Projet 2 SY09

Pierre-Louis Guillot - Capucine PRUDHOMME

July 18, 2018

1 Introduction

Cette partie a pour objectif de présenter des principes qui ne dépendend pas d'un jeu de données précis. On peut notamment évoquer le fait que pour toutes les données, nous ne prenons jamais en compte la première variable X, car celle-ci correspond à l'index des données, et pourrait donc fausser les résultats.

1.1 Méthodologie de sélection de modèle

Pour l'ensemble des jeux de données suivants et pour chaque méthode, nous séparons les données en données d'entrainement (80% des données) et données de test (les 20% restants).
Pour la sélection de modèles nous avons plus précisement fait 50 fois les tests d'erreur de classification en redivisant à chaque fois le jeu de données entier en un ensemble d'apprentissage et un ensemble de test.
Cela aura pour conséquence de légèrement biaiser l'erreur qu'on pourrait trouver à la fin puisque la sélection de modèle a été en partie faite sur tout le jeu de donné. Ici non seulement certains jeux de donnés sont assez petits mais on a pris le parti de ne pas tant s'intéresser à l'erreur du meilleur modèle qu'à la comparaison des modèles entre eux. On acceptera l'approximation.
En outre une majeure partie de ce projet a été réalisée avant le cours sur la validation croisée, et après avoir testée celle-ci sur quelques données elle semble donner des résultats très similaires à notre méthode. On essaiera cependant en conséquence de ne pas trop interprêter les différences de performances mineures.
De plus, à titre d'indication on représentera l'intervalle sur lequel se répartissent les erreurs de test dans le jeu de données. Cet intervalle ne doit **surtout pas** être confondu avec un intervalle de confiance, qui nous aurait permis de juger avec certitude de différences significatives dans les erreurs sur plusieurs essais. L'intervalle que nous avons décidé de représenter a surtout pour objectif de donner un indicateur simple – tout aussi imparfait soit-il – du degré de variabilité des erreurs.

1.2 Indicateurs clés dans l'analyse exploratoire

Pour chacun des jeux de données on a calculé plusieurs indicateurs qui nous aideront dans nos analyses et que l'on a préalablement résumé :

- Les matrices de covariance étant souvent assez grandes (p très large), il est difficile de déterminer si elles sont **très diagonales** ou **peu diagonales** avec notre seule observation – ce qui nous aurait permis de déterminer que les matrices sont très ou très peu indépendantes.
  Il nous a de plus semblé, après les quelques recherches que l'on a pu faire, qu'il est **très difficile** de donner un indicateur pertinent de l'indépendance des données au sein d'une classe. On a imaginé des indicateurs autour de différences de normes matricielles entre la matrice restreinte à sa diagonale et la matrice entière mais ceux-ci nous ont paru beaucoup trop aritraire – déjà car il aurait fallu décider d'une norme – et très difficiles à interpreter.

- De même, il est difficile de déterminer facilement si les matrices de covariance sont égales.
  Pour ce faire on va simplement produire plusieurs statistiques – que l'on a pu trouver dans la littérature et que l'on peut interpêter – qui pourront nous indiquer en les regardant dans leur ensemble si les matrices de covariances sont très ou peu similaires.
  On calculera le rapport des volumes formés par les matrices de covariance Vol $= |V|$ (mis au logarithme et à pondérer par la dimension dans laquelle on se trouve), le rapport des variances individuelles sommées $\sum_{i=1,\dots,p} V_{ii}$, le rapport des variances maximales $\max_{i=1,\dots,p} V_{ii}$ et enfin le rapport des précisions sommées

$\sum_{i=1,\dots,p} V^{-1}{}_{ii}$ (que l'on aura plus de mal à interprêter mais qui devrait en toute logique peut varier pour peu que les matrices soient similaires).

```r
var_1 <- var(X[z==1,])
var_2 <- var(X[z==2,])
X_1.pr <- princomp(X[z==1,])
X_2.pr <- princomp(X[z==2,])

cat("Log relative volume 1/2 : ", log(det(var_1)/det(var_2)),
    "\nDimension : ", p,
    "\nRelative average variance 1/2 : ", sum(diag(var_1))/sum(diag(var_2)),
    "\nRelative average precision 1/2 : ", sum(diag(solve(var_2)))/sum(diag(solve(var_1))),
    "\nRelative maximum variance 1/2 : ", max(X_1.pr$sdev)/max(X_2.pr$sdev))
```

Listing 1: calcul des indicateurs

	breastcancer	ionosphere	sonar	spam	spam2
Log du rapport des volumes	25.9554	2.019812	-88.82956	-57.36919	-11.98373
p (dimension de l'ellipse)	30	60	34	57	57
Rapport des variances moyenne	11.00766	0.9752217	0.4621454	6.030819	1.056974
Rapport des variances maximales	3.273212	0.9720172	1.209263	2.356161	1.145705
Rapport des précisions moyenne	0.6077293	1.420996	0.02989612	0.04248436	0.4655525

Table 1: Similarité des matrices de covariances

- Il est aussi difficile de déterminer si les variables explicatives suivent une loi normale et forment un nuage de points de forme elliptique lorsque l'espace de représentation dépasse $\mathbb{R}^2$ ou $\mathbb{R}^3$.
  Une approximation intéressante est alors de considérer des tests d'hypothèse sur la normalité de chaque variable **indépendamment** et de prendre la **moyenne** des *p-values* correspondantes. Cet indicateur est loin d'être infaillible puisqu'il nous aurait fallu construire ou trouver un test plus complet qui tiendrait compte de la non-indépendance des données, voir de la repartition des *p-values* autour de la moyenne.
  Cependant faute de mieux cela nous permettra d'associer une façon de quantifier numériquement cette notion difficile à évaluer.

	breastcancer	ionosphere	sonar	spam	spam2
p-value moyenne	0.064	0.011	0.006	7.70e-26	9.76e-54

Table 2: Calcul de la p-value moyenne pour cette classe.

  On considérera bien évidemment que les données avec une p-value elevée seront probablement mieux représentées par des modèles gaussiens.

La construction de ces données/tables a constitué une partie importante de l'analyse exploratoire et elles seront referencées tout au long du rapport.

1.3 Méthodes de discrimination

On a modélisé dans ce projet différentes méthodes pour chaque jeu de données :

- Les méthodes gaussiennes LDA, QDA, CBN pour lesquelles on a utilisé les fonctions R correspondantes de la bibliothèque *MASS*
- La régression linéaire pour laquelle on a utilisé la fonction *glm* de la bibliothèque *stats*
- Les arbres binaires pour lesquels on a utilisé la fonction *rpart* de la bibliothèque du même nom qui nous produisent un arbre binaire avec la méthode *CART* dont on a parlé en cours.

On garde en tête qu'on aurait pu mettre en place d'autres méthodes, notamment certaines dérivées des méthodes arborescentes, comme les *forêts aléatoires*.
On a décidé de se concentrer sur ce sous ensemble restreint et de mettre l'accent sur les analyses. Les codes qui nous ont servi à faire les différentes prédictions et à calculer les taux d'erreurs se trouvent en annexe.

2 Discrimination sur différents jeux

2.1 Données breastcancer

2.1.1 Analyse préliminaire

Les données breastcancer sont dans un fichier csv, avec 569 observations de 30 variables quantitatives explicatives et une variable qualitative à expliquer Z (prenant deux valeurs, 1 ou 2, correspondant à la classe réelle, donc à l'état "malade" ou "sain").

Nous avons effectué l'ACP sur ce jeu de données pour observer la façon dont les composantes principales pouvaient expliquer les données.

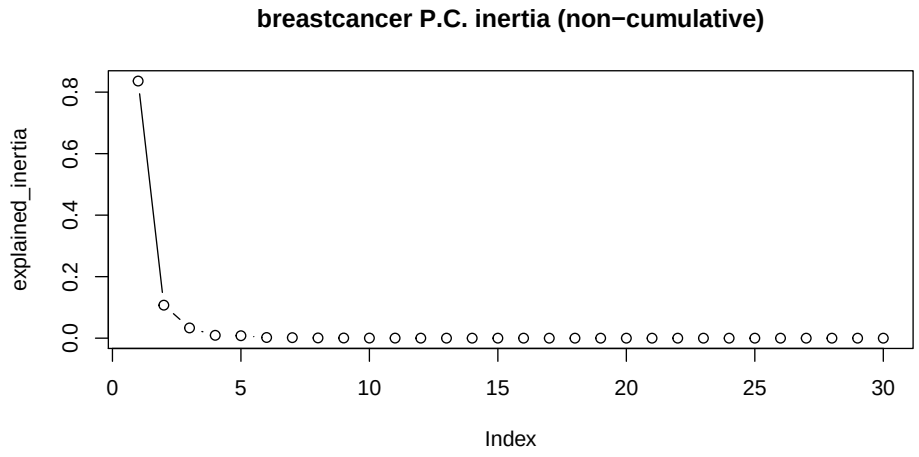

Figure 1: Inertie non-cumulée des composantes principales de breastcancer

On remarque globalement dans la figure 1 que **le premier plan de l'ACP représente très bien les données**.
On va confirmer ça en affichant la proportion d'inertie expliquée cumulée :

```
> cumsum(X.pca$sdev)/sum(X.pca$sdev)
 [1] 0.8360433 0.9433447 0.9766397 0.9859172 0.9938436 0.9960190 0.9977099 0.9984748 0.9989697
     0.9993336 0.9995567 0.9996654 0.9997360 0.9997943 0.9998400 0.9998718 0.9998961 0.9999153
     0.9999324 0.9999484 0.9999595 0.9999691 0.9999765 0.9999832 0.9999882 0.9999926 0.9999951
     0.9999972 0.9999989 1.0000000
```

Qui nous permet de dire que le premier plan de l'ACP représente $\approx 94\%$ de l'inertie. En raison de cela nous avons pensé qu'on pouvait y représenter les données et observer leurs formes. Pour cela on a décomposé en valeurs et en vecteurs propre les matrices de covariances empiriques calculées **dans le plan**, et avons affiché les ellipses associées en plus du nuage de points, grâce à un script R que l'on a mis en annexe. Résultat figure 2.

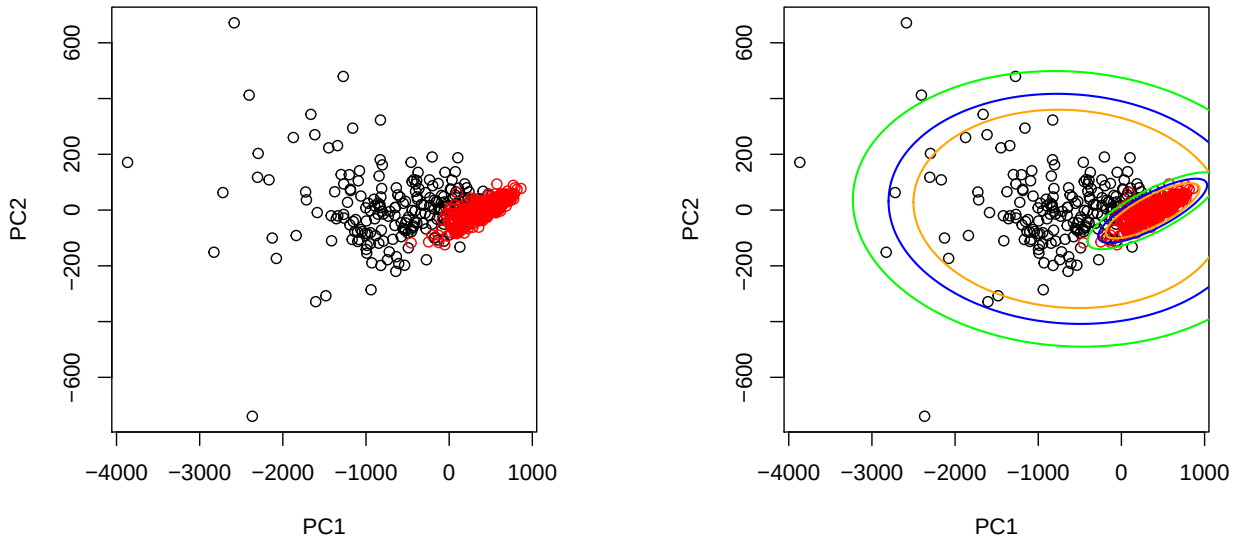

Figure 2: Ellipses de confiances (0.95,0.8,0.6) dans le premier plan de l'ACP

On observera, malgré le fait que l'ACP minimise l'inertie expliquée par les axes et pas forcément les axes qui sont les meilleurs pour la discrimination, qu'étant donné la très grande proportion ($\approx 94\%$) d'inertie expliquée par ce premier plan de projection celui-ci pourra nous fournir une visualisation intéressante de la *forme* des données. Le fait que les données des deux classes y soient aussi bien séparées nous amènent aussi à valider l'interêt de la représentation.
La matrice de covariance semble être différente dans les deux classes : que ce soit au niveau de la taille, de l'orientation, ou même de la largeur de l'ellipse.
On s'attendra donc – avec des matrices de covariances intra-classes aussi différentes – à ce que l'approximation faite par l'analyse discriminante linéaire nous donne de moins bons résultat que l'analyse discriminante quadratique. Les ellipses n'étant pas sphériques on peut aussi s'attendre à ce que le classifieur bayesien naïf fonctionne mal.
Attention : On a ici affaire à un indice et pas à une preuve formelle sur la forme des données. On sait que dans les problèmes de classification il faut toujours se méfier de la représentation dans les premiers plans de l'ACP. Dans les faits, comme on a pu le confirmer en faisant des lectures de notre côté (http://www.datavis.ca/papers/EqCov-rev2.pdf, *"Visualizing Tests for Equality of Covariance Matrices"* - Friendly & Sigal) ou avec des exemples de TD (les crabes dont le premier axe de l'ACP représentait la taille totale et dont les tailles relatives associées aux axes de moindre variances permettaient de mieux discriminer les espèces), c'est parfois les derniers axes de l'ACP qui ont le meilleur pouvoir de discrimination.
Cela ne constitue donc pas une preuve définitive que pour bien discriminer on doit avoir des formes différentes des matrices de covariance intra-classes.
On validera cependant notre interpretation graphique par le fait qu'elle aille dans le sens des statistiques qu'on avait présentées dans la table 1. On va aussi garder en tête que cela présuppose que malgré le faible nombre de données ($n = 569$) les formes des matrices de covariances ainsi estimées soient représentatives et soient bien estimées lors d'un apprentissage.

2.1.2 Apprentissage

Nous appliquons ensuite les méthodes de discrimination et régression, afin de déterminer laquelle minimise l'erreur d'affectation moyenne pour N partitions de l'ensemble d'apprentissage en un sous-ensemble d'apprentissage et un ensemble de validation.

Nous avons obtenu pour les différentes méthodes les résultats suivants :

Methode	Taux d'erreur moyen	intervalle
LDA	0.045	[0;0.089]
QDA	0.044	[0.009;0.08]
Classifieur bayésien	0.046	[0;0.088]
Régression logistique	0.060	[0.024;0.111]
Arbre binaire	0.072	[0.034,0.139]

Table 3: Résultats sur les données breastcancer.

Nous pouvons immédiatement remarquer que les méthodes LDA, QDA et CBN qui supposent que les données suivent une loi normale ont une bonne précision moyenne de l'ordre de 96%, ce qui est en accord avec les tests qu'on avait effectué en début de rapport (Table 2) sur la normalité de la loi.
Nous voyons que la QDA qui ne présuppose ni l'indépendance des variables au sein d'une classe (classes sphériques) ni l'homogéneité des variances pour les différentes classes (classes de formes similaires) obtient (de peu) les meilleurs résultats, semblant les hypothèes qui ont été faites plus haut.
La methode du classifieur bayésien naïf supposant l'indépendance des variables au sein d'une classe ne fonctionne qu'à peine moins bien malgré le fait que ladite indépendance soit remise en cause par nos indicateurs. On peut expliquer cela par le faible nombre de paramètres que fait intervenir la méthode CBN dans un contexte où $n = 569$ est relativement petit, $p = 30$ est assez grand relativement à n, et où l'overfitting peut très facilement s'avérer problématique.
On peut dire la même chose de la LDA qui fonctionne relativement bien malgré l'hypothèse qu'elle fait sur l'homogénéité des matrices de covariance intra-classe, certainement car elle demande elle aussi d'estimer moins de paramètres que la méthode QDA.
Il est aussi intéressant de noter que la régression logistique, qui présente une région de décision similaire à celle de l'analyse discriminante linéaire et propose un modèle simple n'optimisant qu'un nombre restreint de paramètres a des performances significativements moindres que toutes les méthodes discriminantes gaussiennes. On interpretera cela comme une preuve supplémentaire de la normalité du jeu de données.

2.2 Ionosphere

2.2.1 Analyse préliminaire

Le jeu de données Ionosphère contient 350 observations de 36 variables.
Parmis celles-ci on éliminera la variable *X2*, de valeur constante 0 pour tous les individus
Reste alors la variable à expliquer Z binaire, et 33 variables expliquatives :

- 32 variables quantitatives : *X3, X4, ..., X32, X33*
- La variable binaire *X1*

On peut essayer de transformer les variables explicatives en composantes principales pour observer d'éventuelles redondances dans les données.
Observons les variables expliquées par les composantes principales successives grâce à la fonction R *prcomp*.

```
> X.pca$sdev/sum(X.pca$sdev)
 [1] 0.11877028 0.07425647 0.05798326 0.05581950 0.04685454 0.04065639 0.03676863 0.03597682
     0.03483535 0.03183029 0.03048192 0.02877639 0.02800249 0.02722180 0.02549658 0.02508502
     0.02400563 0.02310232 0.02258375 0.02214952 0.02130926 0.02069154 0.01873876 0.01803771
     0.01771256 0.01672998 0.01610761 0.01528823 0.01479093 0.01390543 0.01352900 0.01231553
     0.01018649
> cumsum(X.pca$sdev)/sum(X.pca$sdev)
 [1] 0.1187703 0.1930268 0.2510100 0.3068295 0.3536841 0.3943404 0.4311091 0.4670859 0.5019212
     0.5337515 0.5642335 0.5930098 0.6210123 0.6482341 0.6737307 0.6988157 0.7228214 0.7459237
     0.7685074 0.7906570 0.8119662 0.8326578 0.8513965 0.8694342 0.8871468 0.9038768 0.9199844
     0.9352726 0.9500636 0.9639690 0.9774980 0.9898135 1.0000000
```

On remarque immédiatement qu'il y a peu de redondance dans les données – la décroissance des valeurs propres et de l'inertie expliquée étant assez faible. On peut donc abandonner l'idée d'utiliser l'ACP pour ce jeu de données. Cela nous aurait pourtant probablement été utile étant donné que le nombre de variable explicatives est assez grand comparé au faible nombre de données.
On peut s'attendre à ce que les modèles les plus simpes fonctionnent le mieux, et on essaiera de faire une sélection de variable rudimentaire par régression logistique.

2.2.2 Apprentissage

Methode	Taux d'erreur moyen	Intervalle
LDA	0.131	[0.062;0.219]
QDA	0.133	[0.028;0.217]
Classifieur bayésien	0.099	[0.029;0.157]
Classifieur bayésien (avec réduction de variable)	0.081	[0.014;0.185]
Régression logistique	0.132	[0.047; 0.215]
Arbre binaire	0.121	[0.048,0.212]

Table 4: Résultats sur les données Ionosphere.

On a noté uniquement les résultats du classifieur bayésien pour la réduction de variable car c'était la méthode pour laquelle les résultats d'erreurs changeaient le plus et étaient les plus intéressants.

Pour la QDA, nous avons eu un problème causé par la variable explicative binaire *X1*.
En effet **celle-ci a pour valeur 1 dans toutes les observations de la classe** $\{Z = 1\}$. En raison de cela il n'y a pas de variance intra-classe pour la variable *X1* et la classe $\{Z = 1\}$ et la matrice de covariance associée $\Sigma_{\{Z=1\}}$ n'est pas de plein rang et ne peut pas être inversée !
Il faut donc faire attention lorsqu'on inclus à l'analyse discriminante une variable quantitative à ce que celle-ci n'ait pas toujours la même valeur au sein d'une classe particulière, ce qui peut arriver si on a peu d'observations ou que la variable en question est répartie d'une certaine manière.
On peut aussi se questionner sur la pertinence de la décision d'inclure une variable quantitative à l'analyse discriminante, qui repose sur l'hypothèse que les variables explicatives suivent une loi gausienne multi-dimensionnelle (c'est un problème dont on reparlera à la fin du rapport).
La méthode qui s'avère être la plus efficace sans sélection de variable pour ce jeu de données est le classifieur bayésien naïf.
La *p-value* moyenne de 0.011 nous indique en effet que les variables semblent assez proche d'une gaussienne (comparativement aux autres données, et par rapport à la signification d'une p-value elevée : il est impossible de rejetter la normalité avec un degré de confiance supérieur à 1 moins celle-ci).
Outre les hypothèses, on retiendra surtout que c'est probablement la **simplicité** du modèle CBN relativement à tous les autres modèles qui l'ont rendue particulièrement efficace sur ce jeu de données où l'apprentissage peut être difficile.

Enfin, évoquons la sélection de variable rudimentaire que nous avons fait.
On s'est basé – dans cette expérience à vocation exploratoire – sur un indicateur que l'on sait biaisé ; la valeur absolue du coefficient de chaque variable dans la régression logistique.
On a simplement décidé de retirer du modèle les variables dont la valeur absolue du coefficient rapportée à la valeur absolue du coefficient maximal (en excluant l'intercept) est inférieure à un seuil s donné.
Le seuil $s = \frac{1}{4}$ est celui qui donnait les meilleurs résultats et que l'on a sélectionné dans notre étude.
A noter : Il faut cependant se rappeler qu'on a créé un biais optimiste dans le taux d'erreur en essayant sur l'ensemble du jeu plusieurs valeurs de seuil pour voir celles qui fonctionnaient bien.
Cet exercice avait simplement pour but de montrer qu'un tel jeu de donné peut très facilement profiter d'une simplification de la sorte était donné qu'il est composé de beaucoup de variables pour peu de données et donc très sensible à l'overfitting.

2.3 Sonar

Le jeu de données Sonar contient 208 observations de 60 variables quantitatives et une variable qualitative Z correspondant à la classe (1 ou 2).
Encore plus que pour les jeux de données précedent, le nombre d'observation est très très faible relativement au nombre de variable : on aura probablement de très mauvais résultats sur des modèles complexes et de mauvais résultats sur les modèles simples. D'après la table 1, les matrices de covariance dans les deux classes sont similaires, on peut s'attendre à ce que les méthodes ayant comme hypothèse la normalité avec égalité des variances (lda) donnent de bons résultats.
Observons les résultats de l'ACP sur ce jeu de données

```
> X.pca$sdev/sum(X.pca$sdev)
 [1] 0.1265595483 0.1010532946 0.0654706424 0.0568865059 0.0508645802 0.0472241544 0.0459132419
     0.0363172136 0.0334250283 0.0310292899 0.0280037259 0.0260957036 0.0242122478 0.0218940763
     0.0209172037 0.0195961303 0.0189851148 0.0168332717 0.0162536318 0.0156379162 0.0135393986
     0.0121239363 0.0115855154 0.0109651695 0.0107819763 0.0104000890 0.0088510771 0.0083897554
     0.0080632357 0.0077019682 0.0069610345 0.0067494441 0.0061754833 0.0060213593 0.0059219821
     0.0057130999 0.0052390591 0.0051297051 0.0049421594 0.0045001068 0.0044457244 0.0042643461
     0.0037117480 0.0034199688 0.0031059427 0.0027284893 0.0021426887 0.0019855425 0.0018866689
     0.0015190486 0.0012622794 0.0011153365 0.0009643132 0.0008706099 0.0007841565 0.0007305274
     0.0006269761 0.0005816209 0.0005179426 0.0004079929
> cumsum(X.pca$sdev)/sum(X.pca$sdev)
 [1] 0.1265595 0.2276128 0.2930835 0.3499700 0.4008346 0.4480587 0.4939720 0.5302892 0.5637142
     0.5947435 0.6227472 0.6488429 0.6730552 0.6949493 0.7158665 0.7354626 0.7544477 0.7712810
     0.7875346 0.8031725 0.8167119 0.8288359 0.8404214 0.8513865 0.8621685 0.8725686 0.8814197
     0.8898094 0.8978727 0.9055746 0.9125357 0.9192851 0.9254606 0.9314820 0.9374039 0.9431170
     0.9483561 0.9534858 0.9584280 0.9629281 0.9673738 0.9716381 0.9753499 0.9787699 0.9818758
     0.9846043 0.9867470 0.9887325 0.9906192 0.9921382 0.9934005 0.9945159 0.9954802 0.9963508
     0.9971349 0.9978655 0.9984924 0.9990741 0.9995920 1.0000000
```

Methode	Taux d'erreur moyen	Intervalle
LDA	0.268	[0.170;0.439]
QDA	0.293	[0.121;0.488]
Classifieur bayésien	0.279	[0.170;0.463]
Régression logistique	0.269	[0.167; 0.390]
Arbre binaire	0.287	[0.095,0.52]

Table 5: Résultats sur les données Sonar.

Nous observons que pour le jeu de données Sonar, toutes les méthodes semblent obtenir des résultats de prédiction similaires, tous relativement mauvais.
On peut comprendre cela à travers le nombre de données beaucoup trop faible par rapport au nombre de variable.

En raison de cela nous avons décidé d'implémenter certains modèles dans un plan factoriel de l'ACP de dimension bien choisie (ici on prendra $p = 20$ qui semblait nous donner un rapport entre les tailles p et n *"raisonnable"*, et dont le pourcentage d'inertie expliquée approche 80%). On observe les performances suivantes :

Methode	Taux d'erreur moyen	Intervalle
LDA	0.249	[0.121;0.414]
QDA	0.217	[0.122;0.341]
Classifieur bayésien	0.273	[0.146;0.439]

Table 6: Résultats sur les données Sonar avec ACP.

On réalise que la LDA et la QDA obtiennent de bien meilleurs performances que le modèle CBN dont les performances évoluent peu.
On peut pourtant penser que même si on restreint les données aux 20 premières composantes principales on aura probablement toujours une représentation très sensible à l'overfitting étant donné la faible quantité de données (208).
On attribuera les performances supérieures de la QDA à la justesse du modèle. En effet comme on l'a vu dans la table 1, le jeu de données est celui pour lequel les matrices de covariances semblent le plus dissimilaires. Cela est d'autant plus vrai qu'on s'attend normalement à ce que la QDA fonctionne particulièrement mal sur un nombre aussi restreint de données.
Les performances relativement bonnes de la LDA seront attribuées au nombre moindre de paramètres qui y interviennent. Enfin, on peut s'interroger sur les mauvaises performances du modèle CBN par rapport à la LDA – qui elle même repose pourtant sur des hypothèses qui semblent fausse. On s'attendrait en effet à ce que le CBN fonctionne plutôt bien dans un jeu de données sensible à l'overfitting. On attribuera donc la mauvaise performance au non respect des hypothèses faites par le modèle : les variables sont probablement fortement corrélées au sein d'une même classe.

2.4 Spam

Le jeu de données Spam comprend 4601 observations avec 57 variables quantitatives correspondant à la présence de mots biens choisis dans un e-mail et une variable qualitative Z spécifiant si l'email en question correspond à un Spam.

Pour les données Spam et Spam2, nous utilisons directement la méthode QDA de R avec cross validation.

Methode	Taux d'erreur moyen	Intervalle
LDA	0.111	[0.094;0.131]
QDA	0.169	
Classifieur bayésien	0.264	[0.234;0.300]
Régression logistique	0.073	[0.065;0.089]
Arbre binaire	0.103	[0.086;0.123]

Table 7: Résultats sur les données Spam.

La méthode donnant le meilleur taux d'erreur est la méthode de la régression logistique, suivi par la méthode de l'arbre binaire.
Ces deux méthodes ont en commun d'à la fois reposer sur des modèles assez simples mais aussi de **ne pas supposer de normalité** dans le jeu de données. On avait en effet vu dans la table 2 que la *p-value* moyenne du test de normalité pour les différentes variables explicatives était très faible.
En outre, parmi les méthodes discriminantes gaussiennes on observe que le classifieur Bayésien naïf fonctionne très mal relativement à la QDA et la LDA. On peut attribuer cela au fait que le modèle, moins souple et moins complexe, est moins adapté au jeu de données relativement grand ($N = 4601$), mais surtout au fait que les variables explicatives sont fortement correlées. En effet les variables explicatives représentant la présence de mots biens choisis dans le message il est facile de voir en quoi celles-ci pourraient interragir à travers de nombreux facteurs externes comme le sujet du mail, le ton employé, ...
Enfin on observe que la LDA fonctionne mieux que la QDA malgré le fait que l'on ait un nombre d'observations relativement grand et que la QDA représente un modèle plus complexe. On attribuera cela à deux facteurs. En premier lieu on supposera que la LDA étant plus simple elle a potentiellement un meilleur pouvoir de généralisation en dehors des hypothèses de normalité – visiblement violeés par ce jeu de données – que la QDA. Enfin on observera dans la table 1 plusieurs indicateurs nous laissant penser que les matrices de covariances sont assez similaires et que l'hypothèse faite par la LDA est relativement bonne.

3 Analyse discriminante de données binaires

3.1 Modèle

Soit Z une variable discrète à expliquer à valeurs dans $\{w_1, ..., w_g\}$.
Et soient $X^j_{j\in\{1,...,p\}} \in \{0, 1\}$ les p variables explicatives binaires associées.
On modélise les propabilités conditionnelles **a priori** par des proportions p_{kj} fixées telles que :

$$\mathbb{P}\left(X^j = 1 | Z = w_k\right) = p_{kj}$$

À partir de là on va chercher à modéliser la probabilité **jointe** $\mathbb{P}\left(X = \overrightarrow{x}, Z = w_k\right)$ pour enfin déterminer la probabilité **a posteriori** $\mathbb{P}\left(Z = w_k | X = \overrightarrow{x}\right)$. En calculant ces valeurs avec leurs estimateurs de maximum de vraisemblance on appliquera la règle de Bayes.
On calcule les estimateurs qui maximisent la vraisemblance en annulant les dérivées partielles :

$$\widehat{p_{kj}} = \frac{1}{n_k} \sum_{i=1}^{n} z_{ik} x_{ij}$$

$$\widehat{\pi_k} = \frac{n_k}{n}$$

On peut retrouver les calculs en annexes et les résultats respectivement dans les équations (3) et (2).

3.2 Programmation

La fonction binaryNBCfit permet de faire l'apprentissage sur les données Spam avec le modèle décrit ci dessus. Elle prend comme arguments le tableau individus-variables Xtrain et le vecteur d'indicateurs de classe Ztrain, et retourne les paramètres du modèle :

- Le vecteur des probabilités a priori p_{ik} :
  C'est une matrice $\mathbb{M}_{p,g}$ dont chaque élement correspond à la probabilité pour $\vec{X}^{(i)}$ la i-ème composante de $\vec{X}$ d'être dans la classe k.

- Le vecteur des paramètres π_k :
  Cela renvoie une vecteur dans $\mathbb{R}^g$ correspondant à la probabilité *a priori* d'être dans chacune des g classes.

On a essayé de programmer le calcul de ces paramètres par les formules qu'on a déterminé dans la partie précedente

On a en outre créé une fonction binaryNBCVal qui avec les paramètres que l'on vient d'estimer retourne pour chaque X la classe pour laquelle la probabilité *a priori* est maximale.
Pour se faire on va utiliser la formule

$$\mathbb{P}\left(Z = \vec{z} | X = \vec{x}\right) = \frac{\mathbb{P}\left(X = \vec{x}, Z = \vec{z}\right)}{\mathbb{P}\left(X = \vec{x}\right)}$$

Et ne calculer avec R que le numérateur qu'on a déjà défini dans l'équation (1) présentée en annexe, le dénominateur étant constant selon $\vec{z}$
Les codes R seront eux-aussi mis en annexe.

4 Donnée spam base 2

Les données Spam2 sont les mêmes que les données Spam à la différence que celles ci ont été simplifiées pour être codées comme des variables binaires – probablement sur la base d'un seuil. Il y a ainsi une simplification et donc une perte d'information qu'il est important de ne pas négliger.
De plus, de part leur traitement particulier on peut s'attendre à ce que les données aient une forme de régularité bien à elles dont il s'agira de tenir compte.
Après l'ACP, les deux premières composantes principales ne représentent plus que $\approx 12.5\%$ de l'information. Cependant, nous représentons les données dans le premier plan factoriel afin d'avoir une idée de leur répartition.

```
> cumsum(X.pr$sdev)/sum(X.pr$sdev)
 [1] 0.07550789 0.12475239 0.15787909 0.18869147 0.21740500 0.24429763 0.26909353 0.29302280
     0.31602593 0.33879707 0.36107990 0.38243058 0.40345699 0.42394639 0.44421039 0.46386303
     0.48313668 0.50212413 0.52072216 0.53888974 0.55698784 0.57496903 0.59245380 0.60962640
     0.62638672 0.64301009 0.65936459 0.67534868 0.69121685 0.70689674 0.72209301 0.73698207
     0.75161225 0.76608266 0.78025830 0.79437321 0.80807554 0.82162151 0.83490703 0.84787478
     0.86047603 0.87272829 0.88446589 0.89604039 0.90698719 0.91786648 0.92803807 0.93771369
     0.94655031 0.95519448 0.96352866 0.97144379 0.97869792 0.98577050 0.99204195 0.99757812
     1.00000000
```

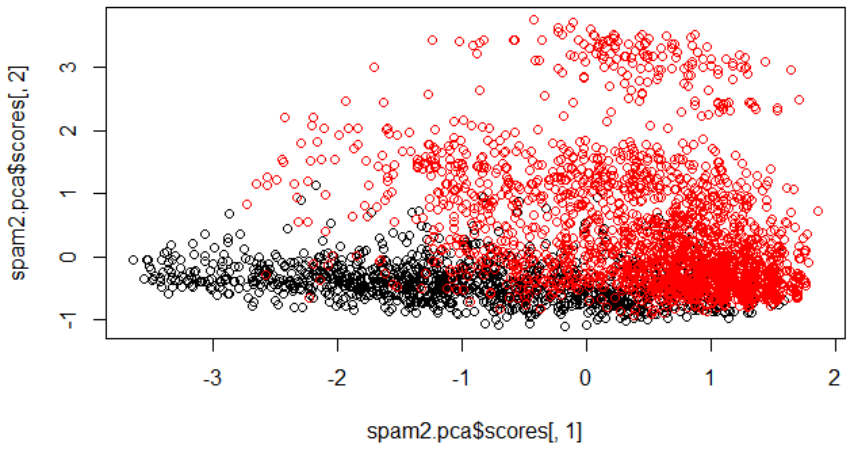

Figure 3: Données Spam2 représentées dans le premier plan factoriel

Nous voyons que les données déjà sont assez différenciables graphiquement dans ce plan qui ne représente que $\approx 12.5\%$ de l'information selon le critère de l'ACP qui n'optimise en plus pas nécessairement la capacité de discrimination. On peut donc facilement imaginer que le jeu de données est d'autant plus facile à discriminer.
Observons donc les résultats des différentes méthodes sur les données *spam2* présentées dans la table 8.

Methode	Taux d'erreur moyen	Intervalle
LDA	0.076	[0.056;0.097]
QDA avec perturbation	0.130	
Classifieur bayésien	0.061	[0.046;0.082]
Régression logistique	0.062	[0.047; 0.079]
Arbre binaire	0.121	[0.097;0.135]
Analyse discriminante *maison*	0.116	

Table 8: Résultats sur les données Spam2.

On notera en premier lieu que l'analyse discriminante que nous venons de conceptualiser et d'implémenter, et qui semble très adaptée à la forme particulière des données binaires fonctionne moins bien que la LDA et que le classifieur bayésien naïf. Cela nous a semblé contre-intuitif au premier abord étant donné qu'il parrait difficile de représenter des données dans $\{0,1\}$ selon une loi continue telle que la loi multinormale.
On peut cependant apporter plusieurs élements pour tenter de l'expliquer.

En premier lieu il est important de noter que la LDA – contrairement au CBN et à notre analyse discriminante – ne fait pas l'hypothèse que les variables explicatives sont indépendantes au sein d'une même classe. Cela pourrait expliquer pourquoi elle fonctionne mieux malgré le fait qu'il est difficile d'imaginer comment l'hypothèse de normalité des données peut être respectée. Il n'est en effet pas difficile à imaginer que la présence des différents mots dans le contenu d'un mail possède de **très fortes correlations** suivant les mots choisis.

Ce qui parait cependant plus étrange, c'est que le CBN fonctionne non seulement mieux que la LDA mais aussi mieux que notre analyse discriminante alors qu'il fait la même hypothèse d'indépendance des variables explicatives. Pire encore les deux méthodes estiment un nombre assez similaire de paramètre avec respectivement $n_{\text{binom}}^{\text{param}} = (p+1) \times g - 1$ et $n_{\text{CBN}}^{\text{param}} = (2p+1) \times g - 1$.
Or si l'on admet que les variables explicatives sont indépendantes au sein d'une même classe, celles-ci ne pouvant valoir que 0 ou 1 y suivent **nécessairement** une loi de Bernouilli d'un paramètre p_{ik} quelconque comme dans notre modèle. Selon ces assomptions, l'hypothèse de normalité devrait nécessairement être rejetée et l'hypothèse

d'une loi de Bernouilli nécessairement admise.
Notre interpretation de la chose est que **l'indépendance des variables explicatives au sein d'une classe n'est en fait pas une bonne hypothèse**, mais que l'analyse Bayesinne naïve – contrairement à notre analyse discriminante – fonctionne quand même. Rien n'indique en effet qu'elle ne doive pas fonctionner en dehors des hypothèses précises pour lesquelles elle a été définie et il n'est pas impossible qu'on assite là au pouvoir de généralisation du modèle en dehors du cadre théorique qu'on lui a donné.
Il est aussi important de mettre cela en lien avec le fait que les données ont été acquises selon un procédé de transformation bien particulier.

Autrement, on peut expliquer que la QDA fonctionne mal par le fait qu'on est obligé de calculer des matrices de covariances sur des sous-ensembles restreints du jeu de données, alors qu'en outre le fait d'avoir des valeurs dans $\{0, 1\}$ fait qu'on a **beaucoup plus facilement** de la redondance dans la matrice des variables explicatives et dans la matrice de covariance associée. On a été contraint de rajouter une perturbation aléatoire normale de $\sigma = 10^{-8}$ sur un dixième des données pour avoir des estimations de la matrice de covariance qui soient toujours de plein rang, mais il est facile d'imaginer que cette astuce ne fasse que rendre compte du problème plus général qui est qu'il est trop difficile d'obtenir des matrices de covariance qui ont du sens sur un sous ensemble restreint du jeu de données alors même que le caractère discret des valeurs crée facilement de la redondance.

Il semble aussi logique que la régression linéaire fonctionne si bien car les données étaient déjà linéarement séparales dans spam et le sont apparement d'autant plus maintenant qu'elles ont été *"rendues plus extrêmes"* en passant d'une plage dans $[0; 1]$ aux valeurs des bornes $\{0, 1\}$.

On notera aussi que l'estimation de l'erreur du meilleur modèle pour les données spam2 (0.061) est plus faible que l'estimation de l'erreur du meilleur modèle pour les données spam (0.073) : la simplification du jeu de données qui peut sembler naïve a donc été fructueuse et a étrangement porté ses fruits lorsqu'on a appliqué des méthodes continues à ces données discrètes.
Attention : C'est là qu'intervient la limite relative au fait de ne pas utiliser un jeu de test séparé sur lequel on ne fait pas de sélection de modèle : la performance du meilleur modèle donne une estimation biaisée optimiste de l'erreur du meilleur modèle.

Annexes A: démonstration mathématique

Déterminer $\mathbb{P}\left(X^j = x_j | Z = w_k\right)$

On part de

$$\begin{cases} \mathbb{P}\left(X^j = 1 | Z = w_k\right) & = p_{kj} \\ \mathbb{P}\left(X^j = 0 | Z = w_k\right) & = 1 - p_{kj} \end{cases}$$

Et on en déduit

$$\mathbb{P}\left(X^j = x_j | Z = w_k\right) = p_{kj}^{x_j}(1 - p_{kj})^{1-x_j}$$

Calculer $\mathbb{P}\left(X = \vec{x} | Z = w_k\right)$

$$\begin{array}{rcl} \mathbb{P}\left(X = \vec{x} | Z = w_k\right) & = & \mathbb{P}\left(X^1 = x_1, ..., X^p = x_p | Z = w_k\right) \\ & = & \prod_{i=1}^{p} \mathbb{P}\left(X^i = x_i | Z = w_k\right) \\ & = & \prod_{i=1}^{p} p_{ki}^{x_i}(1 - p_{ki})^{1-x_i} \end{array}$$

Calculer $\mathbb{P}\left(X = \vec{x}_i, Z = \vec{z}_i\right)$ pour $\left(\vec{x}_i, \vec{z}_i\right)_{i=1,...,n}$

En considérant

$$\vec{z}^{(j)} = \begin{cases} 0 & \text{si } Z \neq w_j \\ 1 & \text{si } Z = w_j \end{cases}$$

et

$$\pi_i = \mathbb{P}\left(Z = \vec{z}_i\right)$$

$$\mathbb{P}\left(X = \vec{x}_i, Z = \vec{z}_i\right) = \prod_{k=1}^{g} \left[\pi_k \prod_{j=1}^{p} \left(p_{kj}^{xj} \left(1 - p_{kj}\right)^{1-x_{ij}} \right) \right]^{z_{ik}} \tag{1}$$

Calculer la vraisemblance $L\left(\left(\vec{x}_i, \vec{z}_i\right)_{i=1,...,n}; \vec{\theta}\right)$

En considérant $\vec{\theta} = \left(\left(p_{ik}\right)_{\substack{i=1,...,n \\ k=1,...,g}}; \pi_k \right)$

$$\begin{array}{rcl} L\left(\left(\vec{x}_i, \vec{z}_i\right)_{i=1,...,n}; \vec{\theta}\right) & = & \prod_{i=1}^{n} \mathbb{P}_{\vec{\theta}}\left(X = \vec{x}_i, Z = \vec{z}_i\right) \\ & = & \prod_{i=1}^{n} \prod_{k=1}^{g} \left[\pi_k \prod_{j=1}^{p} \left(p_{kj}^{x_j} \left(1 - p_{kj}\right)^{1-x_{ij}} \right) \right]^{z_{ik}} \\ \ln L\left(\left(\vec{x}_i, \vec{z}_i\right)_{i=1,...,n}; \vec{\theta}\right) & = & \sum_{i=1}^{n} \sum_{k=1}^{g} z_{ik} \left[\ln\left(\pi_k\right) + \sum_{j=1}^{p} \left(x_j \ln\left(p_{kj}\right) + \left(1 - x_{ij}\right) \ln\left(1 - p_{kj}\right) \right) \right] \end{array}$$

Trouver les estimateurs $\widehat{p_{kj}}$ et $\widehat{\pi_k}$ qui minimisent L et $\ln L$

On cherche $\vec{\theta}_{min} = \left(\left(\widehat{p_{kj}}\right), \left(\widehat{\pi_k}\right) \right)$ qui minimise $\ln L\left(\left(\vec{x}_i, \vec{z}_i\right), \vec{\theta}\right)$. On cherche à optimiser :

$$\begin{cases} \min\left(\ln L\left(\vec{\theta}\right)\right) \\ \sum_{k=1}^{g} \pi_k = 1 \end{cases}$$

Dans lesquelles on écrit les contraintes qui ne sont pas des égalités strictes et qui peuvent se traduire par le problème d'optimisation non contraint suivant en intégrant un multiplicateur de lagrange.

$$\begin{cases} \min\left(J\left(\vec{\theta}\right)\right) \\ J(\vec{\theta}) = \ln L(\vec{\theta}) - \lambda\left(\sum_{k=1}^{g} \pi_k - 1\right) \quad \lambda \in \mathbb{R} \end{cases}$$

On va donc chercher $\vec{\theta}_* = \left(\left(\widehat{p_{ik}}\right)_{\substack{i=1,...,n \\ k=1,...,g}}; \widehat{\pi_k} \right)$ tel que $\frac{\partial J}{\partial \vec{\theta}}\left(\vec{\theta}_*\right) = \vec{0}$

Dérivons tout d'abord par un p_{kj} quelconque en supposant les proportions non triviales différentes de 0 et 1:

$$\begin{array}{rcl} \frac{\partial J(\vec{\theta})}{\partial p_{kj}} & = & \sum_{i=1}^{n} z_{ik} \left[x_{ij} \frac{1}{p_{kj}} - \left(1 - x_{ij}\right) \frac{1}{1 - p_{kj}} \right] \\ & = & \frac{1}{p_{kj}\left(1 - p_{kj}\right)} \left[\left(1 - p_{kj}\right) \sum_{i=1}^{n} z_{ik} x_{ij} - p_{kj} \sum_{i=1}^{n} z_{ik} \left(1 - x_{ij}\right) \right] \end{array}$$

On a donc $\frac{\partial J(\vec{\theta})}{\partial p_{kj}} = 0$ si et seulement si

$$\left[(1-p_{kj})\sum_{i=1}^{n} z_{ik}x_{ij} - p_{kj}\sum_{i=1}^{n} z_{ik}\left(1-x_{ij}\right)\right] = 0$$

$$\sum_{i=1}^{n} z_{ik}\left[x_{ij} - p_{kj}x_{ij} - p_{kj} + p_{kj}x_{ij}\right] = 0$$

$$\sum_{i=1}^{n} z_{ik}p_{kj} = \sum_{i=1}^{n} z_{ik}x_{ij}$$

Or

$$\sum_{i=1}^{n} z_{ik}p_{kj} = p_{kj}\sum_{i=1}^{n} z_{ik} = p_{kj}n_k$$

Donc

$$\widehat{p_{kj}} = \frac{1}{n_k}\sum_{i=1}^{n} z_{ik}x_{ij} \tag{2}$$

L'unique valeur de p_{kj} qui annule la dérivée.
Essayons maintenant de dériver selon λ le multiplicateur de lagrange :

$$\frac{\partial J\left(\vec{\theta}\right)}{\partial \lambda} = -\left(\sum_{k=1}^{g}\pi_k - 1\right)$$

On a donc $\frac{\partial J(\vec{\theta_*})}{\partial \lambda} = 0$ si et seulement

$$\left(\sum_{k=1}^{g}\pi_k - 1\right)$$

Dérivons alors selon un quelconque π_k

$$\frac{\partial J\left(\vec{\theta}\right)}{\partial \pi_k} = \frac{1}{\pi_k}\sum_{i=1}^{n} z_{ik} - \lambda$$

Celle ci s'annule lorsque

$$\begin{aligned}
\lambda &= \tfrac{1}{\pi_k}\textstyle\sum_{i=1}^{n} z_{ik} \\
&= \tfrac{n_k}{\pi_k} \\
\pi_k &= \tfrac{n_k}{\lambda}
\end{aligned}$$

Or

$$\begin{aligned}
\textstyle\sum_{k=1}^{g}\pi_k &= 1 \\
\textstyle\sum_{k=1}^{g}\tfrac{n_k}{\lambda} &= 1 \\
\lambda &= \textstyle\sum_{k=1}^{g} n_k \\
\lambda &= n
\end{aligned}$$

On a donc une annulation uniquement en

$$\widehat{\pi_k} = \frac{n_k}{n} \tag{3}$$

Annexes B: codes intéressants

Apprentissage des paramètres $\widehat{\pi_k}$ et $\widehat{p_{kj}}$ pour notre analyse discriminante

```
binaryNBCfit <- function(X,z)
{
  n <- dim(X)[1]
  p <- dim(X)[2]
  z <- as.factor(z)
  z_levels <- levels(z)
  g <- nlevels(z)

  fit <- data.frame(prop=rep(0,g))
  fit$prob <- matrix(nrow=g,ncol=p)
  for(k in 1:g)
  {
    fit$prop[k] <- sum(z==z_levels[k]) / n
    fit$prob[k,] <- colMeans(X[z==z_levels[k],])
  }
  return(fit)
}
```

Listing 2: Apprentissage des paramètres

Prédiction par la règle de Bayes pour notre analyse discriminante

```
binaryNBCval <- function(X, fit)
{
  n <- dim(X)[1]
  p <- dim(X)[2]
  g <- length(fit$prop)

  scores <- matrix(nrow = n, ncol = g)

  for(k in 1:g)
  {
    M1 <- t(log(fit$prob[k,]^t(X)))
    term1 <- apply(as.matrix(M1),1,sum)
    M2 <- t(log((1-fit$prob[k,])^t(1-X)))
    term2 <- apply(as.matrix(M2),1,sum)
    scores[,k] <- (term1+term2+log(fit$prop[k]))
  }
  predict <- max.col(scores)
  return(predict)
}
```

Listing 3: Prédiction pour notre analyse discriminante

Inertie des composantes principales

```
X.pca <- prcomp(X)
explained_inertia <- X.pca$sdev/sum(X.pca$sdev)
explained_inertia.cumul <- cumsum(explained_inertia)
plot(explained_inertia.cumul, type='b',main='breastcancer P.C. inertia (cumulative)')
plot(explained_inertia, type='b',main='breastcancer P.C. inertia (non-cumulative)')
```

Listing 4: Afficher l'inertie des composantes principales

Afficher les ellipses de confiance

```
# http://www.visiondummy.com/2014/04/draw-error-ellipse-representing-covariance-matrix/
# https://stats.stackexchange.com/questions/9898/how-to-plot-an-ellipse-from-eigenvalues-and-
    eigenvectors-in-r
draw_ellipse <- function(cov, mu, add, confidence=0.95, col="green")
{
  angles <- seq(0, 2*pi, length.out=200)

  eigVal  <- eigen(cov)$values
  eigVec  <- eigen(cov)$vectors
  eigScl  <- eigVec %*% diag(sqrt(eigVal))
  xMat    <- rbind(mu[1] + eigScl[1, ], mu[1] - eigScl[1, ])
  yMat    <- rbind(mu[2] + eigScl[2, ], mu[2] - eigScl[2, ])
  coef = sqrt(qchisq(confidence, df=7))
  ellBase <- cbind(sqrt(eigVal[1])*coef*cos(angles),
                   sqrt(eigVal[2])*coef*sin(angles))
  ellRot  <- eigVec %*% t(ellBase)
  if(!add)
  {
    plot((ellRot+mu)[1, ], (ellRot+mu)[2, ], asp=1, type="l", lwd=1.5, col=col)
  }
  else
  {
```

```
    lines((ellRot+mu)[1, ], (ellRot+mu)[2, ], asp=1, lwd=1.5, col=col)
  }
}

X.pca <- prcomp(X)
X.proj <- X.pca$x[,1:2]
mu_1 <- colMeans(X.proj[z==1,])
var_1 <- var(X.proj[z==1,])
mu_2 <- colMeans(X.proj[z==2,])
var_2 <- var(X.proj[z==2,])

plot(X.proj, col=z)
draw_ellipse(var_1, mu_1, add=T, confidence=0.95)
draw_ellipse(var_1, mu_1, add=T, confidence=0.8, col="blue")
draw_ellipse(var_1, mu_1, add=T, confidence=0.6, col="orange")
draw_ellipse(var_2, mu_2, add=T, confidence=0.95)
draw_ellipse(var_2, mu_2, add=T, confidence=0.8, col="blue")
draw_ellipse(var_2, mu_2, add=T, confidence=0.6, col="orange")
```

Listing 5: Afficher les ellipses de confiance dans le premier plan de l'ACP

Algorithme de test des méthodes

```
require(MASS)
require(ggplot2)
library(caret)
library(e1071)
library(rpart)

#exemple de calcul du degre de correlation des donnees Spam
mean(abs(cor(spam[,-1])))

#exemple de calcul de similitude des matrices de variance sur les donnees Spam2
spam2.Z1 <- spam2[spam2$Z==1,]
spam2.Z2 <- spam2[spam2$Z==2,]
cov1 <- cov(spam2.Z1[,-c(1,59)])
cov2 <- cov(spam2.Z2[,-c(1,59)])
all.equal(cov1, cov2,0.02)
#[1] TRUE
#si les matrices etaient difference, l'appel nous aurait renvoye le pourcentage de difference entre
    les 2.


#exemple de calcul de la pvalue moyenne pour les donnees ionosphere
iono.Z1 <- iono[iono$Z==1,]
iono.Z2 <- iono[iono$Z==2,]
lsharp.iono1 <- lapply(iono.Z1[,-c(1,3,36)], shapiro.test)
lres.iono1 <- sapply(lsharp.iono1, `[`, c("p.value"))
tot <- 0
for(i in 1:length(lres.iono1)){
  tot <- tot + lres.iono1[[i]]
}
group1 <- tot / length(lres.iono1)

lsharp.iono2 <- lapply(iono.Z2[,-c(1,3,36)], shapiro.test)
lres.iono2 <- sapply(lsharp.iono2, `[`, c("p.value"))
tot <- 0
for(i in 1:length(lres.iono2)){
  tot <- tot + lres.iono2[[i]]
}
group2 <- tot / length(lres.iono2)

pvaluemoyenne <- (group1 + group2 )/2

#exemple de calcul de la difference relative des probabilites a priori des donnees spam2:
pik <- abs((dim(spam2.Z1)[1]-dim(spam2.Z2)[1])/dim(spam2)[1])

#algorithme de test des methodes

repPredLDA <- function(N_essais,X, nz){
min <- 1
max <- 0
err <- 0
res <- list()
for(i in 1:N_essais)
{
  tst <- sample(2, nrow(X), replace = T, prob = c(0.8, 0.2))
  X.app <- X[tst == 1,]
  X.tst <- X[tst == 2,]


  fit <- lda(Z~.-X, X.app)
  pred <- predict(fit, newdata=X.tst[,-nz])
  val <- mean(pred$class!=X.tst[,nz])
  if (val<min) {min <- val}
  if(val>max) {max <- val}
```

```
  err <- err + val/N_essais
}
res$err <- err
res$min <- min
res$max <- max
return(res)
}


repPredQDA <- function(N_essais,X, nz){
  min <- 1
  max <- 0
  err <- 0
  res <- list()
  for(i in 1:N_essais)
  {
    tst <- sample(2, nrow(X), replace = T, prob = c(0.8, 0.2))
    X.app <- X[tst == 1,]
    X.tst <- X[tst == 2,]


    fit <- qda(Z~.-X, X.app)
    pred <- predict(fit, newdata=X.tst[,-nz])
    val <- mean(pred$class!=X.tst[,nz])
    if (val<min) {
      min <- val
    }
    if(val>max) {max <- val}
    err <- err + val/N_essais
  }
  res$err <- err
  res$min <- min
  res$max <- max
  return(res)
}


#faire as.factor() avant
repPredNBC <- function(N_essais,X, nz){
  min <- 1
  max <- 0
  err <- 0
  res <- list()
  for(i in 1:N_essais)
  {
    tst <- sample(2, nrow(X), replace = T, prob = c(0.8, 0.2))
    X.app <- X[tst == 1,]
    X.tst <- X[tst == 2,]


    fit <- naiveBayes(Z ~ . -X, data = X.app)
    pred <- predict(object=fit, newdata=X.tst)
    val <- mean(pred!=X.tst[,nz])
    if (val<min) {min <- val}
    if(val>max) {max <- val}
    err <- err + val/N_essais
  }
  res$err <- err
  res$min <- min
  res$max <- max
  return(res)
}

repPredRL <- function(N_essais,X, nz){
  min <- 1
  max <- 0
  err <- 0
  res <- list()
  for(i in 1:N_essais)
  {
    tst <- sample(2, nrow(X), replace = T, prob = c(0.8, 0.2))
    X.app <- X[tst == 1,]
    X.tst <- X[tst == 2,]


    fit <- train(Z ~ .-X, data = X.app,method="glm")
    pred <- predict(object=fit, newdata=X.tst)
    val <- mean(pred!=X.tst[,nz])
    if (val<min) {min <- val}
    if(val>max) {max <- val}
    err <- err + val/N_essais
  }
  res$err <- err
  res$min <- min
  res$max <- max
  return(res)
}
```

```
150
151 repPredBT <- function(N_essais,X, nz){
152   min <- 1
153   max <- 0
154   err <- 0
155   res <- list()
156   for(i in 1:N_essais)
157   {
158     tst <- sample(2, nrow(X), replace = T, prob = c(0.8, 0.2))
159     X.app <- X[tst == 1,]
160     X.tst <- X[tst == 2,]
161
162
163     fit <- rpart(Z ~. -X, method="class", data=X.app)
164     #prune <- prune(prune, cp=fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])
165     pred <- predict(fit,newdata=X.tst, type="class")
166     val <- mean(pred!=X.tst[,nz])
167     if (val<min) {min <- val}
168     if(val>max) {max <- val}
169     err <- err + val/N_essais
170   }
171   res$err <- err
172   res$min <- min
173   res$max <- max
174   return(res)
175 }
176
177 #exemple d'appel e ces fonctions pour breastcancer
178 bc$Z <- as.numeric(bc$Z)
179 resLDA <- repPredLDA(50,bc,32)
180 resLDA$err
181 resLDA$min
182 resLDA$max
183
184 repPredQDA(50,bc,32)
185 bc$Z <- as.factor(bc$Z)
186 repPredNBC(50,bc,32)
187 repPredRL(50,bc,32)
188 repPredBT(50,bc,32)
189
190 #autres exemples particuliers
191 #Iono: pour chaque methode on ne prend pas en compte X2
192 repPredQDA(50,iono[,-3],35)
193
194 #sauf pour QDA oe on le prend pas non plus en compte la variable binaire X1
195 repPredQDA(50,iono[,-c(2,3)],34)
196
197 #Spam et Spam2, QDA avec CV=TRUE plutot que l'utilisation de repPredQDA car
198 #selon la repartitions des donnees en train/test, nous pouvons avoir l'erreur Group rank defficiency
199 spam.qda.CV <- qda(Z~. - X, spam, CV=TRUE)
200 table(spam.qda.CV$class, spam$Z)
```

Listing 6: Algorithme de test des méthodes pour la partie 1