Rapport Pôle Projet Santé

Analyse d'une base de données sur le gliome pédiatrique
CentraleSupélec

Noms :	Damien Auprêtre, Elina Bouinot, Mehdi Hellal, Ayoub Kahi, Théodore Van Moere
Titre du projet de recherche :	Pôle projet santé
Professeur encadrant :	Arthur Tenenhaus

1 Introduction

Les tumeurs cérébrales sont les tumeurs solides les plus fréquentes chez l'enfant et représentent la principale cause de mortalité parmi les cancers pédiatriques. Malgré les progrès réalisés dans les traitements multimodaux, le taux de survie globale à 5 ans pour les enfants atteints de gliomes pédiatriques de haut grade reste limité, avoisinant les 20

Selon leur localisation — par exemple dans le tronc cérébral, les noyaux centraux ou la région supratentorielle — ces tumeurs présentent des caractéristiques variées sur les plans radiologique, histologique et pronostique. Il est supposé [6] que ces tumeurs ont des origines génétiques distinctes et empruntent des voies oncogéniques différentes selon leur emplacement. Par conséquent, les mécanismes biologiques impliqués dans leur développement pourraient différer d'une région à l'autre, comme cela a souvent été suggéré.

Objectif du projet

Dans le cadre de ce projet, nous disposons d'une base de données gliomaData recensant les données multimodales de 53 patients atteints du gliome pédiatrique. Ce jeu de données comprend notamment les données d'expression de certains gènes suspectés, dans un bloc GE (15 702 variables), ainsi que des données sur l'altération de l'ADN des patients, dans un bloc appelé CGH (1 229 variables). Un 3ème bloc comprend 3 classes (midl, cort et dipg) correspondant à la localisation de la tumeur chez le patient, il s'agit du bloc cible pour nos modèles. L'objectif du projet est de trouver les variables génétiques responsables de l'emplacement de la tumeur chez les patients atteints du gliome, et de comparer l'efficacité et les variables obtenues par différentes méthodes disponibles dans la littérature.

Travail et contribution

Nous avions pour référence la méthode RGCCA/SGCCA, qui avait déjà été utilisée sur le jeu de données gliomaData. Nous devions tester différentes méthodes qui n'avaient jamais été testées sur gliomaData, et comparer leur efficacité par rapport à RGCCA/SGCCA. Pour cela nous nous sommes appuyées la documentation disponibles sur les différentes méthodes abordées : RGCCA/SGCCA [5], Sparse Discriminant Analysis [1], Régression logistique et LASSO [3] , Multiview [2]. Nous avons également utilisé les sections dédiées à la régression logistique et à l'analyse linéaire discriminantes dans ce livre [4] afin de nous former lors du projet.

Déroulé du projet

Ce projet vise à identifier les variables génétiques responsables de la localisation des tumeurs dans le cadre des gliomes pédiatriques à partir de données multimodales. Le jeu de données étudié, gliomaData, regroupe des informations d'expression génique (bloc GE), d'altération de l'ADN (bloc CGH) ainsi qu'une variable cible indiquant la localisation de la tumeur (midl, cort ou dipg). Après une phase exploratoire sur le jeu de données Iris, plusieurs méthodes de classification et de sélection de variables ont été mises en œuvre : régression logistique avec pénalisation LASSO, Sparse Discriminant Analysis (SDA), SGCCA, et Multiview. Ces approches permettent à la fois la réduction de dimension et l'interprétation des variables influentes. Les performances des méthodes ont été comparées, en mettant en évidence les atouts de Multiview et SGCCA dans un contexte à très haute dimension. Ce travail met en lumière des gènes potentiellement liés à des localisations tumorales spécifiques, ouvrant des pistes pour des analyses biomédicales plus poussées.

2 Iris de Fischer

Dans le but de prendre en main les concepts clés et de nous former sur la classification et l'apprentissage supervisé, nous nous sommes tout d'abord focalisés sur un jeu de données moins complexe que gliomaData.

Le jeu de données des iris de Fisher comporte 150 observations sur 3 espèces différentes d'iris : setosa, versicolor et virginica. Les 4 variables recensées dans ces observations sont : la longueur/largeur des sépales, ainsi que la longueur/largeur des pétales.

2.1 Régression Logistique (RL)

Dans la régréssion logistique, on va définir des probabilités grâce à la logistique:

$$f(z) = \frac{1}{1 + \exp(-z)}$$

Le but va être de trouver les paramètres β précisés juste après. Le modèle est de cette forme:

$$\forall k \in \{1, \ldots, K-1\} \ \log\left(\frac{\Pr(G = k|X = x)}{\Pr(G = K|X = x)}\right) = \beta_{k,0} + \beta_k^T x$$

où G est le "prédicteur". Le numérateur dans la fraction est la probabilité que le modèle choisisse la classe k sachant qu'on a les observations x. On utilise le log pour se ramener à quelque chose de linéaire. On peut en déduire chaque probabilité:

$$\Pr(G = k|X = x) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)}, \quad k = 1, \ldots, K-1,$$

$$\Pr(G = K|X = x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)},$$

On va définir une fonction de "**vraisemblance**" qu'on va chercher à maximiser. On prend ensuite son logarithme et la fonction devient alors concave et bornée supérieurement et admet donc un maximum. On note cette nouvelle fonction "**log_vraisemblance**":

$$\ell(\theta) = \sum_{i=1}^{N} \log p_{g_i}(x_i; \theta),$$

$$\text{où } p_k(x_i; \theta) = \Pr(G = k|X = x_i; \theta) = \frac{\exp(\beta_{k0} + \beta_k^T x)}{1 + \sum_{\ell=1}^{K-1} \exp(\beta_{\ell 0} + \beta_\ell^T x)}$$

Il est plus commode de prendre $K = 2$, car alors on a un problème binaire et on peut écrire la fonction log_vraisemblance comme ceci:

$$\ell(\beta) = \sum_{i=1}^{N} \{y_i \log p(x_i; \beta) + (1 - y_i) \log(1 - p(x_i; \beta))\}, = \sum_{i=1}^{N} \left\{y_i \beta^T x_i - \log(1 + \exp(\beta^T x_i))\right\}.$$

Il faut noter que β est un vecteur de dimension N+1
Ici on définit $y_i = 1$ si $g_i = 1$ et $y_i = 0$ si $g_i = 0$ et $\beta = \{\beta_{1,0}, \beta_1\}$ On a pris la convention $x_i' = \{1, x_i\}$ pour prendre en compte le biais (le terme $\beta_{1,0}$).

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{N} x_i' \left(y_i - p(x_i'; \beta)\right) = 0,$$

En notant n la dimension de x_i, on a $n+1$ équations car on doit prendre en compte le biais (il s'agit ici de x_i'). On utilise l'algorithme de Newton-Raphson pour résoudre l'équation (1) et on prend la méthode de Newton pour $K > 2$ On a donc accès aux paramètres de notre modèle. On définit une fonction seuil par rapport aux probas

2.2 Analyse Linéaire Discriminante (LDA)

2.2.1 Analyse Linéaire Discriminante de Bayes

Notons :

- X "l'entrée", l'ensemble des données qui sont dans $\mathbf{R}^p$. On notera les valeurs prises par X : $x \in \mathbf{R}^p$, de coordonnées x_i.
- $k \in \{1, \ldots, K\}$ les différentes classes possibles, et G la va de l'événement appartenir à une de ces classes k.
- p la dimension (taille du vecteur X qui est dans $\mathbf{R}^p$, i.e. le nombre de variables utilisées pour classifier les données).
- $f_k(x) = Pr(X = x|G = k)$ la densité de classe conditionelle de X dans la classe $G = k$, indiquant comment les données sont réparties au sein d'une classe si X $\in Cl_k$. On va modéliser ces densités par des gaussiennes :

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k)\right)$$

  avec les μ_k les vecteurs moyennes (dans $\mathbf{R}^p$) au sein de chaque classe (on prendra $\mu_k = \sum\limits_{x \in Cl_k} \frac{x}{N_{x \in Cl_k}}$), et les Σ_k leurs matrices de covariances respectives (dans $\mathbf{R}^{p \times p}$). On fera l'hypothèse que chaque classe possède la même matrice de covariance, i.e. $\Sigma_k = \Sigma = \sum\limits_{k=1}^{K} \frac{\sum_{i|g_i=k}(x_i-\mu_k)(x_i-\mu_k)^T}{N_{tot}-K}$. On obtient donc :

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T\Sigma^{-1}(x-\mu_k)\right).$$

- $\pi_k = Pr(X = k)$ la probabilité a priori d'appartenance de X à la classe k, telle que $\sum\limits_{i=1}^{N} \pi_k = 1$. Avec nos données d'entraînement, on prendra : $\pi_k = \frac{N_{X \in Cl_k}}{N_{tot}}$ le rapport du nombre de données dans la classe k, sur le nombre total de données.

On cherche à trouver les probabilités d'appartenance à une classe a posteriori sachant que notre entrée $X = x, Pr(G = k|X = x)$. En utilisant le théorème de Bayes, on obtient :

$$\Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum\limits_{l=1}^{N} f_l(x)\pi_l}$$

On veut calculer la frontière de décision entre classes, c'est-à-dire trouver l'équation régissant la frontière $Pr(G = k|X = x) = Pr(G = l|X = x)$, en séparant ainsi notre espace par des hyperplans entre les différentes paires de classes. En passant au log dans cette équation, et après remplacement des expressions de Pr et des f_k, f_l :

$$\log\frac{Pr(G = k|X = x)}{Pr(G = l|X = x)} = \log\frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T\Sigma^{-1}(\mu_k - \mu) + x^T\Sigma^{-1}(\mu_k - \mu_l)$$

On obtient ainsi une équation linéaire en x pour chaque frontière.

La règle de décision est équivalente à trouver :

$$G(x) = \arg\max_k \delta_k(x)$$

avec δ_k la "fonction score" associée à la classe k :

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

On affecte à x la classe k maximisant cette fonction score.

2.2.2 Analyse Linéaire Discriminante de Fischer

Le problème de Fischer revient à trouver le vecteur a permettant de maximiser :

$$\max_a \quad \frac{a^T B a}{a^T W a}$$

Avec :

1. La matrice de dispersion entre les classes B mesure la variance entre les différentes classes. Elle est définie par :

$$B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

où :

- c est le nombre de classes
- N_i est le nombre d'échantillons dans la classe i,
- μ_i est le vecteur moyen de la classe i,
- μ est le vecteur moyen global.

2. La matrice de dispersion intra-classe W mesure la variance des échantillons à l'intérieur de chaque classe. Elle est définie comme :

$$W = \sum_{i=1}^{c} \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T$$

où :

- x est un échantillon appartenant à la classe C_i,
- μ_i est le vecteur moyen de la classe i.

On trouve que le max est atteint en un a vecteur propre associé à la plus grande vap de $W^{-1}B$. La projection des données sur a donne la meilleure séparation linéaire possible entre les classes. (On cherche à maximiser la distance entre les moyennes de chaque classe à la moyenne globale des données tout en minimisant la dispersion pour bien séparer les classes).

Formulation par optimal scoring

Une formulation équivalente de l'analyse discriminante linéaire (LDA) peut être obtenue via le *problème d'optimal scoring*, où l'on reformule la LDA comme une régression multivariée contrainte. On suppose :

- $\mathbf{X} \in R^{n \times p}$: les données centrées,
- $\mathbf{Y} \in R^{n \times K}$: la matrice d'indicateurs de classe (1 si l'observation appartient à la classe k, 0 sinon),
- $\mathbf{\Theta} \in R^{K \times (K-1)}$: les scores des classes à apprendre,
- $\boldsymbol{\beta} \in R^{p \times (K-1)}$: les vecteurs discriminants. La donnée de $\boldsymbol{\beta}$ est équivalente à la donnée du vecteur a dans la LDA de Fischer ($X\boldsymbol{\beta}$ est la projection).

Le problème d'optimisation s'écrit :

$$\min_{\boldsymbol{\beta}, \mathbf{\Theta}} \|\mathbf{Y}\mathbf{\Theta} - \mathbf{X}\boldsymbol{\beta}\|_F^2 \quad \text{sous la contrainte} \quad \mathbf{\Theta}^\top \mathbf{Y}^\top \mathbf{Y} \mathbf{\Theta} = \mathbf{I}$$

où $\|\cdot\|_F$ désigne la norme de Frobenius. Cette contrainte assure que les scores sont orthonormés.

Cette approche, bien que différente dans sa forme, aboutit aux mêmes directions discriminantes que la LDA de Fisher. Elle a l'avantage d'être facilement généralisable, notamment pour intégrer des pénalisations (ex. Lasso, Ridge) ou pour traiter des problèmes multiblocs et multi-vues.

2.2.3 Résultats avec R

Pour entraîner le modèle, nous avons séparé le jeu de données d'Iris de Fisher en 2 parties : 70 % des données sont utilisées pour l'apprentissage, et 30 % pour les tests. [1]

```
[1] "Répartition des classes dans le train set :"

    setosa versicolor  virginica
        36         32         37
[1] "Répartition des classes dans le test set :"

    setosa versicolor  virginica
        14         18         13
```

Figure 1: Répartition des classes

On peut aussi accéder aux coefficients des combinaisons linéaires des variables utilisés pour projeter les données sur les axes de la LDA : [2]

```
Coefficients of linear discriminants:
                    LD1        LD2
Sepal.Length  0.887242 -0.1977697
Sepal.Width   1.267866 -1.8705232
Petal.Length -2.101235  1.1717820
Petal.Width  -2.934756 -3.1681626
```

Figure 2: Coefficients de la LDA

Cela signifie que :
$\text{LD}_1 = 0{,}887 \cdot \text{Sepal.Length} + 1{,}268 \cdot \text{Sepal.Width} - 2{,}101 \cdot \text{Petal.Length} - 2{,}935 \cdot \text{Petal.Width}$
$\text{LD}_2 = -0.198 \cdot \text{Sepal.Length} - 1.871 \cdot \text{Sepal.Width} + 1.172 \cdot \text{Petal.Length} - 3.168 \cdot \text{Petal.Width}$

[3]

```
Proportion of trace:
   LD1    LD2
0.9937 0.0063
```

Figure 3: Importance des deux directions

Ici, on voit que LD1 explique 99.37 % de la séparation entre les espèces alors que LD2 n'en explique que 0.63 %. La majorité est portée par LD1. On peut également afficher les corrélations entre les différentes variables observées selon les espèces :

[4]

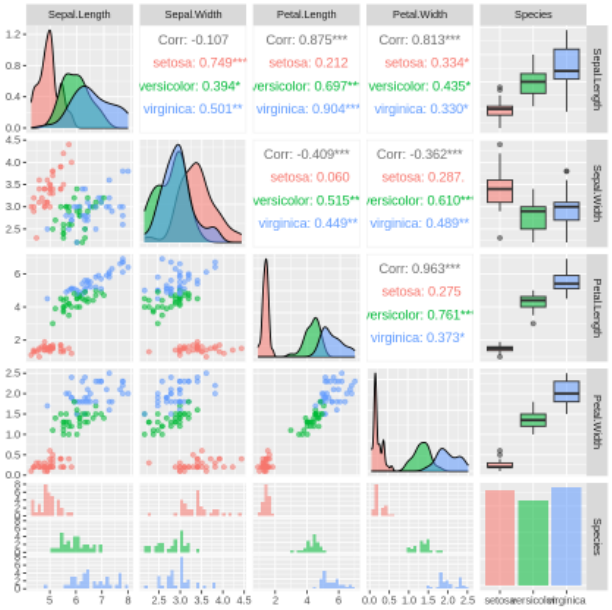

Figure 4: Corrélation entre différentes variables

On voit par exemple que la taille et la longueur des pétales sont très corrélées (positivement) globalement chez les iris, avec une corrélation de 0.963. On remarque aussi que la répartition des variables au sein de chaque espèce ressemblent à des répartitions normales, ce qui est rassurant au vu des hypothèses de la LDA. Cela peut donc expliquer le résultat proche de 100% de précision que l'on obtient, car les hypothèses sont très proches de la réalité : [5]

```
[1] "Matrice de confusion :"
           Actual
Predicted    setosa versicolor virginica
  setosa         14          0         0
  versicolor      0         17         0
  virginica       0          1        13
[1] "Accuracy: 97.78 %"
```

Projection des données sur LD1 et LD2

Figure 5: Matrice de confusion

La matrice de confusion indique que la LDA ne s'est trompée qu'une seule fois dans la classification des espèces sur les données de test (coefficient 1 hors diagonal). Pour finir, on peut afficher les données séparées au mieux par espèce par la méthode LDA par projection sur les directions LD1 et LD2 :

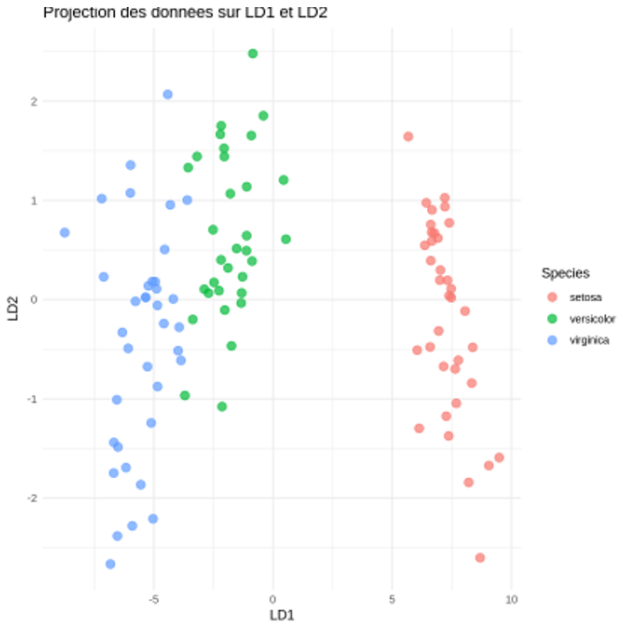

Figure 6: Projection des données sur LD1 et LD2

Les données ont l'air bien séparé par la LDA d'un point de vue graphique.

3 Applications sur dataset Glioma Data

3.1 Régression Logistique + LASSO

3.1.1 Fonctionnement de LASSO

LASSO signifie Least Absolute Shrinkage and Selection Operator. C'est une méthode de régression linéaire utilisée pour sélectionner automatiquement les variables les plus pertinentes. Le LASSO résout le problème suivant :

$$\hat{\beta} = \arg\min_{\beta} \left(\sum_{i=1}^{n} (y_i - X_i\beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right)$$

où :

- y_i : la variable à prédire (cible)
- $X_i\beta$: la prédiction faite par le modèle
- λ : le paramètre de régularisation
- $\sum |\beta_j|$: la pénalité L1 (somme des valeurs absolues des coefficients)

La pénalité L1 (somme des valeurs absolues des coefficients) pousse certains coefficients β_j à devenir exactement zéro.

Résultat : seules les variables les plus utiles conservent un coefficient non nul. Les autres sont automatiquement exclues du modèle. Le Fused Lasso est une extension du LASSO qui sert à la fois à : Sélectionner des variables comme le LASSO classique

Imposer de la similarité entre des coefficients voisins Le Fused Lasso résout le problème suivant :

$$\hat{\beta} = \arg\min_{\beta} \left(\sum_{i=1}^{n} (y_i - X_i\beta)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=2}^{p} |\beta_j - \beta_{j-1}| \right)$$

On remarque qu'une pénalité a été ajoutée par rapport au LASSO : c'est la pénalité de fusion. Elle encourage les coefficients voisins à être proches. Le fused LASSO va détecter des zones où des coefficients changent soudainement Le Fused Lasso choisit quelles variables garder ET force les coefficients à être proches de leurs voisins, sauf quand un vrai "changement" est détecté.

3.1.2 Résultats

Nous avons appliqué la méthode du LASSO pour la sélection de variables et l'évaluation de performance, d'abord sur le bloc GE seul, puis sur les deux blocs combinés GE et CGH.

```
Accuracy moyenne sur 100 itérations : 0.874

     Classe Sensibilité Spécificité
cort   cort       0.931       0.880
dipg   dipg       0.938       0.912
midl   midl       0.644       1.000
```

Figure 7: Accuracy, Spécificité, Sensibilité

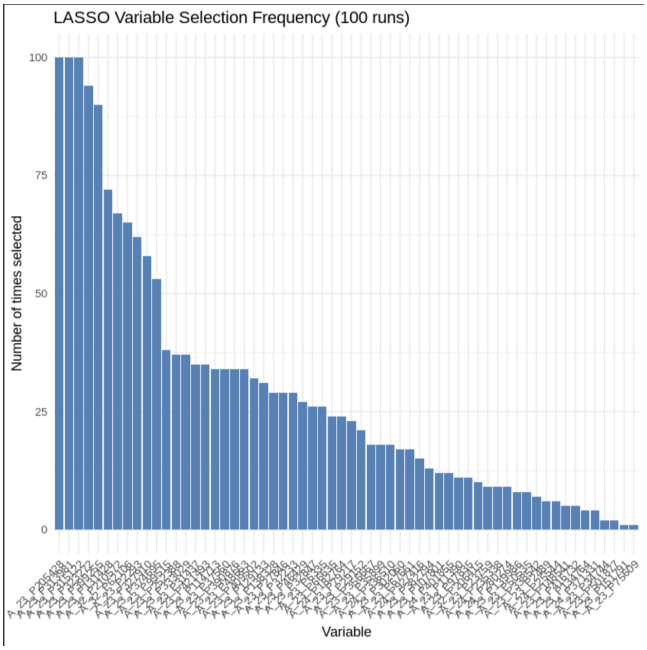

Figure 8: Fréquence d'apparition des variables lors de 100 LASSO sur le bloc GE

Pour le bloc GE seul, les résultats montrent une accuracy moyenne d'environ 88%, ce qui traduit une bonne capacité du modèle à prédire correctement les classes. Les sensibilités et spécificités sont globalement satisfaisantes pour la plupart des classes. Toutefois, on note une faiblesse particulière de la sensibilité pour la classe midl, ce qui signifie que cette classe est souvent mal identifiée ou confondue avec d'autres. Cela pourrait être lié à une moindre représentativité de cette classe dans les données ou à des signaux moins discriminants.

Nous pouvons voir sur la deuxième figure que 11 variables apparaissent sur plus de 50% des itérations, et nous pouvons donc considérer que celles-ci sont des variables discriminantes.

Pour les deux blocs combinés nous obtenons ces résultats :

```
Accuracy moyenne sur 100 itérations : 0.886

     Classe Sensibilité Spécificité
cort   cort       1.000       0.924
dipg   dipg       1.000       0.886
midl   midl       0.451       1.000
```

Figure 9: Accuracy, Spécificité, Sensibilité

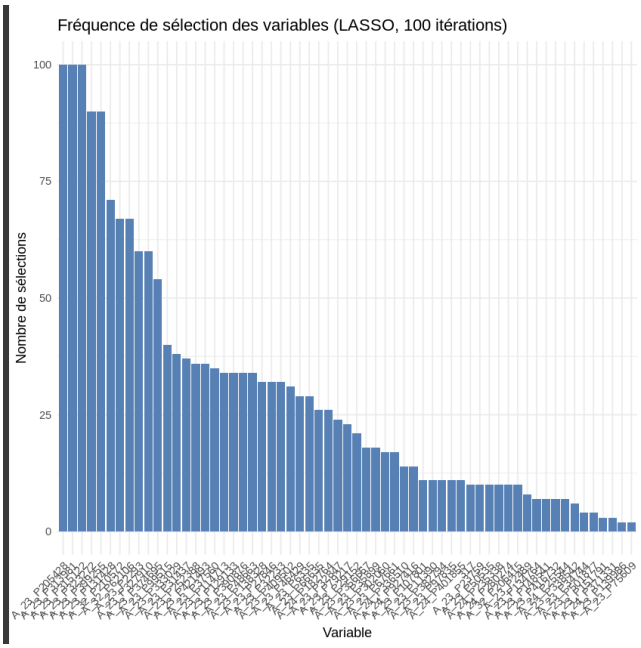

Figure 10: Fréquence d'apparition des variables lors de 100 LASSO sur les blocs GE et CGH combinés

Lorsqu'on combine les blocs GE et CGH, on observe une amélioration légère mais cohérente de la performance globale. L'accuracy reste autour de 88En analysant plus attentivement les résultats, on constate que la similarité des performances entre l'utilisation du bloc GE seul et celle des blocs GE et CGH combinés s'explique par le fait que toutes les variables sélectionnées proviennent exclusivement du bloc GE. Cette observation est confirmée par les graphiques de fréquence de sélection, où aucune variable issue du bloc CGH n'apparaît. Partant de ce constat, nous avons envisagé une approche plus sophistiquée : appliquer un fused LASSO sur le bloc CGH, afin de tirer parti de la structure potentiellement ordonnée de ses variables, tout en conservant un LASSO classique sur le bloc GE. L'objectif était de favoriser une sélection conjointe plus équilibrée entre les deux types de données. Cependant, cette tentative n'a pas donné les résultats escomptés : les variables sélectionnées provenaient à nouveau exclusivement du bloc GE, indiquant que le bloc CGH, même avec une régularisation adaptée, n'apportent pas d'information discriminante supplémentaire pour la tâche de classification considérée.

3.2 Sparse Discriminant Analysis (SDA)

Dans les problèmes de classification supervisée en haute dimension (où le nombre de variables p est largement supérieur au nombre d'observations n), la méthode classique de Linear Discriminant Analysis (LDA) devient inapplicable. En effet, la matrice de covariance intra-classe estimée est alors singulière, ce qui empêche le calcul des vecteurs discriminants.

L'approche *Sparse Discriminant Analysis* (SDA), proposée par Clemmensen *et al.* (2011), repose sur la formulation dite *optimal scoring* de la LDA. Cette reformulation du problème de classification en régression permet l'introduction naturelle de pénalités favorisant la sélection de variables.

3.2.1 Formulation du problème

Soit $X \in R^{n\times p}$ la matrice des observations (centrées), et $Y \in R^{n\times K}$ une matrice de variables indicatrices représentant l'appartenance de chaque observation à l'une des K classes. Pour chaque direction discriminante k, le problème SDA consiste à résoudre :

$$\min_{\beta_k,\theta_k} \|Y\theta_k - X\beta_k\|^2 + \gamma\beta_k^\top \Omega\beta_k + \lambda\|\beta_k\|_1 \tag{1}$$

sous les contraintes :

$$\frac{1}{n}\theta_k^\top Y^\top Y\theta_k = 1, \quad \theta_k^\top Y^\top Y\theta_l = 0 \quad \forall l < k \tag{2}$$

où :

- $\beta_k \in R^p$ est le vecteur discriminant recherché,

- $\theta_k \in R^K$ est un vecteur de scores,
- λ et γ sont des paramètres de régularisation,
- Ω est une matrice symétrique définie positive (souvent $\Omega = I$).

L'ajout de la pénalité $\|\beta_k\|_1$ encourage la *sparsité*, c'est-à-dire l'utilisation d'un nombre restreint de variables pour la classification. La régularisation de type ridge via $\beta_k^\top \Omega \beta_k$ améliore la stabilité numérique, notamment lorsque des variables sont corrélées.

3.2.2 Algorithme de résolution

Le problème (1)–(2) est résolu par une stratégie itérative alternée :

1. Initialiser θ_k de manière aléatoire et l'orthonormaliser par rapport aux directions précédentes ;
2. Pour un θ_k fixé, résoudre en β_k le problème d'elastic net :
$$\min_{\beta_k} \|Y\theta_k - X\beta_k\|^2 + \gamma \beta_k^\top \Omega \beta_k + \lambda \|\beta_k\|_1 \tag{3}$$
3. Pour un β_k fixé, actualiser θ_k :
$$\theta_k \propto (I - Q_k Q_k^\top D_\pi) D_\pi^{-1} Y^\top X \beta_k \tag{4}$$
où $D_\pi = \frac{1}{n} Y^\top Y$ est une matrice diagonale contenant les proportions de classes, et Q_k contient les θ_l précédents (ainsi que le vecteur trivial de 1 si nécessaire).

3.2.3 Classification et visualisation

Une fois les vecteurs discriminants $\beta_1, \ldots, \beta_q$ obtenus, on projette les données originales sur l'espace de dimension réduite :

$$X_{\text{proj}} = XB \quad \text{où } B = [\beta_1 \cdots \beta_q] \tag{5}$$

Une classification LDA standard est ensuite effectuée dans cet espace réduit. Ces directions projetées peuvent aussi servir à la visualisation graphique des classes (scatterplots).

3.2.4 Intérêts de SDA

La méthode SDA présente plusieurs avantages pratiques :

- Elle est applicable lorsque $p \gg n$;
- Elle produit des vecteurs discriminants *sparse*, ce qui améliore l'interprétabilité ;
- Elle est extensible au cas non-linéaire via des mixtures de Gaussiennes (SMDA) ;
- Elle est adaptée aux applications bio-informatiques (e.g. microarray), à l'analyse d'images ou encore aux signaux spectraux.

3.2.5 Résultats

Évaluation des performances par bootstrap OOB

Les hyperparamètres du modèle (`lambda`, `stop`) ont préalablement été sélectionnés par validation croisée pour chaque bloc.

Approche naïve SDA avec bloc GE et double cross-validation

Une première approche naïve a été de réappliquer une cross-validation pour trouver la spécificité, la sensibilité et l'accuracy du modèle, avec le bloc GE seul en entier (en effet, l'ajout du bloc CGH n'avait pas l'air d'avoir beaucoup d'influence sur l'erreur globale, comme ce n'est pas une méthode multibloc). Nous avons obtenu ces métriques, ainsi que les 235 variables les plus sélectionnées lors de la cross-validation à paramètres optimaux fixés. Cependant cette méthode est trop biaisée et optimiste, car on refait les mesures sur les mêmes données. De plus, après comparaison avec RGCCA/SGCCA, les variables sélectionnées par les deux modèles se recoupent très peu.

```
Confusion Matrix and Statistics

          Reference
Prediction cort dipg midl
      cort   20    0    2
      dipg    0   19    5
      midl    0    3    4

Overall Statistics

               Accuracy : 0.8113
                 95% CI : (0.6803, 0.9056)
    No Information Rate : 0.4151
    P-Value [Acc > NIR] : 4.128e-09

                  Kappa : 0.6995

 Mcnemar's Test P-Value : NA
```

Figure 11: Matrice de confusion et accuracy

	Class: cort	Class: dipg	Class: midl
Sensitivity	1.0000	0.8636	0.36364
Specificity	0.9394	0.8387	0.92857
Pos Pred Value	0.9091	0.7917	0.57143
Neg Pred Value	1.0000	0.8966	0.84783
Prevalence	0.3774	0.4151	0.20755
Detection Rate	0.3774	0.3585	0.07547
Detection Prevalence	0.4151	0.4528	0.13208
Balanced Accuracy	0.9697	0.8512	0.64610

Figure 12: Spécificité, sensibilité

Les variables se trouvant dans plus de 70% des sélections de sda :

```
 [1]   59   63  103  117  125  133  146  168  173  217  254  283  319  336
343  492  518  529  545  551  552
 [22]  563  602  620  626  658  710  734  840  885  889  891  903  953  983
1057 1070 1104 1152    2    7   11
 [43]   34   47   52   65   70   79   93   94   96  107  108  124  128  129
131  153  156  158  160  162  165
 [64]  172  181  187  188  192  194  196  203  213  214  219  220  230  232
243  248  255  261  272  273  276
 [85]  280  302  304  306  314  316  345  350  353  354  370  374  383  387
391  399  405  409  418  421  430
[106]  431  435  441  444  446  451  467  470  480  484  485  493  495  502
504  509  519  527  533  536  537
[127]  540  550  554  564  567  582  593  603  610  613  614  616  621  625
629  632  634  652  656  660  664
[148]  669  681  685  694  696  702  705  709  712  716  717  729  733  737
741  746  747  766  767  771  773
[169]  778  779  784  797  798  802  811  818  827  833  838  846  850  854
856  871  875  884  887  907  908
[190]  910  921  922  923  928  932  938  941  943  945  946  947  948  952
961  972  986  992  995 1007 1009
[211] 1014 1021 1025 1030 1046 1052 1055 1058 1062 1067 1080 1088 1090 1092
1093 1106 1114 1115 1125 1132 1135
[232] 1136 1156 1166
```

Figure 13: Variables sélectionnées à plus de 70% lors de la cross-validation

Méthode cross-validation + Boostrap

Les valeurs optimales obtenues ont été fixées pour l'ensemble des itérations bootstrap.

Interprétation des paramètres `lambda` et `stop` dans `sda`

- `lambda` : paramètre de régularisation L1 (lasso), il contrôle le degré de parcimonie du modèle. Une valeur plus élevée de `lambda` impose une plus forte pénalisation, ce qui conduit à la sélection d'un nombre plus restreint de variables explicatives (coefficients nuls pour les variables non pertinentes).
- `stop` : seuil de convergence utilisé dans l'algorithme itératif d'estimation. Il définit le critère d'arrêt en fonction de la variation du critère d'optimisation (par exemple, la log-vraisemblance) entre deux itérations successives. Une valeur de `stop` proche de zéro impose une convergence plus stricte, au prix d'un temps de calcul potentiellement plus long.

Afin d'évaluer la capacité de généralisation du modèle `sparseLDA`, les performances prédictives ont été estimées via une procédure de bootstrap avec estimation *Out-Of-Bag* (OOB), appliquée séparément à chacun des blocs de données : GE, CGH, et GE+CGH.

Méthodologie

La procédure mise en œuvre repose sur 10 rééchantillonnages bootstrap ($B = 10$). Pour chaque tirage :

1. Un échantillon d'apprentissage est généré par tirage avec remise parmi les individus initiaux.
2. Le modèle `sparseLDA` est entraîné sur cet échantillon avec les paramètres optimaux fixés.
3. Les individus non sélectionnés dans le bootstrap (*Out-Of-Bag*) sont utilisés pour évaluer l'erreur de classification.
4. L'erreur de prédiction est enregistrée pour chaque itération.

À l'issue des 10 répétitions, l'erreur OOB moyenne est calculée, accompagnée d'un intervalle de confiance à 95 % basé sur les quantiles empiriques.

Cette démarche permet de simuler un comportement en validation externe sans perte d'information pour l'entraînement.

Résultats

Bloc GE

- Paramètres fixés : `lambda` = 0.3, `stop` = -300
- Erreur OOB moyenne : 0,3012
- Intervalle de confiance (95 %) : [0,1796 ; 0,4611]

Le modèle entraîne une erreur de classification moyenne d'environ 30 % sur des données non vues. L'intervalle de confiance indique que l'erreur réelle pourra varier entre 18 % et 46 %, ce qui traduit une certaine variabilité liée au rééchantillonnage.

Bloc CGH

- Paramètres fixés : `lambda` = 0.3, `stop` = -250
- Erreur OOB moyenne : 0,2964
- Intervalle de confiance (95 %) : [0,1796 ; 0,4242]

Nous avons cependant rencontré des problèmes en essayant de trouver les variables les plus fréquemment sélectionnées par bootstrap, car le code mettait plusieurs heures à s'exécuter et n'aboutissait pas. Idem lorsque l'on combinait les blocs GE-CGH (pas de résultat au bout de 6 heures). Si le temps nous le permet, il faudrait revenir sur l'implémentation de la sélection de variables de sorte à ce que l'algorithme puisse finir.

3.3 RGCCA/SGCCA

RGCCA est un package de R permettant de traiter les données multimodales (plusieurs types de données pour dans notre un même patient) et d'entraîner des modèles dessus.
Dans notre cas, on va surtout utiliser le module SGCCA (Sparse General Canonical Correlation Analysis).

3.3.1 Le modèle mathématique

SGCCA va construire 2 composantes par blocs (car on a 3 classes, raisonnement similaire dans la LDA) en imposant 3 contraintes:

- Chaque composante doit bien représenter son propre bloc (elle doit expliquer au mieux les variables du bloc dont elle est extraite).
- Les composantes issues de différents blocs doivent être aussi corrélées que possible, si ces blocs sont connectés dans la structure du modèle (par exemple via une matrice de design)
- Les composantes doivent être sparse, c'est-à-dire construites à partir d'un petit nombre de variables (beaucoup de poids sont nuls)
- on forme ces vecteurs avec des combinaisons linéaires des variables dans les blocs correspondants (les coefficients étant les poids).

On va donc ici chercher J vecteurs de poids vérifiants ces contraintes et permettant de prédire la localisation de la tumeur.

On pose $\mathbf{X} = [\mathbf{X}_1, \ldots, \mathbf{X}_J]$, où $\mathbf{X}_j \in R^{n \times p_j}$, avec p_j le nombre de variables observées sur les n patients pour le bloc j. Le problème d'optimisation est donc le suivant:

$$\max_{\mathbf{w}_1,\ldots,\mathbf{w}_J} \sum_{j,k=1}^{J} c_{jk}\, g\left(\mathrm{cov}(\mathbf{X}_j\mathbf{w}_j, \mathbf{X}_k\mathbf{w}_k)\right) \quad s.t \quad \|\mathbf{w}_j\|_2^2 = 1 \quad \text{et} \quad \|\mathbf{w}_j\|_1 \leq s_j \quad \text{pour } j = 1, \ldots, J$$

Les s_j correspondent au degré de sparsité qu'on accorde aux vecteur: plus s_j est petit, plus le degré de sparsité est grand (ie on augmente le nombre de coefficients nuls dans le modèle) à contrario du λ de multiview où on a un min.
Dans notre cas $J = 2$ $c_{j,k} = 0$ si X_j et X_k sont liés, 0 sinon. On a la relation entre blocs suivante:

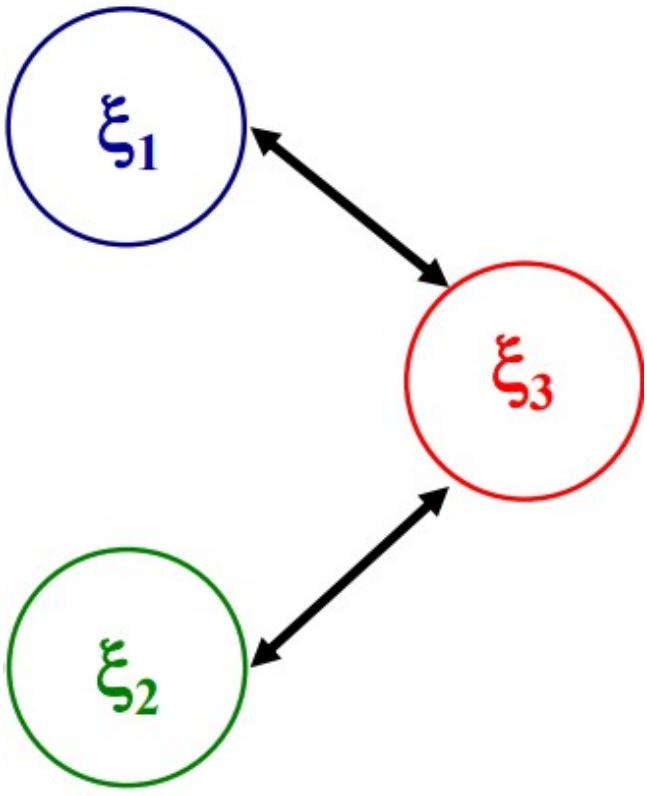

Figure 14: Relation entre blocs

avec ξ_3 la cible: localisation de la tumeur et les 2 autres correspondant aux blocs CGH et GE.

3.3.2 Approche

Ici ce sont les paramètres de sparsités qui vont jouer un grand rôle dans notre modèle, mais déjà faut-il savoir lesquels choisir... On va commencer par trouver les paramètres optimaux par cross-validation.
On entraîne ensuite notre modèle avec ces paramètres optimaux sur nos données d'entrainement. On procède enfin à une méthode type bootstrap pour déterminer les variables les plus importantes (on fait une boucle sur plein d'échantillon et on voit les variables qui apparaissent le plus souvent).

3.3.3 Code

On initialise les données d'entrainement et de test:

```
in_train <- caret::createDataPartition(blocks[[3]], p = .75, list = FALSE)
training <- lapply(blocks, function(x) as.matrix(x)[in_train, , drop = FALSE])
testing <- lapply(blocks, function(x) as.matrix(x)[-in_train, , drop = FALSE])
```

On trouve les paramètres optimaux:

```
cv_out <- rgcca_cv(blocks = training, response = 3,par_type = "sparsity",par_value = c(.2, .2, 1),
                        par_length = 10,
                        prediction_model = "lda",
                        validation = "kfold",
                        k = 7, n_run = 3, metric = "Balanced_Accuracy",
                        n_cores = 2)
```

On fait de la cross-validation en comparant la "Balanced accuracy" $= \frac{\text{specifité + recall}}{2}$ avec un modèle de prédiction type LDA. On borne les coefficients de sparsité par 0.2 pour être sûr de ne récupérer que les variables significatives. Le coefficient de sparsité pour la cible vaut 0, car on veut uniquement que les composantes soient corrélées avec les autres 2 blocs, mais pas qu'elles expliquent elles mêmes le bloc cible (c'est le rôle des 2 autres blocs).

```
fit <- rgcca(cv_out)
fit_stabi <- rgcca_stability(fit,
                             keep = vapply(
                               fit$a, function(x) mean(x != 0),
                              FUN.VALUE = 1.0
                               ),
                             n_boot = 100, verbose = TRUE, n_cores = 2)
```

On peut déjà plot les résultats de fit pour la 2e composante de chaque bloc par exemple, sans avoir fait de bootstrap.

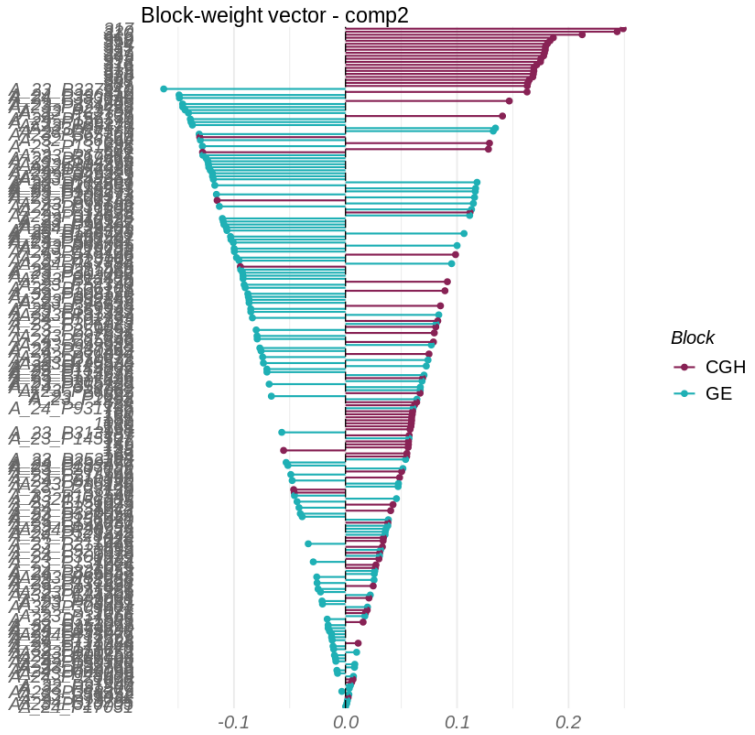

Figure 15: Coefficients trouvé avant stabilité et bootstrap

On observe que le modèle avec seulement 2 composantes sur 4 peut séparer la classe 'cort' des autres On pourrait à première vue penser que les coefficients du bloc CGH sont plus importants, mais il ne faut pas oublier que l'on ne compare l'importance des coefficients que bloc et à bloc et que comme les vecteurs doivent être de norme 1, on ne peut dire que les coefficients maximum de CGH trouvés ici sont plus importants que ceux de GE.

Si maintenant on affiche les samples avec notre modèle fit-stabi dans l'espace des 2 vecteurs optimaux:

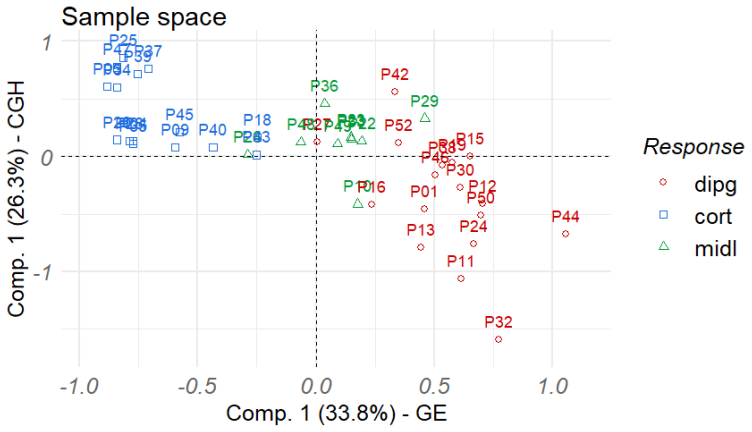

Figure 16: Sample space

On peut enfin regarder la significativé des variables trouvées (p-value et intervalles de confiance) pour en extraire les meilleures.

```
boot_out <- rgcca_bootstrap(fit_stabi, n_boot = 500)
```

3.3.4 Résultats

On extrait donc les 20 variables les plus importantes pour chaque blocs.

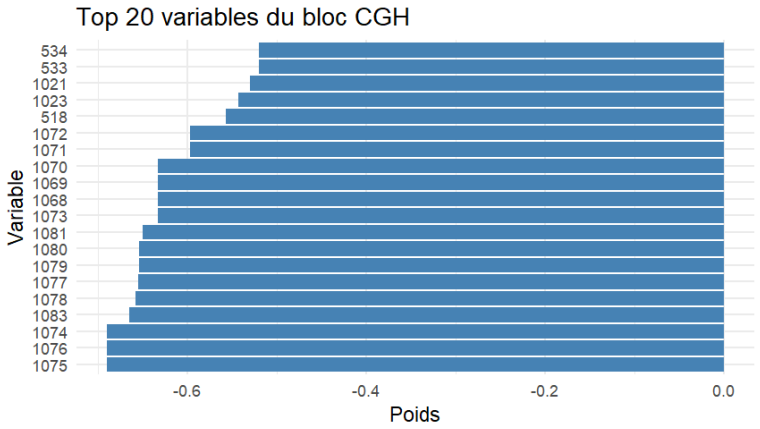

Figure 17: Plot des 20 coefficients les plus importants dans le block CGH

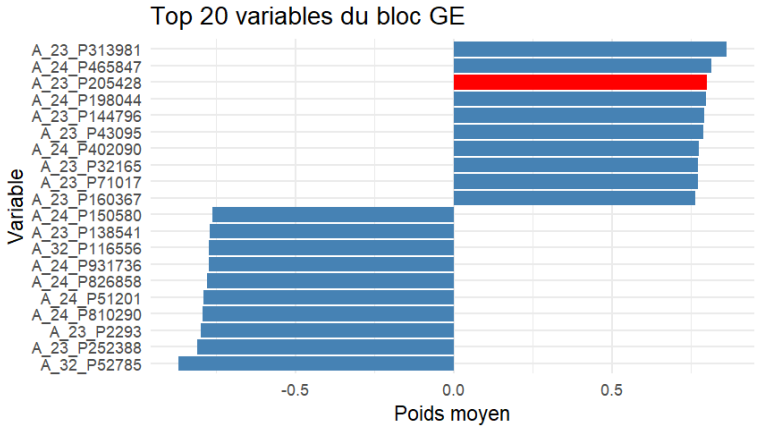

Figure 18: Plot des 20 coefficients les plus importants dans le block GE

On retrouve ici en rouge la variable A-23-P205428 qui sera énoncée juste après

3.4 Multiview

Multiview, tout comme RGCCA, est un package R permettant d'entrainer des modèles sur des données multi-modales hétérogènes (notre cas ici) tout en respectant les "points de vue" (d'où le nom du package) de chaque bloc de données (ici d'une part le bloc GE et de l'autre le bloc CGH). De plus Multiview fait de la sélection de variables et intègre de la cross-validation, le package est donc bien adapté à notre étude.

3.4.1 Formulation Mathématique

Pour deux vues de données, considérons les matrices de caractéristiques $X \in R^{n \times p_x}$, $Z \in R^{n \times p_z}$, et notre variable cible $y \in R^n$. Nous supposons ici que les colonnes de X et Z ont été centrées et réduites, et que y est centrée (nous pouvons donc omettre l'intercept ci-dessous). Pour une valeur fixée de l'hyperparamètre $\rho \geq 0$, `multiview` cherche à déterminer $\beta_X \in R^{p_x}$ et $\beta_Z \in R^{p_z}$ qui minimisent :

$$\frac{1}{2}\|y - X\beta_X - Z\beta_Z\|^2 + \frac{\rho}{2}\|(X\beta_X - Z\beta_Z)\|^2 + \lambda \left[(1-\alpha)(\|\beta_X\|_1 + \|\beta_Z\|_1) + \alpha \left(\frac{\|\beta_X\|_2^2}{2} + \frac{\|\beta_Z\|_2^2}{2} \right) \right].$$

La **pénalité d'accord** entre les 2 "vues" est donc controlée par le paramètre ρ :

1. $\rho = 0$ revient à faire fusionner les 2 blocs données avant l'entrainement (ce qui a été fait précédémment dans SparseLDA et la RL)
2. $\rho = 1$ revient à faire fusionner les résultats après l'entrainement indivuel de chaque blocs de données (méthode aussi utilisée avec la RL)

3. $\rho = 0.5$ a été gardé pour les résultats

α correspond au paramètres d'**elastic net mixing** permettant d'allier pénalité Ridge et pénalité Lasso.
Le λ optimal est trouvé avec la cross-validation proposée par multiview (cv.multiview)

3.4.2 Approche

Multiview ajuste des modèles logistiques, mais uniquement binaires! Or nous avons 3 classes... On contourne le problème en optant pour une stratégie **One vs ALL**, où on entraine une classe contre toutes les autres en ajustant un modèle binomial. On trouve aussi la possibilité de plus facilement interpréter les résultats: quelle variable influe sur quelle emplacement de la tumeur.

3.4.3 Code

Voici un exemple de code générique où au préalable on a chargé les blocs GE et CGH dans x_{list}, binarisé y en y_{bin} et chargé celui-ci.

```
cv_fit <- cv.multiview(
  x_list       = X_list,
  y            = y_bin,
  family       = binomial(),
  rho          = 0.5,
  trace.it     = 1,
  alpha=0.5,
  nfolds=4,
  type.measure = 'class')
```

On utilise ici comme paramètre de mesure pour lambda 'class' qui correspond à trouver λ tel que l'erreur de classification soit minimisée: $\lambda min = 0.22$ sera donc utilisé.

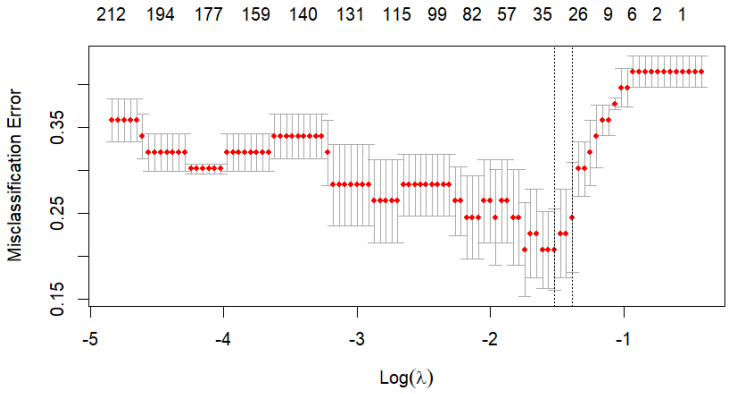

Figure 19: Graphe de λ fourni par la cross-validation avec comme cible la classe 'dipg' de y

On a observé que allier pénalité Lasso et Ridge avec donc $\alpha = 0.5$ diminuait l'erreur de classification (erreur plus grande avec uniquement du Lasso).

Une autre feature intéréssante de multiview qui permet de se faire une idée des coefficients est un plot des coefficients en faisant varier lambda (en utilisant néanmoins multiview sans cross-validation!):

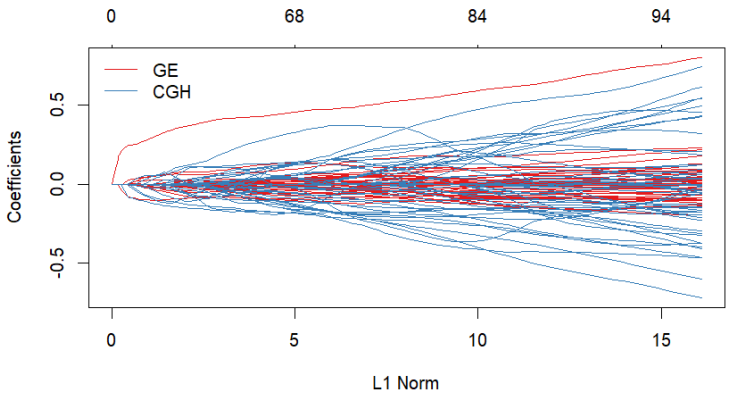

Figure 20: Plot des coefficients avec comme cible y='dipg'

Chaque courbe représente une variable du modèle, on observe bien le lien entre λ et la norme L_1: plus λ est petit, plus la contrainte est faible sur le modèle et le nombre de coefficients non nuls élevé.
Pour extraire au maximum les variables importantes, on utilisera λ_{1se} qui au correspond au λ tel qu'on a le moins de coefficients et le moins d'écart par rapport à l'erreur minimale lié à λ_{min} On remarque par exemple qu'un des paramètres du bloc GE a l'air de se démarquer peu importe la valeur de λ. Il s'agit en fait de A-23-P205428, une expression génique, qui apparaît comme paramètre dominant dans chaque modèle pour prédire la classe 'dipg'.
Il est le seul paramètre de GE avec autant de poids d'après le graphe.
On voit l'importance et l'utilité d'un tel graphe en première approximation pour avoir une vue d'ensemble.

3.4.4 Résultats

Enfin les résultats qui correspondent

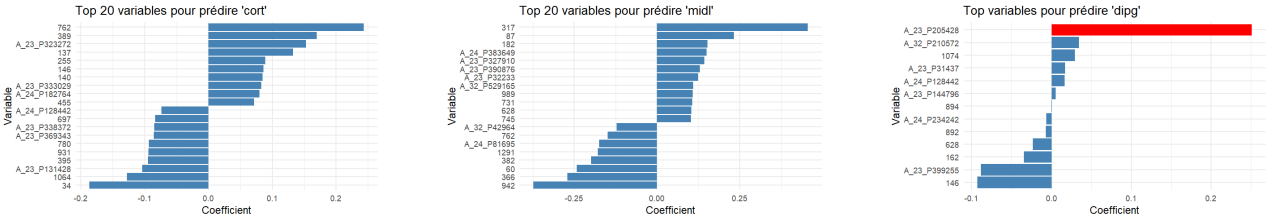

Figure 21: Poids et variables pour chaque classe

Il est intéréssant d'observer que le modèle a isolé uniquement 13 variables pour prédire la classe "dipg', tandis qu'il en a trouvé plus de 20 pour 'midl' et 'cort'. De plus à priori la donnée de 'A-23-P205428' permet de dire quasi à elle seule l'appartenance à 'dipg'.
On peut apprécier le fait qu'on ait réduit le nombre de dimensions de plus de 15000 !

4 Conclusion

Modèle	Sensibilité cort	Sensibilité dipg	Sensibilité midl
RGCCA	1	0.8	1
SparseLDA GE (CV successives, moyennement fiable)	1	0.86	0.36
RL + LASSO GE + CGH	1	1	0.45
RL + LASSO GE	0.93	0.93	0.64

Table 1: Récapitulatif sensibilité des modèles

Modèle	Spécificité cort	Spécificité dipg	Spécificité midl
RGCCA	1	1	0.9
SparseLDA GE	0.93	0.83	0.92
RL + LASSO GE + CGH	0.92	0.88	1
RL + LASSO GE	0.88	0.91	1

Table 2: Récapitulatif spéficité des modèles

Modèle	Accuracy
RGCCA	0.91
SparseLDA GE	0.81
RL + LASSO GE + CGH	0.88
RL + LASSO GE	0.87

Table 3: Récapitulatif ccuracy obtenues

Les 2 méthodes multiblocks Multiview et RGCCA sont similaires qualitativement: regarder comment chaque bloc est lié à la réponse (ou point de vue) et pondérer les coefficients de chaque bloc par l'importance de celui-ci. Néanmoins, peu nombreux sont les coefficients signifcatifs semblables entre les 2 méthodes. On a en effet entraîner 3 modèles différents en One vs All pour Multiview tandis qu'on a entrainé un modèle générale pour RGCCA. La classe 'midl' étant minoritaire, 11 patients comparé à 20 et 22 pour les 2 autres classes. Les modèles binaires de multiview sont donc qualitativement moins significatifs avec ces données.

La SparseLDA et la Régréssion Logistique couplé à du Lasso, déterminent certes des variables et permettent de se faire une idée potentielle, mais elles ne rendent pas compte de la corrélation entre blocs, on observe par exemple que pour la RL, seul des variables du blocs GE ressortent, car le bloc GE (15 702 variables) est bien plus important que CGH (1300). En effet, il y a en a beaucoup plus, le modèle aura plus de chance de les choisir.

5 Perspectives

L'un des principaux freins à une interprétation robuste des résultats réside dans la taille limitée de l'échantillon ($n = 53$ patients). Cette faible quantité de données peut engendrer un sur-apprentissage, en particulier dans des contextes de haute dimension comme celui du bloc GE (15 702 variables). Cela nous permettrait aussi d'avoir des tests de nos modèles plus fiables avec ces nouvelles données. Une perspective possible consisterait donc à collaborer avec d'autres centres hospitaliers ou laboratoires pour enrichir la base de données, même si cela est malheureusement difficile à faire en réalité.

Avec plus de temps, d'autres méthodes disponibles dans la littérature pourraient être également explorées.

De plus, le projet actuel repose sur l'analyse de deux blocs de données génétiques (GE et CGH). Toutefois, les tumeurs cancéreuses sont des pathologies complexes résultant d'interactions multi-niveaux (génétique, épigénétique, transcriptionnelle, phénotypique). Il serait donc pertinent d'intégrer des blocs supplémentaires.

References

[1] Line Clemmensen, Trevor Hastie, Daniela Witten, and Bjarne Ersbøll. Sparse discriminant analysis, 2011. Preprint.

[2] Daisy Yi Ding, Robert J. Tibshirani, Balasubramanian Narasimhan, Trevor Hastie, Kenneth Tay, and James Yang. *multiview: Cooperative Learning for Multi-View Analysis*. CRAN, 2023. R package version 0.8.

[3] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[4] Jerome H. Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2 edition, 2009.

[5] Fabien Girka, Eric Camenen, Claire Peltier, Anne Gloaguen, Vincent Guillemot, Louis Le Brusquet, and Arthur Tenenhaus. Multiblock data analysis with the rgcca package. *Journal of Statistical Software*, 100(1):1–22, 2023.

[6] Stéphanie Puget, Cathy Philippe, Dorine A. Bax, Bastien Job, Pascale Varlet, Marie-Pierre Junier, Felipe Andreiuolo, Dina Carvalho, Ricardo Reis, Lea Guerrini-Rousseau, Thomas Roujeau, and Jacques Grill. Mesenchymal transition and pdgfra amplification/mutation are key distinct oncogenic events in pediatric diffuse intrinsic pontine gliomas. *PLOS ONE*, 7(2):e30313, 2012.