Projet du cours « Introduction au machine learning »

ROHIMUN SHAKIL

Utilisation des packages :

```
library(MASS)
library(e1071)     # Pour Naive Bayes
library(dplyr)    # Pour les variables aléatoire
library(ggplot2) # Meilleur rendu visuel pour les graphiques
```

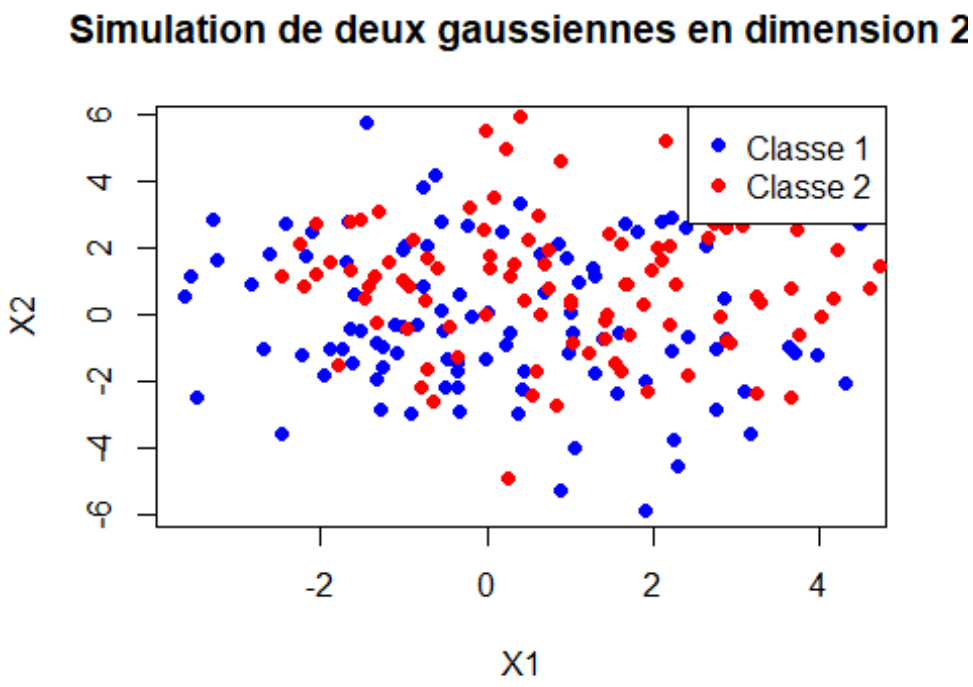
Etude de simulation

Vous simulerez deux gaussiennes en dimension 2 de matrice de variance $\sigma^2 I$, et de vecteurs moyennes respectifs $\mu_1^T = (0,0)$ et $\mu_2^T = (\epsilon, \epsilon)$.

1) Evaluer empiriquement l'erreur de classification commise par la méthode un classifieur de Bayes naif en fonction de ϵ lorsque les deux classes sont de proportion identique et que le nombre d'observations total est n = 200.

Avant de répondre à la question, pour avoir une meilleur idée de ce que représente ces classes, je décide de les représenter selon des paramètres précis.

```
sigma <- 2
epsilon <- 1

mu1 <- c(0, 0)
mu2 <- c(epsilon, epsilon)

matrice_covariance <- diag(sigma^2, 2)

classe1 <- mvrnorm(n = 100, mu = mu1, Sigma = matrice_covariance)
classe2 <- mvrnorm(n = 100, mu = mu2, Sigma = matrice_covariance)

plot(classe1, col = "blue", pch = 19, xlab = "X1", ylab = "X2", main =
"Simulation de deux gaussiennes en dimension 2")
points(classe2, col = "red", pch = 19)
legend("topright", legend = c("Classe 1", "Classe 2"), col = c("blue",
"red"), pch = 19)
```



Après avoir eu un visuel de ce que représente les classes, on répond à la question avec une fonction de paramètre epsilon pour obtenir mes deux classes. On choisit $\sigma^2 = 3$.

```
n = 200

simulation_gaussienne <- function(epsilon) {
  matrice_covariance <- 3 * diag(2)
  mu1 <- c(0, 0)
  mu2 <- c(epsilon, epsilon)

  classe1 <- mvrnorm(n = n/2, mu = mu1, Sigma = matrice_covariance)
  classe2 <- mvrnorm(n = n/2, mu = mu2, Sigma = matrice_covariance)

  return(list(classe_1 = classe1, classe_2 = classe2))
}
```

On crée un vecteur d'epsilon allant de 1 à 10 pour évaluer l'erreur et une liste vide qui représentera la liste des erreurs en fonction de epsilon. On utilise la fonction "set.seed(123)" pour la reproductibilité des résultats car les variables d'entraînements et de tests sont choisit de manière aléatoire.

```
valeur_epsilon <- seq(1, 10, by = 1)
resultats_liste <- list()

for (epsilon in valeur_epsilon) {
  data <- simulation_gaussienne(epsilon)
```

```
  all_data <- rbind(data$classe_1, data$classe_2)
  labels <- rep(c("classe 1", "classe 2"), each = 100)
  dataset <- data.frame(X1 = all_data[, 1], X2 = all_data[, 2], Class =
labels)

  set.seed(123)

  training.versus.testing <- rbinom(100, 1, 1/2)
  training.versus.testing <- as.factor(training.versus.testing)
  levels(training.versus.testing) <- c("test", "train")
  Z <- data.frame(dataset, training.versus.testing)
  Z.split <- split(Z, training.versus.testing)
  Z.train <- Z.split$train
  Z.test <- Z.split$test

  naive_bayes_classifier <- naiveBayes(Class ~ X1 + X2, data = Z.train)
  predictions <- predict(naive_bayes_classifier, newdata = Z.test)
  prediction_classe <- as.factor(predictions)
  erreur_test <- mean(prediction_classe != Z.test$Class)
  resultats_liste[[as.character(epsilon)]] <- list(data = data, erreur_test =
erreur_test)
}
for (result in resultats_liste) {
  print(result$erreur_test)
}
```

```
## [1] 0.4150943
## [1] 0.2264151
## [1] 0.1320755
## [1] 0.05660377
## [1] 0.009433962
## [1] 0.009433962
## [1] 0.009433962
## [1] 0
## [1] 0
## [1] 0
```

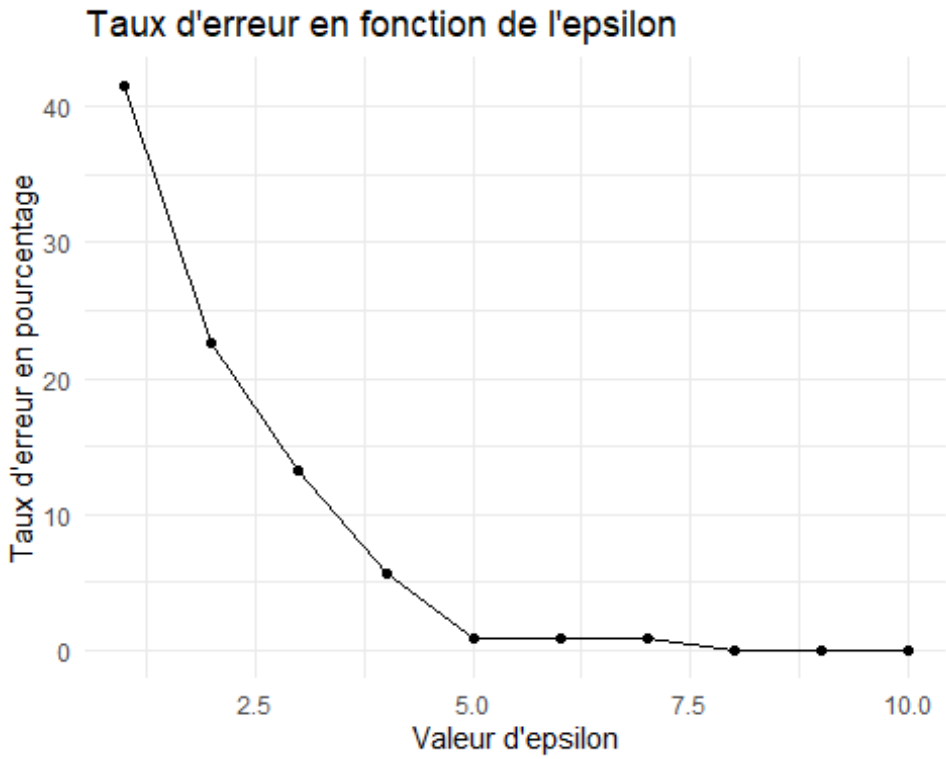
```
resultats_dataframe <- data.frame(epsilon =
as.numeric(names(resultats_liste)),
                                  erreur_test = sapply(resultats_liste,
function(x) x$erreur_test))
```

On constate qu'à partir de $\epsilon = 8$ l'erreur de prédiction est nulle ou suffisamment petite pour la considérer comme nulle.

On trace la courbe d'erreur en fonction d'epsilon.

```
ggplot(resultats_dataframe, aes(x = epsilon, y = erreur_test*100)) +
  geom_line() +
  geom_point() +
  labs(title = "Taux d'erreur en fonction de l'epsilon",
```

```
      x = "Valeur d'epsilon",
      y = "Taux d'erreur en pourcentage") +
 theme_minimal()
```



Ce résultat est logique car plus le vecteur moyen de la classe 2 s'éloigne de la classe 1 plus il est facile de distingué les deux classes pour l'algorithme.

2) Evaluer théoriquement l'erreur de classification d'un classifieur de Bayes (lois théoriques connues) en fonction de ϵ.

a) Montrer que la fonction de décision prend la forme :

$$\hat{f}(x) = \begin{cases} 1 & \text{si } A = (\mu_1 - \mu_2)^t \Sigma^{-1} \left(x - \frac{1}{2}(\mu_1 - \mu_2) \right) > \log \frac{\pi_1}{\pi_2}, \\ 2 & \text{sinon.} \end{cases}$$

La densité de probabilité conditionnelle pour une distribution gaussienne multivariée est donnée par :

$$\forall i \in \{1,2\} \quad P(X \mid C_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(X - \mu_i)^T \Sigma_i^{-1} (X - \mu_i) \right)$$

Avec : - X est le vecteur aléatoire représentant les caractéristiques du point à classifier - μ_i est le vecteur moyen de la classe C_i
- Σ_i est la matrice de covariance de la classe C_i - d est la dimension de X.

On prend le logarithme des densité conditionnelles, ce qui donne :

$$\forall i \in \{1,2\} \quad \log P(X \mid C_i) = -\frac{d}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma_i| - \frac{1}{2}(X-\mu_i)^T\Sigma_i^{-1}(X-\mu_i)$$

On obtient A avec la soustraction des logarithmes des deux classes :

$$A = \log\big(\pi_2 P(X \mid C_2)\big) - \log\big(\pi_1 P(X \mid C_1)\big)$$

Avec π_i les probabilités a priori des classes C_i

Comme les deux gaussiennes ont la même matrice de variance covariance, on obtient :

$$A = (\mu_1 - \mu_2)^t \Sigma^{-1}\left(x - \frac{1}{2}(\mu_1 - \mu_2)\right) - \log\frac{\pi_1}{\pi_2}$$

On définit donc la fonction de décision avec :

$$\hat{f}(x) = \begin{cases} 1 & \text{si } A > 0, \\ 2 & \text{sinon.} \end{cases}$$

D'où le résultat souhaité.

b) Si X appartient à la classe 1, montrer que :

$$A \sim \mathcal{N}\left(\frac{1}{2}\delta^2, \delta^2\right)$$

et donner la forme de δ.

Présentation des données

Vos données

Installer le package mlbench

Vous considérerez le jeu de données Ozone: library(mlbench); data(Ozone).

```
library(mlbench)
```

```
## Warning: le package 'mlbench' a été compilé avec la version R 4.3.2
```

```
data(Ozone)
```

Présentez le jeu de données et le décrire brièvement en utilisant des techniques de statistiques descriptives univariées et bivariées (histogrammes, boxplot, pair plot. . .).

```
summary(Ozone)
```

```
##        V1            V2      V3           V4              V5
##  1      : 31   1      : 12   1:52   Min.   : 1.00   Min.   :5320
##  3      : 31   2      : 12   2:52   1st Qu.: 5.00   1st Qu.:5700
##  5      : 31   3      : 12   3:52   Median : 9.00   Median :5770
##  7      : 31   4      : 12   4:53   Mean   :11.53   Mean   :5753
```

```
##  8      : 31   5      : 12   5:53   3rd Qu.:16.00   3rd Qu.:5830
##  10     : 31   6      : 12   6:52   Max.   :38.00   Max.   :5950
##  (Other):180   (Other):294   7:52   NA's   :5       NA's   :12
##        V6               V7              V8              V9
##  Min.   : 0.000   Min.   :19.00   Min.   :25.00   Min.   :27.68
##  1st Qu.: 3.000   1st Qu.:49.00   1st Qu.:51.00   1st Qu.:49.73
##  Median : 5.000   Median :65.00   Median :62.00   Median :57.02
##  Mean   : 4.869   Mean   :58.48   Mean   :61.91   Mean   :56.85
##  3rd Qu.: 6.000   3rd Qu.:73.00   3rd Qu.:72.00   3rd Qu.:66.11
##  Max.   :11.000   Max.   :93.00   Max.   :93.00   Max.   :82.58
##                   NA's   :15      NA's   :2       NA's   :139
##       V10            V11             V12             V13
##  Min.   : 111   Min.   :-69.0   Min.   :27.50   Min.   :  0.0
##  1st Qu.: 890   1st Qu.:-10.0   1st Qu.:51.26   1st Qu.: 70.0
##  Median :2125   Median : 24.0   Median :62.24   Median :110.0
##  Mean   :2591   Mean   : 17.8   Mean   :60.93   Mean   :123.3
##  3rd Qu.:5000   3rd Qu.: 45.0   3rd Qu.:70.52   3rd Qu.:150.0
##  Max.   :5000   Max.   :107.0   Max.   :91.76   Max.   :500.0
##  NA's   :15     NA's   :1       NA's   :14
```

Présentation du jeu de données : Ozone.

Il s'agit d'un jeu de données avec 13 composantes :

- V1 : Un nombre allant de 1 à 12 correspondant au mois où la donnée à été prélevé. (Variable qualitative)
- V2 : Un nombre allant de 1 à 31 correspondant au date où la donnée à été prélevé. (Variable qualitative)
- V3 : Un nombre allant de 1 à 7 correspondant au jour où la donnée à été prélevé. (Variable qualitative)
- V4 : La concentration maximale quotidienne d'ozone mesurée en moyenne sur une heure. (Variable quantitative)
- V5 : La hauteur de la pression atmosphérique à 500 millibars mesurée à la base aérienne de Vandenberg en mètre. (Variable quantitative)
- V6 : La vitesse du vent mesurée à l'aéroport de Los Angeles en mille par heure. (Variable quantitative)
- V7 : Le pourcentage d'humidité mesuré à l'aéroport de Los Angeles. (Variable quantitative)
- V8 : La température mesurée à Sandburg, en Californie en degré Fahrenheit. (Variable quantitative)
- V9 : La température mesurée à El Monte, en Californie en degré Fahrenheit. (Variable quantitative)

- V10 : La hauteur de la base d'inversion mesurée à l'aéroport de Los Angeles en pieds. (Variable quantitative)
- V11 : Le gradient de pression entre LAX et Daggett, en Californie en millimètres de mercure. (Variable quantitative)
- V12 : La température à la base d'inversion mesurée à l'aéroport de Los Angeles en degré Fahrenheit. (Variable quantitative)
- V13 : La visibilité mesurée à l'aéroport de Los Angeles en miles. (Variable quantitative)

Remarque 1 : La hauteur de la base d'inversion représente la distance verticale entre la surface terrestre et le point où une inversion thermique commence dans l'atmosphère. Une inversion thermique est un phénomène où la température de l'air augmente avec l'altitude au lieu de diminuer normalement. Cette mesure est importante pour évaluer la qualité de l'air, car une base d'inversion basse peut piéger les polluants près du sol, impactant ainsi la qualité de l'air dans une région.

Remarque 2 : Le gradient de pression mesure la variation de la pression atmosphérique sur une distance horizontale donnée. Calculé comme la différence de pression divisée par la distance, il indique à quelle vitesse la pression change dans l'atmosphère sur l'horizontalité. Un gradient élevé signifie une variation rapide de la pression sur une courte distance, tandis qu'un gradient bas indique une variation plus graduelle. En météorologie, un fort gradient de pression est associé à des conditions dynamiques, influençant les vents et les modèles météorologiques régionaux.

On constate que nous avons 3 variables qualitatives et 10 variables quantitatives.

Le but est de prédire la variable V4 (La concentration maximale quotidienne d'ozone mesurée en moyenne sur une heure), par rapport aux 9 autres variables quantitatives.

Dans un premier temps essayons de "nettoyé" la base de donnée, le but étant de retirer les lignes (ie les données prise une certaine date) qui ont peu d'observation.

On définit un seuil pour laquelle si la ligne contient plus de valeurs manquantes que la valeur du seuil donnée on retire la ligne. On regarde le nombre de valeur manquante par lignes en fonction du seuil.

```
seuil <- 1
valeur_NA <- rowSums(is.na(Ozone))
indices_a_supprime <- which(valeur_NA > seuil)
length(indices_a_supprime)
```

```
## [1] 21
```

On remarque si on met le seuil à 0, on a 163 lignes à supprimer ce qui risque de biaisé notre jeu de donnée, en revanche lorsque le seuil est égale à 1, on doit supprimer uniquement 21 lignes ce qui me paraît assez petit pour ne pas biaisé mon jeu de donnée. Donc je supprime ces 21 lignes.

```r
indices_a_garder <- which(valeur_NA <= seuil)
Ozone_1 <- Ozone[indices_a_garder, ]
nrow(Ozone_1)

## [1] 345
```

La base de donnée Ozone ayant 366 lignes pour les 366 jours de l'année, on a bien supprimé les 2 lignes que l'on souhaitait.

Maintenant pour éviter des problème future, notamment pour l'utilisation du K-means, on va remplacer les valeurs manquantes (NA) par la moyenne des valeurs de sa colonnes (ie la moyenne des valeurs observées). Les colonnes V1, V2, V3 ne peuvent pas avoir de valeurs manquantes car il s'agit des dates d'observation.

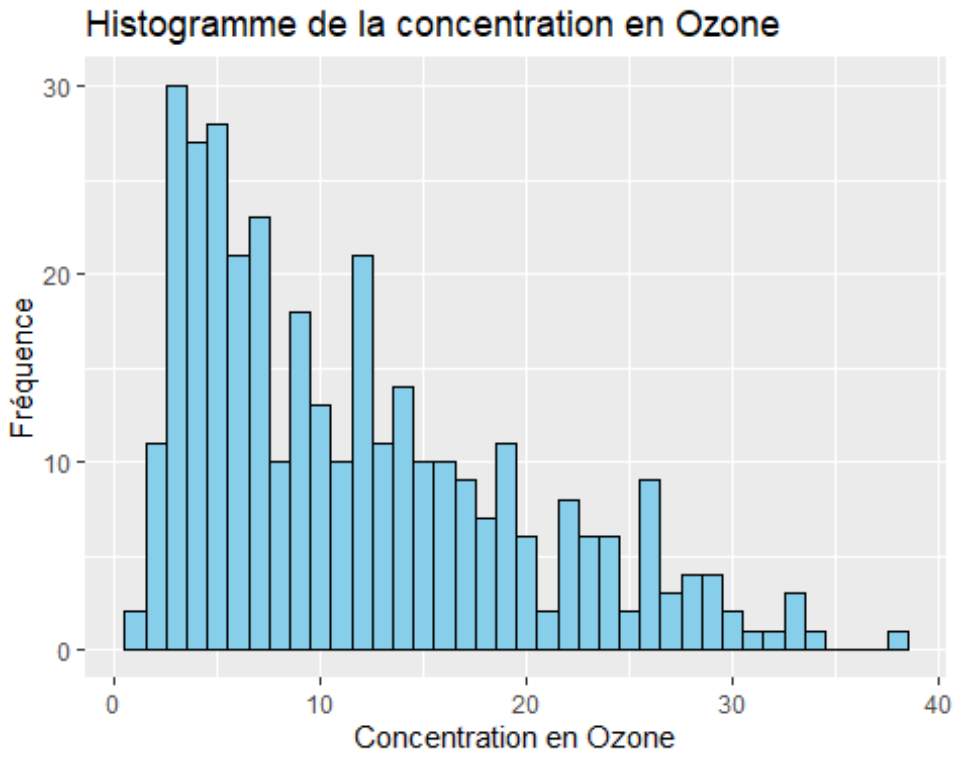
```r
for (i in 4:13) {
  if (any(is.na(Ozone_1[,i]))) {
  moyenne <- mean(as.numeric(Ozone_1[,i]), na.rm = TRUE)
  Ozone_1[,i] <- replace(Ozone_1[,i], is.na(Ozone_1[,i]), moyenne)
}
}
```

Vérifions qu'il ne reste aucune valeurs NA.

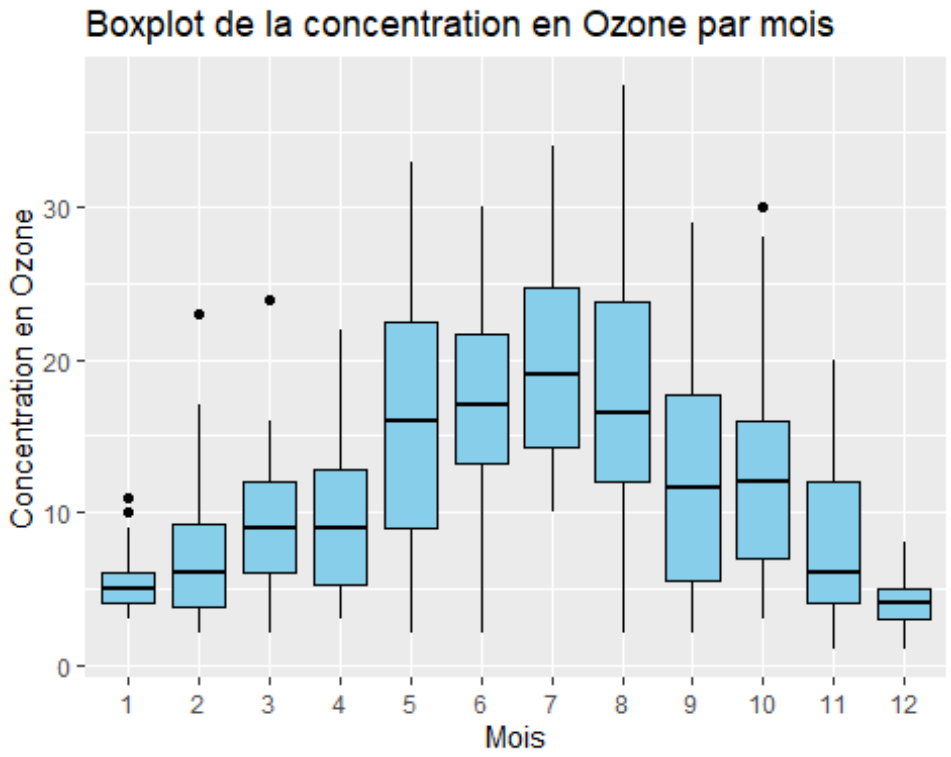
```r
n = 0
for (i in 1:13) {
  if (any(is.na(Ozone_1[,i]))) {
  n <- n + 1
}
}
n

## [1] 0
```

Passons à la description du jeu de donnée :

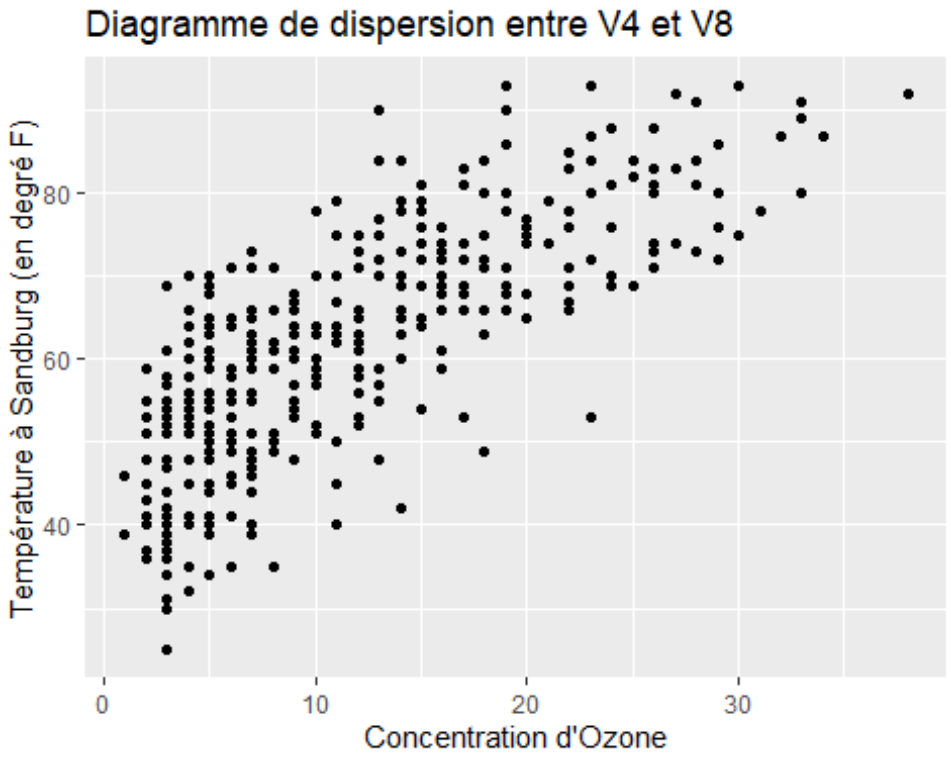
```r
ggplot(Ozone_1, aes(x = V4)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black", alpha = 1)
+
  labs(title = "Histogramme de la concentration en Ozone",
       x = "Concentration en Ozone",
       y = "Fréquence")
```




```r
ggplot(Ozone_1, aes(x = factor(V1), y = V4)) +
  geom_boxplot(fill = "skyblue", color = "black", alpha = 1) +
  labs(title = "Boxplot de la concentration en Ozone par mois",
       x = "Mois",
       y = "Concentration en Ozone")
```

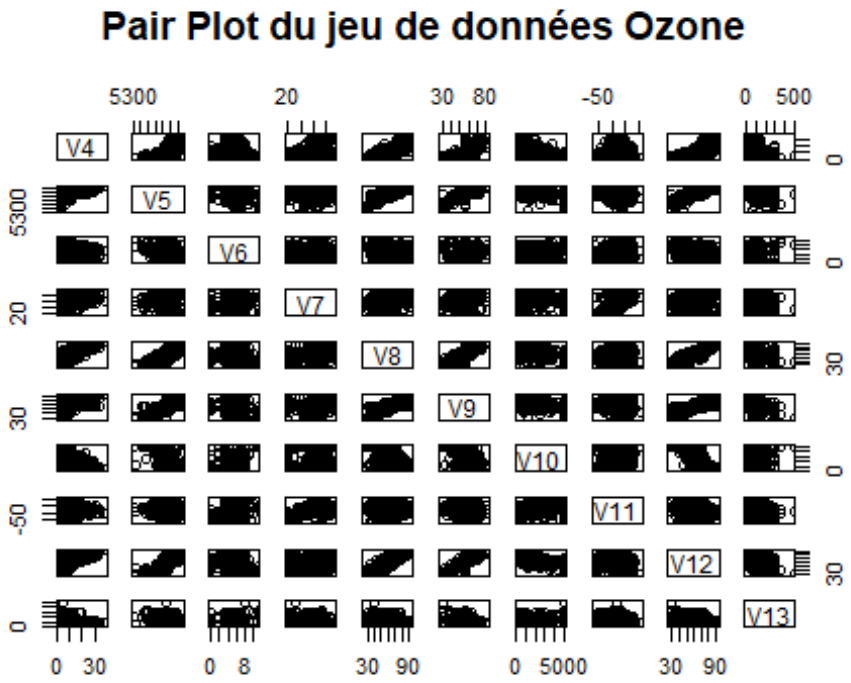



```
ggplot(Ozone, aes(x = V4, y = V8)) +
  geom_point() +
  labs(title = "Diagramme de dispersion entre V4 et V8", x = "Concentration
d'Ozone", y = "Température à Sandburg (en degré F)")

## Warning: Removed 7 rows containing missing values (`geom_point()`).
```



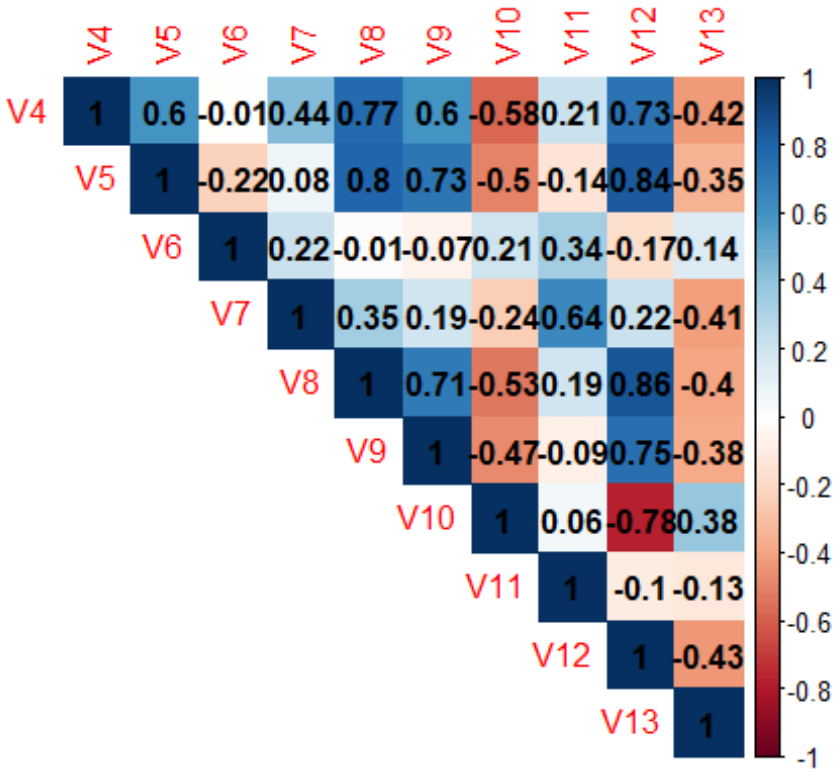
On peut aussi regarder si les variables sont liées. Pour cela on regarde la corrélation entre les variables quantitatives.

```
pairs(Ozone_1[, 4:13], main = "Pair Plot du jeu de données Ozone")
```



Enfin on peut aussi aussi les corrélations entre les varaibles grâce au packages “corrplot”

```
library(corrplot)

## Warning: le package 'corrplot' a été compilé avec la version R 4.3.2

## corrplot 0.92 loaded

matrice_corr <- cor(Ozone_1[,4:13], use = "complete.obs")
corrplot(matrice_corr, method = "color", type = "upper", addCoef.col =
"black")
```



Ce qui nous permet de distingué une forte corrélation entre les variables V5, V8, V9 et V12 par rapport à V4 qui est la variable que l'on cherche à estimé.

Exploration non supervisée avec l'algorithme des k-means

Utiliser l'algorithme des k-means pour continuer votre analyse exploratoire. Attention de sélectionner et ou transformer vos variables avant d'appliquer la méthode. Vous pourrez suivant les cas utiliser la technique sur les individus ou les variables.

Commenter

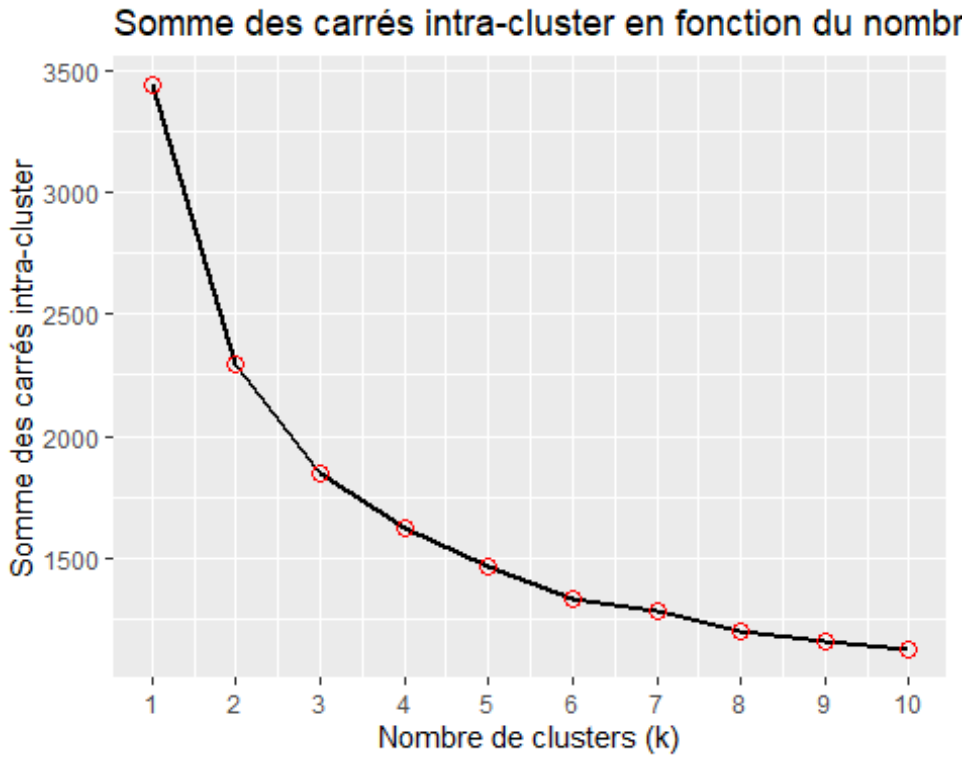
Dans un premier temps, on normalise (ie centre et réduit) notre jeu de donnée pour éviter que la méthode des K-means soit biaisé car une valeur est beaucoup trop éloigner des autres.

```
K_means_data <- scale(Ozone_1[,4:13])
```

Le but maintenant est de déterminer le nombre optimal de "Clusters" pour utiliser notre K-means, on va procéder par la méthode du coude.

```
vecteur_nul <- numeric(10)

for (i in 1:10) {
  kmeans_n <- kmeans(K_means_data, centers = i)
  vecteur_nul[i] <- sum(kmeans_n$withinss)
}
```

```r
ggplot() +
  geom_line(aes(x = 1:10, y = vecteur_nul), linewidth = 1) +
  geom_point(aes(x = 1:10, y = vecteur_nul), color = "red", size = 3, shape = 1) +
  labs(title = "Somme des carrés intra-cluster en fonction du nombre de clusters (k)",
       x = "Nombre de clusters (k)",
       y = "Somme des carrés intra-cluster") +
  scale_x_continuous(breaks = 1:10)
```



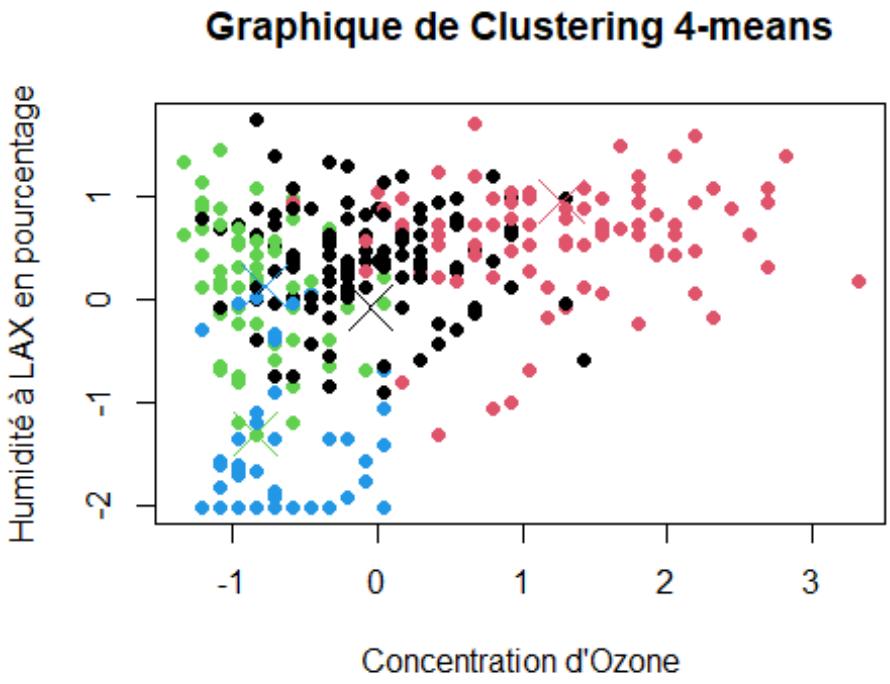
On remarque que d'après la méthode du coude $k = 4$ est un bon choix. Donc on applique la méthode des 4-means pour.

```r
k <- 4
kmeans_n <- kmeans(K_means_data, centers = k)

print(kmeans_n)
```

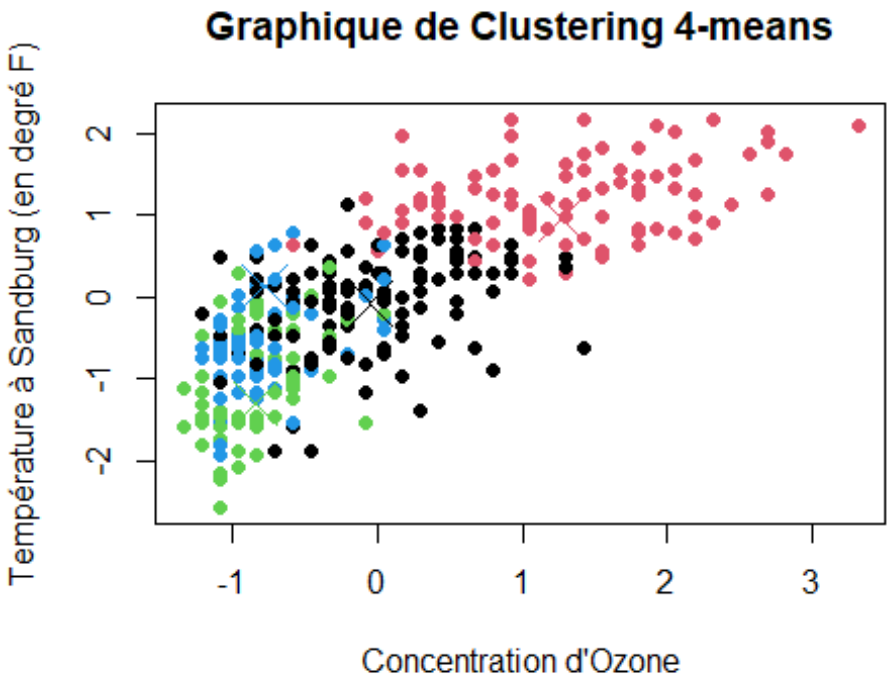
```
## K-means clustering with 4 clusters of sizes 119, 91, 67, 68
##
## Cluster means:
##            V4          V5          V6         V7          V8          V9
## 1 -0.04906841 -0.07176857 -0.08944645  0.3936800 -0.04524232 -0.06927585
## 2  1.26795101  0.95237605  0.08353597  0.5993610  1.22014565  0.98176365
## 3 -0.84413473 -1.29622799  0.74911370  0.1106185 -1.04250358 -1.13524926
## 4 -0.77922609  0.12825758 -0.69335682 -1.6000179 -0.52649527 -0.07404362
```

```
##          V10        V11         V12        V13
## 1 -0.3300919  0.2480233  0.03235575 -0.2771075
## 2 -0.7581275  0.2941891  1.12730978 -0.5903469
## 3  1.2180769  0.6028879 -1.38884997  0.7465659
## 4  0.3920499 -1.4217569 -0.19680259  0.5393742
##
## Clustering vector:
##   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
21  22
##   4   4   1   1   4   4   1   1   4   1   4   4   4   4   4   1   1   4
4   4
##  23  24  25  26  27  29  30  31  32  33  34  35  36  37  38  39  40  41
42  43
##   1   3   1   1   4   4   4   4   4   1   1   3   3   3   3   3   3   3
4   3
##  44  45  46  47  48  49  51  52  53  54  55  56  57  58  59  60  61  62
63  64
##   1   3   3   1   1   1   3   4   4   3   3   1   1   1   1   3   3   3
3   3
##  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82
83  84
##   3   4   3   3   3   3   3   4   4   1   1   1   2   1   1   4   4   4
1   1
##  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101 102
103 104
##   1   4   3   1   1   4   1   1   1   3   3   3   3   1   3   3   1   3
3   3
## 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
123 124
##   3   3   3   3   1   1   2   1   3   1   1   1   1   3   3   1   2   2
1   3
## 125 126 127 128 129 130 131 132 133 134 135 136 138 139 140 141 143 145
146 147
##   1   3   3   1   1   1   2   2   2   2   2   2   2   1   1   1   1   3
1   2
## 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165
166 167
##   2   1   1   1   1   1   2   1   1   1   1   1   1   3   3   3   1   1
2   2
## 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185
186 187
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   1   2
2   2
## 188 189 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206
207 208
##   2   2   2   2   1   2   2   2   2   2   2   2   2   2   2   2   2   2
2   2
## 209 210 211 212 213 214 215 216 222 223 224 225 226 227 228 229 230 231
232 233
##   2   2   2   1   1   1   1   1   2   2   2   2   2   1   3   3   1   3
```

```
3   1
## 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251
252 253
##   1   1   2   2   2   2   2   2   2   2   2   2   2   2   2   1   1   2
2   1
## 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271
272 273
##   1   3   1   2   2   1   3   1   1   1   1   2   2   1   1   3   3   1
1   1
## 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291
292 293
##   3   3   3   1   2   2   2   2   4   4   2   2   2   2   2   1   1   2
1   1
## 294 295 296 297 298 299 300 301 302 303 304 305 308 310 312 314 315 316
317 318
##   1   1   3   1   1   1   4   4   4   1   1   4   4   4   2   1   1   3
3   3
## 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336
337 338
##   3   4   4   4   4   1   1   1   1   1   1   1   3   4   4   4   4   4
4   4
## 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356
357 358
##   1   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4   4
4   1
## 359 362 363 364 365
##   1   4   4   1   3
##
## Within cluster sum of squares by cluster:
## [1] 469.2579 342.1072 411.7400 386.3772
##  (between_SS / total_SS =  53.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
"tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"

plot(K_means_data[, 1], K_means_data[, 4], col = kmeans_n$cluster, pch = 16,
main = "Graphique de Clustering 4-means" ,xlab = "Concentration d'Ozone",
ylab = "Humidité à LAX en pourcentage")
points(kmeans_n$centers[, 1], kmeans_n$centers[, 2], col = 1:k, pch = 4, cex
= 3)
```




```
plot(K_means_data[, 1], K_means_data[, 5], col = kmeans_n$cluster, pch = 16,
main = "Graphique de Clustering 4-means" ,xlab = "Concentration d'Ozone",
ylab = "Température à Sandburg (en degré F)")
points(kmeans_n$centers[, 1], kmeans_n$centers[, 2], col = 1:k, pch = 4, cex
= 3)
```



On remarque que quatre groupes représente nos données, on peux donc prédire la concentration en Ozone d'un future points en fonction de son appartenance aux groupes.

Prédiction avec un classifieur de Bayes naif

Utiliser un classifieur de Bayes naif pour prédire **Daily maximum one-hour-average ozone reading** à partir de certaines autres variables du jeu de données. Vous donnerez une estimation de l'erreur de prédiction.

```
library(e1071)
```

Dans un premier temps, on sépare notre jeu de donnée entre ce que l'on doit prédire et les variables qui vont nous aider dans la prédiction. On prend que des variables quantitatives pour prédire.

```
predicteurs <- Ozone_1[,5:13]
variable_a_predire <- Ozone_1[,4]
matrice_vap <- as.matrix(as.numeric(variable_a_predire))
```

On divise notre jeu de donnée de prédicteurs en deux pour avoir une partie qui entraîne l'algorithme (50%) et l'autre pour le testé (50%). On utilise la fonction "set.seed(123)" pour reproduire les résultats car le choix de la divisions des données est aléatoire.

```
set.seed(123)
indices <- sample(1:nrow(Ozone_1), 0.5 * nrow(Ozone_1))
```

```
train_data <- Ozone_1[indices, ]
test_data <- Ozone_1[-indices, ]
```

On s'assure que que les dimension sont correcte.

```
dim(predicteurs)

## [1] 345   9

dim(matrice_vap)

## [1] 345   1
```

On entraîne notre algorithme et on constate l'estimation de l'erreur de prédiction.

```
BayesNaif <- naiveBayes(V4 ~ ., data = train_data)
predictions <- predict(BayesNaif, test_data)
error.e1071 <- mean(predictions == test_data$V4)
cat("Test Error : ", error.e1071)

## Test Error :  0.132948
```

On obtient alors un erreur de 13% ce qui est assez grand, cela est peux être dû à la division dans notre jeu de donnée en 50/50, une répartition de 75% pour l'entraînement et 25% pour les tests aurait sûrement été plus judicieux.