Initiation à l'analyse de données

Jeu de données : « YEAST – Protein Localization Sites »

Vanna BOUNGNALITH, Michael HAJAGE et Mathieu PONT

1 Introduction

Dans le cadre du master informatique 1ère année de l'université Paris Descartes il nous a été demandé, dans l'ECUE Initiation à l'analyse de données, d'analyser le jeu de données « YEAST – Protein Localization Sites ». Ce rapport retrace de façon synthétique les travaux et les choix effectués durant ce projet.

2 Description des données

Le jeu de données est issu d'une étude du docteur en informatique Paul Horton et du docteur en biologie cellulaire Kenta Nakai dans laquelle ils ont défini un système de classification probabiliste qui permet de prédire les sites de localisations de protéines [Horton et al., 1996]. Leur modèle de classification se base sur les séquences d'acides aminés des protéines et l'utilisation d'une méthode de *cross-validation*.

Chacun des 1484 échantillons est constitué du nom d'une séquence protéique et de sa localisation (qui est sa classe, son label), ainsi que 8 variables décrivant la séquence :

- mcg : résultat de la méthode de McGeoch pour la reconnaissance du signal de la séquence.
- gvh : résultat de la méthode de von Heijne pour la reconnaissance du signal de la séquence.
- alm : score du programme de prédiction des régions transmembranaires.
- mit : score de l'analyse discriminante du contenu de l'acide aminé de la région N-terminale des protéines mitochondriales ou non-mitochondriales.
- erl : indique la présence ou non d'une sous-chaine « HDEL » dans la séquence. Attribut binaire (valeur de 0.5 ou 1).
- pox : valeur du signal PTS (Peroxysome Transfert Signal) dans l'extrémité C-terminale (une des deux extrémités d'une protéine ou d'un peptide). Les protéines destinées aux péroxysomes sont porteuses de ce signal PTS.

- vac : score de l’analyse discriminante de la teneur en acides aminés des protéines vacuolaires et extracellulaires.
- nuc : score de l’analyse discriminante des signaux de localisation nucléique de protéines nucléiques et non nucléiques.

Ce sont toutes des variables quantitatives sauf *erl* qui est une variable qualitative et le jeu de données ne contient pas de valeurs manquantes.

Ci-dessous la répartition des échantillons en fonction des classes est représentée :

Classe	CYT	NUC	MIT	ME3	ME2	ME1	EXC	VAC	POX	ERL
Nombre	463	429	244	163	51	44	35	30	20	5

Tableau 1: Nombre d'échantillons par classes.

On remarque que les échantillons ne sont pas équitablement distribués sur l’ensemble des 10 classes. Cela peut poser des problèmes si l’on veut concevoir un système de classification par exemple, les échantillons les moins représentés pourront être plus complexe à classifier.

	Min.	**1st qu.**	**Median**	**Mean**	**3rd qu.**	**Max.**	**Std**
mcg	0.11	0.41	0.49	0.50	0.58	1.00	0.14
gvh	0.13	0.42	0.49	0.50	0.57	1.00	0.12
alm	0.21	0.46	0.51	0.50	0.55	1.00	0.09
mit	0.00	0.17	0.22	0.26	0.32	1.00	0.14
erl	0.5	0.5	0.5	0.5	0.5	1.0	0.05
pox	0.00	0.00	0.00	0.08	0.00	0.83	0.08
vac	0.00	0.48	0.51	0.50	0.57	1.00	0.06
nuc	0.00	0.22	0.22	0.28	0.30	1.00	0.11

Tableau 2: Description des variables (critères de position et de dispersion).

On voit que les différentes variables n’ont pas la même moyenne ni le même écart-type, si nous voulons concevoir un système de classification il sera intéressant de centrer-réduire les données (normalisation). La différence sur l’écart type n’est pas flagrante contrairement à la moyenne, donc seulement centrer les données pourrait suffire.

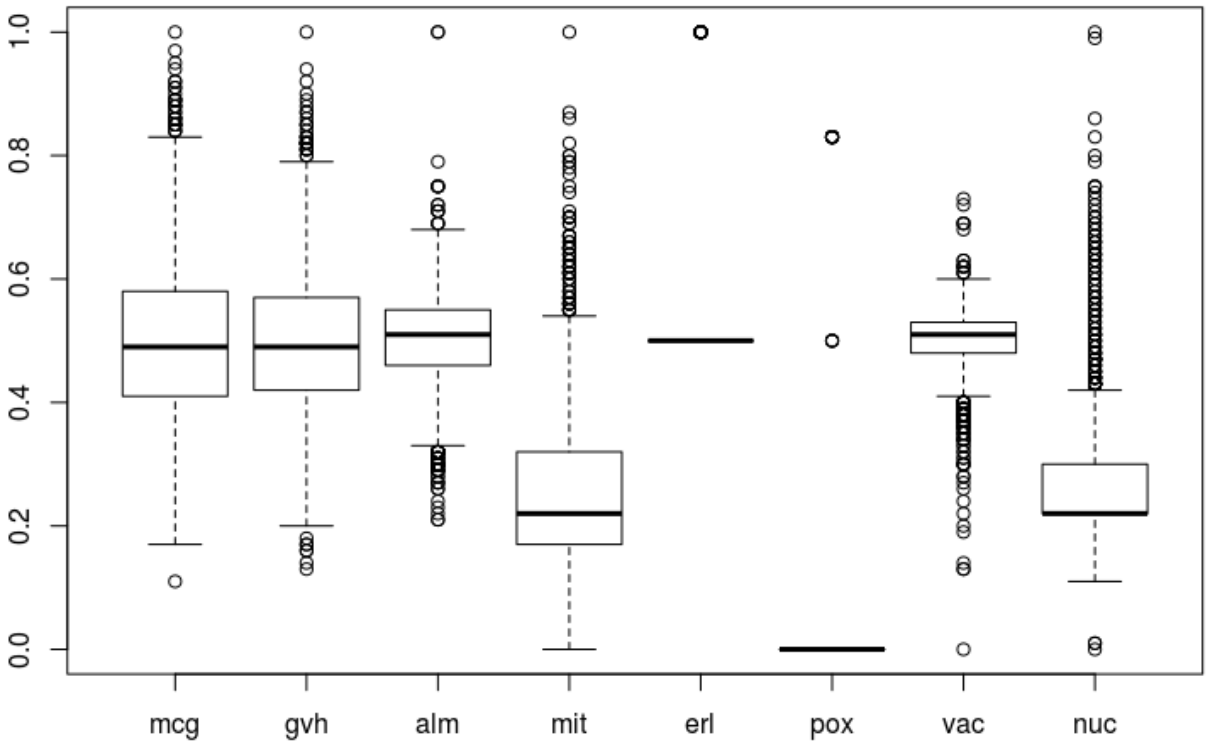

Figure 1 : Boxplot pour les 8 variables descriptives.

Ce boxplot nous indique que l'on a beaucoup de valeurs extrêmes (potentiellement des valeurs aberrantes). Il faudra donc faire attention lors de l'analyse ou classification car elles pourraient influer sur les résultats.

	mcg	**gvh**	**alm**	**mit**	**erl**	**pox**	**vac**	**nuc**
mcg	*1*	0.58	-0.16	0.16	0.06	0.01	0.08	-0.12
gvh	0.58	*1*	-0.27	0.14	0.06	~0	0.09	-0.10
alm	-0.16	-0.27	*1*	0.06	-0.01	0.01	-0.19	-0.02
mit	0.16	0.14	0.06	*1*	-0.01	-0.01	-0.10	-0.05
erl	0.06	0.06	-0.01	-0.01	*1*	-0.01	0.04	~0
pox	0.01	~0	0.01	-0.01	-0.01	*1*	0.02	-0.04
vac	0.08	0.09	-0.19	-0.10	0.04	0.02	*1*	0.09
nuc	-0.12	-0.10	-0.02	-0.05	~0	-0.04	0.09	*1*

Tableau 3: Matrice de corrélation entre les variables.

Les variables *mcg* et *gvh* semblent légèrement corrélées, cela fait sens car les données sont tirées de deux méthodes presque identique (se basant sur le même principe).

La méthode McG prend en compte la terminaison amine chargé et la zone hydrophobe de la chaîne peptidique d'une protéine. A partir de la taille de la zone hydrophobe et de la charge de la terminaison amine, un discriminant est alors calculé. Un discriminant positif et élevé signifie que la chaîne peptidique aura une grande probabilité d'être fendue.

La méthode GvH est, comme la méthode MvG, une méthode de reconnaissance se basant sur d'autres propriété de la chaîne peptidique qui calcule également un déterminant donnant la probabilité que la chaîne soit fendue.

Les deux méthodes étant similaire elle donnent des critères de position et de dispersion du même ordre de grandeur voire identiques. On peut donc directement supposer qu'elles sont corrélées. Cette supposition est confirmée dans la matrice de corrélation des variables ou le coefficient de corrélation entre Mcg et Gvh vaut 0,58.

Hormi elles, il ne semble pas avoir de corrélation entre les autres variables.

3 Analyse

Pour analyser les données nous avons commencé par réaliser une analyse en composante principle (ACP). Etant donné que les moyennes et ecart-types sont différents entre les variables nous avons opté dans un premier temps pour une ACP normée. Comme la variable *erl* est qualitative nous l'avons ajouté aux variables supplémentaires.

Le premier résultat n'était pas très satisfaisant, avec une inertie de seulement 43,9 %. Etant donné que les écarts-types sont peu différents et que les variables sont dans des gammes de valeurs plus ou moins identiques nous avons finalement décidé de faire une ACP non normée.

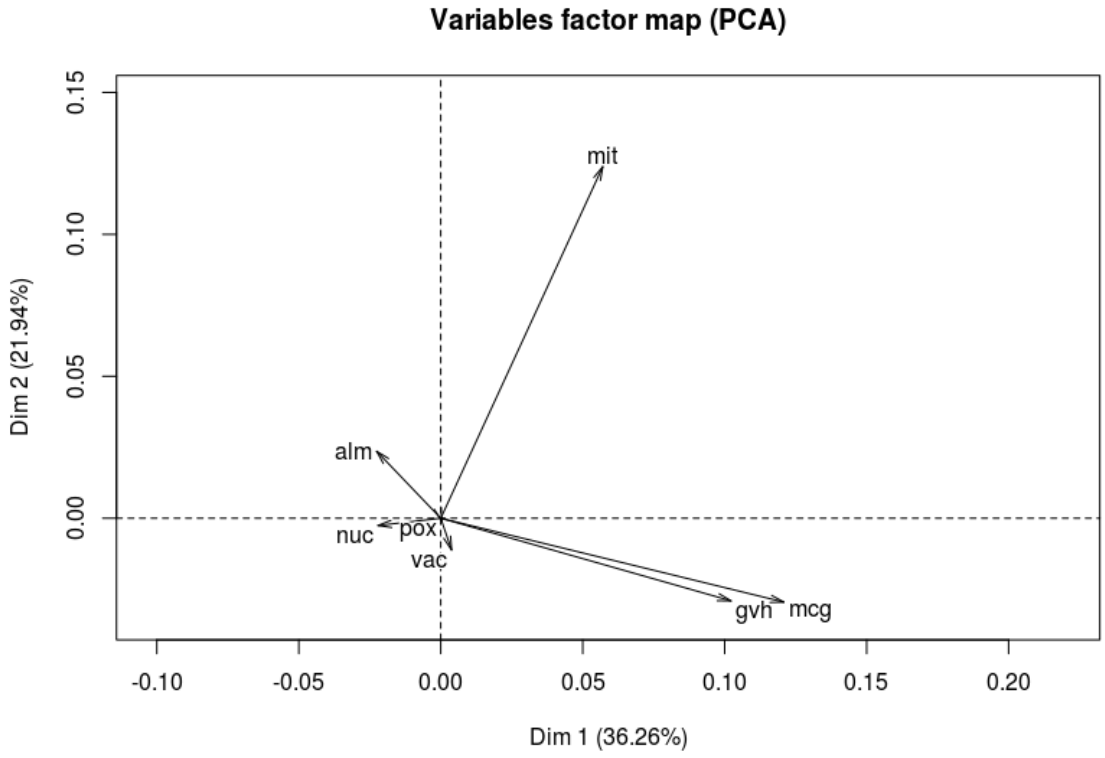

Figure 2 : Première ACP non normée.

Avec l'ACP non normée l'inertie est plus grande qu'avec l'ACP normée, avec une valeur de 58,2 %.

Comme prévu, les variables *mcg* et *gvh* sont fortement corrélées. Les variables *pox*, *nuc*, *vac* et *alm* semblent ne pas contribuer beaucoup aux composantes principales.

On voit que la variable *pox* n'est pas du tout représentée par l'ACP. Nous avons essayé de l'ajouter en variable qualitative supplémentaire pour qu'elle ne soit pas prise en compte dans les calculs.

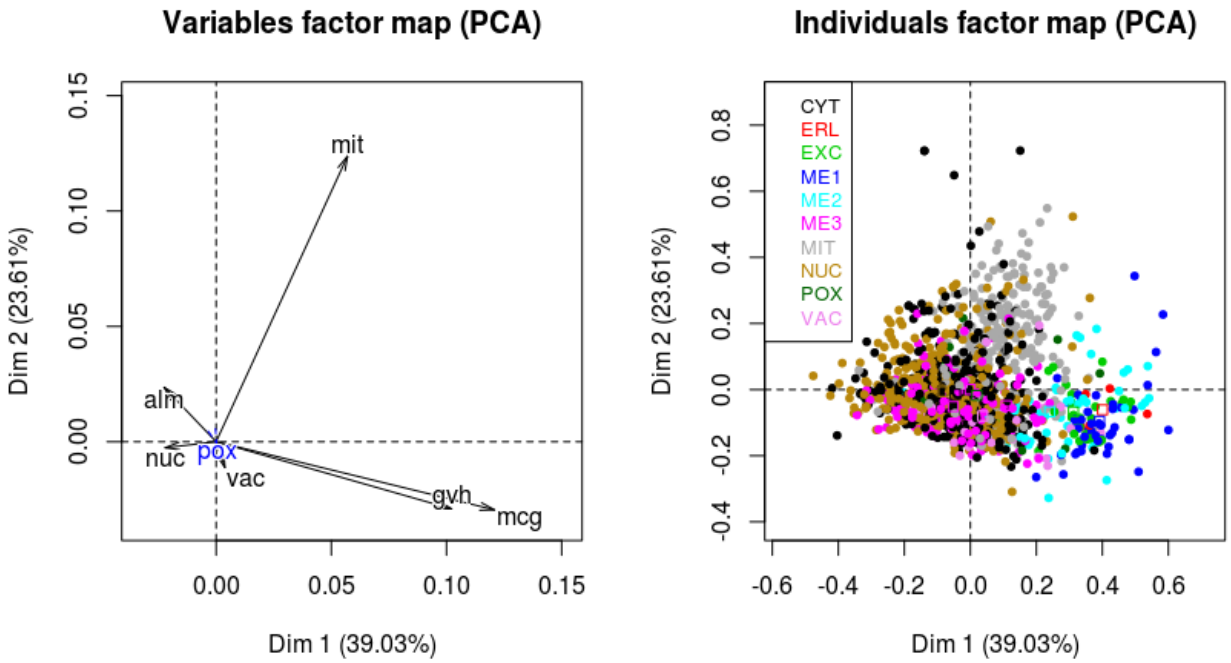

Figure 3 : Deuxième analyse en composante principale, en ajoutant pox comme variable qualitative supplémentaire.

On remarque qu'en ajoutant *pox* comme variable supplémentaire nous avons augmenté l'inertie à 62,64 %. C'est un meilleur résultat mais toujours pas des plus satisfaisant.

Etant donné que l'ACP peut être influencée par les valeurs extrêmes nous nous devions d'essayer de les traiter. Malheureusement nous n'avons pas d'expert en protéine et nos connaissances dans ce domaine sont extremement limitées, nous ne pouvions pas correctement décidé des actions à réaliser quant à ces valeurs.

Nous avons finalement décidé de réaliser une ACP en enlevant les échantillons ayant une variable à l'extérieur des moustaches du boxplot de la *figure 1*. Nous aurions pu tout aussi bien les remplacer par une valeur typique comme la moyenne ou la médiane par exemple. Avec cette ACP nous avons obtenu une inertie de 65,72 %.

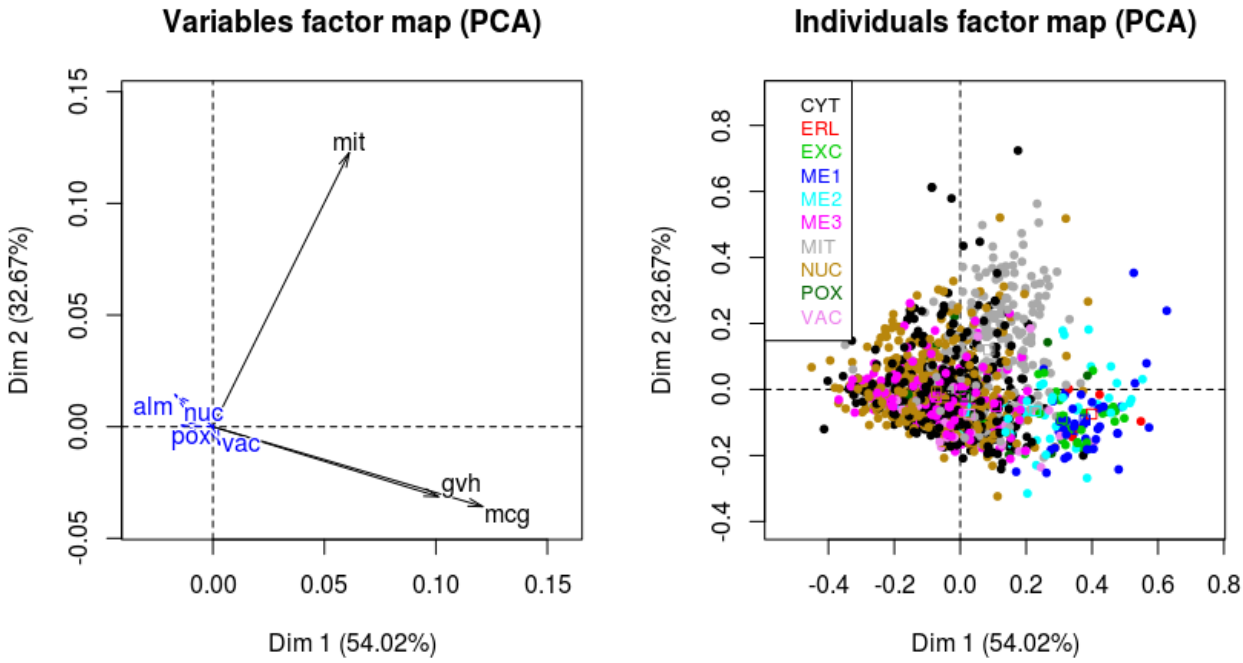

Figure 4 : Troisième analyse en composante principale, en ajoutant pox, vac, nuc et alm comme variable qualitative supplémentaire.

Nous avons aussi ensuite mis les variables autres que *mit*, *gvh* et *mcg* en tant que variables supplémentaires et avons obtenu une inertie de 86,68 %. Ce résultat est obtenu avec les valeurs extrêmes, sans, le résultat est plus bas (83,11 %).

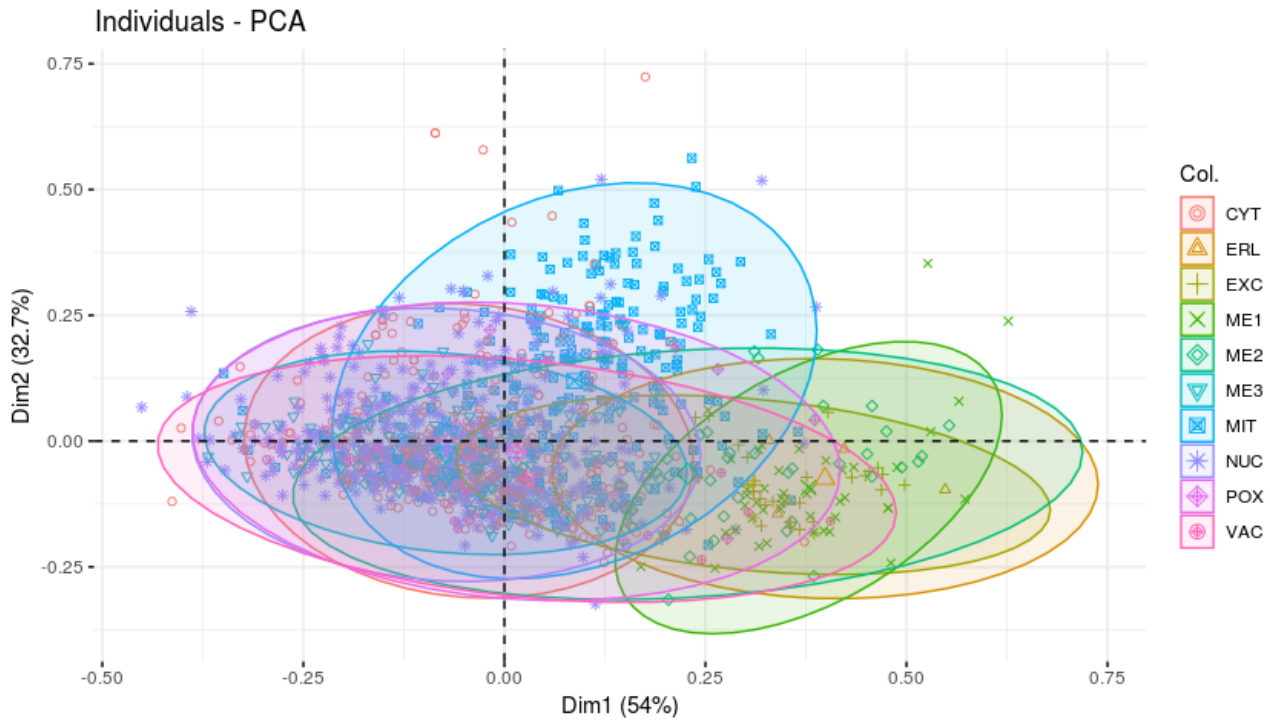

Figure 5 : Ellipse des classes.

On voit sur cette représentation du plan factoriel des individus que les classes ne sont pas bien séparées sur les axes principaux de l'ACP.

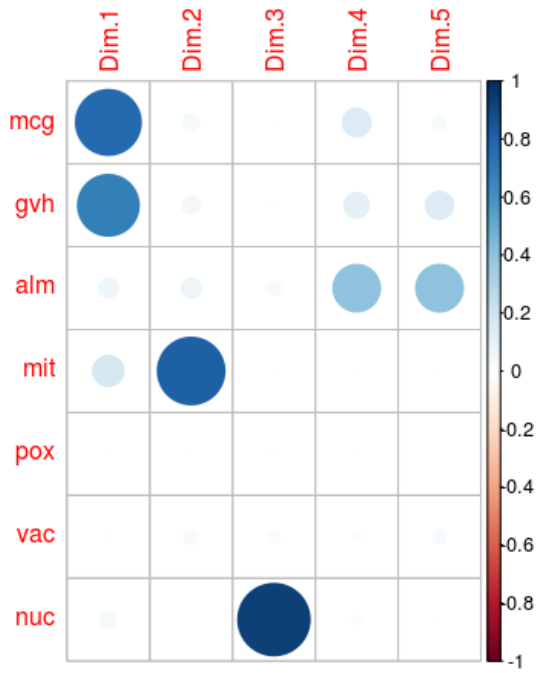

Figure 6 : Qualité de représentation cos2 de chaque variable.

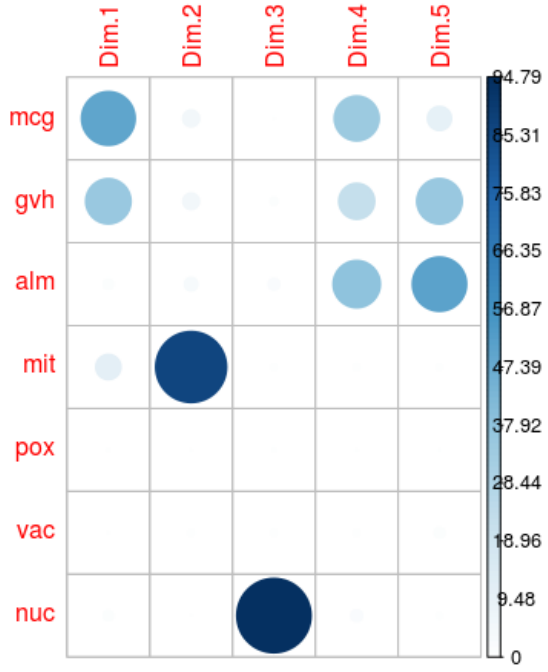

Figure 7 : Contribution de chaque variable aux composantes.

Ces figures ont été réalisées à partir de la première ACP (*figure 2*). Comme on le voit les variables *pox*, *vac*, *alm* et *nuc* ont une qualité de représentation et une contribution presque nulles pour tous les axes. On note tout de même que la variable *alm* ne l'est pas pour les axes 4 et 5. Il est intéressant de voir que la variable *nuc* possède une très bonne qualité de représentation et de contribution pour l'axe 3. Les variables les plus importantes aux deux axes principaux semblent être *mcg* et *gvh* pour l'axe 1 et *mit* pour l'axe 2.

4 Classification

Nous avons décidé (même si cela n'était pas forcément demandé) de concevoir un système de classification sur ce jeu de données. Nous devions donc choisir un algorithme de classification multi-classes.

Dans [Horton et al., 1997] l'algorithme kNN a été choisi. Nous trouvions intéressant d'essayer de comparer ses performances à un autre type d'algorithme. En nous intéressant à la catégorie des algorithmes supervisés (étant donné que nous avons les labels) nous avons hésité entre une machine à vecteur de support et un réseau de neurones. Ayant déjà implanté ces derniers nous nous sommes plutôt tourné vers cette solution.

Nous avons implanté cet algorithme en python et à l'aide de la librairie tensorflow, le code est disponible sur le répertoire github de l'un de nous, le lien est en [1].

Dans un premier temps nous avons construit un modèle qui faisait du sur-apprentissage sur le jeu de données (100 % de précision sur le jeu d'entrainement). Ensuite à l'aide de regularisation comme dropout [Hinton et al., 2012] et de la regularisation l2 nous avons homogénéisé la précision sur le jeu d'entrainement et le jeu de test.

Pour valider les performances du système nous utilisons la méthode des *k-fold cross-validation*. Elle consiste à séparer le jeu de données en *k* ensembles et utiliser chacun d'eux comme jeu de test pour un système ayant appris sur les *k-1* autres ensembles. Nous avons une architecture de la forme 8-256-10 (Input-Hidden-Output) avec l'optimiseur Adam.

k	0	1	2	3	4	**Moyenne**	**Ecart-type**
Précision	62,84	61,82	61,49	62,50	61,82	**62,09**	0,56

Tableau 4: Précision totale de classification pour chaque ensemble de test (k-fold cross-validation).

Nous atteignons, pour la précision générale, un score moyen de 62,09 % sur les 5 ensembles de test. Notre précision est plus basse que [Anastasiadis et al., 2003] qui atteint, avec un réseau de neurones également, une précision moyenne de 69,5 %. Nous n'avons pas réussi à reproduire leur résultat. Cependant nous avons une meilleure précision que l'algorithme kNN de [Horton et al., 1997] qui atteint les 59,5 %.

Classe	**CYT**	**NUC**	**MIT**	**ME3**	**ME2**	**ME1**	**EXC**	**VAC**	**POX**	**ERL**
Précision	70,4	54,5	59,4	**85,9**	29,4	79,5	31,4	0	50	**100**
kNN [Horton et al., 1997]	**70,7**	50,7	57,8	74,8	21,6	79,5	**62,9**	0	**55**	0
NN [Anastasiadis et al., 2003]	66,7	**57,7**	**61,3**	85,6	**47,8**	**82,9**	62,7	**4,1**	54,6	99,6

Tableau 5: Précision par classe sur tous les k ensembles de test. Comparaison des résultats avec deux autres résultats.

On voit que notre modèle n'arrive pas à identifier les échantillons de la classe VAC, elle semble être complexe à classifier au vu des résultats des deux autres papiers. Quant à la classe ERL le modèle arrive à classer 100 % des échantillons contrairement à 0 % pour le kNN.

	CYT	NUC	MIT	ME3	ME2	ME1	EXC	VAC	POX	ERL
CYT	**326**	97	32	4	2	0	1	0	1	0
NUC	152	**234**	25	14	1	0	2	0	0	1
MIT	68	13	**145**	7	5	3	1	0	2	0
ME3	9	12	1	**140**	1	0	0	0	0	0
ME2	9	3	5	8	**15**	9	2	0	0	0
ME1	0	0	1	2	2	**35**	4	0	0	0
EXC	6	1	3	0	3	11	**11**	0	0	0
VAC	13	5	3	7	1	0	1	**0**	0	0
POX	6	0	3	0	1	0	0	0	**10**	0
ERL	0	0	0	0	0	0	0	0	0	**5**

Tableau 6: Matrice de confusion pour tous les k ensembles de test.

Cette matrice nous montre que le modèle confond souvent la classe CYT et NUC en ne classant pas correctement un échantillon de l'une de ces classes dans l'autre. On revoit le même problème dans [Horton et al., 1997] et [Anastasiadis et al., 2003].

5 Commentaires et conclusions

Nous avons vu au travers des résultats de l'ACP que certaines variables semblent ne pas beaucoup influencer les résultats. Au final ce sont principalement trois variables : *mit*, *mcg* et *gvh* qui contribuent aux axes principaux.

Encore une fois, la signifcation de chacune de ces variables est relativement complexe pour pouvoir en tirer des conclusions sur l'importance de l'une d'elles sur la localisation du site de la protéine (sa classe). Un expert du domaine aurait été utile pour mieux analyser l'influence de chacune d'elles.

Pour le système de classification nous obtenons de meilleurs résultats que l'algorithme kNN mais de moins bon que le réseau de neurones. Il aurait été surement plus intéressant de concevoir une machine à vecteur de support afin de comparer trois algorithmes différents.

Cependant nous avons tout de même appris dans l'ajustement d'un réseau de neurones en concevant le notre.

Références

[Horton et al., 1996] Paul Horton and Kenta Nakai. "A Probabilistic Classification System for Predicting the Cellular Localization Sites of Proteins". 1996.

[Horton et al., 1997] Paul Horton and Kenta Nakai. "Better Prediction of Protein Cellular Localization Sites with the k Nearest Neighbors Classifier".

[Anastasiadis et al., 2003] Aristoklis D. Anastasiadis, George D. Magoulas and Xiaohui Liu. "Classification of Protein Localisation Patterns via Supervised Neural Network Learning". 2003.

[Hinton et al., 2012] Hinton Geoffrey E., Srivastava Nitish, Krizhevsky Alex, Sutskever Ilya, Salakhutdinov Ruslan R. "Improving neural networks by preventing co-adaptation of feature detectors". 2012.

[1] https://github.com/MatPont/Yeast_Classification

Annexes

```
dfYeast <- read.table("yeast.data")
dfYeast
names(dfYeast) <- c("Sequence Name", "mcg", "gvh", "alm", "mit", "erl", "pox", "vac", "nuc",
"Localization (label)")
dfYeast
summary(dfYeast)
#matYeast <- data.matrix(dfYeast)
#matYeast <- matYeast[,-1]
matYeast <- dfYeast[,-1]
matYeast
varYeast <- matYeast[,-9]
varYeast
for(i in 1:ncol(varYeast)){
  print(sd(varYeast[,i]))
}
mcor <- cor(varYeast)
mcor
write.csv(mcor,"yeast_correlation-matrix.csv")
boxplot(varYeast)

# ALL PCA
library("FactoMineR")
library("corrplot")
library("factoextra")
help(PCA)

matYeast

valeurs_aberrantes <- function(x){
  tmp <- ncol(x) -1
  for(i in 1:tmp){
    vector_indice=c()
    q1=quantile(x[,i])[2]
    q3=quantile(x[,i])[4]
    ecart = q3-q1
    val_extremes_min = q1-1.5*ecart
    val_extremes_max = q3+1.5*ecart
    for (j in 1:nrow(x)){
      if (x[j,i]>val_extremes_max || x[j,i]<val_extremes_min){
        vector_indice = c(vector_indice,j)
      }
    }
    vector_indice=unique(vector_indice)
    print(length(vector_indice))
    x = x[-c(vector_indice),]
    print(x)
  }
  return(x)
```

```
}
#matYeast <- valeurs_aberrantes(matYeast)
#matYeast

# PCA
layout(matrix(c(1,2), ncol=2))
resPCA <- PCA(matYeast, quali.sup = c(5, 9), scale.unit = FALSE)
plot.PCA(resPCA, choix="ind", habillage = 9, label = "none")

var <- get_pca_var(resPCA)
corrplot(var$cos2)
corrplot(var$contrib, is.corr=FALSE)

# PCA avec pox en variable supplémentaire
layout(matrix(c(1,2), ncol=2))
resPCA <- PCA(matYeast, quali.sup = c(5, 9), quanti.sup = c(6), scale.unit = FALSE)
plot.PCA(resPCA, choix="ind", habillage = 9, label = "none")

# PCA avec pox, vac, nuc et alm en variables supplémentaires
layout(matrix(c(1,2), ncol=2))
resPCA <- PCA(matYeast, quali.sup = c(5, 9), quanti.sup = c(3,6,7,8), scale.unit = FALSE)

resPCA$eig
dimdesc(resPCA)

plot.PCA(resPCA, choix="ind", habillage = 9, label = "none")
fviz_pca_ind(resPCA, col.ind = matYeast[,9], label = "none", addEllipses = TRUE)
```