marked in red had actually misplaced or disordered anatomical landmarks. These 4 appear as outliers because of how much heat-induced warping and fractures affected their shape. Together, CEI/XXI 65, 32, 5, and 50 represent the most extremely modified-by-heat *humerii* within our sample. This can be confirmed visually with the 3D meshes files, but is also sustained by performing a PCA on our data.

## Principal Component Analysis

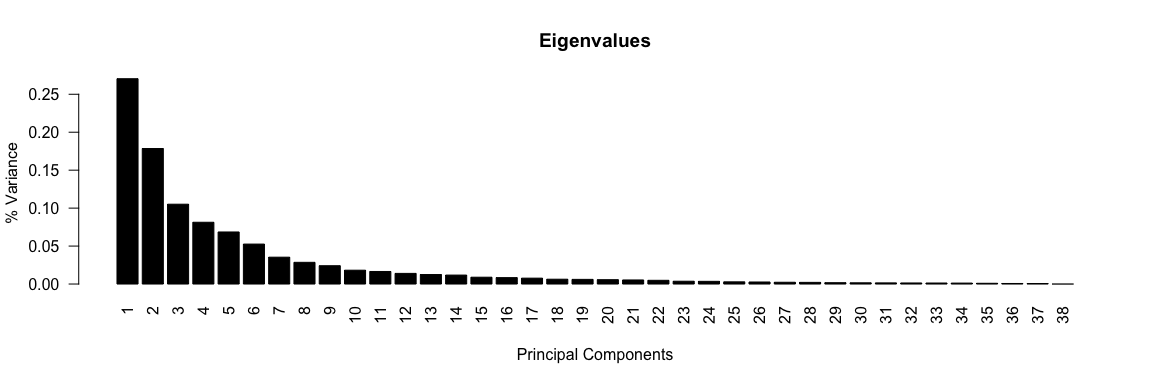
The following function plots a set of Procrustes-aligned specimens in tangent space along their principal axes. This plot illustrates that bones before heating and bones that were not much affected by the heating experiment tend to cluster around the Origin of the Cartesian plot of the two Principal Components. The more extremely modified by heat are dispersed in the periphery in what seems to be pseudorandom directions. A bigger dataset is needed to understand those patterns, which are possibly describing different ways in which the bone is being deformed by heat.

# Define graphical proprieties of the PC plot:  
  
gp <- as.factor(SW$temperature)  
col.gp <- c('#bdc3c7', '#f1c40f', '#f39c12', '#e67e22', '#d35400', '#e74c3c', '#c0392b')  
pch.gp <- c(12:18)  
names(col.gp) <- levels(gp)  
names(pch.gp) <- levels(gp)  
col <- col.gp[match(gp, names(col.gp))]  
pch <- pch.gp[match(gp, names(pch.gp))]  
  
# Calculate a PCA:  
  
y <- two.d.array(Shapes)  
pc.res <- prcomp(y)  
pcdata <- pc.res$x  
  
# Code our plot:  
  
plot(pcdata[, 1], pcdata[, 2], pch = pch, asp = 1, col = col, cex = 1.5, xlab = paste('PC ', 1), ylab = paste('PC ', 2))  
segments(min(pcdata[, 1]), 0, max(pcdata[, 1]), 0, lty = 2, lwd = 1)  
segments(0, min(pcdata[, 2]), 0, max(pcdata[, 2]), lty = 2, lwd = 1)  
text(pcdata[, 1], pcdata[, 2], dimnames(rawdata)[[3]], adj = c(-0.05, -0.7), cex = 0.7)  
  
# Add a legend:  
  
legend(0.12, 0.04, legend = levels(gp), pch = pch.gp, col = col.gp)



One should never forget that ultimately, PCA is nothing more than a rotation of the original data. Its utility lies in the fact that features measured in a study will exhibit covariances because they interact and are influenced by common processes.

# To plot a graph of the proportion of variance explained by each PC:  
  
pvar <- (pc.res$sdev^2)/(sum(pc.res$sdev^2))  
names(pvar) <- seq(1:length(pvar))  
barplot(pvar, main = 'Eigenvalues', xlab= 'Principal Components', ylab = '% Variance', col = 'black', las = 2, cex.names = 1)

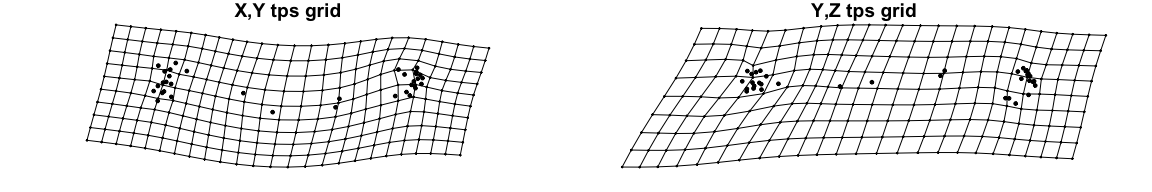


# Shape eigenvalues in descending order. The comulative porportion of the first 13 PCs explain more than 90% of the variance in the data.

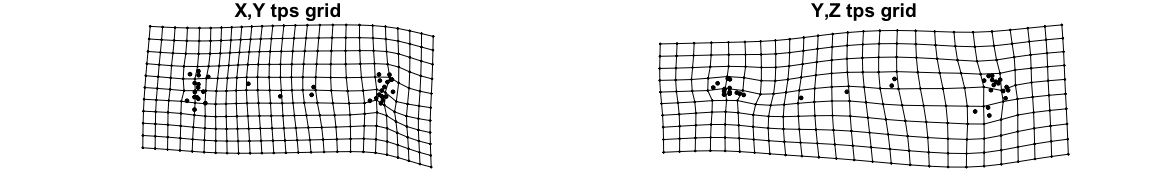
## Thin-Plate Spline Interpolaton

Also important is to visualize the changes within a particular bone, between its pre-heating shape and deformed-by-heat shape through TPS. For achieving this, we generate thin-plate spline deformation grids. To visualize 3D data deformations into paper format that bounds this thesis, thin-plate spline deformations are show in the x-y and x-z axis. The y-z axis is avoided, because it is very difficult to understand anything from that perspective. Also, because the other two views in 2D are sufficient to show all the 3 axes from 3D already. Thus, a third view is redundant.

GP <- gridPar(pt.bg = 'black', pt.size = 0.5, n.col.cell = 25) # defines general graphical proprieties of the following plots.  
  
plotRefToTarget(Shapes[,,1], Shapes[,,2], gridPars = GP, method = 'TPS') #CEIXXI05F



plotRefToTarget(Shapes[,,3], Shapes[,,4], gridPars = GP, method = 'TPS') #CEIXXI08F



plotRefToTarget(Shapes[,,5], Shapes[,,6], gridPars = GP, method = 'TPS') #CEIXXI17M



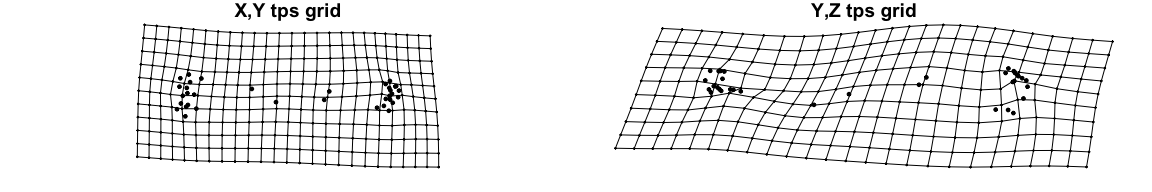
plotRefToTarget(Shapes[,,7], Shapes[,,8], gridPars = GP, method = 'TPS') #CEIXXI24F



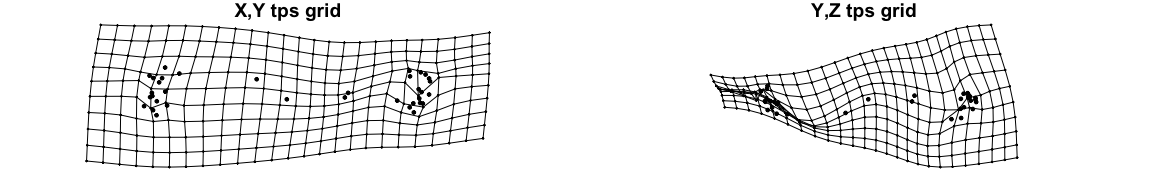
plotRefToTarget(Shapes[,,9], Shapes[,,10], gridPars = GP, method = 'TPS') #CEIXXI26F



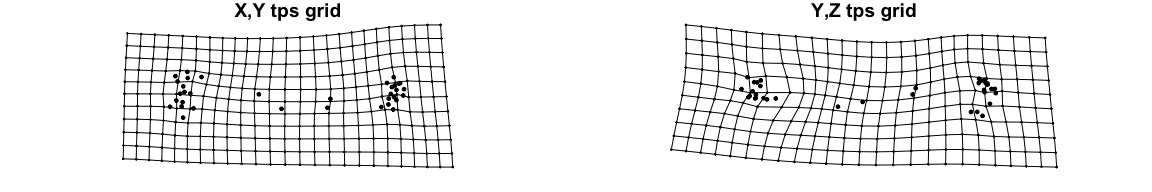
plotRefToTarget(Shapes[,,11], Shapes[,,12], gridPars = GP, method = 'TPS') #CEIXXI29M



plotRefToTarget(Shapes[,,13], Shapes[,,14], gridPars = GP, method = 'TPS') #CEIXXI32F



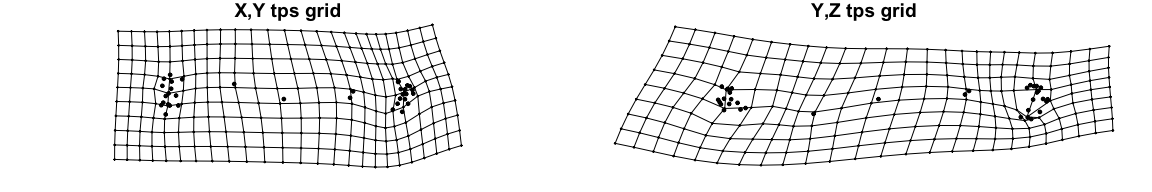
plotRefToTarget(Shapes[,,15], Shapes[,,16], gridPars = GP, method = 'TPS') #CEIXXI35M



plotRefToTarget(Shapes[,,17], Shapes[,,18], gridPars = GP, method = 'TPS') #CEIXXI43M



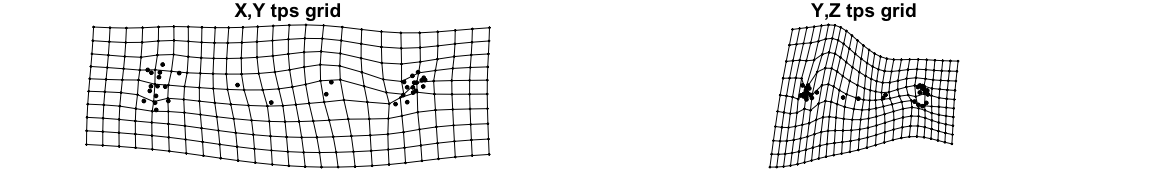
plotRefToTarget(Shapes[,,19], Shapes[,,20], gridPars = GP, method = 'TPS') #CEIXXI49F



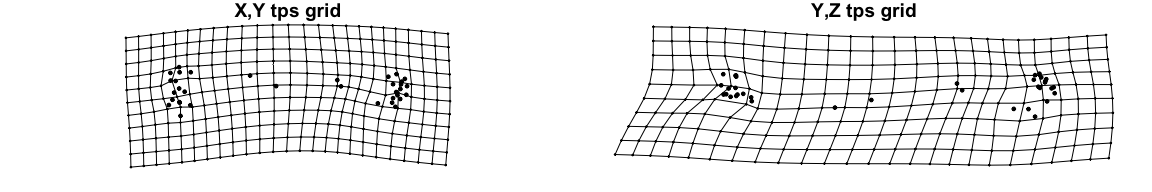
plotRefToTarget(Shapes[,,21], Shapes[,,22], gridPars = GP, method = 'TPS') #CEIXXI50F



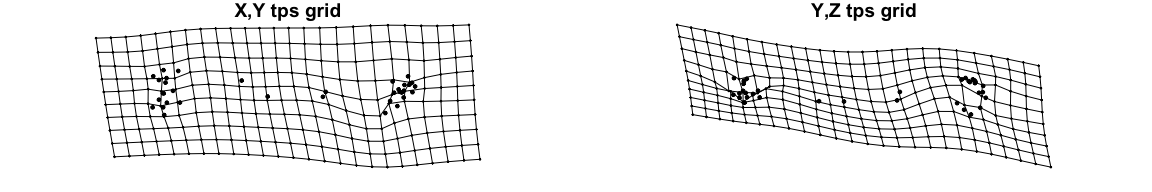
plotRefToTarget(Shapes[,,23], Shapes[,,24], gridPars = GP, method = 'TPS') #CEIXXI51M



plotRefToTarget(Shapes[,,25], Shapes[,,26], gridPars = GP, method = 'TPS') #CEIXXI53F



plotRefToTarget(Shapes[,,27], Shapes[,,28], gridPars = GP, method = 'TPS') #CEIXXI57M



plotRefToTarget(Shapes[,,29], Shapes[,,30], gridPars = GP, method = 'TPS') #CEIXXI64M



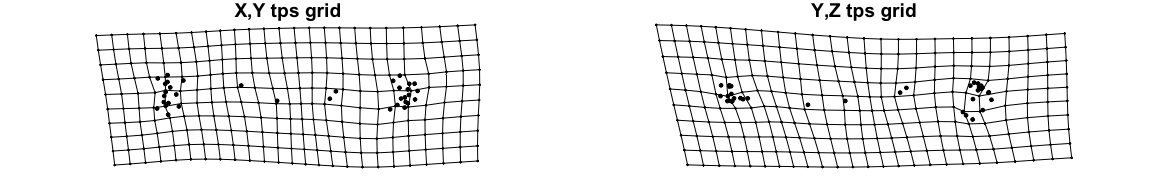
plotRefToTarget(Shapes[,,31], Shapes[,,32], gridPars = GP, method = 'TPS') #CEIXXI65F



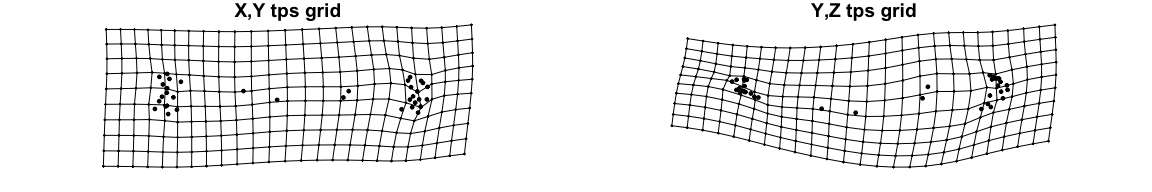
plotRefToTarget(Shapes[,,33], Shapes[,,34], gridPars = GP, method = 'TPS') #CEIXXI79M



plotRefToTarget(Shapes[,,35], Shapes[,,36], gridPars = GP, method = 'TPS') #CEIXXI86M



plotRefToTarget(Shapes[,,37], Shapes[,,38], gridPars = GP, method = 'TPS') #CEIXXI97F



## Predictive Modelling

“Essentially, all models are wrong, but some are useful.” — (George Box in Box & Draper, 1987: 424)

Creating models with such reduced sample sizes is usually not a good decision. Here, we create them, to show how easy it is to implement and test predictive models with our kind of data. While all models presented are already fully operational, it is hard to truly estimate their actual degree of accuracy. This is due to deficient implementation of cross-validation algorithms when dealing with badly represented factors and small *n*.

data.models <- cbind(SW[,-c(1, 2)], logSize, pcdata[,1:8])  
  
lmtFit1 <- LMT(burnt ~ PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8, data = data.models)  
summary(lmtFit1)

##   
## === Summary ===  
##   
## Correctly Classified Instances 35 92.1053 %  
## Incorrectly Classified Instances 3 7.8947 %  
## Kappa statistic 0.8421  
## Mean absolute error 0.2245  
## Root mean squared error 0.297   
## Relative absolute error 44.8947 %  
## Root relative squared error 59.4075 %  
## Coverage of cases (0.95 level) 100 %  
## Mean rel. region size (0.95 level) 97.3684 %  
## Total Number of Instances 38   
##   
## === Confusion Matrix ===  
##   
## a b <-- classified as  
## 18 1 | a = no  
## 2 17 | b = yes

The model lmtFit1 uses Logistic Model Trees to predict if a humerus was burnt or not with 92.1% accuracy, by being trained on the first 8 Principal Components of our sample. So how should nonmorphometricians interested in burnt remains proceed to apply this model to a humerus from their own sample? By following the whole protocol: (1) 3D Scanning a humerus; (2) Processing it in MeshLab; (3) Obtain our 35 Landmarks for that bone; (4) Perform an OPA on the obtained configuration by using the mean shape of our sample as the reference shape; (5) perform a PCA on the configuration, (6) apply the model by using the R function predict() on his first 8 PCs. Alternatively, step 1, 2, and 3 could be substituted by obtaining landmarks directly from a MicroScribe or a similar device. However, in most cases, for the trained osteologist it is easy to understand if a bone was burnt or not by just looking at it. So why would a model like this be useful? It wouldn’t, it is too time consuming. A far more useful model would be a similar one that could not only determine if a bone was burnt but at which temperature it was burnt. Next, we implement such model in a similar fashion.

lmtFit2 <- LMT(as.factor(temperature) ~ PC1+PC2+PC3+PC4+PC5+PC6+PC7+PC8, data = data.models)  
summary(lmtFit2)

##   
## === Summary ===  
##   
## Correctly Classified Instances 32 84.2105 %  
## Incorrectly Classified Instances 6 15.7895 %  
## Kappa statistic 0.7415  
## Mean absolute error 0.1091  
## Root mean squared error 0.2116  
## Relative absolute error 54.8523 %  
## Root relative squared error 68.3659 %  
## Coverage of cases (0.95 level) 100 %  
## Mean rel. region size (0.95 level) 56.391 %  
## Total Number of Instances 38   
##   
## === Confusion Matrix ===  
##   
## a b c d e f g <-- classified as  
## 19 0 0 0 0 0 0 | a = 0  
## 0 1 0 0 0 0 0 | b = 700  
## 4 0 2 0 0 0 0 | c = 800  
## 0 0 0 1 0 0 0 | d = 850  
## 1 0 0 0 8 0 0 | e = 900  
## 1 0 0 0 0 0 0 | f = 1000  
## 0 0 0 0 0 0 1 | g = 1050

Despite the lack of values for each factor, which of course causes overfitting, and the low diversity of the factors themselves, which obligates us to use a classification algorithm for temperatures instead of a regression, we’ve obtained what seems to be very promising results (Overall Accuracy = 84.21%). Here is shown a stepping-stone from where one could eventually create a very powerful model by increase n in each of the factors of variables in order to reduce overfitting while increasing overall accuracy.

## Procrustes Analysis of Variance

A Procrustes ANOVA is used to quantify the relative amount of shape variation attributable to one or more variables in a linear model and assesses this variation through permutation. In geomorph the function procD.lm() allow us to input data by a `y~X formula, where 'y' specifies the response variables (shape data), and 'X' contains one or more independent variables (Adams et al., 2015). procD.lm() follows the philosophy that:

* Randomization procedures are used to generate empirical sampling distributions to assess significance of effects.
* Effect sizes are estimated as standard deviates from such sampling distributions.

The function performs statistical assessment of the terms in the model using Procrustes distances among specimens, instead of explained covariance matrices among variables. With this approach, the sum-of-squared Procrustes distances are used as a measure of SS. Permutation is used to evaluate observed SS (Adams & Otárola-Castillo, 2013).

# The residual SS (RSS) of a linear model (also called the sum of squared error, SSE) is found as follows:  
  
RSS <- function(fit) sum(diag(resid(fit)%\*%t(resid(fit))))   
  
Sex <- SW$sex  
  
fit1 <- lm(y ~ 1) # model contaning just an intercept  
fit2 <- lm(y ~ logSize) # allometric scaling of x  
fit3 <- lm(y ~ logSize + Sex) # previous model + sexual dimorphism  
fit4 <- lm(y ~ logSize \* Sex) # previous model + interaction between sex and log(CS)  
  
# For any model 'fit', we can summarize the error of prediction by calculating RSS.  
RSS(fit1)

## [1] 0.06791796

RSS(fit2)

## [1] 0.06537494

RSS(fit3)

## [1] 0.06288619

RSS(fit4)

## [1] 0.05736431

kable(procD.lm(Shapes ~ logSize\*Sex, RRPP = TRUE), digits = 4)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | df | SS | MS | Rsq | F | Z | P.value |
| logSize | 1 | 0.0025 | 0.0025 | 0.0374 | 1.5073 | 1.2721 | 0.168 |
| Sex | 1 | 0.0025 | 0.0025 | 0.0366 | 1.4751 | 1.2891 | 0.155 |
| logSize:Sex | 1 | 0.0055 | 0.0055 | 0.0813 | 3.2728 | 2.9887 | 0.002 |
| Residuals | 34 | 0.0574 | 0.0017 | NA | NA | NA | NA |
| Total | 37 | 0.0679 | NA | NA | NA | NA | NA |

In the literature the model we just created as been described as a Procrustes ANOVA (Goodall, 1991), which is analogous to the popular distance-based ANOVA designs (Anderson, 2001). In procD.lm() two resampling procedures are possible: (1) if RRPP=FALSE, the rows of the matrix of shape variables are resampled in relation to the design matrix; (2) if RRPP=TRUE, a residual randomization permutation procedure is utilized (Collyer et al., 2014). While identical for single-factor designs, when evaluating factorial models it has been shown that RRPP attains higher statistical power and thus is better at pattern-recognition (Anderson & Braak, 2003). Our objective here is to show the ability of Procrustes ANOVA to compare models. As it is shown in Table 2 the model fit4 that corresponds to size and sexual dimorphism interacting, outperforms the other simpler models. This same procedure could be repeated for many other variables and model designs, however having in consideration our unbalanced factors in most of the other relevant variables available it would be quite absurd to expect reliable error estimates (RSS) of most possible models.