SDMA - TP4
Agrégation de modèles
Analyse spectrale et clustering

Maha ELBAYAD

23 Décembre 2015

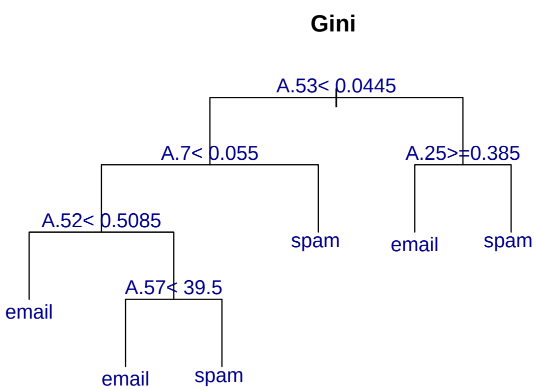
Agrégation des modèles

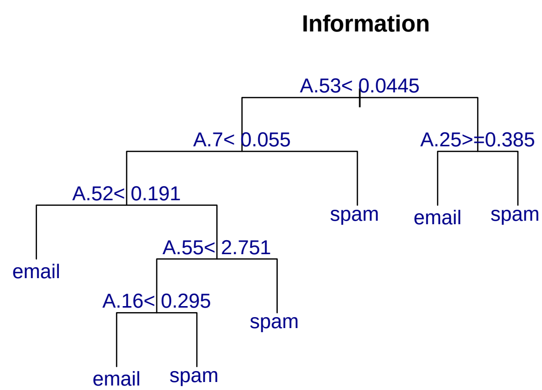
Analyse préliminaire

On dispose pour cette étude de 4601 observations avec 58 variables. D'après `spambasenames.txt`:

- A1:55 numériques à valeurs flottantes:
  - A1:48 `word_freq_WORD` i.e fréquence des mots `WORD` dans le mail.
  - A49:54 `char_freq_CHAR`, fréquence du caractère `CHAR` dans le mail, l'étude s'interesse ici aux carctères spéciaux {; ([!#$}
  - A55 longueur moyenne des séquences toutes en majuscules
- A56:57 numériques entières:
- A56 la longueur maximale d'une séquence toute en majuscules
- A57 le nombre total des lettres majuscules dans l'email
- spam: catégorique indiquant si le mail en question est un spam (1) ou pas (0).

Arbres de Classification





La variable la plus influente est `A53` qui correspond à la fréquence du caractère `$` dans l'email.

Seules 5 ou 6 variables interviennent dans cet arbre:

- Gini `A.25 A.52 A.53 A.57 A.7`: correspondant aux mots/caractètres: hp, !, $, remove et le nombre total des lettres majuscules.
- Entropie `A.16 A.25 A.52 A.53 A.55 A.7`: correspondant aux mots/caractètres: free, hp, remove, !, $, remove et la longueur moyenne des séquences toutes en majuscules.

Erreur de classification - Gini

Base d'apprentissage:

L'erreur de classification: **9.91%**

Taux des fausses alarames: **18.1%**

Taux des non-detections: **4.67%**

Base de test:

L'erreur de classification: **11%**

Taux des fausses alarames: **21%**

Taux des non-detections: **4.35%**

Erreur de classification - Information/Entropie

Base d'apprentissage:

		True	
		email	spam
Pred	email	1966	190
	spam	133	1161

L'erreur de classification: **9.36%**

Taux des fausses alarames: **14.1%**

Taux des non-detections: **6.34%**

Base de test:

		True	
		email	spam
Pred	email	650	78
	spam	39	384

L'erreur de classification: **10.2%**

Taux des fausses alarames: **16.9%**

Taux des non-detections: **5.66%**

Conclusion:

On constate que la perfomance des arbres de classification est limitée. De plus la construction de l'arbre dépend fortement des données d'apprentissage d'où l'intérêt de considérer des approches plus robustes comme le bagging ou le boosting.

Agrégation de modèles: Bagging

La fonction `bagging` de la libraire `ipred` génère par défaut 25 arbres.

Base d'apprentissage:

		True	
		email	spam
Pred	email	2098	7
	spam	1	1344

L'erreur de classification: **0.232%**

Taux des fausses alarames: **0.518%**

Taux des non-detections: **0.0476%**

Base de test:

		True	
		email	spam
Pred	email	668	46
	spam	21	416

L'erreur de classification: **5.82%**

Taux des fausses alarames: **9.96%**

Taux des non-detections: **3.05%**

Conslusion:

La performance avec un bagging de 25 arbres est remarquablement meilleure que celle d'un seul arbre.

Agrégation de modèles: Random Forest

Par défaul `randomForest()` génère 500 arbres en bootstrappant les données d'apprentissage, on va se limiter à ntree=25. La fonction choisit aléatoirement $\sqrt{p} \approx 7$ variables à considérer pour chaque split.

Base d'apprentissage:

	True	
Pred	email	spam
email	2099	14
spam	0	1337

L'erreur de classification: **0.406%**

Taux des fausses alarames: **1.04%**

Taux des non-detections: **0%**

Base de test:

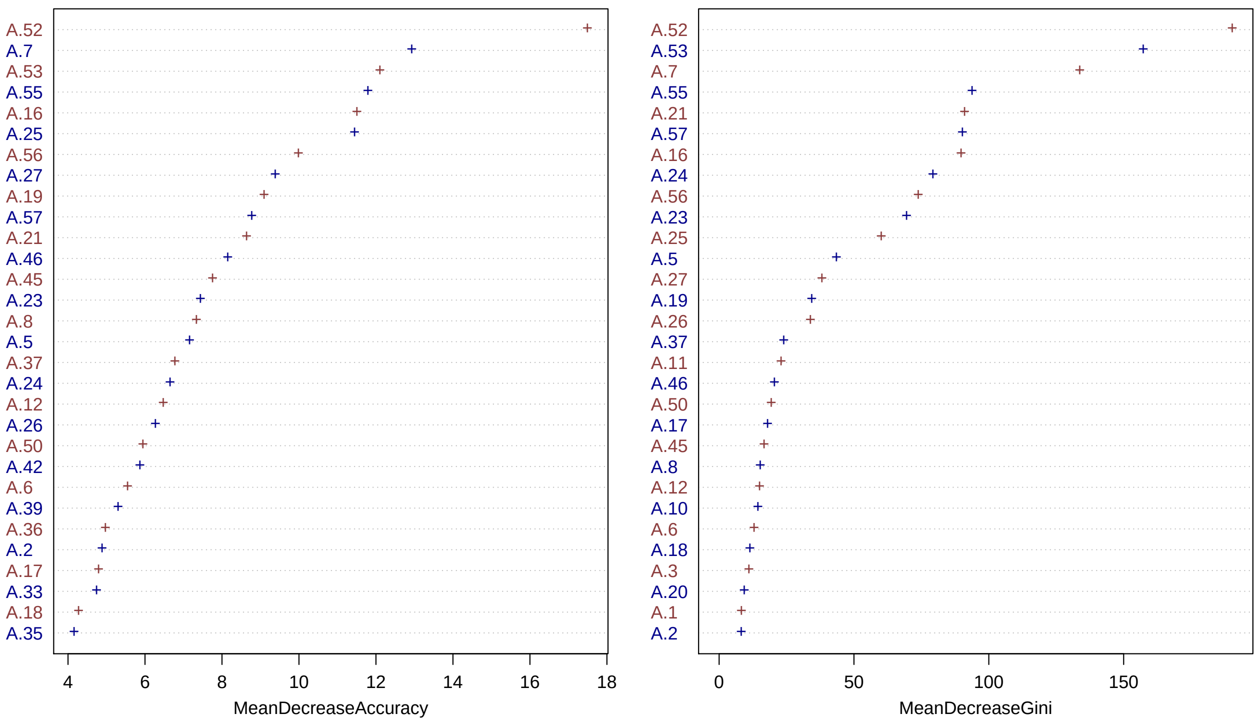
	True	
Pred	email	spam
email	672	44
spam	17	418

L'erreur de classification: **5.3%**

Taux des fausses alarames: **9.52%**

Taux des non-detections: **2.47%**

L'importance des variables



Conslusion:

Les variables les plus importantes de l'arbre de décision (CART): `A.53, A.7, A.52, A.57, A.25` figurent dans les 10 les plus importantes du Random Forest avec un ordre différent vu le bootstrapping des données d'apprentissage.

Côté performance, Random Forest se généralise mieux que le bagging avec 25 arbres.

Scoring: comparaison des modèles de classification

Base d'apprentissage - Logit:

	True		
		mail	spam
Pred	mail	2000	138
	spam	99	1213

L'erreur de classification: **6.87%**

Taux des fausses alarames: **10.2%**

Taux des non-detections: **4.72%**

Base de test - Logit:

	True		
		mail	spam
Pred	mail	658	49
	spam	31	413

L'erreur de classification: **6.95%**

Taux des fausses alarames: **10.6%**

Taux des non-detections: **4.5%**

Base d'apprentissage - LDA:

	True		
		email	spam
Pred	email	2002	298
	spam	97	1053

L'erreur de classification: **11.4%**

Taux des fausses alarames: **22.1%**

Taux des non-detections: **4.62%**

Base de test - LDA:

	True		
		email	spam
Pred	email	665	97
	spam	24	365

L'erreur de classification: **10.5%**

Taux des fausses alarames: **21%**

Taux des non-detections: **3.48%**

Base d'apprentissage - SVM:

	True		
		email	spam
Pred	email	2033	119
	spam	66	1232

L'erreur de classification: **5.36%**

Taux des fausses alarames: **8.81%**

Taux des non-detections: **3.14%**

Base de test - SVM:

	True		
		email	spam
Pred	email	665	45
	spam	24	417

L'erreur de classification: **5.99%**

Taux des fausses alarames: **9.74%**

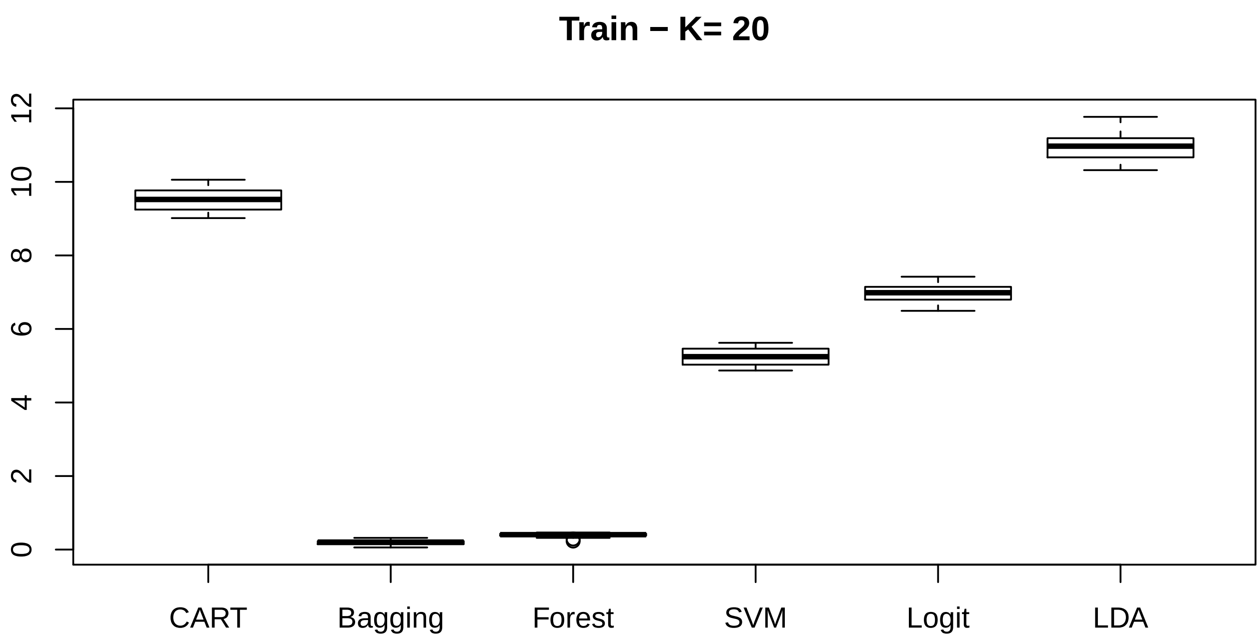
Taux des non-detections: **3.48%**

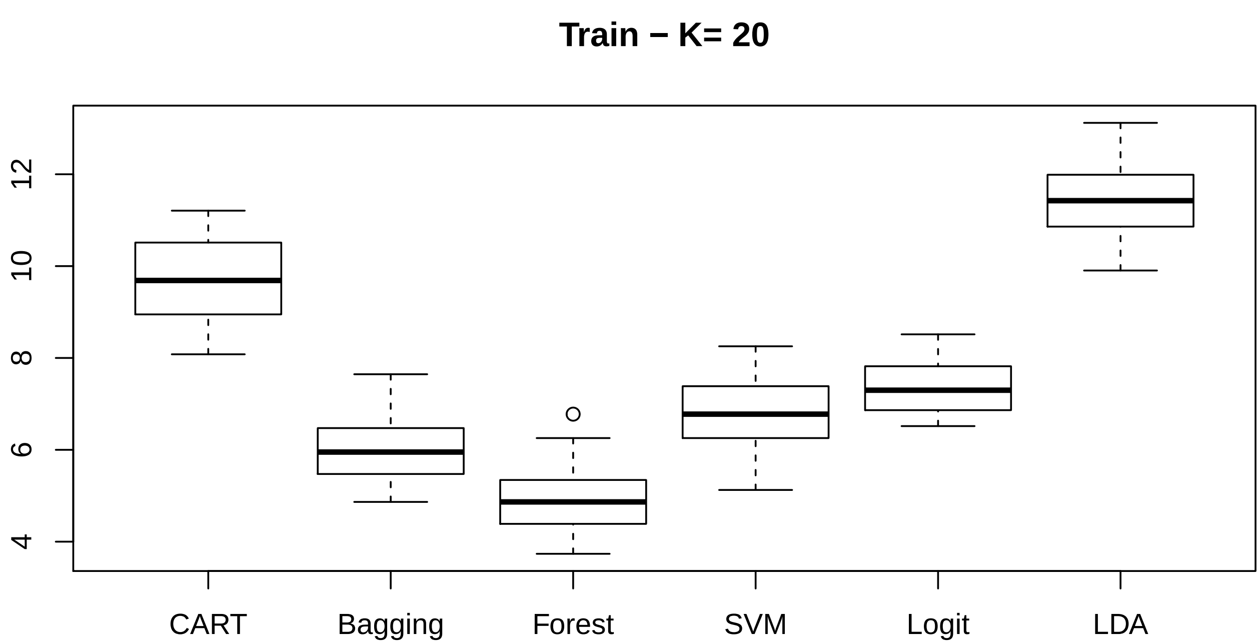
Conclusion

Modèle	Train: Erreur%	FP%	FN%	Test: Erreur%	FP%	FN%
CART	9.36	14.1	6.34	10.2	16.9	5.66
Bagging	0.232	0.518	0.0476	5.82	9.96	3.05
RF	0.406	1.04	0	5.3	9.52	2.47
logit	6.87	10.2	4.72	6.95	10.6	4.5
LDA	11.4	22.1	4.62	10.5	21	3.48
SVM	5.36	8.81	3.14	5.99	9.74	3.48

On constate que le Random Forest et le Bagging surpassent de loin les méthodes classqqiues telles le LDA et la régression logistiques (sans cross-validation pour tuner les modèles)

Comparaison des modèles de classification

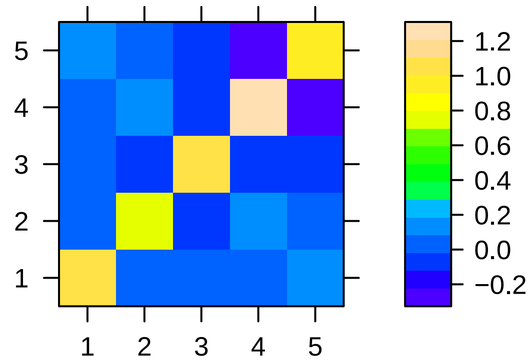




	CART	Bagging	Forest	SVM	Logit	LDA
Moyenne -Train %	9.51	0.197	0.399	5.24	6.97	10.9
Ecart-type -Train %	0.314	0.0656	0.0609	0.234	0.247	0.38
Moyenne -Test %	9.73	6.05	4.96	6.82	7.38	11.4
Ecart-type -Test %	0.941	0.777	0.777	0.798	0.615	0.812

Analyse spectrale et clustering

Exercice 1



On remarque que les variables $X_1..X_p$ sont corrélées.

On décompose S sous la forme $S = U\Sigma V^T$ où U, V deux matrices unitaires et Σ matrice diagonale à éléments diagonaux positifs. On compare les deux matrices U et V en évaluant la norme de leur différence, on choisit ici la norme de Frobenius. On peut aussi évaluer les valeurs propres de $U - V$ et calculer leur norme .

$$\frac{|U - V|_F}{|U|_F} = 6.7452 \times 10^{-16}$$

et

$$|\sigma(U - V)|_2 = 1.5100666 \times 10^{-15}$$

On remarque que:

$$Tr(\Sigma - S) = -4.4408921 \times 10^{-16}$$

Comme la matrice $S \in \mathbf{S}^p_{++}$, la décomposition en valeurs singulières est équivalente à la décompision en vecteurs propres. Ainsi:

$$\Sigma = \Lambda = diag(\sigma(S))$$

et

$$U = V$$

On peut donc écrire

$$S = U\Lambda U^T$$

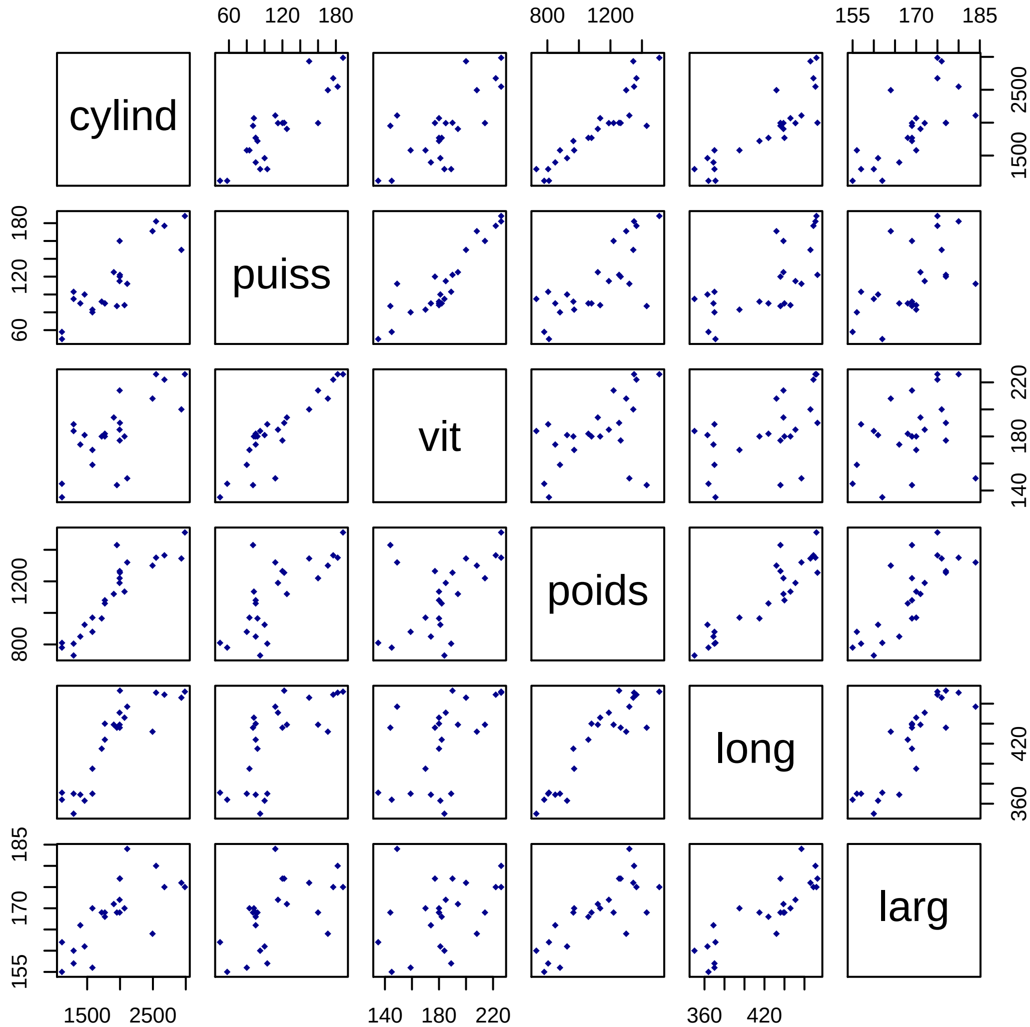
Numériquement:

$$|S - U\Sigma V^T|_F = 1.8068858 \times 10^{-15}$$

Exercice 2: Analyse en composantes principales

Préliminaires:

En analysant le contenu de `cardata.txt`, on constate qu'on dispose de 24 observations de 6 variables et qu'on ne dispose pas d'une variable cible. La commande `plot` affiche des nuages de points entre les 6 variables.



	cylind	puiss	vit	poids	long	larg
cylind	1					
puiss	0.86	1				
vit	0.69	**0.89**	1			
poids	**0.9**	0.75	0.49	1		
long	0.86	0.69	0.53	**0.92**	1	
larg	0.71	0.55	0.36	0.79	0.86	1

On remarque que les variables `(puiss,vit)`,`(poids,cylind)`,`(poids,long)` sont fortement corrélées.

ACP:

On effectue une analyse en composantes principales sur les variables centrées réduites. La sortie de `prcomp` a les attributs suivants:

- `sdev`:

  la déviation des composantes principales i.e., $\sqrt{\lambda_i(cor(X))}$.

- `rotation`:

  Matrices des composantes principales ~ vecteurs propres.

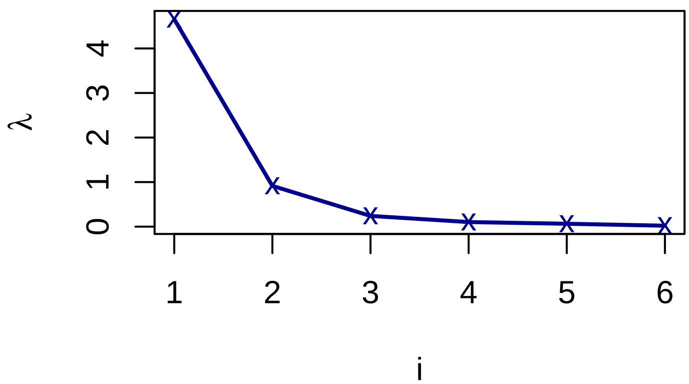
- `center, scale`:

  Les paramètres de centrage et de redimensionnement ou le booléan FALSE si on précise `center=F,scale=F`

- `x`:

  La projection des données d'entrée sur l'espace des PC.

Etude des Valeurs propres:



Variance expliquée par chaque axe principal i et chaque plan {1..i}:

	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
Var par axe%	77.60	15.25	4.01	1.71	1.08	0.35
Var par plan%	77.60	92.85	96.86	98.57	99.65	100.00

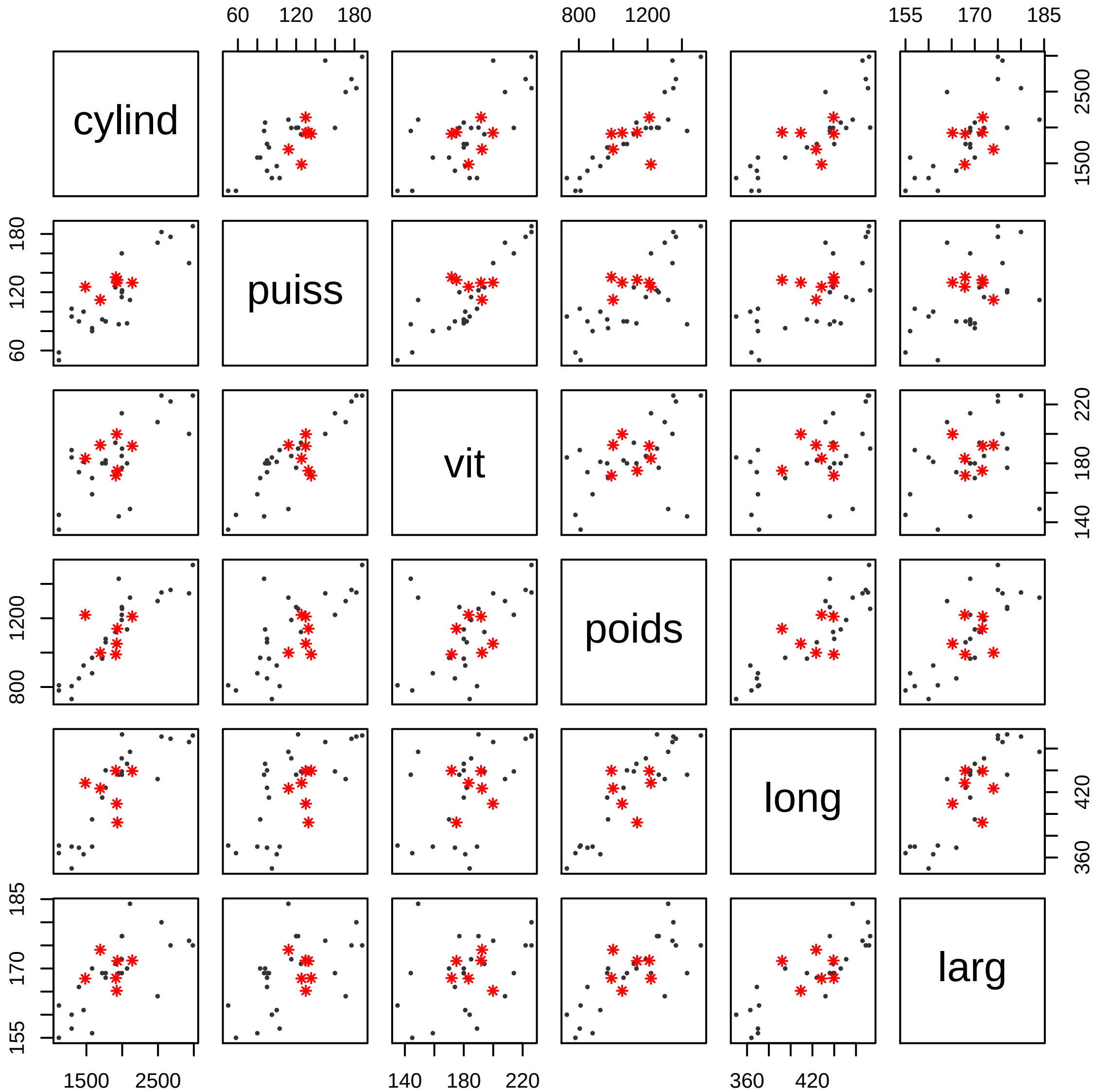
Pour expliquer 95% de variance on se contentera des **3** premières valeurs propres, pour 98% on ajoutera la 4ème aussi.

Etude des Vecteurs propres:

On liste les coordonnées des composantes principales dans la base centrée réduite et dans la base initiale puis on les visualise en les superposant en rouge aux nugaes des points précédents.

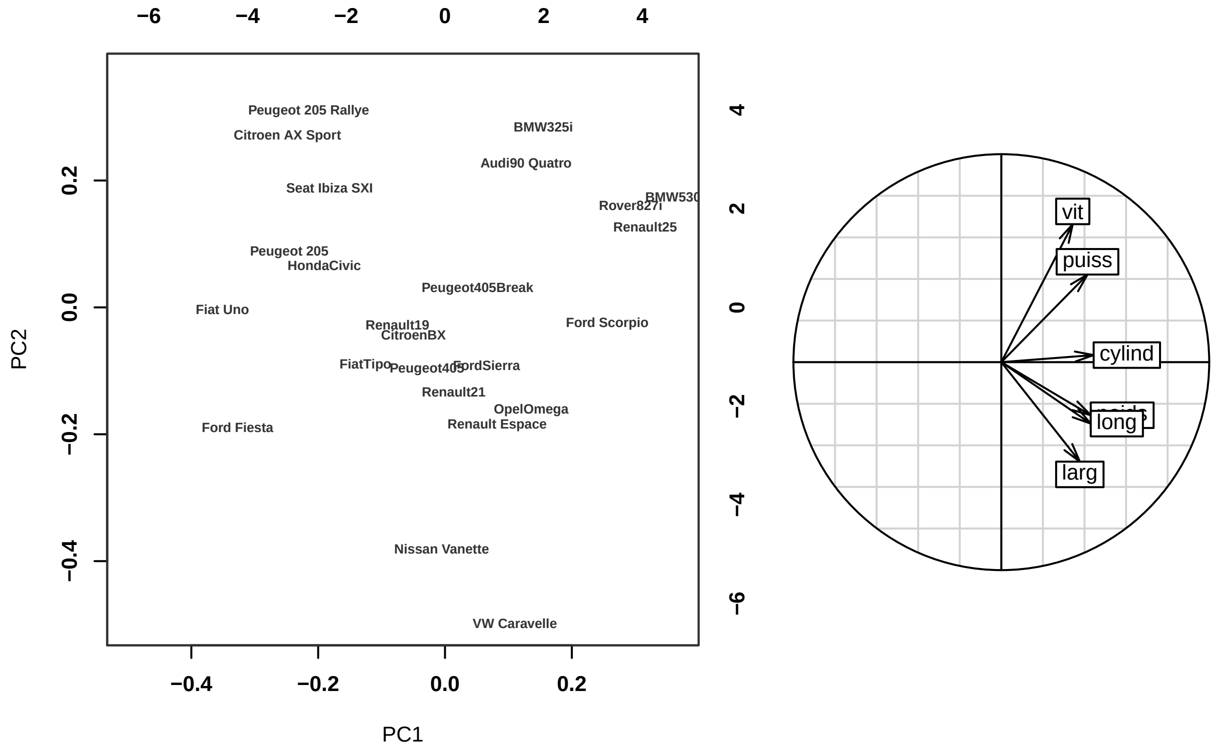
	PC1	PC2	PC3	PC4	PC5	PC6
cylind	0.44	0.03	-0.40	0.05	-0.80	0.01
puiss	0.41	0.42	-0.04	0.49	0.31	0.56
vit	0.34	0.66	0.37	-0.32	0.01	-0.45
poids	0.43	-0.26	-0.48	0.12	0.47	-0.53
long	0.43	-0.30	0.04	-0.71	0.17	0.44
larg	0.38	-0.48	0.68	0.37	-0.13	-0.12

	PC1	PC2	PC3	PC4	PC5	PC6
cylind	2140.62	1924.06	1694.21	1932.53	1484.54	1911.86
puiss	129.74	130.00	112.13	132.66	125.56	135.45
vit	191.75	199.81	192.41	175.02	183.26	171.73
poids	1209.93	1052.07	999.27	1139.19	1219.67	989.74
long	439.37	409.36	423.40	392.16	428.44	439.67
larg	171.72	165.17	174.05	171.63	167.83	167.92

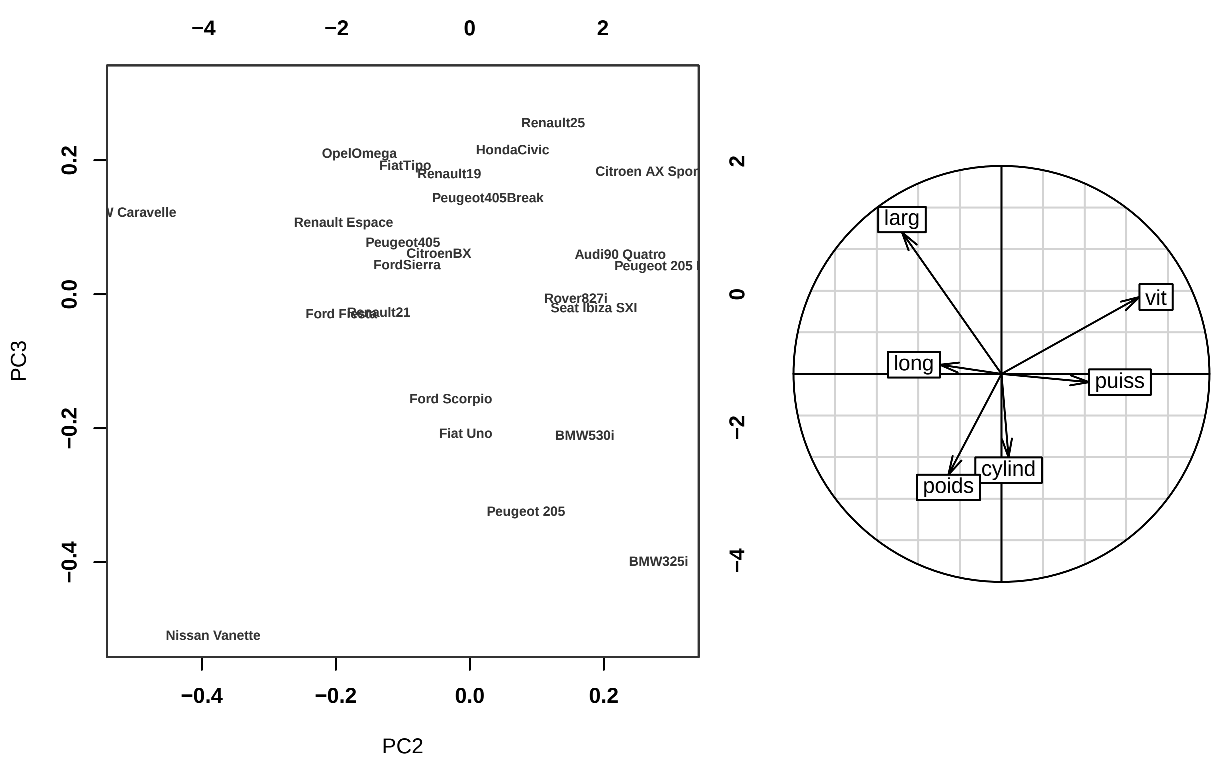


Cercle des corrélations:

Premier plan factoriel



Second plan factoriel



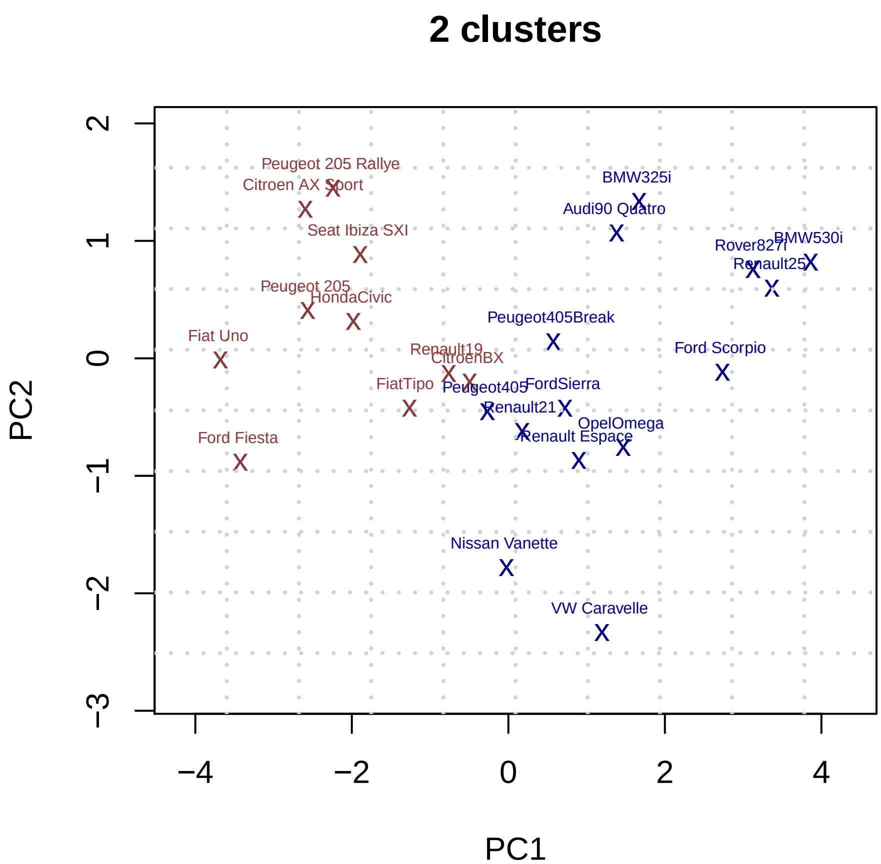
On constate que les points sont uniformément répartis sur le 1er plan, un peu moins sur le second plan. Sur le premier cercle de corrélation, on voit que le 1er quadrant regroupe les voitures à grandes `(puiss,vit)` opposé au 2ème quadrant et le quatrième quadrant comprend les voitures à grand volume `(poids,larg,long)` opposé au 3ème quadrant. Les covariables sont dispersés sur le 2ème cercle, mais comme certaines sont corrélées, des regions du plan restent vides.

Classification non-supervisée - kmeans:

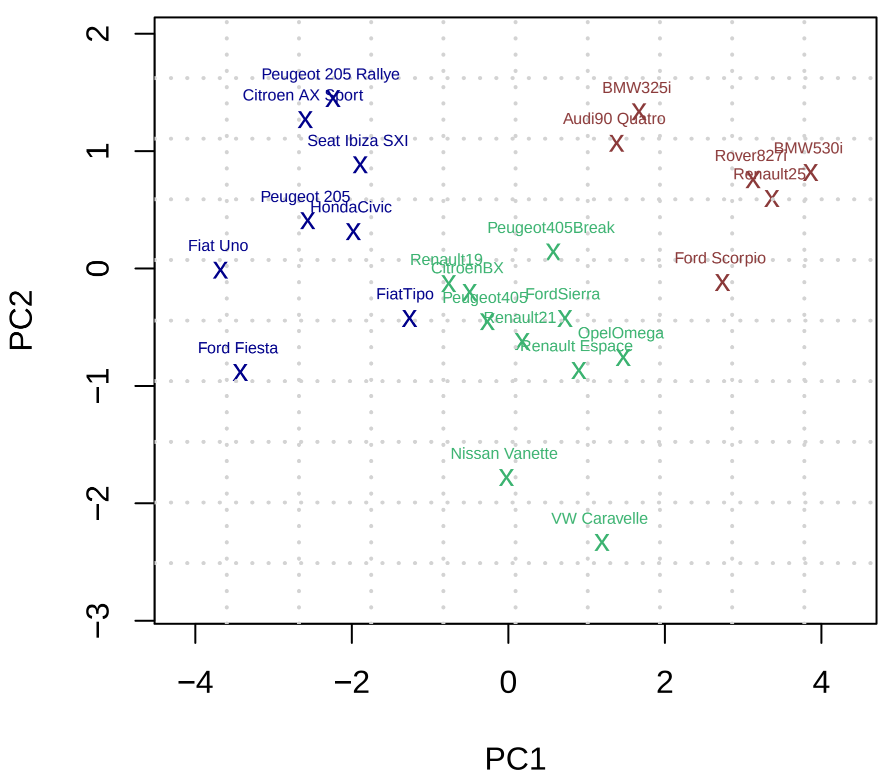
On applique l'algorithme `kmeans` avec 2,3,4 clusters. La sortie de `kmeans` a les attributs suivants:

- `centers`: Les centroides de chaque cluster.
- `cluster`: L'attribution des observations aux clusters.
- `withins`: La somme des carrés dans chaque cluster.
- `betweenss`: La somme des carrés entre les clusters.

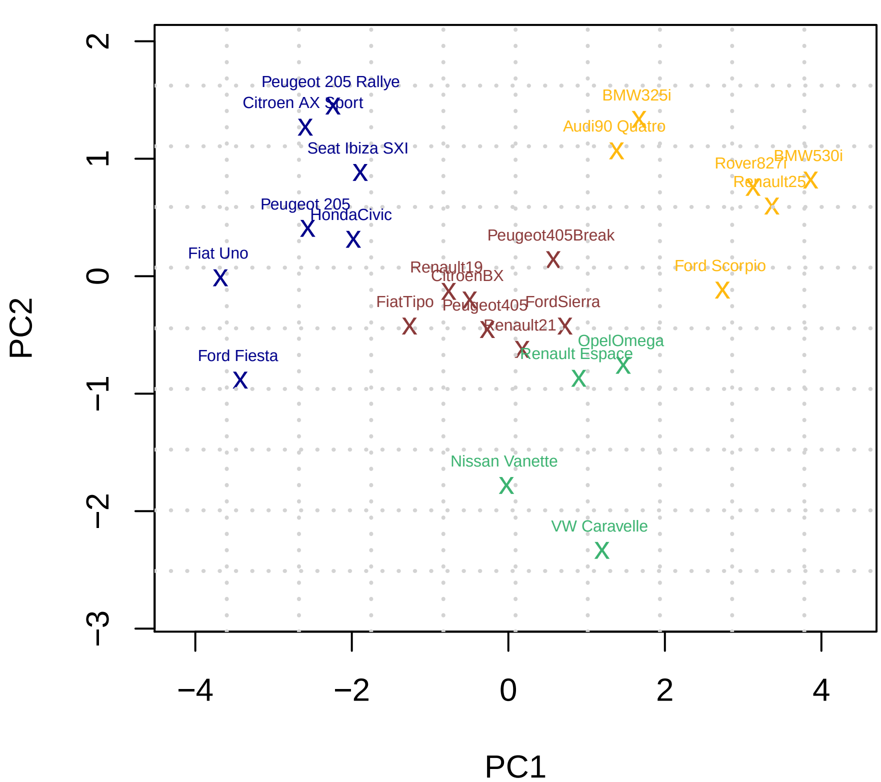
K	SC1	SC2	SC3	SC4	SC-moyenne	SC
4	8.6414	4.462	5.7266	8.4314	6.8153	110.74
3	11.285	8.4314	14.647	-	11.454	103.64
2	44.073	17.683	-	-	30.878	76.244



3 clusters

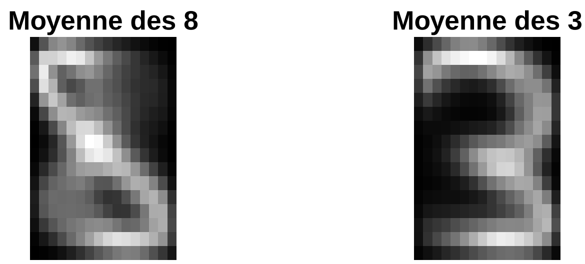


4 clusters



On remarque que les clusters sont déjà regroupés dans le premier plan factoriel et sont linéairement séparables. Avec 2 clusters, on pourrait dire que $C1 = \{x|PC1(x) \geq 0\}$ et $C2 = \{x|PC1(x) \leq 0\}$. On conclut du tableau que le meilleur choix de k serait **k=3** avec la plus faible $SC + SC_{moyenne}$.

Exercice 3: Caractères manuscrites



On applique `svd` à X la matrice des données, les vecteurs propres à droite V vérifient:

$$Xv = \lambda v$$

De plus, avec $X = USV^T$, $U^T U = V^T V = I$:

$$X^T X = VSU^T USV^T = V(S^T S)V^T = VDV^T$$

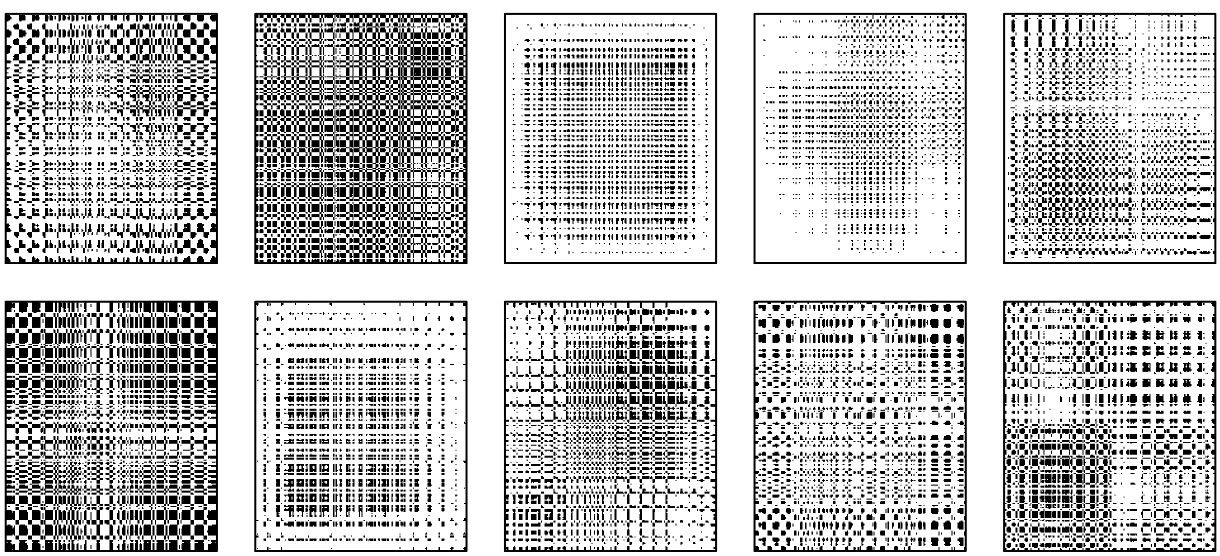
Où $D = S^2$ est diagonale.

Ainsi $(X^T X)V = VD$ i.e V sont des vecteurs propres de $X^T X$ que l'on calcule aussi avec `eigen` appliquée à la matrice de covariance $C \propto X^T X$

Numériquement $\| |V_{eig-d3}| - |V_{svd-d3}| \|_F = 3.98 \times 10^{-12}$

Modes propres:

La 1ère ligne regroupe les 5 premiers modes propres de "3" et la deuxième ceux de "8".



Matrice de projection sur le sous-espace vectoriel engendré par les 5 premières composantes principales:

On cosnidère U la matrice des composantes principales. La matrice de projection sur l'espace des composantes est définie par:

$$P = U(U^T U)^{-1} U^T = UU^T \text{ comme } U^T U = I_5$$

P est une matrice de projection carrée de taille (256x256), symétrique et idempotente ($P^2 = P$).

Pour $P_5 = UU^T$, où U la matrices des 5 composantes, on vérifie numériquement que:

$\|P_5^2 - P_5\|_F = 1.2343 \times 10^{-15}$ et $P_5^T = P_5$

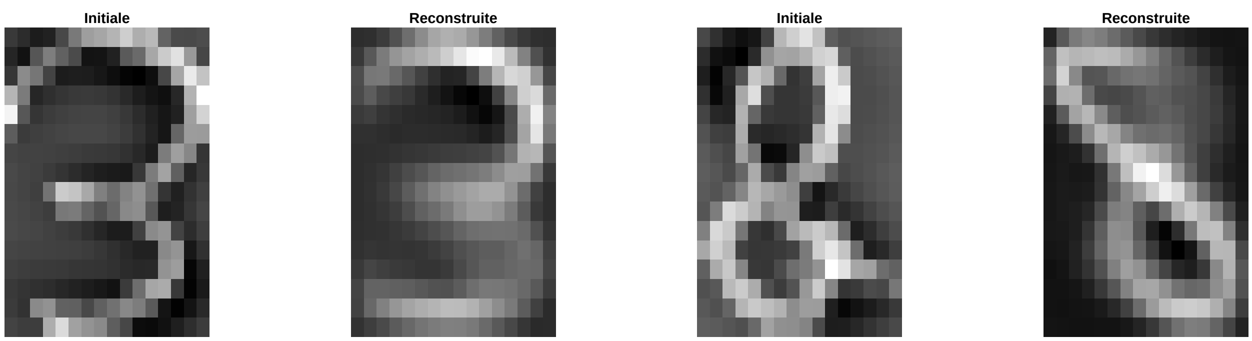
On projette les images centrées normalisées dans l'espace des 5 premières composantes principales:

$$\forall i \; p_{5,i} = U^T Image_{cn}^{(i)}$$

Pour reconstruire les images de base on applique[1]:

$$\hat{Image}^{(i)} = \mathbb{E}[Image_3] + \sigma[Image_3] \odot (Up_{5,i})$$

Exemple:



Il suffit donc de stocker le vecteur des moyennes (256×1), le vecteur des écarts-types (256×1), la matrice des composantes U (256×5) et les projections $(p_{5,i})_i$ chacune de dimension 5×1 ce qui permet de réduire l'espace de stockage de manière considérable (Pour 1000 images on passe de 256000# à 6792#).

[1] $\odot$ dénote la multiplication terme à terme