TP/TD 1/2 Morphométrie[1] M2 AGE

Introduction

Ce TP a pour but de se familiariser avec le traitement des points homologues (landmarks) en R. Le jeu des données consiste de 5 statères (monnaies grecques), prise 2 fois en photo (images sont dans le dossier 'Statere'). La première partie du TD a pour but d'explique comment faire l'acquisition des points homologues, la deuxième partie est consacré à leur traitement par « L'analyse de Procruste (GPA) ».

	a	b
LeRider_022		
LeRider_026		
LeRider_083		
LeRider_091		
LeRider_092		

[1] La plupart du code de ce TP a été créé par Sébastien Couette et Nicolas Navarro. Les corrections et le texte ont été ajoutés par Josef Wilczek. Dernière modification : 2016-10-25.

Acquisition des points homologues

A. Possibilités

Depuis les années 90, plusieurs logiciels de morphométrie géométrique ont été créés :

- IMP (?) – plus mis à jour
- TPS (F. Rohlf) – sur Windows ; Contient un software pour l'acquisition de pts (TPSdig2)
- NTSYS (extersoftware) – logiciel payant,
- Shape (Hiroyoshi Iwata) – pour les contours fermes (EFA)
- MorphoJ (Chris Klingenberg) - cross-platforms (Java)
- l'EVAN Toolbox (Evan consortium) – toutes les plateformes

En R, plusieurs packages pour traiter les informations morphométriques sont disponibles :

- shapes (I. Dryden)
- Rmorph (M. Baylac) - non dispo sur le site du CRAN
- geomorph (D. Adams) - le plus récent
- Morhpo (Schlatter?) - le plus récent (3D)

B. Initialisation

Avant de commencer, définissons notre répertoire de travail 'Seance 3', c.à.d. le dossier où se trouvent les fichiers dont on aura besoin

```
setwd("C:/Users/Josef/Desktop/Seance 3/")
```

Nous allons travailler avec le package 'geomorph' car il permet l'acquisition des points homologues. Sinon il faudrait soit acquérir les données avec TPSdig2, soit avec par exemple ImageJ puis importer les données sous R. Mais travaillons seulement en R en utilisant le package 'geomorph'

```
install.packages("geomorph")
library(geomorph)
```

En plus, nous auront besoin de quelques fonctions complémentaires. Elles sont stockées dans le fichier 'archeoMorpho.R'. On peut les charger aussi.

```
source("archeoMorpho.R")
```

C. Acquisition des points homologues

Avant de commencer la digitalisation, il nous faut définir le nombre de points homologues avec lesquels nous voulons travailler.

On travaille avec des images qui sont en 2D, donc chaque point aura 2 coordonnées (x et y).

```
n.dim <- 2
```

Sur les statères (monnaie grecque), on va travailler avec 16 points :

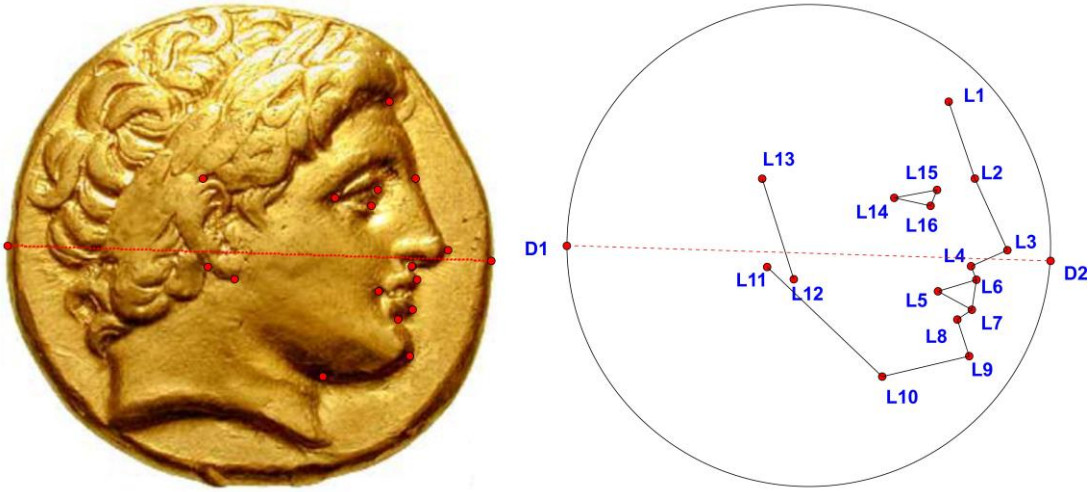
```
n.land <- 16
```

Des points homologues peuvent être digitalisés par la fonction 'digitize2d'. Faisons donc un test sur des images 'aLeRider_091@16.jpg' et 'aLeRider_091@16_b.jpg', en sauvegardant les résultats dans le fichier 'test.tps'.

```
digitize2d(c("aLeRider_091@16.jpg","aLeRider_091@16_b.jpg"), n.land, 1, "test.tps")
```

Note : si une erreur se produit ('Error in digstart:length(filelist) : result would be too long a vector'), il faut d'abord effacer le fichier 'test.tps' du dossier 'Seance 3'.

Les 16 points homologues (landmarks) sont définis par de simple « cliques gauches » de la souris. Avant de commencer la digitalisation des points, il faut néanmoins définir l'échelle. Comme les images ne contiennent pas de réelles échelles, on va définir l'échelle à l'aide du diamètre des Statères.
Cliquer donc sur deux points, qui définissent le diamètre de la monnaie (D1 et D2 sur l'image) et ensuite, vectorisez les 16 landmarks d'après le schéma.

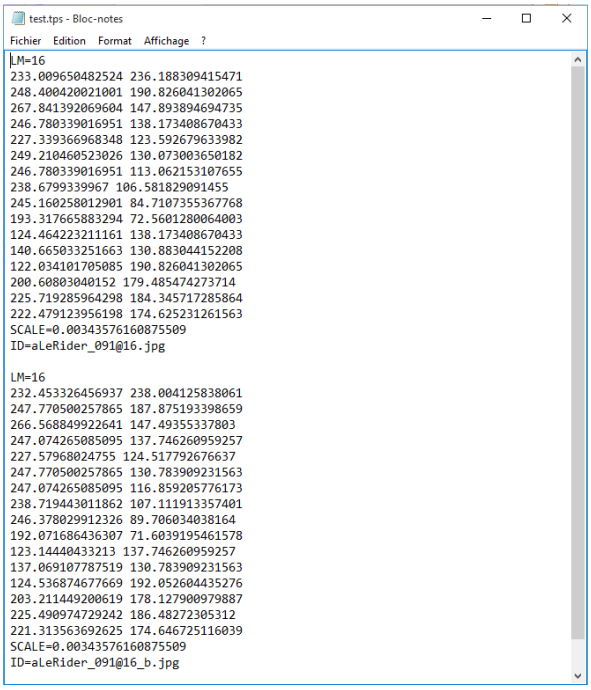


Notez, que la fonction vous guide. Elle va toujours demander si votre digitalisation a été correcte :
`Keep scale (y/n)?`
`Keep Landmark  1 (y/n/a)?`
Il faut toujours répondre. En acceptant, vous poursuivez au point suivant, en rejetant, vous pouvez le corriger. A la fin, la fonction va vous poser la question, si vous voulez procéder de même avec le deuxième individu.
`Continue to next specimen (y/n)?`
Acceptez, et faites la digitalisation du deuxième spécimen.

Une fois fini, vous pouvez ouvrir (en 'Bloc-notes') le fichier 'test.tps' qui a été créé et regarder sa structure :

LM donne le nombre de landmarks (donc 16)
Les lignes suivantes représentent les coordonnées x, y des 16 landmarks
SCALE correspondant à l'échelle
ID correspond à l'identifiant c.-à-d. le nom du fichier que vous avez digitalisé.

Vous avez ensuite les mêmes informations pour le deuxième individu.


test.tps - Bloc-notes
Fichier Edition Format Affichage ?

```
LM=16
233.009650482524 236.188309415471
248.400420021001 190.826041302065
267.841392069604 147.893894694735
246.780339016951 138.173408670433
227.339366968348 123.592679633982
249.210460523026 130.073003650182
246.780339016951 113.062153107655
238.6799339967 106.581829091455
245.160258012901 84.7107355367768
193.317665883294 72.5601280064003
124.464223211161 138.173408670433
140.665033251663 130.883044152208
122.034101705085 190.826041302065
200.60803040152 179.485474273714
225.719285964298 184.345717285864
222.479123956198 174.625231261563
SCALE=0.00343576160875509
ID=aLeRider_091@16.jpg

LM=16
232.453326456937 238.004125838061
247.770500257865 187.875193398659
266.568849922641 147.49355337803
247.074265085095 137.746260959257
227.57968024755 124.517792676637
247.770500257865 130.783909231563
247.074265085095 116.859205776173
238.719443011862 107.111913357401
246.378029912326 89.706034038164
192.071686436307 71.6039195461578
123.14440433213 137.746260959257
137.069107787519 130.783909231563
124.536874677669 192.052604435276
203.211449200619 178.127900979887
225.490974729242 186.48272305312
221.313563692625 174.646725116039
SCALE=0.00343576160875509
ID=aLeRider_091@16_b.jpg
```

Qualité de digitalisation (comparaison des deux sessions) et différences entre les monnaies

Maintenant nous allons faire une étude plus sérieuse. Avant de commencer une étude morphométrique, il faut s'assurer que les landmarks ont été précisément localisés sur les images. Dans l'exemple suivant on va travailler avec 5 statères et on va estimer la qualité de notre digitalisation. Pour le faire, on va placer les landmarks deux fois sur un individu. Ensuite :

(i) on va visuellement regarder les différences entre ces deux sessions

(ii) on va tester statistiquement si ces différences sont significatives

(iii) on va calculer et visualiser la précision de localisation de chaque des landmarks. Ca nous permettra de voir quelles des points étaient les points « difficilement localisables »

(iv) on va tester s'il y a des différences entre les monnaies

(v) et on va visualiser ses différences

Pour voir avec quelles images (statères) nous allons travailler, regarder dans le dossier 'Statère'

1. Digitalisation

Avant de commencer, nous allons définir le dossier contenant les images :

```
setwd("Statere")
```

Maintenant nous allons faire une boucle pour traiter toutes les images dans le même temps. Listons tous les fichiers dans le dossier Statere (`'rmq: getwd()'` donne l'adresse du dossier), et stockons le nom de toutes ces images 'jpg' du dossier Statere dans l'objet `'img'`

```
img <- list.files(path=getwd(), pattern='.jpg')
print(img)
```

```
[1] "aLeRider_022.jpg" "aLeRider_026.jpg" "aLeRider_083.jpg" "aLeRider_091.jpg" "aLeRider_092.jpg"
[6] "bLeRider_022.jpg" "bLeRider_026.jpg" "bLeRider_083.jpg" "bLeRider_091.jpg" "bLeRider_092.jpg"
```

Il y a donc 10 images de 5 statères pris en photo deux fois. Donnons ces informations dans trois vecteurs…

```
n.img <- 10
n.session <- 2
n.ind <- n.img/n.session
```

…et commençons la prise de données

```
digitize2d(img,n.land,1, "statere.tps")
```

La fonction a créé le fichier 'statere.tps'. Pour vous en assurer, ouvrez-le en 'Bloc-notes' et vérifiez que toutes les coordonnées ont bien été stockées. Le fichier doit contenir 10 paragraphes (pour 10 individus), et aucune valeur ne doit être 'NA' ou 'NaN'. Si un problème arrive, essayer de le corriger.

2. Charger tps

Si tout est bon, on peut lire le fichier 'tps' qu'on vient du créer en R en appelant la fonction `'readland.tps'`

```
xy.pts <- readland.tps("statere.tps", specID = "ID", warnmsg = T)
```

Ensuite créons un facteur `'ind'` et un facteur `'session'`

Juste pour rappel : un facteur est une variable catégorielle qui nous définit des groupes. Par exemple si on compare la taille des hommes et de femmes, le facteur (ou 'grouping factor' ou 'grouping vector')

est ‘genre’. Les ‘niveaux’ ('levels') d’un facteur sont ses valeurs (‘modes’). Par exemple le facteur ‘genre’ a deux ‘niveaux’ – ‘Homme’ et ‘Femme’.
Dans notre cas, le facteur est 'session' et il possède deux niveaux (‘1’ pour première et ‘2’ pour deuxième digitalisation d’un individu) :

```
session <- gl(n.session,n.ind)
```

Le facteur `'ind'` va ensuite contenir les noms des fichiers :

```
ind <- as.factor(rep(1:n.ind,n.session))
```

On peut créer des facteurs en utilisant la fonction `'substring'` appliquée sur les noms d'images. Pour savoir comment cette fonction marche, essayez d'envoyer la ligne suivante

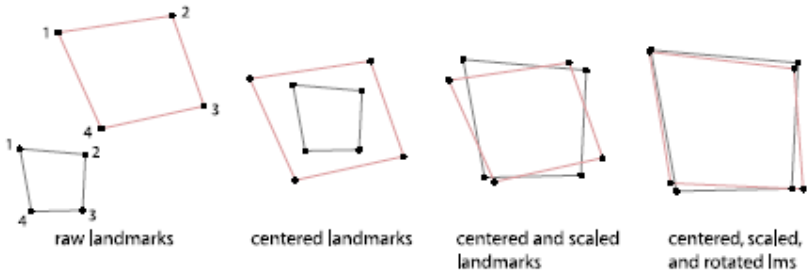
```
substring("Abrakadabrakadabra", 3, 6)
[1] "raka"
```

elle nous a donné "raka" – c’est-à-dire de la troisième à la sixième lettre du mot "Abrakadabrakadabra"

```
ind <- as.factor(substr(img,2,12))
session <- as.factor(substr(img,1,1))
```

3. Performer Superposition Procruste (GPA: Ajustement Procruste généralisé)
L’analyse de Procruste (GPA) a pour but de standardiser la position, la taille et la rotation des objets.

Fig. 3 The three steps of Procrustes superimposition: translation to a common origin, scaling to unit centroid size, and rotation to minimize the sum of squared Euclidean distances among the homologous landmarks. The resulting landmark coordinates are called Procrustes shape coordinates



En supprimant ces trois effets (position, taille et rotation), on observe que les différences de formes qu’il y a entre les objets. Ces différences sont numériquement exprimées par les « distances de Procruste » (Procrustes distances).
En réalité, en faisant le GPA, on perd 4 dégrées de liberté. C'est parce que quand on fait le GPA:

1) on standardise la position - des landmarks originales, on enlève **deux** coordonnées du centroïde (donc on perd 2ddl)
2) on standardise la taille - on divise les landmarks-centrés par **une** taille du centroid (centroid-size) (donc on perd 1ddl)
3) on standardise la rotation - on fait la rotation des landmarks-centrées-réduits par **un** angle phi (donc on perd 1ddl)

L’Analyse de GPA en R peut être fait grâce à la fonction `'gpagen'`

```
gpa <- gpagen(xy.pts)
```

on peut regarder ce qu'elle nous fournit en utilisant la fonction `'str'` qui nous retourne la ‘structure’ d'un objet en R

```
str(gpa)
```

on voit que `'gpa'` contient deux 'lists' - `$coords` et `$Csize`

```
gpa$coords
```

…nous donne un ‘array’ qui contient les landmarks standardisés par GPA pour chaque individu

```
gpa$cSize
```

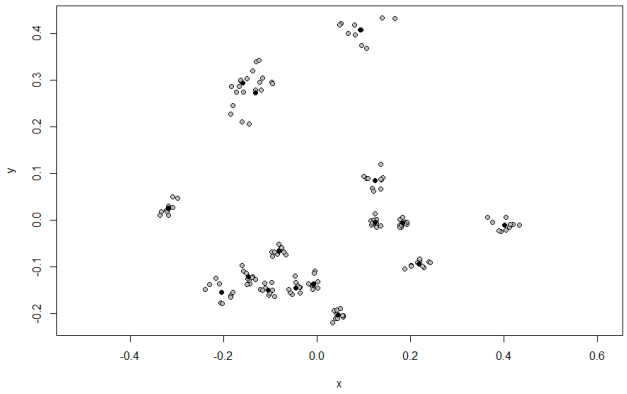
…nous donne l'information sur 'le centroid size' pour chaque individu. Centroid size est la somme des distances entre le centroïde et chacun des landmarks

3.1. Visualisation des résultats

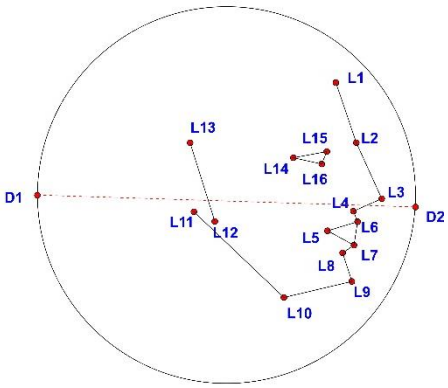
Pour visualiser les résultats, on peut appeler la fonction `'plotAllSpecimens'`

```
plotAllSpecimens(gpa$coords, plot.param=list(pt.cex=1, mean.cex=1))
```

On peut voir que les points en gris correspondent aux landmarks tandis que les points noirs à la 'meanshape'. L'image peut néanmoins paraitre un peu illisible - pour voir les visages qu'on a défini, il faut avoir beaucoup d'imagination... Pour mieux visualiser les visages, on pourra ajouter les lignes qui font les liaisons entre certains landmarks. Pour le faire, on va donc créer une matrice `'myLinks'` telle que dans chaque ligne, on va définir quelle Landmark est liées avec l'autre.



Sur l'image, on voit par exemple que: le 1er landmark est lié au 2eme, le 2em au 3eme, le 3eme au 4eme, le 5eme au 6eme, etc.



On va donc donner toutes ces liaisons dans la matrice `'myLinks'`

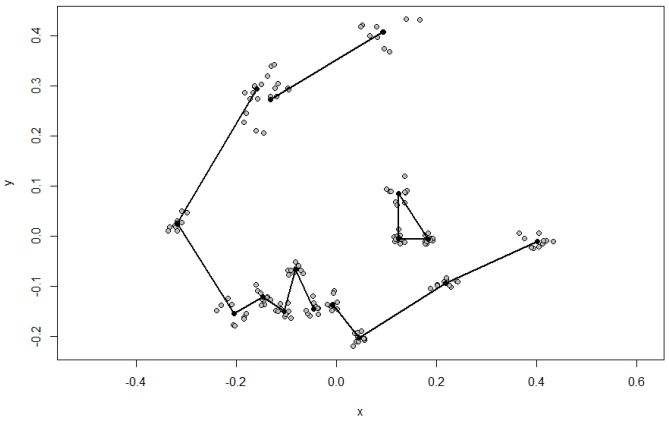
```
myLinks <- cbind(c(1:3,5,5,7:10,12,14:16),c(2:4,6,7,8:11,13,15,16,14))
```

Une fois les liaisons définies, on peut de nouveau appeler la fonction `'plotAllSpecimens'`, mais maintenant en ajoutant le paramètre `'links'` auquel on va associer nos liaisons `'myLinks'`

```
plotAllSpecimens(gpa$coords, plot.param=list(pt.cex=1, mean.cex=1),
links=myLinks)
```

De cette manière on peut voir le visage plus clairement. Sur l'image, on peut observer les disparités des landmarks (en gris) autour de leurs moyennes (en noir). Ces disparités peuvent nous donner des premiers indices sur notre précision de digitalisation, mais il montre aussi la différence entre les individus.

Par exemple sur l'image, on peut voir beaucoup de disparités autour d'oreille (L11, L12 et L13) et autour du landmark d'œil (L14).



3.2. Calcul de la mean shape

La mean shape – c.-à-d. la forme moyenne de tous les individus, a été calculée automatiquement par la fonction `'plotAllSpecimens'`. On peut la voir sur l'image (en points en noirs). Si on veut

connaitre ces coordonnées, on peut les calculer par la fonction 'mshape' appliquée sur les landmarks standardisés par GPA :

```
msh <- mshape(gpa$coords)
```

La matrice 'msh' contient les coordonnées de la mean shape de tous les landmarks, c.-à-d. des deux sessions. Bien sûr si on veut voir cette mean shape, on peut simplement passer par fonction 'plot':

```
plot(msh, asp=1)
```

3.3. Calcul de la mean shape pour chaque session

Si on veut visualiser les différences entre les deux sessions, on peut calculer les mean shape pour deux sessions séparément. On va donc d'abord calculer la mean shape pour la 1ère et puis pour la 2ème session :

```
msh.1 <- mshape(gpa$coords[,,which(session=="a")])
msh.2 <- mshape(gpa$coords[,,which(session=="b")])
```

3.4. Visualisation de la différence de moyenne entre les sessions

Une fois les deux mean shapes calculés, on va les mettre sur le même graphique. On peut passer par la fonction 'plotRefToTarget' qui va faire trois choses:

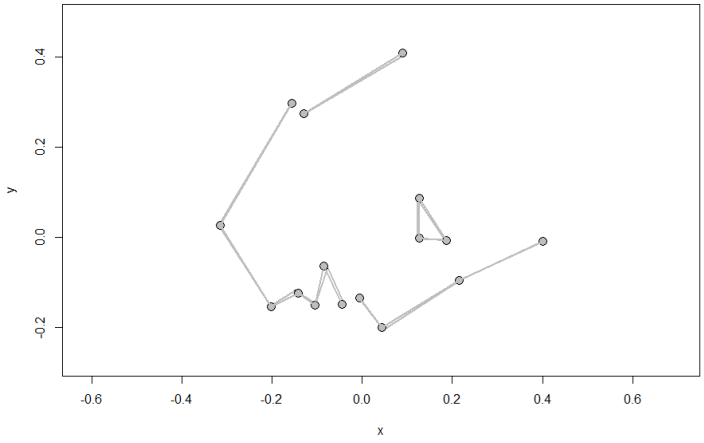
(i) elle va prendre le deuxième mean shape ('msh.2')

(ii) elle va l'ajuster (par GPA) sur la première ('msh.1')

(iii) et ensuite elle va la projeter sur un plot

Ensuite on pourra injecter le premier mean shape ('msh.1') dans le graphique grâce à la fonction 'add.mshape.links'

```
plotRefToTarget(msh.1,msh.2,method="vector",mag=1,links=myLinks)
add.mshape.links(msh.1,myLinks,lwd=2,col="gray")
```



4. Transformation des données d'un array en une matrice

Dans le pas suivant, on va appliquer ACP. La fonction 'prcomp' ne travaille pas avec des objets 'array'. Il faut donc d'abord transformer les données d'un 'array' en une matrice avec $n \times (2k)$ dimensions où n est le nombre des individus (ici $n = 10$ car on a 10 individus) et k est le nombre de dimensions des coordonnées (ici $k = 2$ car on a x- and y- coordonnées). La façon la plus simple est de passer par la fonction 'two.d.array'

```
shape <- two.d.array(gpa$coords)
shape
```

Dans la matrice 'shape' on a donc dans les lignes les individus et dans les colonnes les coordonnées des landmarks. Pour s'en assurer, on peut regarder le nombre des dimensions de la matrice 'shape'

```
dim(shape)
[1] 10 32
```

On voit que la matrice a 10 (individus) x 32 (x et y coordonnées) dimensions

5. Analyse en composante principale (ACP)

Pour visualiser différemment les différences entre nos digitalisations, on peut passer par l'Analyse en composante principale (ACP). Par l'ACP on va créer un espace morphométrique (morphospace), défini par les Composantes Principales (CP), dans lesquelles on va projeter nos individus. Les distances entre les individus dans ce morphospace vont montrer les similarités de leurs formes.

Bien sûr qu'avec l'ACP, on va surtout réduire la dimensionnalité en supprimant les dimensions nulles - c.-à-d. les dimensions qui contiennent aucune (ou peu) des variances. En pratique, parmi plusieurs façon de calculer l'ACP, on peut choisir la fonction `'prcomp'`.

```
pcs <- prcomp(shape)
```

On peut regarder la table des pourcentages de la variance totale par Composantes Principales

```
table.pca <- summary(pcs)$importance
table.pca
```

```
                            PC1         PC2         PC3         PC4         PC5         PC6         PC7
Standard deviation     0.07439169 0.04027529 0.03562874 0.02007485 0.01562456 0.01189202 0.0100744
Proportion of Variance 0.58572000 0.17168000 0.13435000 0.04265000 0.02584000 0.01497000 0.0107400
Cumulative Proportion  0.58572000 0.75740000 0.89175000 0.93440000 0.96024000 0.97521000 0.9859500
                            PC8         PC9          PC10
Standard deviation     0.009045725 0.007136713 4.221039e-17
Proportion of Variance 0.008660000 0.005390000 0.000000e+00
Cumulative Proportion  0.994610000 1.000000000 1.000000e+00
```

On voit, qu'au lieu d'utiliser 32 dimensions (c.-à-d. 32 x et y coordonnées), il nous suffit de prendre quelques (dans ce cas deux ou trois) CP, qui contiennent la plupart de la variance originale.

Maintenant regardons les résultats graphiques d'ACP. Afin de voir tout sur la même image, on peut préparer un espace graphique 2x2 dans lequel on va ensuite injecter les trois résultats d'ACP :

```
layout(matrix(1:4,2,2))
```

On peut faire la même chose avec: `par(mfrow=c(2,2))` - c'est à vous de choisir...

Le premier graphique qu'on va injecter dans 'layout' va montrer le Pourcentage de variance par CP.

Le deuxième graphique sera 'biplot' - c.à.d. la projection des individus sur les deux premiers axes (PC1 et PC2). Il faut dire que 'biplot' est que rarement utilisé dans les études morphométriques.

Le dernier graphique sera le Morphospace – la projection de PC1 versus PC2 avec les 2 sessions coloriées en bleu ou rouge.

D'abord barplot

```
barplot(table.pca[2,],ylab=rownames(table.pca)[2],xlab="PC",names.arg=colnames(table.pca))
```

Ensuit biplot :

```
biplot(pcs)
```

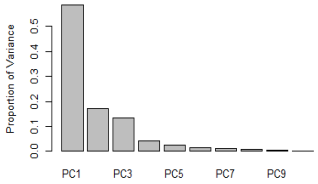
et à la fin PC1 vs PC2 :

```
plot(pcs$x,col=c(rep("blue",n.ind),rep("red",n.ind)),pch=19)
```

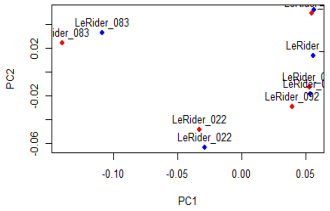
Pour savoir quel point correspond à quel individu, on peut injecter leurs noms

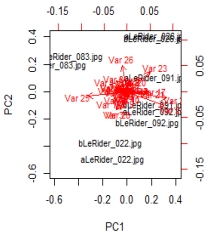
```
text(pcs$x[,1],pcs$x[,2],labels=ind,pos=3)
```

On voit, que la majorité de la variance totale est portée par trois CP (barplot).



Dans PCA vs PC2 graphique on peut observer les différences entre nos digitalisations. On peut observer qu'il existe une différence entre les digitalisations du même individu mais que ces différences semblent moindres que la différence entre les individus.





6. Tests des différences

On voit donc qu'il y a des différences entre la digitalisation des individus des deux sessions. Pour savoir si ces différences sont significatives, on peut utiliser quelques tests statistiques. On pourra aussi tester, s'il y a des différences entre des individus.

En général, on a plusieurs options pour procéder. On peut utiliser par exemple :

1) le Goodall-F test du package 'geomorph'
2) MANOVA
3) Hotelling test

N'importe quel test on choisit, les h0 et h1 sont toujours pareille: H0 dit qu'il n'y a pas des différences entre mesures et H1 dit contraire. Elle dit qu'il a des différences entre mesures

6.1. le Goodall-F test de geomorph:

Goodall-F test ou Procrustes ANOVA sont similaires à l'Analyse de la variance à un facteur (ANOVA). De la même manière, le calcul de F est basé sur le ratio entre 'Between-group SS' et 'Within-group SS'. La différence est que pour mesurer les SS, des "sum-of-squared Procrustes distances" sont utilisées. La distribution des probabilités des valeurs de F pour le test est obtenu par les permutations (pour plus d'informations envoyez `'help("procD.lm")'` ou consultez internet).

En pratique, pour voir si le facteur (d'individu ou de la session) a de l'effet sur la forme, on peut passer par la fonction `'procD.lm'` qui calcule Procrustes ANOVA. Voici comment on applique la fonction dans R : `procD.lm(var.ind~var.dep)`

les variables indépendantes ici sont les landmarks standardisées par GPA (matrix `'shape'`)
les variables dépendantes sont les individus ou session (facteur `'ind'` ou `'session'`)

Notre modèle pour regarder l'effet d'individu est donc...

```
pAoV <- procD.lm(shape~ind,iter=999)
pAoV
```

```
Type I (Sequential) Sums of Squares and Cross-products

Randomization of Raw Values used

          df       SS        MS     Rsq      F      Z P.value
ind        4 0.078015 0.0195038 0.91744 13.891 2.0231   0.001
Residuals  5 0.007020 0.0014041
Total      9 0.085036
```

On voit donc que dans ce cas, on peut rejeter H0 – les individus sont différents.

On peut faire le même test pour savoir s'il y a des différences entre sessions :

```
pAoV2 <- procD.lm(shape~session,iter=999)
pAoV2
```

```
Type I (Sequential) Sums of Squares and Cross-products

Randomization of Raw Values used

          df       SS        MS      Rsq      F       Z P.value
session    1 0.001396 0.0013958 0.016415 0.1335 0.12209  0.9355
Residuals  8 0.083640 0.0104550
Total      9 0.085036
```

6.2. MANOVA

Multivariate analysis of variance (MANOVA) teste si les groupes sont similaires. MANOVA peut être vu comme une extension du F-test (ANOVA). ANOVA teste s'il y a des différences significatives entre deux ou plusieurs groupes sur une variable, MANOVA fait la même chose, elle teste s'il y a des différences significatives entre deux ou plusieurs groupes, mais sur une ou plusieurs variables. Comme les variables des formes sont multidimensionnelles, pour regarder si les groupes sont similaires on applique donc presque toujours MANOVA.

Néanmoins, nous ne pouvons pas appliquer MANOVA directement sur les CP calculées à partir de la GPA. C'est parce que:

(i) certaines des CP peuvent être nulles - c.-à-d. qu'elles ne contiennent aucune variance

(ii) en appliquant la GPA, on perd (dans le cas du 2D) 4 dégrées de liberté (voit au-dessous)

Pour ces raisons, il nous faut d'abord ajuster les données.

De toutes les Composantes principales qu'on a obtenues par ACP, on va choisir celles qui sont non nulles (c.-à-d. celles qui contiennent quelques variances). Parce que les humains, machines et logiciels (R inclus) n'ont pas la même compréhension du "0", il faut d'abord regarder comment la plus petite valeur est définie dans le logiciel R. Cette valeur peut être obtenue avec la ligne suivante

```
.Machine$double.eps
```

De tous les CP on va ensuite sélectionner celles qui ont la variance non nulle

```
non.null <- which(pcs$sdev>.Machine$double.eps)
```

Ensuite on va encore enlever 4 dégrées de libertés – on va donc enlever les 4 dernières valeurs

```
non.null <- non.null[-c((length(non.null)-3):length(non.null))]
```

Après on va sélectionner les CP non nulles et on va appliquer MANOVA. La question ici est de savoir s'il y a des différences de formes entre les individus

```
mod.1 <- manova(pcs$x[,non.null]~ind)
summary(mod.1,test="Pillai") #default: pillai trace
```

```
          Df Pillai approx F num Df den Df    Pr(>F)
ind        4 3.8615   22.312     20     16 3.784e-08 ***
Residuals  5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

On voit qu'on peut rejeter H0.

De même manière, on peut regarder si la session a eu de l'effet sur les différences entre les formes – dit autrement s'il y a des différences de formes entre deux sessions

```
mod.1s <- manova(pcs$x[,non.null]~session)
summary(mod.1s,test="Pillai") #default: pillai trace
```

```
          Df  Pillai approx F num Df den Df Pr(>F)
session    1 0.21204  0.21527      5      4 0.9385
Residuals  8
```

6.3 Hotelling test d'après Claude (2008)

L'approche de Hotelling peut être aussi vue comme un type d'ANOVA. Dans notre cas, on peut se poser la question si la 'session' a l'effet sur 'la forme'. On peut donc définir un modèle linéaire: `'lm(forme~session)'` où `'forme'` contient les coordonnées et `'session'` est le facteur (ou effet).

Ce modèle nous permettra ensuite de décomposer les différentes sources de variances de la variance totale (exprimé par les SS). Cela nous permet de définir :

- combien de variances peuvent être expliquées par le facteur (c.-à-d. par la session) => cette somme de variance on va la noter `'SSef'` (comme Sum-of-squares of effect)
- combien de variances ne peuvent pas être expliquées par ce facteur => cette somme de variance on va la noter `'SSer'` (comme Sum-of-squares of error)

Pour chacune de ces variances on va ensuite définir le nombre de dégrées de liberté (ddl):

- `'dfef'`(ddl pour effet) = $p - 1$ où p est le nombre des niveaux du facteur
- `'dfer'` (ddl pour error) = $n - p$ où p est le nombre des niveaux du facteur et n le nombre des individus

Une fois que l'on a ces quatre valeurs, on peut les utiliser dans la fonction `'Hotellingsp'`

Pour récapituler, la fonction `'Hotellingsp'` nécessite les 4 paramètres suivants :

- `'SSef'`: Sum of squares and cross-products of effect (dans notre cas, l'effet est soit `'ind'`, soit `'session'`)
- `'SSer'`: Sum of squares and cross-products of residual variation
- `'dfef'`: df for the effect term
- `'dfer'`: df for the error term

D'abord il faut appliquer le modèle linéaire. Comme facteur, on va d'abord regarder l'effet 'individu'.

```
library(MASS)
fact <- ind
n <- 10
p <- 5
mod1 <- lm(pcs$x[,1:4]~as.factor(fact))
```

Ensuite, il faut calculer les matrices de dispersion du facteur/effect et de l'erreur et les dégrées de liberté

```
SSef <- (n-1)*var(mod1$fitted.values)
SSer <- (n-1)*var(mod1$residuals)
dfef <- length(levels(fact))-1
dfer <- n - length(levels(fact))
```

A la fin, le test de Hoteling peut être fait.

```
Hotellingsp(SSef, SSer, dfef, dfer)
```

dfeffect	dferror	T2	Approx_F	df1	df2	p
4.00000000	5.00000000	813.26078669	25.41439958	16.00000000	2.00000000	0.03849097

On rejeter H0.

De la même manière on peut se demander, si la session produit un effet sur les différences entre les formes.

```
fact <- session
n <- 10
p <- 2
mod1s <- lm(pcs$x[,1:4]~as.factor(fact))
SSef <- (n-1)*var(mod1s$fitted.values)
SSer <- (n-1)*var(mod1s$residuals)
```

```
dfef <- length(levels(fact))-1
dfer <- n - length(levels(fact))
Hotellingsp(SSef, SSer, dfef, dfer)
```

```
   dfeffect    dferror         T2    Approx_F        df1        df2          p
1.00000000 8.00000000 0.03853439 0.04816799 4.00000000 5.00000000 0.99419865
```

7. Calcul de l'erreur de mesure d'apres Bailey and Byrnes (1990)

L'erreur de mesure peut être exprimée par 'mean square error' (Bailey and Byrnes 1990) :

$$ME = \frac{MS\ between\ (ou\ among)}{MS\ within * p} * 100 = \frac{\frac{SSer}{dfer}}{\frac{SSef}{dfer} - \left(\frac{SSer}{dfer} * p\right)} * 100$$

La fonction 'meas.error' nécessite les mêmes paramètres que la fonction 'Hotellingsp'

```
mod <- lm(shape~ind)
SSCPind <- crossprod(mod$effects[which(mod$assign==1),,drop=F])
SSCPerr <- crossprod(mod$residuals)
SSind <- sum(diag(SSCPind))
SSerr <- sum(diag(SSCPerr))
dferr <- (n.session-1)*n.ind
dfind <- (n.ind-1)
meas.error(SSind,SSerr,dfind,dferr)
```

```
$s2_among
[1] 0.009049879

$s2_within
[1] 0.00140409

$me
[1] 13.43117
```

On voit, que cette erreur est égale à 13 %.

8. Calcul de l'erreur (version per landmarks)

ME peut être calculé pour chaque landmark séparément. Ça nous permettra de quantifier la précision avec laquelle on a localisé les landmark sur les objets (dans le cas de session). Mais comme nous avons trouvé avant qu'il n'y avait pas la différence entre les deux sessions, nous allons donc regarder l'erreur liée aux différences entre individus.

On va préparer une matrix vide 'ME' avec 16 colonnes (pour 16 landmarks) et trois lignes (pour garder l'infos sur s2_among, s2_within et ME). Après, on va faire une boucle : pour chaque landmark, on va calculer chaque source de la variance et ME (voit au-dessous).

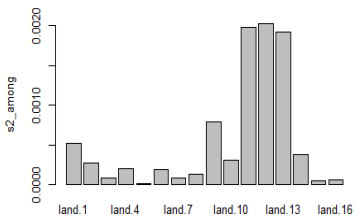
```
ME <- matrix(NA,3,n.land)
rownames(ME) <- c("s2_among","s2_within","%me")
colnames(ME) <- paste("land",1:n.land,sep=".")
land <- seq(1,n.land*n.dim,by=n.dim)
for (l in land){
  SSind <- sum(diag(SSCPind[l:(l+1),l:(l+1)]))
  SSerr <- sum(diag(SSCPerr[l:(l+1),l:(l+1)]))
  tmp <- meas.error(SSind,SSerr,dfind,dferr)
  ME[1,ceiling(l/n.dim)] <- tmp$s2_among
  ME[2,ceiling(l/n.dim)] <- tmp$s2_within
  ME[3,ceiling(l/n.dim)] <- tmp$me
}
ME
```

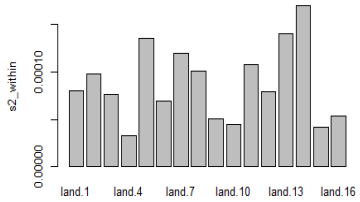
9. Visualisation des sources des variances

On peut projeter trois barplots : deux pour voir deux sources de variations et un pour ME

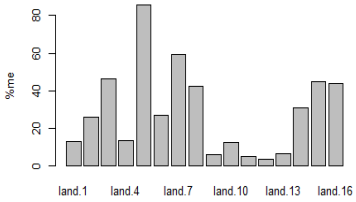
```
par(mfrow=c(2,2))
for (i in 1:3)
  barplot(ME[i,],ylab=rownames(ME)[i])
```

Le premier graphique (S2_among) montre les variances des landmarks entre les individus. On peut observer que les landmarks L9, L11, L12 et L13 varient beaucoup au niveau des individus. Par ces landmarks, on peut donc mieux différencier les individus.





Sur le graphique à droite (S2_within), on a la variation résiduelle, c.-à-d. la variation qui n'était pas expliquée par la différence entre individus. Ces différences peuvent être donc causées par l'effet de la digitalisation (mais aussi par hasard !).



Enfin, sur la troisième image, on a le ratio entre ces deux valeurs, c.à.d. (S2_among/S2_within*p*100). Notez, qu'il y a un bug dans le script (au lieu d'erreur, il montre la qualité…). Le graphique doit donc être regardé « inversement ». Les landmarks L9-L13 ont la marge d'erreur la plus grande.

10. Visualisation des effets d'Ind et Erreur

Pour visualiser les changements des landmarks entre les individus, il faut d'abord obtenir l'axe majeur de variation de la matrice de variation entre les individus.

```
vec.ind <- matrix(prcomp(SSCPind/dfind)$rotation[,1],n.land,n.dim,byrow=T)
```

Ce vecteur `'vec.ind'` donne des directions et quantités du « déplacement » des landmarks. Pour mieux comprendre ce que c'est, on peut le visualiser :

```
plot(vec.ind, type="n",asp=1)
for (i in 1:dim(vec.ind)[1]) {lines(rbind(c(0,0), vec.ind[i,]))}
```

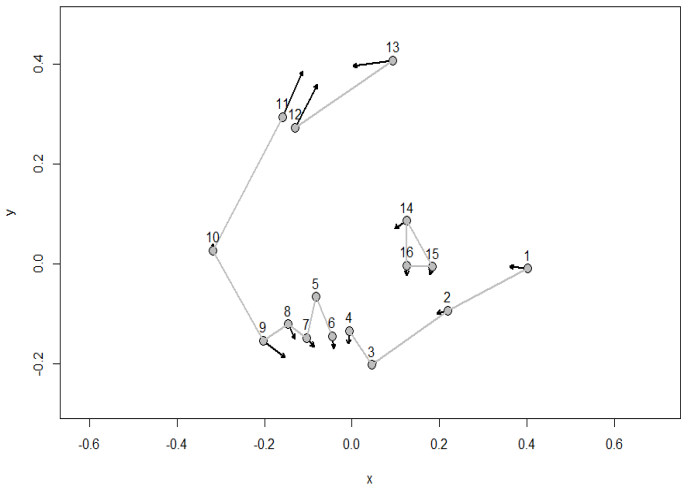
Ensuite, on va prendre les coordonnées de la mean shape (`'msh'`) et on va les « déplacer » par ce vecteur (`'vec.ind'`). Notez, que la valeur 0.2 donne la magnitude du déplacement qui nous permet de mieux voir les changements des formes (au lieu du 0.2, on peut choisir n'importe quel chiffre qu'on veut). Les nouvelles coordonnées vont être données dans l'objet `'target'`

```
target <- msh + vec.ind * 0.2
```

A la fin on va juste projeter `'msh'` et `'target'`

```
plotRefToTarget(msh,target,method="vector",mag=1)
add.mshape.links(msh,myLinks)
text(msh[,1],msh[,2],labels=1:n.land,pos=3)
```

On voit les directions et magnitudes des déplacements des landmarks qui différencient mieux les individus. On voit donc que les individus sont différents au niveau des oreilles (L11-L13) et un peu au niveau du menton (L9).
Juste pour information : la forme moyenne (`'msh'`) est en points gris et le `'target'` est défini par la fin des flèches noirs. Ce graphique nous montre les mêmes tendances que le graphique (S2_among).



De la même manière, on peut visualiser l'axe majeur de variation de la matrice de variation résiduelle

```
vec.err <- matrix(svd(SSCPerr/dferr)$u[,1],n.land,n.dim,byrow=T)
target <- msh + vec.err * 0.2
plotRefToTarget(msh,target,method="vector",mag=1)
add.mshape.links(msh,myLinks)
text(msh[,1],msh[,2],labels=1:n.land,pos=3)
```

Remarque: Si nous avions travaillé sur les scores de l'ACP alors nous aurions eu besoin de retourner dans l'espace des coordonnées, via `mat.Ind <- eigenvec%*%MSCPind%*%t(eigenvec)`

Visualiser l'effet de la session

Si on a trouvé que l'effet de la session était significatif, on peut alors visualiser la différence de mean shape entre sessions

```
msh.s1 <- apply(gpa$coords[,,session=="a"],c(1,2),mean)
msh.s2 <- apply(gpa$coords[,,session=="b"],c(1,2),mean)
target <- msh.s1 + (msh.s2-msh.s1)* 10
plotRefToTarget(msh.s1,target,method="vector",mag=1)
```

Comparaison entre les denars

L'autre jeu des données contient trois lots de denars (monnaies romaines) trouvés en Bohème. Le premier type de denar figure la tête de Antonius Pius (86-161 AD), le deuxième de Marcus Aurelius (121-180 AD) et le troisième de Hadrien (76-138 AD). Choisissez deux types de lots et répondez aux questions suivantes :

1) Y a t'il des différences entre deux types de monnaies ?
2) Figurez ces différences dans des graphiques.

Ecrivez un rapport sous la forme d'un article en 1-2 pages (introduction, méthodes, résultats, interprétations, conclusion, bibliographie).