EDMAinR: basic usage

2019-11-13

Introduction

EDMAinR is an R package for Euclidean Distance Matrix Analysis (EDMA). EDMA is a coordinate-free approach for comparing biological shapes using landmark data as described in Lele and Richtsmeier (1991). The implementation follows Hu (2007).

Install

The package can be installed from GitHub:

```
if (!require(EDMAinR)) {
  if (!require(remotes))
      install.packages("remotes")
  remotes::install_github("psolymos/EDMAinR")
}
```

We can now load the package:

```
library(EDMAinR)
```

```
## EDMAinR 0.0-2     2019-11-07
```

Landmark data

Use the `read_xyz` function to read 2 or 3 D landmark data from `*.xyz` files. First we specify the paths to two xyz files:

```
file1 <- system.file("extdata/crouzon/Crouzon_P0_Global_MUT.xyz",
    package="EDMAinR")

file2 <- system.file("extdata/crouzon/Crouzon_P0_Global_NON-MUT.xyz",
    package="EDMAinR")
```

Note: we use the `system.file()` function to access example files from the package. When defining your own files, you will either set the working directory using `setwd()` or a path like `c:/Users/<user>/<etc>`.

Now we can read in these text files:

```
x1 <- read_xyz(file1)
x1
```

```
## EDMA data: Crouzon P0 MUT
## 3 dimensions, 47 landmarks, 28 replicates
```

```
x2 <- read_xyz(file2)
x2
```

```
## EDMA data: Crouzon P0 UNAFF
## 3 dimensions, 47 landmarks, 31 replicates
```

The data objects are lists with 3 elements:

- `$name` contains info about the file from its header
- `$data` contains the landmark data
- `$notes` contains optional information about the individuals

```
x1$notes
```

```
##  [1] "CZCD1_1 Scanned on 110309"  "CZCD1_2 Scanned on 110309"
##  [3] "CZCD1_7 Scanned on 110309"  "CZCD1_10 Scanned on 110309"
##  [5] "CZCD1_11 Scanned on 110309" "CZCD1_15 Scanned on 092710"
##  [7] "CZCD1_16 Scanned on 092710" "CZCD1_18 Scanned on 091910"
##  [9] "CZCD1_20 Scanned on 091910" "CZCD1_21 Scanned on 092010"
## [11] "CZCD1_23 Scanned on 091710" "CZCD1_24 Scanned on 092010"
## [13] "CZCD1_25 Scanned on 092010" "CZCD1_28 Scanned on 092710"
## [15] "CZCD1_30 Scanned on 092810" "CZCD1_32 Scanned on 092810"
## [17] "CZCD1_36 Scanned on 010511" "CZCD1_37 Scanned on 010511"
## [19] "CZCD1_39 Scanned on 010511" "CZCD1_40 Scanned on 010511"
## [21] "CZCD1_42 Scanned on 010511" "CZCD1_53 Scanned on 122310"
## [23] "CZCD1_56 Scanned on 122310" "CZCD1_59 Scanned on 122310"
## [25] "CZCD1_65 Scanned on 010511" "CZCD1_66 Scanned on 010511"
## [27] "CZCD1_72 Scanned on 010311" "CZCD1_73 Scanned on 010311"
```

Here are the methods that we can use to learn more about the data sets.

Access dimensions (landmarks, K; dimensions, D; replicates, n) and dimension names (`landmark_names` returns the landmark labels):

```
dim(x1)
```

```
## [1] 47  3 28
```

```
dimnames(x1)
```

```
## [[1]]
##  [1] "amsph" "bas"   "cpsh"  "ethma" "ethmp" "laalf" "lasph" "lflac"
##  [9] "lnsla" "lnslp" "locc"  "loci"  "lpalf" "lpfl"  "lpmx"  "lpns"
## [17] "lpsh"  "lpsq"  "lpto"  "lptyp" "lsqu"  "lsyn"  "lzya"  "lzygo"
## [25] "lzyt"  "opi"   "raalf" "rasph" "rflac" "rmaxi" "rnsla" "rnslp"
## [33] "rocc"  "roci"  "rpalf" "rpfl"  "rpmx"  "rpns"  "rpsh"  "rpsq"
## [41] "rpto"  "rptyp" "rsqu"  "rsyn"  "rzya"  "rzygo" "rzyt"
##
## [[2]]
## [1] "X" "Y" "Z"
```

```
landmark_names(x1)
```

```
##  [1] "amsph" "bas"   "cpsh"  "ethma" "ethmp" "laalf" "lasph" "lflac"
##  [9] "lnsla" "lnslp" "locc"  "loci"  "lpalf" "lpfl"  "lpmx"  "lpns"
## [17] "lpsh"  "lpsq"  "lpto"  "lptyp" "lsqu"  "lsyn"  "lzya"  "lzygo"
## [25] "lzyt"  "opi"   "raalf" "rasph" "rflac" "rmaxi" "rnsla" "rnslp"
## [33] "rocc"  "roci"  "rpalf" "rpfl"  "rpmx"  "rpns"  "rpsh"  "rpsq"
## [41] "rpto"  "rptyp" "rsqu"  "rsyn"  "rzya"  "rzygo" "rzyt"
```

Subsetting the data comes handy sometimes. The most general way to subset the data sets is via the `[` function, the 3 indices inside the brackets refer to the landmarks, dimensions, and replicates (most often individuals). The `subset` method subsets the replicates:

```
x1[1:10, , ]  # select the 1st 10 landmarks
```

```
## EDMA data: Crouzon P0 MUT
## 3 dimensions, 10 landmarks, 28 replicates
```

```
x1[ , 1:2, ]  # select 2 of the 2 dimensions
```

```
## EDMA data: Crouzon P0 MUT
## 2 dimensions, 47 landmarks, 28 replicates
```

```
x1[ , , 1:20] # select the 1st 20 individuals
```

```
## EDMA data: Crouzon P0 MUT
## 3 dimensions, 47 landmarks, 20 replicates
```

```
x1[1:10, , 1:20] # combine multiple indices
```

```
## EDMA data: Crouzon P0 MUT
## 3 dimensions, 10 landmarks, 20 replicates
```

The data (`$data`) format inside the objects `x1` and `x2` is list of the $K \times D$ matrices for each individual. Sometimes it is handy to stack these matrices and create a rectangular data (either as a matrix, or data frame, with $n \times K$ rows and D columns):

```
str(as.matrix(x1))
```

```
##  num [1:1316, 1:3] 5.85 2.79 6.86 9.11 8.25 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:1316] "rep1_amsph" "rep1_bas" "rep1_cpsh" "rep1_ethma" ...
##   ..$ : chr [1:3] "X" "Y" "Z"
```

```
str(as.data.frame(x1))
```

```
##  num [1:1316, 1:3] 5.85 2.79 6.86 9.11 8.25 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:1316] "rep1_amsph" "rep1_bas" "rep1_cpsh" "rep1_ethma" ...
##   ..$ : chr [1:3] "X" "Y" "Z"
```

```
str(stack(x1))
```

```
##  num [1:1316, 1:3] 5.85 2.79 6.86 9.11 8.25 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:1316] "rep1_amsph" "rep1_bas" "rep1_cpsh" "rep1_ethma" ...
##   ..$ : chr [1:3] "X" "Y" "Z"
```

Note: we are using `str` to show the structure of these objects, this is not necessary when exploring the data.

Alternatively, we can store the data as an array ($K \times D \times n$):

```
str(as.array(x1))
```

```
##  num [1:47, 1:3, 1:28] 5.85 2.79 6.86 9.11 8.25 ...
##  - attr(*, "dimnames")=List of 3
##   ..$ : chr [1:47] "amsph" "bas" "cpsh" "ethma" ...
##   ..$ : chr [1:3] "X" "Y" "Z"
##   ..$ : NULL
```

Nonparametric estimation

The nonparametric estimator gives the mean form matrix ($\hat{M}$) and $\hat{\Sigma}^*_K$, that we can extract from the fitted model object `fit` using the `Meanform` and `SigmaKstar` functions (using only the first 5 landmarks here):

```
fit <- edma_fit(x1[1:5,,])
fit
```

```
## EDMA nonparametric fit: Crouzon P0 MUT
## 3 dimensions, 5 landmarks, 28 replicates, no bootstrap
```

```
Meanform(fit)
```

```
##                X           Y             Z
## amsph -0.8574818  0.13310052  0.0086353361
## bas   -4.4348102  0.05026895 -0.0044123572
## cpsh   0.5944860 -0.38325079  0.0022689123
## ethma  2.7233543  0.22732694 -0.0009270024
## ethmp  1.9744516 -0.02744562 -0.0055648888
```

```
SigmaKstar(fit)
```

```
##               amsph          bas          cpsh         ethma         ethmp
## amsph  0.0014232224  0.001795735 -0.0004605613 -0.0025008713 -0.0002575251
## bas    0.0017957354  0.012181291 -0.0034234865 -0.0057199529 -0.0048335867
## cpsh  -0.0004605613 -0.003423487  0.0019053156  0.0008965274  0.0010822048
## ethma -0.0025008713 -0.005719953  0.0008965274  0.0076288128 -0.0003045160
## ethmp -0.0002575251 -0.004833587  0.0010822048 -0.0003045160  0.0043134231
```

We can extract the mean form as pairwise Euclidean distances (object class *dist* as it is customary in R, see `?dist` for the details). This is the form matrix in distance matrix format:

```
as.dist(fit)
```

```
##           amsph       bas      cpsh     ethma
## bas   3.5783110
## cpsh  1.5410613 5.0479505
## ethma 3.5820884 7.1603548 2.2146999
## ethmp 2.8365161 6.4097331 1.4251188 0.7910663
```

The Euclidean distances from the form matrix can be stacked, and the stacked distances sorted. The `get_fm` function produces the stacked for matrix, because this is most useful to us as it can be sorted and more easily inspected:

```
get_fm(fit)
```

```
##      row   col      dist
## 1    bas amsph 3.5783110
## 2   cpsh amsph 1.5410613
## 3  ethma amsph 3.5820884
## 4  ethmp amsph 2.8365161
## 5   cpsh   bas 5.0479505
## 6  ethma   bas 7.1603548
## 7  ethmp   bas 6.4097331
## 8  ethma  cpsh 2.2146999
## 9  ethmp  cpsh 1.4251188
## 10 ethmp ethma 0.7910663
```

```
get_fm(fit, sort=TRUE, decreasing=TRUE)
```

```
##      row   col      dist
## 6  ethma   bas 7.1603548
## 7  ethmp   bas 6.4097331
## 5   cpsh   bas 5.0479505
## 3  ethma amsph 3.5820884
## 1    bas amsph 3.5783110
## 4  ethmp amsph 2.8365161
```

```
## 8  ethma  cpsh 2.2146999
## 2   cpsh amsph 1.5410613
## 9  ethmp  cpsh 1.4251188
## 10 ethmp ethma 0.7910663
```

```
get_fm(fit, sort=TRUE, decreasing=FALSE)
```

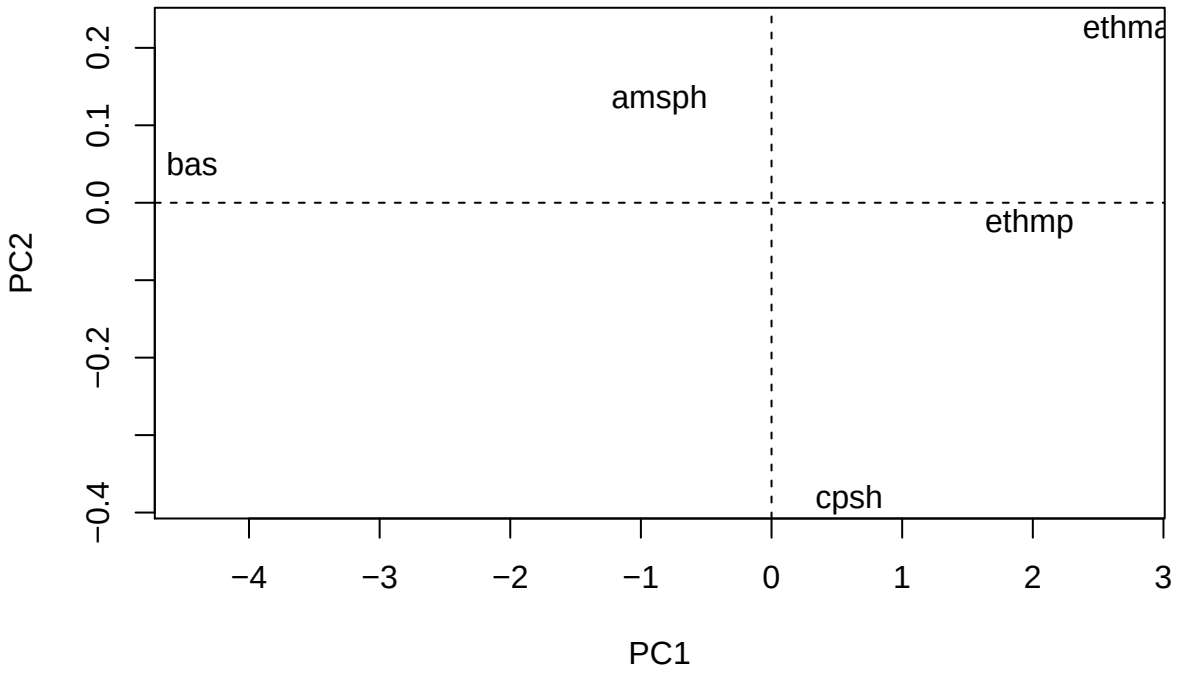
```
##      row   col      dist
## 10 ethmp ethma 0.7910663
## 9  ethmp  cpsh 1.4251188
## 2   cpsh amsph 1.5410613
## 8  ethma  cpsh 2.2146999
## 4  ethmp amsph 2.8365161
## 1    bas amsph 3.5783110
## 3  ethma amsph 3.5820884
## 5   cpsh   bas 5.0479505
## 7  ethmp   bas 6.4097331
## 6  ethma   bas 7.1603548
```

We can turn the Euclidean distance into principal components using the `get_pca` function:

```
pc <- get_pca(fit)
pc
```

```
##              PC1         PC2
## amsph -0.8574818  0.13310052
## bas   -4.4348102  0.05026895
## cpsh   0.5944860 -0.38325079
## ethma  2.7233543  0.22732694
## ethmp  1.9744516 -0.02744562
## attr(,"class")
## [1] "edma_pca" "matrix"
```

```
plot(pc, type="n")
text(pc, labels=rownames(pc))
abline(h=0, v=0, lty=2)
```



Comparing 2 sets of 3D landmark data

We can fit the EDMA object with bootstrap resampling by specifying the `B` argument, this is needed for downstream statistical testing:

```
B <- 99
numerator <- edma_fit(x1[1:25,,], B=B)
denominator <- edma_fit(x2[1:25,,], B=B)
```

The form difference matrix is defined as the ratio of the two Euclidean distances form matrices based on the mean forms from the numerator and denominator model objects. This is in the matrix (`dist`) format:

```
fd <- formdiff(numerator, denominator)
str(fd)
```

```
##  'dist' num [1:300] 1.043 0.983 1.004 0.94 0.988 ...
##  - attr(*, "Size")= int 25
##  - attr(*, "Labels")= chr [1:25] "amsph" "bas" "cpsh" "ethma" ...
##  - attr(*, "Diag")= logi FALSE
##  - attr(*, "Upper")= logi FALSE
##  - attr(*, "method")= chr "euclidean_distance_ratio"
##  - attr(*, "Tval")= num 1.6
```

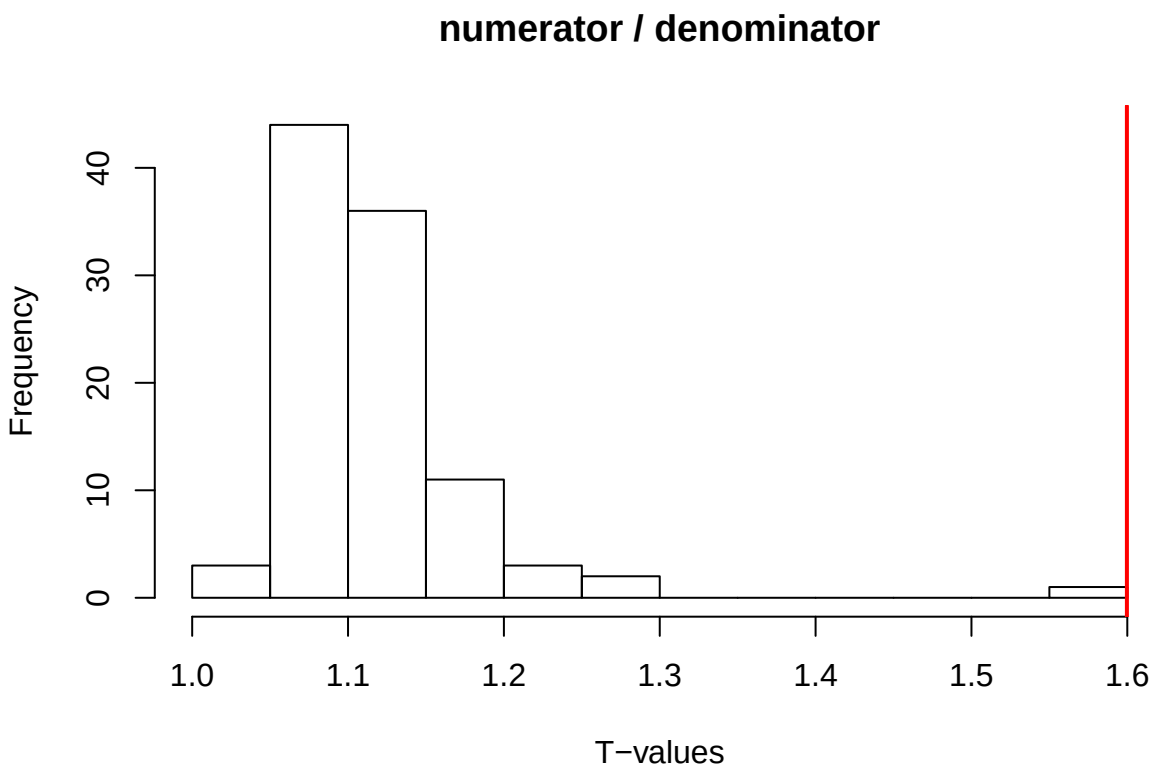
The global T-test assesses if the form difference is significant. It uses a bootstrap distribution that is a mixture of the two samples, thus the bootstrap leads to a 'null' distribution and the observed T values is compared against this set of bootstrap based values:

```
fdm <- edma_fdm(numerator, denominator, B=B)
fdm
```

```
## EDMA form difference matrix
## 99 bootstrap runs, T=1.5997, p=0.01
```

The plot compares the bootstrap distribution (histogram) to the observed value (red line):

```
plot(fdm, type="global")
```



The local test can be achieved in 2 ways. One way is to use the 'mixed' bootstrap distribution from the global test and calculate the probability that a random pairwise distance ratio is lower, higher (one sided p-value), or lower or higher (two sided p-value) than the observed distance ratio. This p-value can be accessed as part of the stacked form difference matrix via the `get_fdm` function:

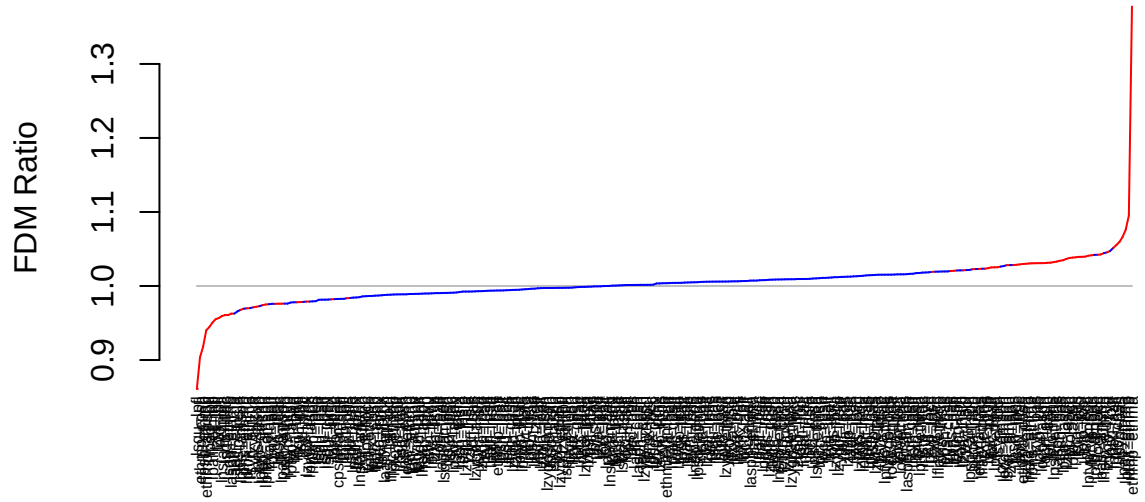
```
head(get_fdm(fdm, sort=TRUE))
```

```
##       row   col      dist pval
## 241  lsqu  lpfl 0.8610910 0.03
## 49  ethmp  cpsh 0.9035048 0.03
## 102  lpsh ethmp 0.9179315 0.03
## 4   ethmp amsph 0.9400968 0.03
## 238  lpsq  lpfl 0.9445835 0.03
## 98  lpalf ethmp 0.9505329 0.03
```

```
tail(get_fdm(fdm, sort=TRUE))
```

```
##       row   col     dist pval
## 60   lpns  cpsh 1.055348 0.03
## 44   lsyn   bas 1.059504 0.03
## 78  lpalf ethma 1.066190 0.03
## 100  lpmx ethmp 1.076121 0.03
## 91  laalf ethmp 1.095120 0.03
## 70  ethmp ethma 1.377518 0.03
```

```
plot(fdm, type="local_p")
```

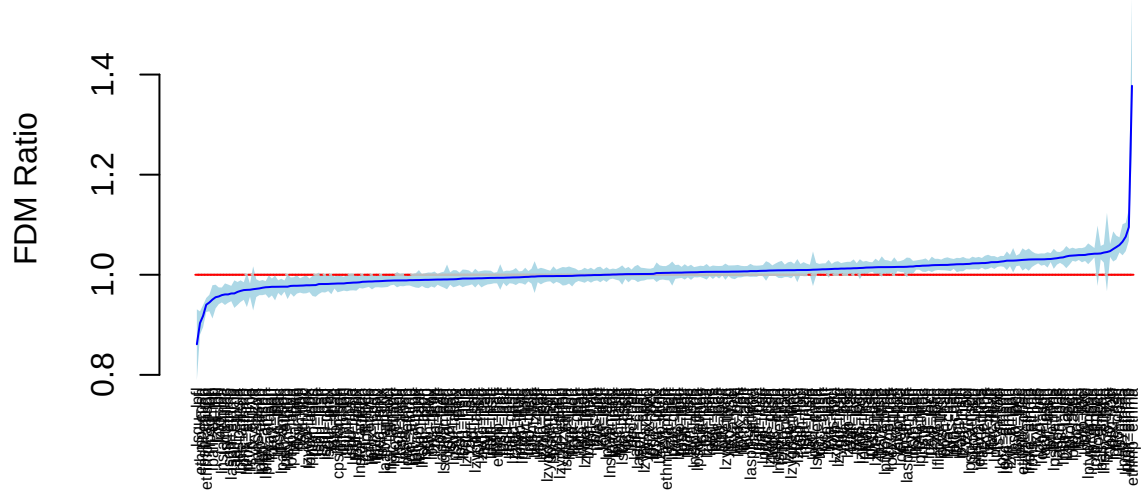


The other way is to use the bootstrap distribution from the individual `edma_fit` objects (this represents uncertainty around the mean form). This provides the confidence intervals around the distance ratios. This local test is significant when the interval does not overlap 1 (no form difference). The coverage of the confidence interval depends on the `level` argument that is set to 95% by default:

```
head(confint(fdm))
```

```
##        2.5%     97.5%
## 1 1.0308493 1.0576396
## 2 0.9649677 0.9989395
## 3 0.9892348 1.0173246
## 4 0.9254901 0.9535495
## 5 0.9749716 0.9999477
## 6 0.9952918 1.0343820
```

```
plot(fdm, type="local_ci")
```



References

Lele, S. R., and Richtsmeier, J. T., 1991. Euclidean distance matrix analysis: A coordinate-free approach for comparing biological shapes using landmark data. American Journal of Physical Anthropology 86:415–27. DOI: 10.1002/ajpa.1330860307.

Lele, S. R., and Richtsmeier, J. T., 1995. Euclidean Distance Matrix Analysis: Confidence Intervals for Form and Growth Differences. American Journal of Physical Anthropology 98:73–86.

Hu, L., 2007. Euclidean Distance Matrix Analysis of Landmarks Data: Estimation of Variance. Thesis, Master of Science in Statistics, Department of Mathematical and Statistical Sciences, University of Alberta, Edmonton, Alberta, Canada. Pp. 49.